МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖНОЙ ПОЛИТИКИ КРАСНОДАРСКОГО КРАЯ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
КРАСНОДАРСКОГО КРАЯ
«НОВОРОССИЙСКИЙ КОЛЛЕДЖ РАДИОЭЛЕКТРОННОГО ПРИБОРОСТРОЕНИЯ»
(ГБПОУ КК НКРП)

Комплект оценочных средств

для проведения, текущего, рубежного контроля и промежуточной аттестации в форме дифференцированного зачета

по МДК **МДК01.02 Прикладное программирование**

(индекс и наименование учебной дисциплины(МДК)

в рамках программы подготовки специалистов среднего звена (ППССЗ)

по специальности СПО

09.02.03 «Программирование в компьютерных системах»

(код и наименование специальности)

2021

СОГЛАСОВАНО

Зам. директора по УМР

_____ Е.В. Кужилева

02 07 2021 г.

УТВЕРЖДАЮ

Зам. директора по УР

_____ Т.В.Трусова

02 07 2020 г.

Одобрен

УМО общепрофессиональных
и специальных дисциплин
специальностей 09.02.03, 10.02.01, 10.01.03
Протокол от 01 июля 2021 г. № 11
Председатель УМО
_____ О.А. Афиногенова

Комплект оценочных средств для проведения промежуточной аттестации по МДК01.02. Прикладное программирование разработан на основе федерального государственного образовательного стандарта среднего профессионального образования по специальности 09.02.03 Программирование в компьютерных системах (утв. приказом Министерства образования и науки Российской Федерации от 28.07.2014 г. № 804, зарегистрирован в Минюст России от 21.08.2014 г. № 33733), рабочей программы ПМ01 Разработка программных модулей программного обеспечения для компьютерных систем (утв. директором колледжа), Положения по организации текущего контроля успеваемости и промежуточной аттестации обучающихся ГБПОУ КК НКРП (утв. директором колледжа), Положения по формированию КОС по дисциплине, МДК (утв. директором колледжа)

Организация-разработчик: ГБПОУ КК «Новороссийский колледж радиоэлектронного приборостроения» (далее ГБПОУ КК НКРП)

Разработчик:

преподаватель ГБПОУ КК НКРП
(должность, место работы)

_____ Барилова С.В.
(подпись)

Рецензенты:

_____ Черникевич Д.В., ведущий инженер-программист
ПАО «Новороссийское морское пароходство»
(ФИО, должность место работы)

_____ Афиногенова О.А., преподаватель высшей категории ГБПОУ КК НКРП
(ФИО, должность место работы)

РЕЦЕНЗИЯ

на комплект оценочных средств МДК.01.02.Прикладное программирование
Направление подготовки (специальность) 09.02.03 Программирование в компьютерных системах
Комплект оценочных средств подготовлен *Бариловой С.В.*

КОС МДК.01.02 Прикладное программирование разработан на основе рабочей программы профессионального модуля.

Предназначен для подготовки оценочных материалов, обеспечивающих проведение дифференцированного зачета по МДК.

КОС состоит из следующих разделов:

- раздел «Паспорт комплекта оценочных средств», характеризующий область применения и нормативные основания разработки КОС; сводные сведения об объектах оценивания, показателях и критериях оценивания, типах заданий; формах аттестации;

- раздел «Комплект оценочных средств», структура которого позволяет разрабатывать и комплектовать разные типы заданий для обучающихся.

В паспорте указаны знания и умения в соответствии с рабочей программой профессионального модуля, показатели и критерии оценки знаний обучающихся по каждому объекту оценивания, а также формы и методы контроля.

Комплект оценочных средств включает в себя типовые задания для проведения дифференцированного зачета по каждому объекту оценивания.

Задания разработаны конкретно, последовательно, технически грамотно и позволяют проверить знания и умения по профессиональному модулю. Предлагаемый программой перечень заданий обеспечивает приобретение умений и навыков у студентов.

Таким образом, КОС МДК 01.02 Прикладное программирование обеспечивает проведение промежуточного контроля знаний студентов и может быть использована в учебном процессе Новороссийского колледжа радиоэлектронного приборостроения.

Рецензент: _____ Черникевич Д.В., ведущий инженер-программист
ПАО «Новороссийское морское пароходство»
01 июля 2021 г.

РЕЦЕНЗИЯ

на комплект оценочных средств МДК 01.02 Прикладное программирование

Направление подготовки (специальность) 09.02.03 Программирование в компьютерных системах

Комплект оценочных средств подготовлен *Бариловой С.В.*

Комплект оценочных средств МДК 01.02 Прикладное программирование на основе рабочей программы профессионального модуля.

КОС состоит из следующих разделов:

- раздел «Паспорт комплекта оценочных средств», характеризующий область применения и нормативные основания разработки КОС; сводные сведения об объектах оценивания, показателях и критериях оценивания, типах заданий; формах аттестации;
- раздел «Комплект оценочных средств», структура которого позволяет разрабатывать и комплектовать разные типы заданий для обучающихся.

В паспорте указаны знания и умения в соответствии с рабочей программой профессионального модуля, показатели и критерии оценки знаний обучающихся по каждому объекту оценивания, а также формы и методы контроля.

Комплект оценочных средств включает в себя типовые задания для проведения дифференцированного зачета по каждому объекту оценивания.

Задания разработаны конкретно, последовательно, технически грамотно и позволяют проверить знания и умения по профессиональному модулю, а также сформированность соответствующих профессиональных компетенций.

КОС МДК 01.02 Прикладное программирование может быть использован в учебном процессе ГБПОУ КК «Новороссийского колледжа радиоэлектронного приборостроения».

Рецензент: _____ О.А. Афиногенова

подпись расшифровка

преподаватель высшей аттестационной категории ГБПОУ КК НКРП

01 июля 2021 г.

1 Паспорт комплекта оценочных средств

1.1 Область применения комплекта оценочных средств

Комплект оценочных средств (КОС) предназначен для оценки результатов освоения МДК.01.01 Прикладное программирование

КОС включает контрольные материалы для проведения текущего контроля и промежуточной аттестации в форме дифференцированного зачета.

Результаты освоения (объекты оценивания)	**Основные показатели оценки результата и их критерии**	**Тип задания; № задания**	**Форма аттестации (в соответствии с учебным планом)**
уметь:			
осуществлять разработку кода программного модуля на современных языках программирования;	**Показатель**: соблюдение требований к проектированию и разработке программного проекта в объектно-ориентированной среде программирования; **Критерий**: программный проект соответствует правилам хорошего стиля программирования и требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практических занятий №1-82 Дифференцированный зачет
создавать программу по разработанному алгоритму как отдельный модуль;	**Показатель**: разработка кода программного модуля в соответствии со спецификацией; **Критерий**: программный модуль выполняет все функции в соответствии со спецификацией и требованиям,	Практические задания	Текущий контроль при выполнении практических занятий №1-82 Дифференцированный зачет

	определенным в практических заданиях		
выполнять отладку и тестирование программы на уровне модуля;	**Показатель**: выполнение и тестирования программы; **Критерий**: совпадение ожидаемых результатов с конечными результатами и соответствие требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практических занятий №1-82 Дифференцированный зачет
оформлять документацию на программные средства	**Показатель**: разработка спецификации программного модуля, документации по сопровождению программного продукта в соответствии с ЕСПД и ISO; **Критерий**: оформление спецификации и документации по сопровождению программного продукта произведено полно и точно	Курсовая работа	Текущий контроль при выполнении курсовой работы
программировать работу с базой данных	**Показатель**: разработка кода программного модуля в соответствии со спецификацией для работы с базой данных; **Критерий**: программный модуль выполняет все функции в соответствии со спецификацией и требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практических занятий №37-44, 57, 58, 81, 82

работать с распределенными базами данных	**Показатель**: разработка кода программного модуля в соответствии со спецификацией для работы с базой данных InterBase; **Критерий**: программный модуль выполняет все функции в соответствии со спецификацией и требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практических занятий №43
работать с фреймворком FireMonkey	**Показатель**: разработка кода мобильного приложения в соответствии со спецификацией; **Критерий**: мобильное приложение выполняет все функции в соответствии со спецификацией и требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практического занятия №46
создавать WPF приложения в VisualStudio	**Показатель**: разработка кода прикладного приложения по технологии WPF в соответствии со спецификацией; **Критерий**: прикладного приложения по технологии WPF выполняет все функции в соответствии со спецификацией и требованиям, определенным в практических заданиях	Практические задания	Текущий контроль при выполнении практических занятий №72-82

знать:			
основные принципы отладки и тестирования программных продуктов;	перечисление и четкое изложение основных принципов отладки и тестирования программных продуктов	Устный опрос	Текущий контроль
методы и средства разработки технической документации	изложение методов и перечисление средств разработки технической документации в соответствии с ЕСТД;	Тестирование	Текущий контроль
работа с базами данных в сети	принципы, компоненты технологии InterBase	Устный опрос	Текущий контроль
операционные системы Android, IOS	особенности операционных систем Android, IOS, разработка приложений под Android, IOS	Тестирование	Текущий контроль
технология WPF	Особенности платформы WPF, введение в язык XAML	Устный опрос	Текущий контроль

2 Комплект оценочных средств

2.1 Задания для проведения контроля практических занятий:

Практическое занятие 1 Знакомство с интегрированной средой разработки C++ Builder

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки эффективного использования интегрированной среды C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

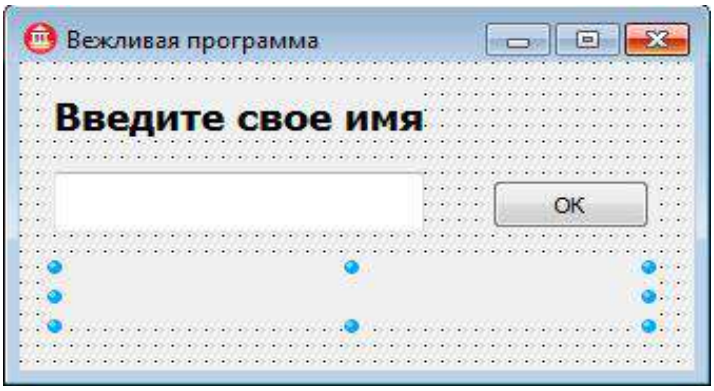
Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1 Создать в своей папке папку «Прикладное программирование»;

2 Загрузить C++ Builder, сохранить новый проект в папку;

3 Разработать форму в соответствии с образцом (рис. 3), для этого установить на форму компоненты (вкладка **Standard**) и изменить им свойства в соответствии с таблицей.



Объект	Свойство	Значение
TLabel	Name	Label1
	Caption	Введите свое имя
	Font…	Полужирный, 14
TLabel	Name	Label2
	Caption	
	Font…	Полужирный, 14, желтый
TEdit	Name	Edit1
	Text	
	Font	полужирный, 14
TButton	Name	Button1
	Caption	OK
	Font	полужирный, 12
Форма	Caption	Вежливая программа

4 Создать процедуру обработки события **OnClick** для объекта Button1, для этого:

- выделить объект;
- на вкладке **Events** инспектора объектов выбрать событие **OnClick**, выполнить двойной щелчок в правом столбце строки;

5 В появившемся окне редактора кода ввести следующий код:

```
if(Edit1->Text==" ") Label2->Caption="Забыл имя?";
else Label2->Caption="Привет, "+Edit1->Text;
```

6 Проверить работу приложения, для этого:

- загрузить приложение (**F9** или в главном меню **Run/Run**);
- в поле текстового окна ввести произвольный текст;
- щелкнуть по кнопке **Ok**;
- изменить введенный текст, **Ok**, просмотреть результат;
- закрыть окно приложения.

7 Добавить на форму новые компоненты с вкладки **Standart** и изменить для них свойства.

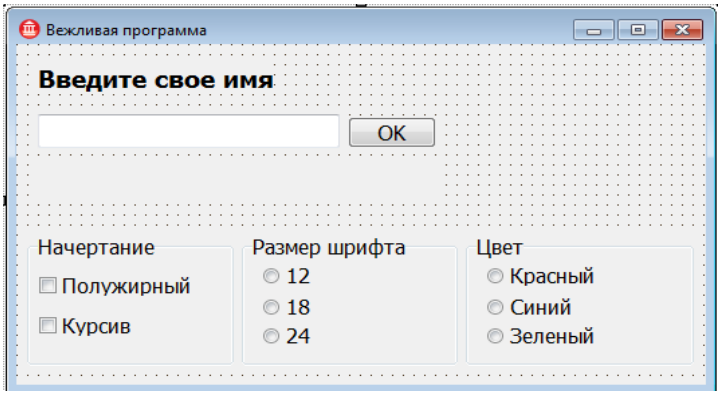


Рисунок 1 – Вид формы

Объект	Свойство	Значение
TGroupBox	Name	GroupBox1
	Caption	Начертание
	Font…	12
	Name	GroupBox2
	Caption	Размер шрифта
	Font…	12
	Name	GroupBox3
	Caption	Цвет
	Font…	12
TCheckBox	Name	CheckBox1
	Caption	Полужирный
	Font…	12
	Name	CheckBox2
	Caption	Курсив
	Font…	12
TRadioButton	Name	RadioButton1
	Captio	12
	Font	12
	Name	RadioButton2
	Caption	18
	Font…	12
	Name	RadioButton3
	Caption	24
	Font…	12
	Name	RadioButton4
	Caption	Красный
	Font…	12
	Name	RadioButton5
	Caption	Синий
	Font…	12
	Name	RadioButton6
	Caption	Зеленый
	Font…	12

8 Для объектов CheckBox1 и CheckBox2 введите код обработки события OnClick

```
if(CheckBox1->Checked==true)
Edit1->Font->Style=TFontStyles()<<fsBold;

if(CheckBox2->Checked==true)
Edit1->Font->Style=TFontStyles()<<fsItalic;
```

9 Для объекта RadioButton1 введите код обработки события OnClick

```
Edit1->Font->Size=12;
```

10 Для объекта RadioButton4 введите код обработки события OnClick

```
Edit1->Font->Color=clRed;
```

11 Самостоятельно создать функции для остальных объектов RadioButton, записать обработчики событий в тетрадь.
12 Проверить работу приложения.
13 Сделать всплывающую подсказку для текстового окна, для этого:
- Выделите объект **Edit1**;
- Для свойства **Hint** введите значение **ПОЛЕ ВВОДА**;
- Для свойства **ShowHint** выберите значение **True**;

14 Измените вид курсора при наведении на командную кнопку, для этого:
1. Выделите объект Button1;
2. Для свойства Cursor выберите из списка любое значение.

15 Сохранить проект.

Практическая часть

1 Создать новый проект.

2 Добавить на форму компоненты с вкладки Standart.

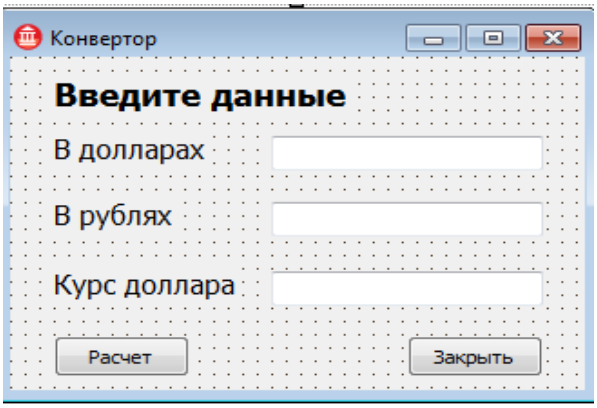


Рисунок 2 – Форма

3 Для объекта Button1 (Расчет) ввести код обработки события OnClick:

```
float a,b,c;
a=StrToFloat(Edit1->Text);
c=StrToFloat(Edit3->Text);
b=a*c;
Edit2->Text=FloatToStr(b);
```

4 Для объекта Button2 ввести код обработки события OnClick:

```
Form1->Close();
```

5 Проверка ввода значений ShowMessage(“Сообщение”);

```
if ((Edit1->Text==””)&&(Edit3->Text==””))
        ShowMessage(“Введите данные”);
else
        {…}
```

6 Добавить кнопку Очистить.

Контрольные вопросы:

1. Как можно открыть существующий проект в C++ Builder?

2. Сколько файлов создается при сохранении проекта?
3. Как выделить несколько объектов на форме?
4. Как можно изменить значение свойства объекта?
5. Как загрузить объект на выполнение?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 2 Первые прикладные приложения (программы "Приветствие", "Конвертор")

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки эффективного использования интегрированной среды C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Спроектировать окно:

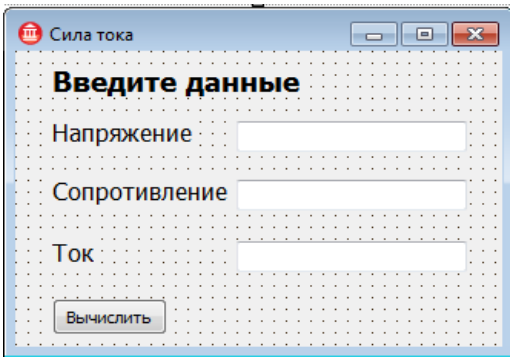


Рисунок 1 – Форма

2. Вычислить силу тока по формуле I=U/R;
3. Добавить кнопку Завершить.
4. Проверить введены ли данные:

```
if(Edit1->Text.Lenght()==0||Edit2->Text.Lenght()==0)
{
MessageDlg("Надо ввести напряжение и сопротивление", mtInformation, TMsgDlgButtons()<<mbOK,0);
If (Edit1->Text.Lenght()==0)
Edit1->SetFocus();
else
Edit2->SetFocus();
}
else {…}
```

5. Сохранить и показать результат преподавателю.
6. Создать новое приложение.
7. Спроектировать окно:

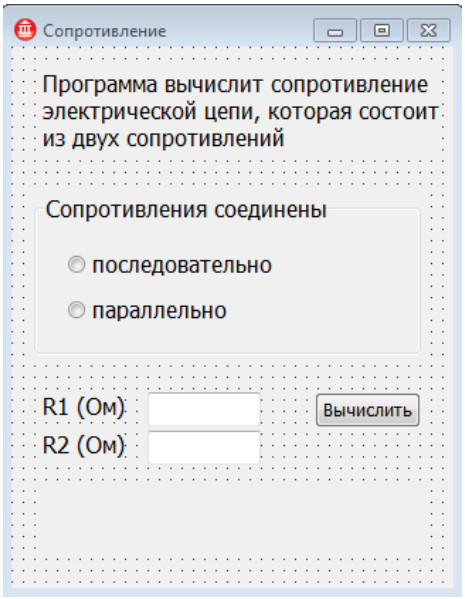


Рисунок 2 – Форма

8. Добавить проверку ввода данных и передачу фокуса.
9. Сохранить и показать результат преподавателю.

Практическая часть

10. Создать новый проект.
11. Спроектировать окно:

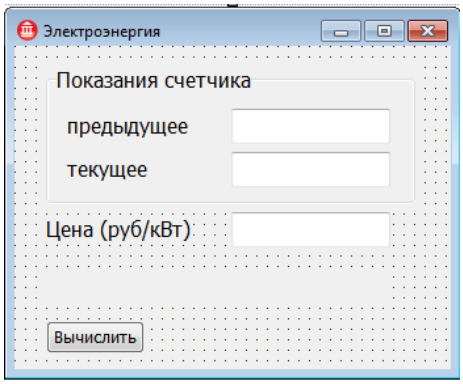


Рисунок 3 – Форма

12. Добавить проверку ввода данных.

Контрольные вопросы:

1. Как программно закрыть выполняющееся приложение?
2. С помощью какого свойства можно проверить выбрана ли радиокнопка?
3. С помощью какого события программируется реакция приложения на нажатие клавиш клавиатуры?
4. С помощью какого метода можно передать фокус элементу?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 3 Обработка навыков визуального программирования

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки эффективного использования интегрированной среды C++ Builder, работа с компонентом TTimer

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Проект 1. Создать программу, выполняющую действия:

- По щелчку мышью на кнопке кнопка либо останавливается, либо двигается;
- Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

1 Открыть новый проект.

2 Разместить на форме экземпляры компонентов: кнопку Button, таймер Timer. Кнопка включает и выключает таймер, а таймер двигает кнопку.

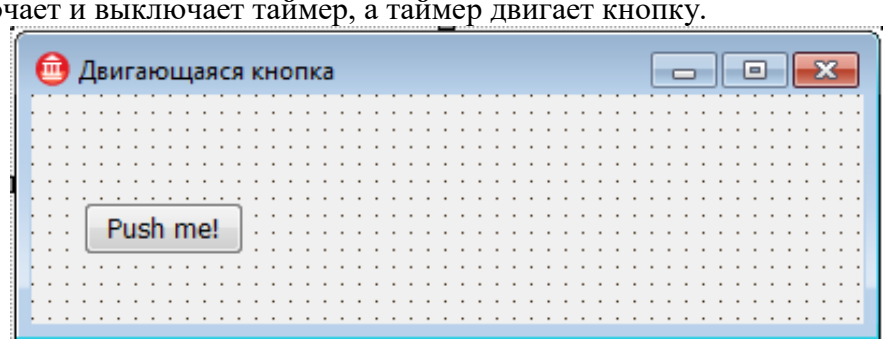


Рисунок 1 – Вид формы

3 Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
Form1	Properties	Caption	Установка имени формы «Двигающаяся кнопка»
Timer1	Properties	Enabled	Установить значение свойства Enabled=false. Свойство Enabled определяет включен или выключен таймер (по умолчанию он выключен)
		Interval	Interval=100 Свойство Interval определяет интервал в миллисекундах между возникновением событий OnTimer (по умолчанию интервал равен 1 секунде)
	Events	OnTimer	Button1->Left=Button1->Left-5; if(Button1->Left<10) Button1->Left=100;
Button1	Properties	Caption	Установка имени кнопки «Сменить заголовок окна»
		Default	Выбрать в раскрывающемся списке значение True
	Events	OnClick	Timer1->Enabled=false;

4 Сохранить проект, запустите и протестируйте его.

Проект 2. Создать программу-игру, выполняющую следующие действия:

– После запуска программы в окне изображается беспорядочно прыгающая кнопка;
– Необходимо успеть щелкнуть по ней;
– Кнопка перепрыгивает из одного места в другое по сигналу, полученному от таймера;
– Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

5 Открыть новый проект;

6 Разместить на форме экземпляры компонентов: командная кнопка Button, таймер Timer.

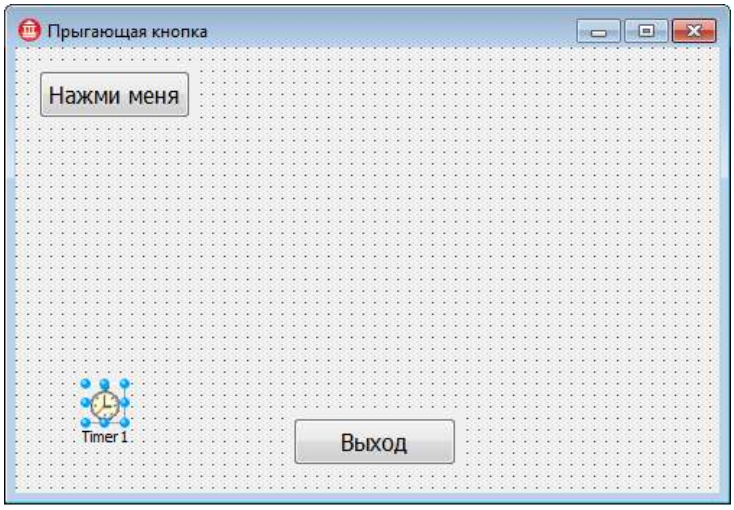


Рисунок 2 – Вид формы

7 Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/имя события	Действие
Form1	Properties	Caption	Установка имени формы «Прыгающая кнопка»
		ClientWidth (Внутренняя ширина)	Присвоить значение 300
		ClientHeight (Внутренняя высота)	Присвоить значение 200
		BorderStyle (тип границы)	Выбрать значение bsSingle (тонкая)
	Events	OnCreate	Random();
Timer1	Properties	Interval (интервал)	Присвоить значение 500 (промежуток времени измеряется в миллисекундах)

	Events	OnTimer	int i;
			{
			i=random(9);
			Button1->Visible=true;
			Button1->Top=10+50*(i/3);
			Button1->Left=10+100*(i%3);
			}

Button1	Properties	TabStop	Присвоить значение false. Это свойство разрешает выбрать данный элемент управления клавишей Tab. Клавиатурой пользоваться запрещается.
		Visible	Присвоить значение false. Сначала кнопка невидима.
		Height	Присвоить значение 30
		Width	Присвоить значение 80

	Events	OnClick	Button1->Caption=«готово»; Button1->Enabled=false; Timer1->Enabled=false;
Button2	Properties	Caption	Ввести надпись «Выход»
		Default(по умолчанию)	Выбрать значение true
		Left (слева)	Присвоить значение 110
		Top (сверху)	Присвоить значение 160
		Width (ширина)	Присвоить значение 80
		Height (высота)	Присвоить значение 30
	Events	OnClick	Close;

8 Сохранить проект, запустите и протестируйте его.

Практическая часть

1. Изменить игру так, чтобы скорость можно было настраивать в процессе игры.
2. Создайте две кнопки: Медленнее и Быстрее. Щелчок на одной из них будет увеличивать или уменьшать значение свойства Timer1->Interval на 100 миллесекунд.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 4 Базовые компоненты

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

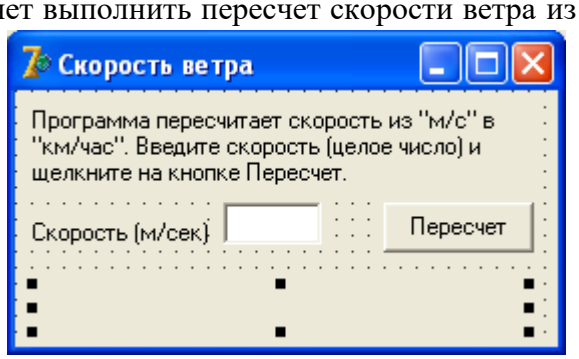
Цель: Получить навыки использования в программе базовых компонентов

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Разработать приложение, которое позволяет выполнить пересчет скорости ветра из «метров в секунду» в «километры в час».



Порядок выполнения:

1. Загрузить C++ Builder;
2. Установить на форму 3 компонента **Label**, компонент **Edit** и **Button**.
3. Привести форму в соответствие с образцом;
4. Для командной кнопки ввести следующий программный код:

```
int   ms;                          // скорость м/с
float kmh;                         // скорость км/час
 ms  = StrToInt(Edit1->Text);      // ввести исходные данные
 kmh = ms * 3.6;                   // пересчитать
 Label3->Caption = IntToStr(ms) + " м/с - это " + FloatToStr(kmh) + " км/час";
                                   // вывести результат
```

5. Проверить работу приложения на различных значениях скорости. Проверить реакцию программы, если поле ввода осталось пустым.
6. Модернизировать приложение таким образом, организовать проверку ввода значения в обработчик события Button1Click добавить следующий код (выделен полужирным шрифтом):

```
if (Edit1->Text. Length()== 0 )
   ShowMessage("Надо ввести скорость");
```

7. Проверить работу приложения на различных значениях (целых и вещественных).
8. Модернизировать приложение для того, чтобы вычисление выполнялось не только при щелчке по кнопке **Пересчет**, но и при нажатии **Enter** после ввода последней цифры в поле **Скорость**.
9. Проверить работу приложения.
10. Сохранить проект. Результат показать преподавателю.

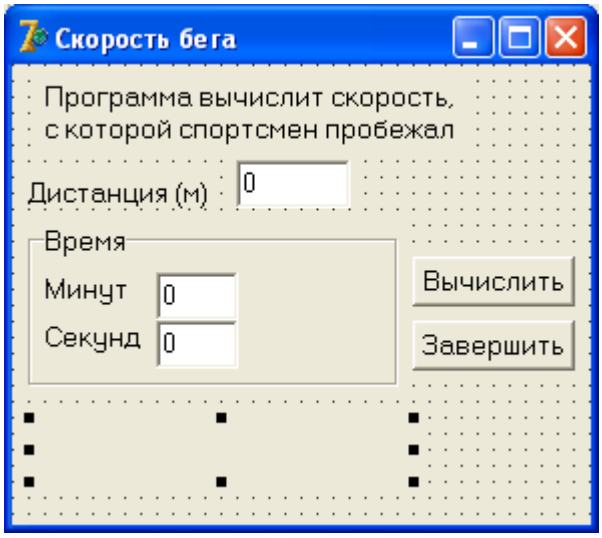
Практическая часть

Задание №1

Разработать приложение для пересчета массы из фунтов в килограммы (1 фунт = 409,5 грамм). Кнопка **Пересчет** должна быть доступна только в том случае, если пользователь ввел исходные данные. Разрешается вводить целые и вещественные значения (разделитель точка и только одна). Добавить на форму кнопку, при щелчке по которой удаляются значения из полей ввода и вывода.

Задание №2

Разработать приложение, которое вычисляет скорость (км/час), с которой спортсмен пробежал дистанцию. Рекомендуемый вид формы приведен на рисунке. Количество минут задается целым числом, секунд – вещественным.



Контрольные вопросы:

1. Когда происходит событие OnKeyPress?
2. Значения какого типа возвращают объекты Edit, Label?
3. Какие объекты позволяют вводить (выводить) значения?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 5 Применение компонентов VCL

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования в программе компонентов библиотеки VCL
Оборудование: ноутбук Dell Inspiron 15 5000 Series
Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Создать компьютерную версию одной из головоломок Самуэля Ллойда: из заданного набора чисел надо выбрать те, сумма которых составит 50. Числа, которые избрал Ллойд для своей головоломки: 25, 27, 3, 12, 6, 15, 9, 30, 21, 19.

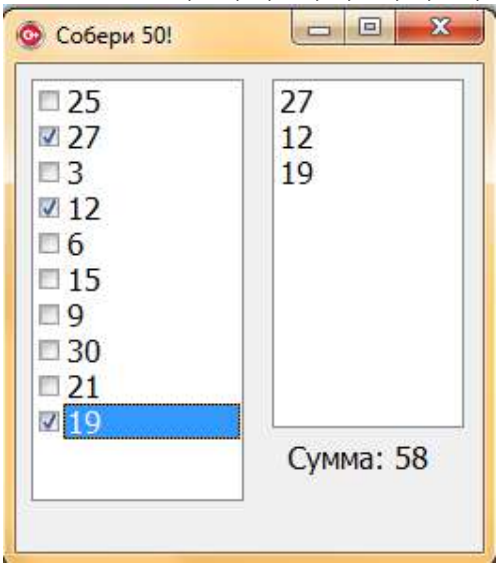


Рисунок 1 – Образец формы

Работа программы:

— После запуска программы в окне изображается список чисел Ллойда.
— Выбираем с помощью флажков числа и помещаем в правое окно.
— Сумма выбранных чисел представлена в виде надписи.
— Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

2. Открыть новый проект.
3. Разместить на форме экземпляры компонентов: список флажков CheckListBox, надпись Label, список ListBox.
4. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Установка имени формы "Головоломка"
		BorderStyle	Задайте значение bsSingle
CheckListBox 1 (Вкладка Additional)	Properties	Items	Задаем состав списка. Щелкнуть на кнопке построителя. Откроется окно String List editor (Редактор списка строк). Введите в список заданные числа через Enter. Нажмите кнопку OK.
		Height	Подобрать значение так, что все числа поместились в список (без полос прокруток).
	Events	OnClickCheck	Описать очистку списка. Проверить, установлен флажок или нет. После обновления списка необходимо подсчитать сумму выбранных чисел. Элементы списка выглядят как числа, но являются текстовыми строками (воспользоваться функцией StrToInt).

5. Сохраните проект, запустите и протестируйте его.

Листинг OnClickCheck для CheckListBox1

```
int s=0;
ListBox1->Clear();
for (int i=0; i<CheckListBox1->Items->Count-1;i++)
{
        if (CheckListBox1->Checked[i])
                ListBox1->Items->Add(CheckListBox1->Items->Strings[i]);
}

for (int i=0;i<ListBox1->Items->Count-1;i++)
        s=s+StrToInt(ListBox1->Items->Strings[i]);

Label1->Caption = "Сумма: " + IntToStr(s);

if (s==50)
{
        Label1->Caption = "Сумма: " + IntToStr (s);
        CheckListBox1->Enabled = false;
        ListBox1->Enabled = false;
 }
```

6. Результат показать преподавателю.

7. Создать компьютерную версию головоломки: из изображенных пяти сброшенных флажков установить все. Но при выборе одного флажка меняется состояние двух следующих.

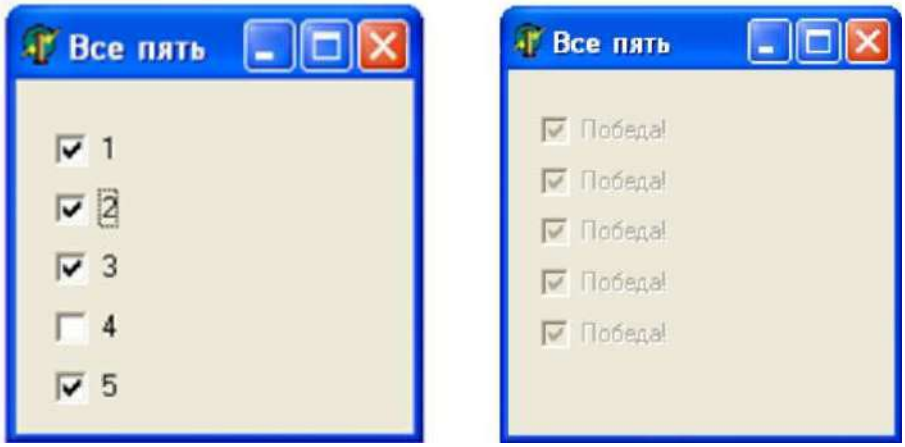


Рисунок 2 – Образец формы

Работа программы:

- После запуска программы в окне изображаются пять сброшенных флажков.
- Щелкать разрешено только на сброшенных флажках. Щелчок на установленном флажке не действует.
- При установке какого-то флажка меняется состояние двух следующих флажков. При этом сброшенные флажки устанавливаются, а установленные - сбрасываются.
- Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

8. Открыть новый проект.
9. Разместить на форме экземпляры компонентов: список флажков CheckBox.

10. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Установка имени формы "Все пять"
	Events	OnCreate	Создать глобальную логическую переменную s. В то время, когда идет обработка события, переменная имеет значение True (да).
CheckBox1	Properties	Caption	Задать значение "1".
	Events	OnClick	Необходимо работать с флажками, как с массивом. Форма является контейнером для флажков. У объектов-контейнеров имеется свойство Controls (Элементы управления) - массив элементов управления, находящихся в данном контейнере: ((TCheckBox *)Controls[Index])
Для создания еще пяти флажков можно использовать метод копирования через буфер обмена.			
CheckBox2	Properties	Caption	Задать значение "2".
CheckBox3	Properties	Caption	Задать значение "3".
CheckBox4	Properties	Caption	Задать значение "4".
CheckBox5	Properties	Caption	Задать значение "5".

11. Сохраните проект, запустите и протестируйте его.

Листинг программы:

```
void __fastcall TForm2::FormCreate(TObject *Sender)
{
s=false;
}
void __fastcall TForm2::CheckBox1Click(TObject *Sender)
{
 int Index, i, num;
/* Если программа снова вызовет процедуру обработки, будет выполнен оператор Exit - немедленный выход из процедуры*/
if (s) exit;
s =true;
// Оператор break прерывает выполнение цикла
for (Index = 0; Index <= 4; Index++)  // Определяется, какой флажок был переключен
/*Когда выполнение цикла завершается, значение переменной Index соответствует переключенному флажку*/
/* Если значение свойства Checked (Установлен) равно False (Нет), флажок сброшен, а если True (Да) - установлен. Номер флажка в массиве определяется переменной Index*/
 if (Sender==Controls[Index]) break;
if ( !((TCheckBox *)Controls[Index])->Checked  )
  ((TCheckBox *)Controls[Index])->Checked=true;
  /*Условие выполнено, если флажок сейчас сброшен, т.е. до щелчка он был установлен*/\
}
 else {
/*Программирование изменения состояния "дополнительных флажков". Текущий флажок уже переключен*/
      num = Index + 1;           //Переключение двух следующих флажков
      if( Index == 4) num = 0; //Изменение состояния нулевого флажка
      // Состояние флажка надо поменять на противоположное
      ((TCheckBox *)Controls[num])->Checked =! ((TCheckBox *)Controls[num])->Checked;
      //Выполнение проверки на выход за пределы массива
      num = num + 1;
```

```
        if( num == 3) num= 0;
        ((TCheckBox *)Controls[num])->Checked = ! ((TCheckBox *)Controls[num])->Checked;
        }
bool e;
e =true; //Головоломка решена, если установлены все пять флажков
for ( int i =0;i<=4;i++)
  e = e && ((TCheckBox *)Controls[i])->Checked;
// После цикла значение останется равным True, если все флажки установлены
 if (e)
// Головоломка решена
        for ( int i =0;i<=4;i++)
          {
          ((TCheckBox *)Controls[i])->Caption ="Победа!";
          ((TCheckBox *)Controls[i])->Enabled =false; /*Отключение флажков*/
           }
s=false;
}
```

Контрольные вопросы:

1. Что такое VCL?
2. Перечислите компоненты VCL, которые вы знаете?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 6 Применение визуальных компонентов

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

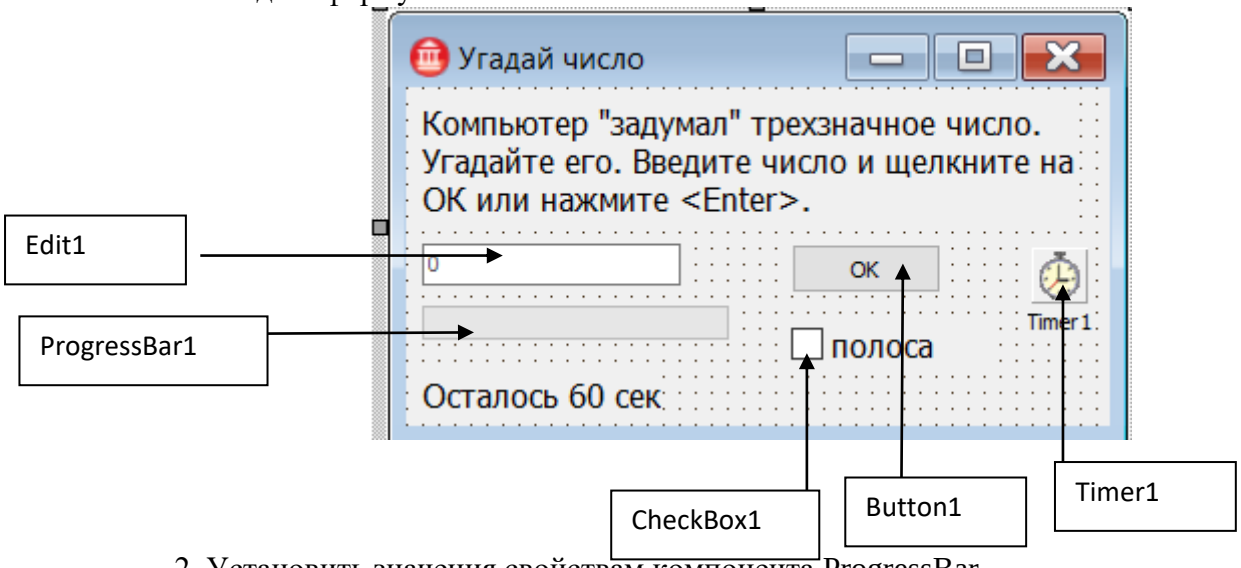
Цель: Получить навыки использования визуальных компонентов.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Создать форму



2. Установить значения свойствам компонента ProgressBar

Min=0

Max=60

Step=1

Position=60

Smooth=true

3.Ниже проведен листинг программы: сгенерировать соответствующие события и записать код:

```
#define TR 60          //время, отведенное на решение задачи
int pw;                // «секретное» число
int rem = TR;          //остаток времени на выполнение задания
//конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)          : TForm(Owner)
{
Label4->Caption= IntToStr(TR);
//**настройка индикатора**
//обратный отсчет, поэтому начальное значение равно максимальному
 ProgressBar1->Max = TR;
 ProgressBar1->Position = TR;
 ProgressBar1->Step = 1;
 ProgressBar1->Smooth = true;
//загадать число
 Randomize();
 {
 pw =Random (899)+999; }}
//сигнал от таймера
```

```
        void __fastcall TForm1::Timer1Timer(TObject *Sender)
        {
          rem--;
         ProgressBar1->Position--;
         Label4->Caption=IntToStr(rem);
         if (rem == 0)
         {
          Timer1->Enabled=false;
          Edit1->Enabled=false;
          Button1->Enabled=false;
          ShowMessage("К сожалению, Вы не справились с поставленной задачей \n
\Секретное\ число:"+IntToStr(pw));
         }
        }
        //щелчок на переключателе Полоса
        void __fastcall TForm1::CheckBox1Click(TObject *Sender)
        {
          if (CheckBox1->Checked)
          {
          ProgressBar1->Step =1;
          ProgressBar1->Smooth =true;
          }
          else
          {
                  ProgressBar1->Step =6;
                  ProgressBar1->Smooth =false;
          }
        }
        //проверяет, правильное ли число ввел игрок
        // функцию добавить в класс формы (см после листинга)
        void __fastcall TForm1::IsRightToLeft(void)
        {
                if (StrToInt(Edit1->Text)==pw)
                {
                  Timer1->Enabled = false;
                  Button1->Enabled = false;
                  Edit1->Enabled = false;
                  ShowMessage ("Поздравляю!\nВы угадали число за "+ IntToStr(TR - rem)+"
сек");
                }
        }
        //щелчок на ОК
        void __fastcall TForm1::Button1Click(TObject *Sender)
        {
          IsRightToLeft();
        }
        //Нажатие клавиши в поле редактирования
        void __fastcall TForm1::Edit1KeyPress(TObject *Sender, System::WideChar &Key)
        {
         if ((Edit1->Text.Length()<3) &&
                 ((Key>= '0') &&
                  (Key<= '9')))
                 return;
         if (Key == VK_RETURN)
```

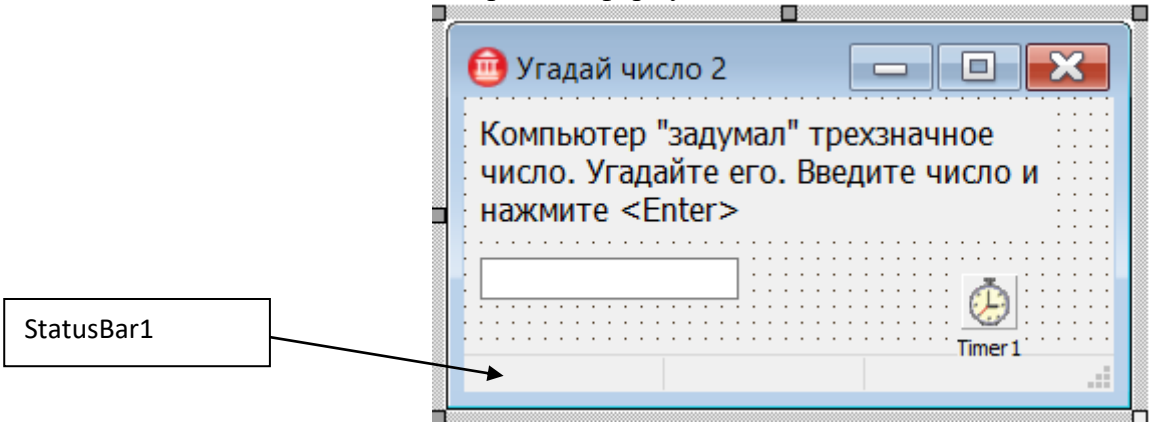
```
 {
  IsRightToLeft();
  return;
 }
 if (Key == VK_BACK) return;
          Key = 0;
}
//Содержимое поля редактирования изменилось
void __fastcall TForm1::Edit1Change(TObject *Sender)
{
        if (Edit1->Text.Length()==3)
                Button1->Enabled = true;
        else
                Button1->Enabled = false;
}
```

4. Функцию добавить в класс формы (Unit1.h)

```
class TForm1 : public TForm
{
__published:   // IDE-managed Components
        TLabel *Label1;
        TEdit *Edit1;
        TButton *Button1;
        TCheckBox *CheckBox1;
        TLabel *Label2;
        TProgressBar *ProgressBar1;
        TTimer *Timer1;
        void __fastcall Timer1Timer(TObject *Sender);
        void __fastcall CheckBox1Click(TObject *Sender);
        void __fastcall Button1Click(TObject *Sender);
        void __fastcall Edit1KeyPress(TObject *Sender, System::WideChar &Key);
        void __fastcall Edit1Change(TObject *Sender);
private:        // User declarations
public:         // User declarations
        __fastcall TForm2(TComponent* Owner);
        void __fastcall IsRightToLeft(void);
};
```

5. Сохранить, протестировать.

6. Создать новый проект и форму:



8. Листинг программы:

```
#define TR 60 //время, отведенное на решение задачи
int pw;         // «секретное» число
int rem = TR;  //остаток времени
```

```
int p =0;                    //количество попыток
//-конструктор формы
__fastcall TForm1::TForm2(TComponent* Owner)          : TForm(Owner)
{
 Randomize();
 pw=Random(100)+999;
}
//сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 rem--;
 StatusBar1->Panels->Items[2]->Text = " Осталось: " + IntToStr(rem) + " сек";
  if (rem == 0)
  {
   Timer1->Enabled=false;
   Edit1->Enabled=false;
   ShowMessage ("К сожалению, Вы не справились с поставленной
задачей\n\"Секретное\" число:" + IntToStr(pw));
  }
}
//Нажатие клавиши в поле редактирования
void __fastcall TForm1::Edit1KeyPress(TObject *Sender, System::WideChar &Key)
{
  if ((Edit1->Text.Length()<3) && ((Key>='0') &&(Key<='9')))
          return;
  if ((Key == VK_RETURN) && (Edit1->Text.Length() == 3))
  {
        if (StrToInt(Edit1->Text)==pw)
        {
                Timer1->Enabled=false;
                Edit1->Enabled=false;
                ShowMessage("Поздравляю!\nВы угадали число за" + IntToStr(TR-
rem)+" сек");
        }
        else
        {
         p++;
         StatusBar1->Panels->Items[1]->Text=" Попыток: " + IntToStr(p);
        }
        return;
  }
        if(Key==VK_BACK) return;
        Key=0;
}
//содержимое поля редактирования изменилось
void __fastcall TForm2::Edit1Change(TObject *Sender)
{
 StatusBar1->Panels->Items[0]->Text= " Символов: " + IntToStr (Edit1-
>Text.Length());
}
```

Контрольные вопросы:

1. Какие визуальные компоненты Вы знаете?
2. Какой формулой характеризуется компонент?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 7 Применение невизуальных компонентов

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования невизуальных компонентов.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. В окне программы Часы находится изображение идущих часов, которые показывают текущее время.

Заголовок для окна установить Часы, на форму перенести компонент таймер.

2.Ниже проведен листинг программы: сгенерировать соответствующие события и записать код:

```
#include "DateUtils.hpp" //для доступа к SecondOf, MinuteOf и HourOf
#include "math.h"        //для доступа к sin и cos
#define R 75             // радиус циферблата часов
int x0, y0;              //центр циферблата
int ahr, amin, asec;     //положение стрелок (угол)
__fastcall TForm1::TForm1(TComponent* Owner)        : TForm(Owner)
```

```
{
TDateTime t;
//зададим размер формы в соответствии с размером циферблата
ClientHeight = (R+30)*2;
ClientWidth = (R+30)*2;
x0=R+30;
y0=R+30;
t=Now();
/*Определить положение стрелок.
Угол между метками (цифрами) часов, например, цифрами 2 и 3, -30 градусов.
Угол между метками минут -6 градусов.
Угол отсчитывается от 12-ти часов */
ahr=90-HourOf(t)*30-(MinuteOf(Today())/12)*6;
amin= 90-MinuteOf(t)*6;
asec=90-SecondOf(Today())*6;
Timer1->Interval=1000;                    //период сигнала от таймера 1 сек
Timer1->Enabled=true;
}
//рисует вектор из точки (x0,y0) под углом a относительно оси X.
//длина вектора l
//пишем процедуру сами и добавляем в класс в раздел public
void __fastcall TForm1::Vector(int x0,int y0,int a, int l)
{
 #define TORAD 0.0174532  //коэф-ты пересчета угла
                                        // из градусов в радианы
 int x,y;                               //координаты конца вектора
 Canvas->MoveTo(x0,y0);
 x=x0+1*cos(a*TORAD);
 y=y0-1*sin(a*TORAD);
 Canvas->LineTo(x,y);
}
// пишем процедуру сами и добавляем в класс в раздел public
void __fastcall TForm1::DrawClock(void)
{
  TDateTime t;
//шаг секундной и минутной стрелок 6 градусов
//часовой -30
//стереть изображение стрелок
  Canvas->Pen->Color =  clBtnFace;
  Canvas->Pen->Width  = 3;;
//часовую
  Vector(x0,y0, ahr, R-20);
//минутную
  Vector(x0,y0, amin, R-15);
//секундную
  Vector(x0,y0, asec, R-7);
  t  =  Now();
//новое положение стрелок
  ahr  =  90 - HourOf(t)*30-(MinuteOf(t)%12)*6;
  amin  =  90 - MinuteOf(t)*6;
  asec  =  90 - SecondOf(t)*6;
//нарисовать стрелки
//часовая стрелка
  Canvas->Pen->Width  =  3;
```

```
  Canvas->Pen->Color =  clBlack;
  Vector(x0,y0,ahr,  R-20);
//минутная стрелка
  Canvas->Pen->Width  =  2 ;
 Canvas->Pen->Color =  clBlack;
 Vector(x0,y0,amin,  R-15);
//секундная стрелка
 Canvas->Pen->Width  =  1;
Canvas->Pen->Color =  clYellow;
Vector(x0,y0, asec, R-7);
}
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 DrawClock();
}
//прорисовка циферблата
void __fastcall TForm1::FormPaint(TObject *Sender)
{
 int x, y;                      //координаты маркера на циферблате
  int a;                        //угол между OX и прямой (x0,y0) (x,y)
  int h;                        //метка часовой риски
  TBrushStyle bs;               //стиль кисти
  TColor pc;                    //цвет карандаша
  int pw;                       //ширина карандаша
  bs  =  Canvas->Brush->Style;
  pc  =  Canvas->Pen->Color;
  pw  =  Canvas->Pen->Width;
  Canvas->Brush->Style  =  bsClear;
  Canvas->Pen->Width  =  1;
  Canvas->Pen->Color =  clBlack;
  a = 0;          //метки ставим от 3х часов, против часовой стрелки
  h = 3;          //угол 0 градусов – это 3 часа
//циферблат
while  ( a < 360 )
{
  x = x0 +  R * cos(a * TORAD);
  y = y0 -  R * sin(a * TORAD);
  Form1->Canvas->MoveTo(x,y);
 if  ( (a % 30) == 0 )
 {
        Canvas->Ellipse(x-2,y-2,x+3,y+3);
//цифры по большому радиусу
        x = x0 +  (R+15) * cos(a * TORAD);
        y = y0 -  (R+15) * sin(a * TORAD);
        Canvas->TextOut(x-5,y-7,IntToStr(h));
        h--;
   if  ( h == 0 ) h = 12;
 }
 else
        Canvas->Ellipse (x-1, y-1, x+1, y+1) ;
  a = a + 6;                            // 1 минута – 6 градусов
}
//восстановить карандаш и кисть
  Canvas->Brush->Style  =  bs;
```

```
  Canvas->Pen->Width  =  pw;
  Canvas->Pen->Color =  pc;
  DrawClock();
}
```

3. Сохранить, протестировать.

Контрольные вопросы:

3. Какие невизуальные компоненты Вы знаете?
4. Чем невизуальные компоненты отличаются от псевдовизуальных?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 8 Отладка и тестирование в C++ Builder

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки отладки и тестирования приложений в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. На форму поместить только один компонент - Timer.

2.Ниже проведен листинг программы: сгенерировать соответствующие события и записать код:

```
//глобальные переменные
int x, y;                //положение мячика
int dx, dy;              //приращение координат
int r;                   //радиус мячика
TColor cBall;  //цвет мячика
TColor cBack;            //цвет поля
int wp, hp;              //размер поля (формы)
//это переменные для управления движением ракетки
int rd;                  //0 – не движется, 1 – движется влево, 2 - вправо
int rx1, rx2;            //координаты X концов ракетки
int ry;                  //координата Y концов ракетки
int rdx;          //шаг перемещения ракетки
//конструктор формы
```

```
__fastcall TForm1::TForm1(TComponent* Owner)          : TForm(Owner)
{
r = 5;                    //радиус мячика
x = r;y= 50;              //начальное положение мячика
dx = 1;
dy = 1;
cBall = (TColor)RGB(217, 217, 25);          //цвет мячика
cBack = (TColor)RGB(33, 94, 33);            //цвет поля
Form1->Color = cBack;
wp = Form1->ClientWidth;
hp = Form1->ClientHeight;
//управление ракеткой
rd = 0;                                     //ракетка на месте
rx1 = 100;
rx2=125;
ry = Form1->ClientHeight - 20;       //расстояние до нижней границы
                                            //формы 20 пикселей
rdx = 2;                                    //шаг движения ракетки
//настройки таймера
Timer1->Interval = 10;
Timer1->Enabled = true;
}
//событие OnPaint
void __fastcall TForm1::FormPaint(TObject *Sender)
{
//нарисовать ракетку
  Form1->Canvas->Pen->Color = clRed;
Form1->Canvas->Rectangle(rx1, ry, rx2, ry+1) ;
}
//сигнал от таймера
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
//стереть изображение мяча
  Form1->Canvas->Pen->Color = cBack;
Form1->Canvas->Ellipse(x,y,x+r,y+r);
//вычислить новое положение мяча
if  (dx > 0 )
{
//мяч движется вправо
if  ( x+ dx+ r > wp )  dx = -dx;
}
Else
//мяч движется влево
if  ( x + dx - r < 0 ) dx = -dx;
if  ( dy > 0 )
{
//мяч движется вниз
if ((x>= rx1) && (x <= rx2 -r) && (y== ry - r - 1) )
//мячик попал в ракетку
dy = - dy;
else
if  ( y + dy + r > Form1->ClientHeight ) dy = -dy;
}
else
```

```
{
//мяч движется вверх
if  ((x >= rx1) && (x <= rx2) && (y >= ry - r) && (y <ry + r))
{
//мячик отскочил от нижней стенки и попал в ракетку снизу
//чтобы не было дырок в ракетке, перерисуем ее
Form1->Canvas->Pen->Color = clRed;
Form1->Canvas->Rectangle(rx1, ry, rx2, ry+1);
}
if ( y+ dy - r < 0 ) dy = -dy;
}
x += dx;
y += dy;
//нарисовать мяч в новой точке
Form1->Canvas->Pen->Color = cBall;
Form1->Canvas->Ellipse(x,y,x+r,y+r);
//ракетка
  if  ( rd != 0 )
{
//игрок нажал и удерживает одну из клавиш управления
if  ( rd == 1 )
{
//вправо
if  (rx2 < wp )
{
//стереть часть слева
Form1->Canvas->Pen->Color  =  cBack;
Form1->Canvas->Rectangle(rx1, ry, rx1 + rdx,ry + 1);
//дорисовать часть справа
Form1->Canvas->Pen->Color  = clRed;
Form1->Canvas->Rectangle  (rx2, ry, rx2 + rdx,ry + 1) ;
rx1 += rdx;
rx2 += rdx;
}
}
else
//влево
if  (rx1 > 1 )
{
//стереть часть справа
Form1->Canvas->Pen->Color  = cBack;
Form1->Canvas->Rectangle  (rx2, ry, rx2 - rdx,ry+1);
//дорисовать слева
Form1->Canvas->Pen->Color  = clRed;
Form1->Canvas->Rectangle(rx1 - rdx, ry, rx1 +rdx,  ry+1);
rx1-=rdx;
rx2-=rdx;}}}
//нажата клавиша событие OnKeyDown
void __fastcall TForm1::FormKeyDown(TObject *Sender, WORD &Key, TShiftState
Shift)
{
  if  (rd != 0 )
||gjkmpjdfntkm elth;bdftn rkfdbie? Hfrtnrf ldb;tncz
return;
```

```
switch ( Key )
{
case VK_LEFT :
rd = 2; break;
case VK_RIGHT :
rd = 1; break;
}
}
//клавиша отпущена событие OnKeyUp
void __fastcall TForm1::FormKeyUp(TObject *Sender, WORD &Key, TShiftState
Shift)
{
  rd = 0;
}
```

3. Сохранить, отладить и протестировать.

Контрольные вопросы:

1. Что такое точки останова?
2. Что такое трассировка?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 9 Применение полос прокрутки

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки применения полос прокрутки в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

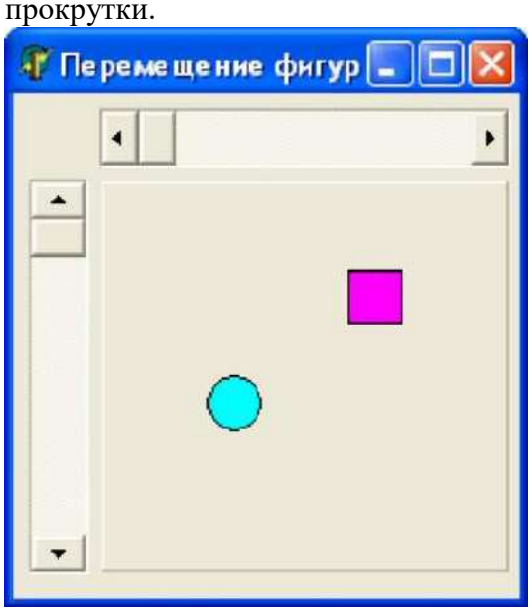
Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Создать программу, выполняющую следующие действия.

— После запуска программы в окне изображается две полосы прокрутки. Вертикальная полоса будет управлять движением по вертикали, горизонтальная - по горизонтали.

— Наводя указатель мыши на одну из двух фигур, можно выбирать, какая из этих фигур связана с полосами прокрутки.



— Требуются дополнительные объекты, с помощью которых ограничивается область движения фигур в окне.

— Если полоса прокрутки активная, то она должна реагировать на клавиши ВВЕХ, ВНИЗ, ВЛЕВО, ВПРАВО, PAGE UP, PAGE DOWN.

— Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

1. Открыть новый проект.
2. Разместить на форме экземпляры компонентов: панель Panel, полоса прокрутки ScrollBar, фигура Shape.
3. Ввести дополнительную переменную логического типа num. Если она принимает значение True (Да), то текущей считается первая фигура. Значению False (Нет) соответствует вторая фигура. Эта переменная должна быть доступна во всех процедурах.
4. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Forml	Properties	Caption	Установка имени формы "Перемещение фигур"
Panel (Вкладка Standard)	Properties	Height	Присвоить значение 161
		Width	Присвоить значение 161
		Caption	Оставить значение свойства пустым
ScrollBarl (Вкладка Standard)	Properties	Min	Присвоить значение 5
		Max	Присвоить значение 145
		Position	Присвоить значение 76
		SmallChange	Присвоить значение 2
		LargeChange (Большое изменение)	Присвоить значение 20

	Events	OnChange	If(num) Shape1->Left = ScrollBar1->Position; else Shape2->Left = ScrollBar1->Position;
ScrollBar2 (Вкладка Standard)	Properties	Kind	Выбрать значение sbVertical. Горизонтальная полоса прокрутки станет вертикальной.
		Min	Присвоить значение 5
		Max	Присвоить значение 145
		Position	Присвоить значение 76
		SmallChange (Малое изменение)	Присвоить значение 2
		LargeChange	Присвоить значение 20
	Events	OnChange	if (num) Shape1->Top = ScrollBar2->Position; else Shape2->Top = ScrollBar2->Position;
Shapel (Вкладка Additional)	Properties	Height	Присвоить значение 11
		Width	Присвоить значение 11
		Left	Присвоить значение 76
		Top	Присвоить значение 76
		Shape (Форма)	Выбрать значение stCircle (Круг)
		Brush (Кисть)	Выбрать для подсвойства Color (Цвет кисти) значение clAqua (голубой цвет)
	Events	OnMouseMove (При движении мыши)	Shape1->Brush->Color = clAgua; Shape1->Brush->Color =clFuchsia; Num = True; ScrollBar1->Position=Shape1->Left; ScrollBar2->Position=Shape1->Top;
Shape2 (Вкладка Additional)	Properties Events	Height	Присвоить значение 11
		Width	Присвоить значение 11
		Left	Присвоить значение 76
		Top	Присвоить значение 76
		Shape	Выбрать значение stSquare (Квадрат)
		Brush OnMouseMove	Выбрать для подсвойства Color (Цвет кисти) значение clFuchsia (фиолетовый цвет) Аналогично Shape2

5. Сохраните проект, запустите и протестируйте его.

Листинг события OnScrollBar1Change

```
if( num) Shape1->Left = ScrollBar1->Position;
else Shape2->Left = ScrollBarl1->Position;
```

Листинг события OnScrollBar2Change

```
if( num) Shape1->Top = ScrollBar2->Position;
else Shape2->Top = ScrollBar2->Position;
```

Листинг события OnShape1MouseMove

```
Shape1->Brush->Color = clAgua;
Shape1->Brush->Color = clFuchsia;
Num = True;
ScrollBar1->Position= Shape1->Left;
ScrollBar2->Position= Shape1->Top;
```

Листинг события OnShape2MouseMove

```
Shape2->Brush->Color = clFuchsia;
Shape2->Brush->Color = clAgua;
Num = False;
ScrollBar1->Position= Shape2->Left;
ScrollBar2->Position = Shape2->Top;
```

Листинг события OnFormCreate

```
num = True;
```

Контрольные вопросы:

1. Для чего нужна полоса прокрутки?
2. Какой еще компонент C++ Builder позволяет организовать полосы прокрутки?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 10 Использование компонентов ввода/вывода информации

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования компонентов ввода/вывода в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Пользователь должен ввести данные о себе и своих домашних животных, а затем нажать кнопку «Обработать». После нажатия на кнопку результаты опроса выводятся в расположенное справа текстовое поле.

1. Открыть новый проект.
2. Разместить на форме экземпляры компонентов как показано на рисунке 1.

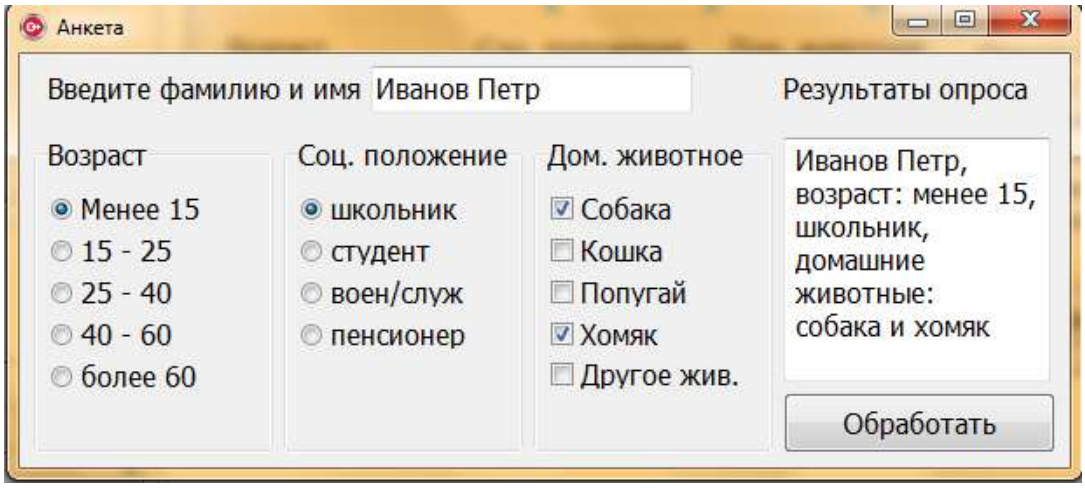


Рисунок 1 – Образец формы

3. Результат показать преподавателю.

Контрольные вопросы:

1. Какие компоненты ввода и вывода информации использованы в программе?
2. В чем отличие TRadioButton (зависимые переключатели) от TCheckBox (независимые переключатели)?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 11 Интерфейс и компоненты внешнего оформления

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

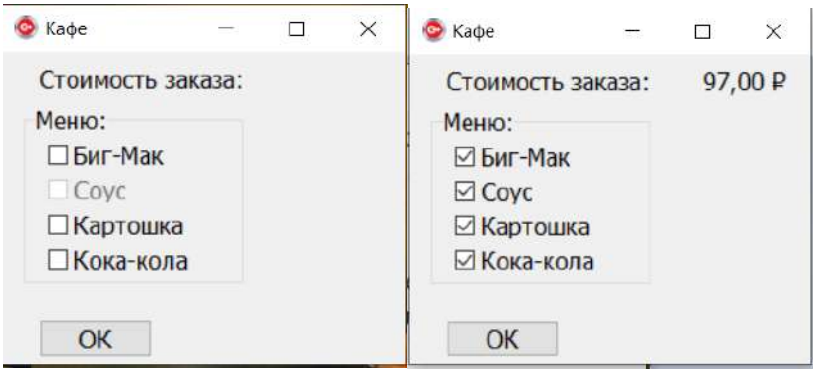
Цель: Получить навыки проектирования интерфейса в C++ Builder.

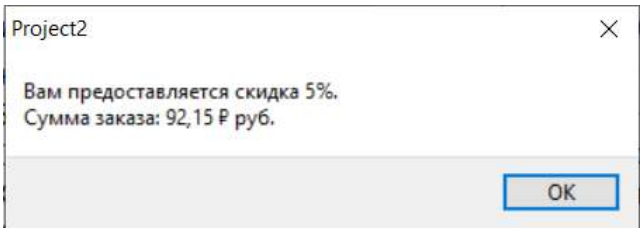
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Разработать приложение «Кафе» с интерфейсом (один соус выбрать нельзя):





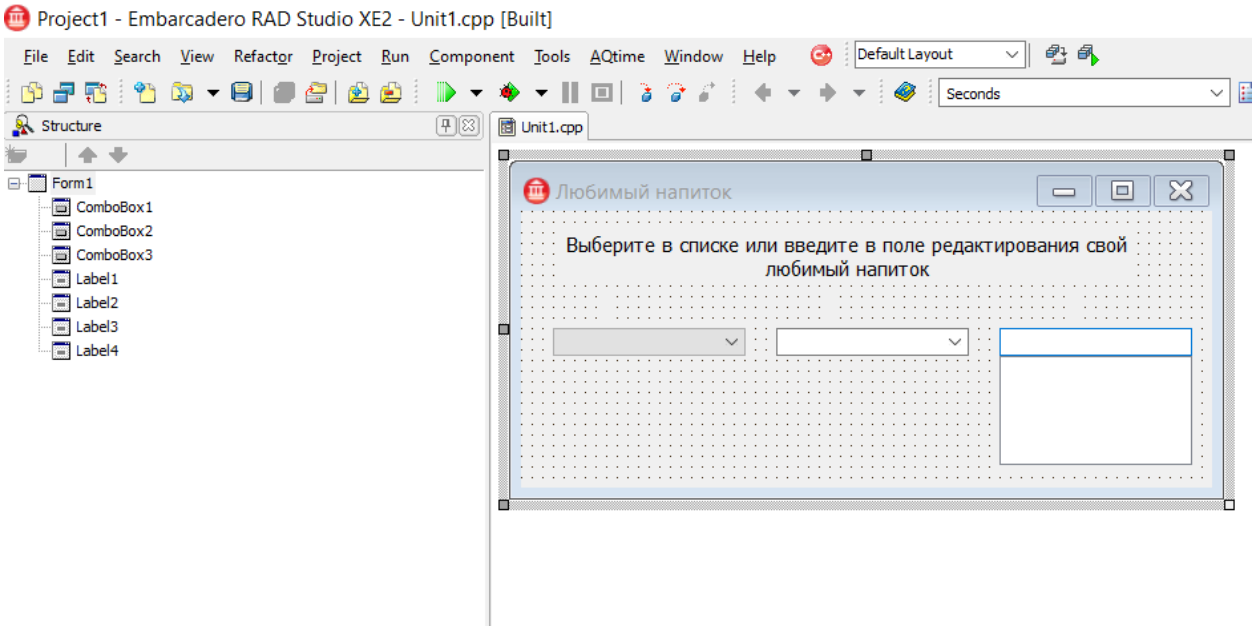
Листинг программы:

```
float summ;                    //глобальная переменная
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner): TForm(Owner)
{
 CheckBox2->Enabled=false;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::CheckBox1Click(TObject *Sender)
{
  if (CheckBox1->Checked)
  {
          summ+=54;
          CheckBox2->Enabled=true;
  }
  else
  {
           summ-=54;
           if(CheckBox2->Checked)
                   CheckBox2->Checked=false;
            CheckBox2->Enabled=false;
  }
  Label1->Caption=FloatToStrF(summ, ffCurrency, 6,2);
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm2::CheckBox2Click(TObject *Sender)
{
 if (CheckBox2->Checked)
        summ+=10.5;
 else
        summ-=10.5;
        Label1->Caption=FloatToStrF(summ,ffCurrency,6,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::CheckBox3Click(TObject *Sender)
{
  if(CheckBox3->Checked)
        summ+=18.5;
  else
        summ-=18.5;

        Label1->Caption=FloatToStrF(summ, ffCurrency,6,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::CheckBox4Click(TObject *Sender)
{
  if (CheckBox4->Checked)
        summ+=14;
  else
        summ-=14;
        Label1->Caption=FloatToStrF(summ,ffCurrency,6,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
  if                    ((CheckBox1->Checked)&&(CheckBox2->Checked)&&(CheckBox3-
>Checked)&&(CheckBox4->Checked))
   {
    summ=summ*0.95;
    ShowMessage("Вам предоставляется скидка 5%.\n"
    "Сумма заказа: " + FloatToStrF(summ, ffCurrency,6,2)+ " руб.");
    }
    else
        if((CheckBox1->Checked)||(CheckBox3->Checked)||(CheckBox4->Checked))
         ShowMessage("Сумма заказа: " + FloatToStrF(summ,ffGeneral,6,2)+ " руб.");
    else
    ShowMessage("Вы ничего не заказали");
}
```

2. Разработать приложение «Любимый напиток» с интерфейсом:



Листинг программы;

```
__fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner)
{
 ComboBox1->Sorted  =  true;
 ComboBox1->Items->Add("Кока-Кола");
 ComboBox1->Items->Add("Меринда")  ;
 ComboBox1->Items->Add("Пепси-Кола");
 ComboBox1->Items->Add("Спрайт");
 ComboBox1->Items->Add("Фанта");

 ComboBox2->Sorted  = true;
ComboBox3->Items->Add("Чай")  ;
ComboBox3->Items->Add("Чай с лимоном");
ComboBox3->Items->Add("Кофе черный") ;
ComboBox3->Items->Add("кофе со сливками");
ComboBox3->Items->Add("Какао");
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
{
  Label1->Caption = ComboBox1->Text;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox2Click(TObject *Sender)
{
   Label2->Caption = ComboBox2->Items->Strings[ComboBox2->ItemIndex];
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox3Click(TObject *Sender)
{
        Label3->Caption =ComboBox3->Items->Strings[ComboBox3->ItemIndex];
}
//---------------------------------------------------------------------------
void __fastcall TForm1::ComboBox2KeyPress(TObject *Sender, System::WideChar
&Key)

{
```

```
            if  (Key == VK_RETURN)
            {
                    int n = ComboBox2->Items->Add(ComboBox2->Text);
                     ComboBox2->ItemIndex = n;
                       Label2->Caption = ComboBox2->Items->Strings[n];
            }
        }
        //---------------------------------------------------------------------------
        void __fastcall TForm1::ComboBox3KeyPress(TObject *Sender, System::WideChar
&Key)

        {
            AnsiString st;
            if  (Key == VK_RETURN)
            {
             st = ComboBox3->Text.Trim();
             if  ( ComboBox3->Items->IndexOf(st) == -1 );
             {
          int n = ComboBox3->Items->Add(st);
                    ComboBox3->ItemIndex = n;
                     Label3->Caption = ComboBox3->Items->Strings[n];
             }
            }
        }
        //---------------------------------------------------------------------------
        void __fastcall TForm1::ComboBox1KeyPress(TObject *Sender, System::WideChar
&Key)

        {
         if(Key == VK_RETURN);
          ComboBox2->SetFocus();
        }
```

Контрольные вопросы::

1. Для чего используется свойство формы AutoSize?
2. Как запретить изменять размеры окна приложения?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 12 Работа с меню

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования компонентов-менюа в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Создать программу, выполняющую следующие действия:

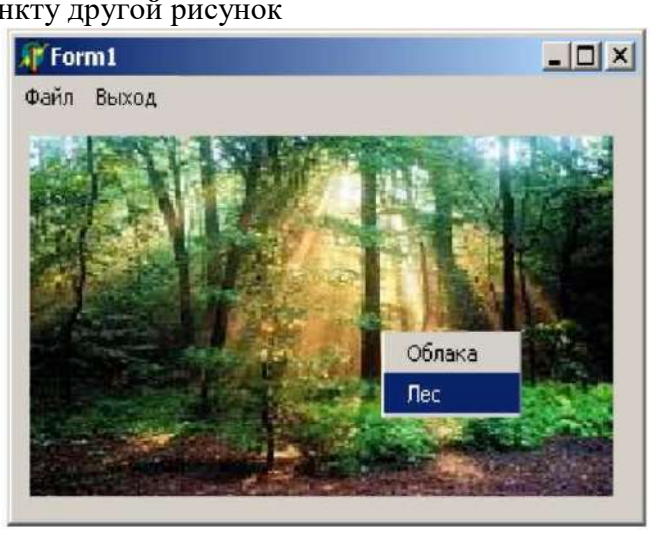
После запуска программы в окне изображается строка меню (Файл, Выход).

При выборе пункта меню Файл появляются пункты меню (Рисунки, Выход).

При выборе пункта меню Рисунки появляется вложенное меню, состоящее из двух пунктов (Облака, Лес).

По щелчку правой кнопки мыши появляется контекстное меню.

Выбрать по пункту другой рисунок



Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

Если выбрать любой из пунктов Выход, работа программы завершается.

Описание плана разработки программы

1. Открыть новый проект.

2. Разместить на форме экземпляры компонентов: панель Panel, рисунок Image, диалоговое окно OpenDialog.

3. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Установка имени формы "Мое меню"

	Events	OnMouseDown	TPoint p;
			p->X =X;
			p->Y =Y;
			p = ClientToScreen (p);
			PopupMenu1->Popup (p->X, p->Y);

Запустить редактор меню (дважды щелкнуть на значке меню на форме)			
Form1->MainMenul	Properties (в окне Object Inspector не выбран никакой объект)	Caption	Ввести текст пункта меню -Файл, и нажать Enter. Система присвоит ему имя N1
Между существующими и будущими пунктами меню можно переключаться с помощью щелчка мыши или курсорных клавиш.			
Form1->MainMenul	Properties	Caption	Ввести текст пункта меню -Выход, и нажать Enter. Система присвоит ему имя N2.
	Events (щелкнуть на пункте Выход в строке меню)	N2Click	Close;
Щелкните на пункте Файл. Редактор меню создал еще одну заготовку под этим пунктом. Это заготовка для меню, которое откроется при выборе пункта Файл в работающей программе. Используя заготовки, создайте в этом меню два пункта: Рисунки (система присвоит ему имя N3) и Выход (N4). Выберите в редакторе меню пункт Рисунки и нажмите комбинацию клавиш Ctrl + Вправо.			
N4: TMenuItem	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N2Click
Form1->MainMenul	Properties	Caption	Ввести текст пункта меню -Облака, и нажать Enter. Система присвоит ему имя N5.
N5: TMenuItem	Events (выбрать в строке меню на форме пункт Облака)	OnClick	Image1->Picture->LoadFromFile ("C:\Облака.jpg");
Form1->MainMenul	Properties	Caption	Ввести текст пункта меню - Лес, и нажать Enter. Система присвоит ему имя N6.
N6: TMenuItem	Events (выбрать в строке меню на форме пункт Лес)	OnClick	Image1->Picture->LoadFromFile ("C:\Лес.jpg");
Закройте окно редактора меню и убедитесь, что теперь строка меню появилась в основной форме программы.			

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
PopupMenu (Вкладка Standard)	Properties	Caption	Ввести текст пункта меню -Облака, и нажать Enter. Система присвоит ему имя N7.
		Caption	Ввести текст пункта меню - Лес, и нажать Enter. Система присвоит ему имя N8.
N7	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N5Click
N8	Events	OnClick	Выберем из раскрывающегося списка уже существующую процедуру-обработчик N6Click
Image (Вкладка Additional)	Properties	Stretch	Присвоить значение True

4. Сохраните проект, запустите и протестируйте его.

Контрольные вопросы:

1. Для чего используется компонент MainMenu?
2. Для чего используется компонент PopupMenu?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 13 Табулирование функций

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования компонента StringGrid в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

Функция 1/(5-3cos(x)).

Первый Edit – вводится минимальное значение интервала, второй Edit – вводится максимальное значение интервала, третий Edit – шаг табуляции.

StringGrid, на две строки x и y.

Нужно сделать табуляцию функции, значения которой будут отображаться в StringGrid.

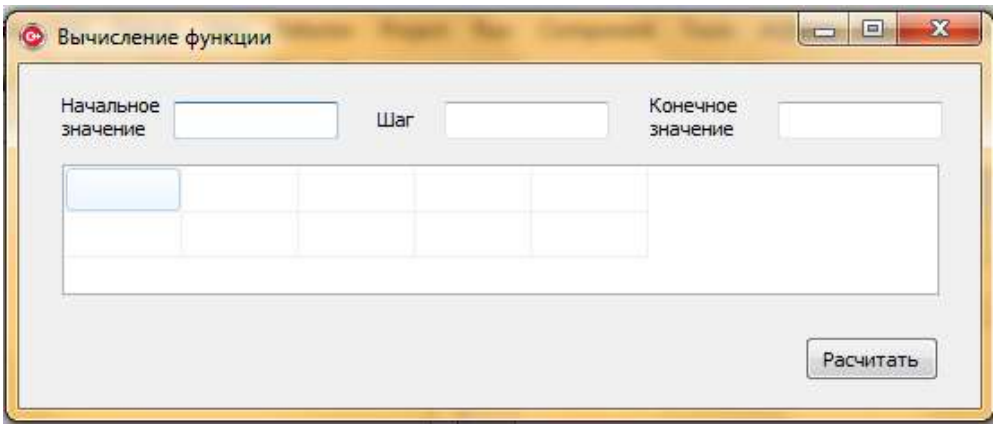


Рисунок 1 – Пример формы

Контрольные вопросы:

1. Как обратиться к ячейке StringGrid?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 14 Практика работы с визуальными компонентами

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки работы с визуальными компонентами в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Создать программу, выполняющие следующие действия:
   - После запуска программы в окне изображается три поля

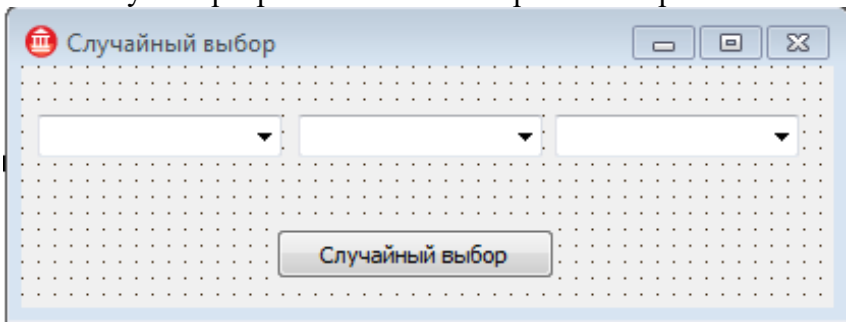


Рисунок 1 – Демонстрация запущенной программы

   - По щелчку мышью на кнопке «Случайный выбор» из трех слов составляется предложение случайным образом

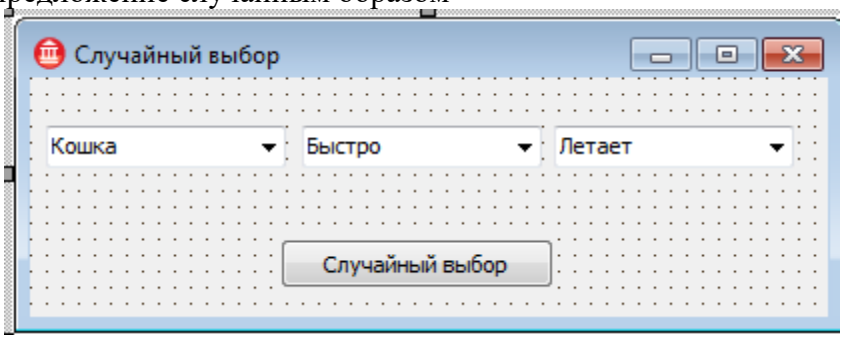


Рисунок 2 – Работа программы

   - Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка
2. Открыть новый проект
3. Разместить на форме экземпляры компонентов: поле со списком ComboBox, командная кнопка Button
4. Выполнить следующие действия:

Объект	Object Inspector	Имя	Действия
Form1	Properties	Caption	Установка имени формы «Сочинитель»
ComboBox1 ComboBox2 ComboBox3	Events	OnCreate	ComboBox1->ItemIndex=0; ComboBox2->ItemIndex=0; ComboBox3->ItemIndex=0;
	Properties	Style	Выберите значение cSDropDownList из раскрывающегося списка
		Items	Щелкните на кнопке построителя. Откроется окно String List Editor. Ввести пункты списка по одному в каждую строчку, завершая ввод нажатием клавиши Enter. После того как список готов, щелкните на кнопке OK.
Button1	Properties	Caption	Установка имени кнопки «Случайный выбор»

5. Сохраните проект и протестируйте его

Таблица для заполнения:

Кошка	Быстро	Плавает
Змея	Высоко	Бегает
Кузнечик	Медленно	Летает
Дельфин	Сильно	Ползает
Черепаха	Хорошо	Прыгает
Ласточка	Плохо	прячется

6. Создать программу, выполняющие следующие действия:

— После запуска программы в окне изображается два движка.

— Необходимо выбрать два числовых значения и найти их произведение.

— Если выбирается одно число, то находится его квадрат.

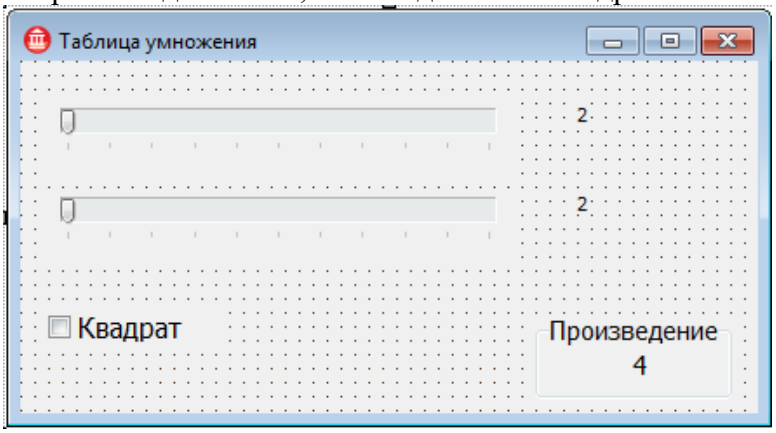


Рисунок 3 – Вид запущенной программы

— Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

7. Открыть новый проект
8. Разместить на форме экземпляры компонентов: командная кнопка Button, движок TrackBar, рамка GroupBox, надпись Label, флажок CheckBox.
9. Выполнить следующие действия:

Объект	Object Inspector	Имя	Действия
Form1	Properties	Caption	Установка имени формы «Таблица умножения»
TrackBar1	Properties	Min	Присвоить значение 2
		Max	Присвоить значение 99
		Position (Положение)	Присвоить значение 2
		LineSize (малое изменение)	Присвоить значение 1
		PageSize (Постраничное изменение)	Присвоить значение 7
		Frequency (частота засечек)	Присвоить значение 7
	Events	OnChange	Label1->Caption=IntToStr(TrackBar1->Position); Label3->Caption=IntToStr(TrackBar1->Position*TrackBar2->Position); If (CheckBox1->Checked) { TrackBar2->Position=TrackBar1->Position;}
TrackBar2	Properties	Min	Присвоить значение 2
		Max	Присвоить значение 99
		Position (Положение)	Присвоить значение 2
		LineSize (малое изменение)	Присвоить значение 1
		PageSize (Постраничное изменение)	Присвоить значение 7
		Frequency (частота засечек)	Присвоить значение 7
	Events	OnChange	Label2->Caption=IntToStr(TrackBar2->Position); Label3->Caption=IntToStr(TrackBar1->Position*TrackBar2->Position); If (CheckBox1->Checked) { TrackBar2->Position=TrackBar1->Position;}
GroupBox1	Properties	Caption	Ввести подпись «Произведение»
Label1	Properties	AutoSize	Установить значение False

		Caption	Присвоить значение 2
		Aligment	Установить значение taRightJustify
Label2	Properties	AutoSize	Установить значение False
		Caption	Присвоить значение 2
		Aligment	Установить значение taRightJustify
Label3	Properties	AutoSize	Установить значение False
		Caption	Присвоить значение 4
		Aligment	Установить значение taRightJustify
CheckBox1	Properties	Caption	Ввести подпись «Произведение»
		Aligment	Установить значение taLeftJustify
	Events	OnClick	Trackbar2->Position=TrackBar1->Position;

10. Сохраните проект и протестируйте его

Листинг подпрограммы событие OnChange для TrackBar1

```
Label1->Caption= IntToStr(TrackBar1->Position);
Label3->Caption= IntToStr(TrackBar1->Position * TrackBar2.Position);
if (CheckBox1->Checked ) TrackBar2.Position = TrackBar1.Position;
```

событие OnChange для TrackBar2

```
Label2.Caption = IntToStr(TrackBar2.Position);
 Label3.Caption= IntToStr(TrackBar1->Position * TrackBar2->Position);
 if (CheckBox1->Checked)  TrackBar2->Position = TrackBar1->Position;
```

событие OnClick для CheckBox1

```
TrackBar2->Position = TrackBar1->Position;
```

Задание для самостоятельного выполнения

1. Изменить программу так, чтобы находить произведения не только двузначных, но и трехзначных чисел от 2 до 199.

2. Изменить программу так, чтобы находить сумму двух чисел.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 15 Практика работы с невизуальными компонентами

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки работы с невизуальными компонентами в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий

1. Создать программу, выполняющие следующие действия:

— После запуска программы в окне изображается рисунок

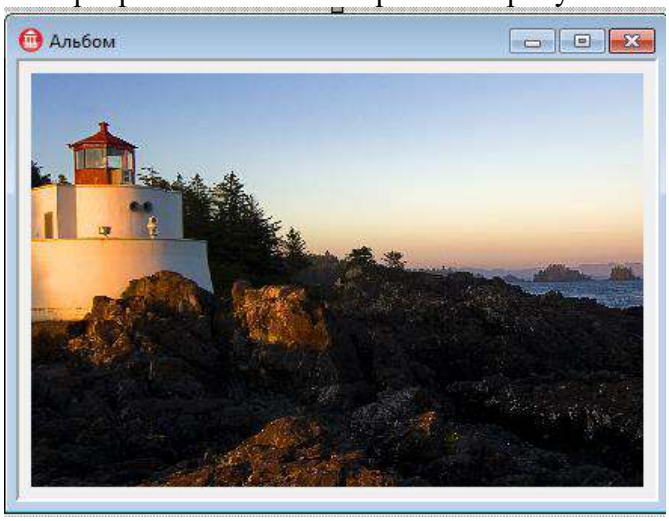


Рисунок 1 – Изображение запущенной программы

— По щелчку мыши на рисунке появляется диалоговое окно;

— Выбрать в диалоговом окне любой другой рисунок;

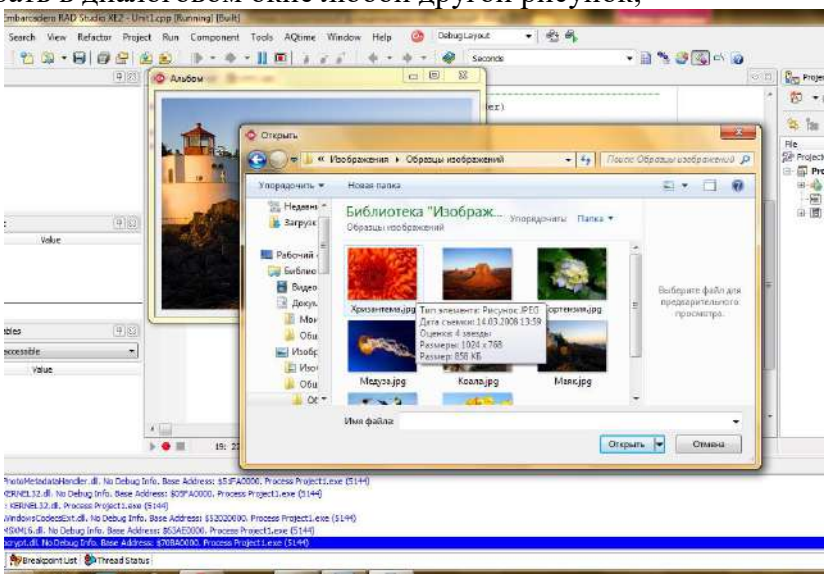


Рисунок 2 – Диалоговое окно

— Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

2. Открыть новый проект
3. Разместить на форме экземпляры компонентов: панель Panel, рисунок Image, диалоговое окно OpenDialog;
4. Выполнить следующие действия:

Объект	Object Inspector	Имя	Действия
Form1	Properties	Caption	Установка имени формы «Альбом»
Panel1	Properties	Caption	Очистить значение свойства
		BevelOuter	Выбрать значение bvLowered
		BevelInner	Выбрать значение bvNone
		BevelWidth	Присвоить значение 2
		Width	Присвоить значение 241
		Height	Присвоить значение 185
Image1	Properties	Left	Присвоить значение 2
		Top	Присвоить значение 2
		Width	Присвоить значение 237
		Height	Присвоить значение 181
		Stretch	Включить свойство True
		Picture	С помощью кнопки-построителя открыть диалоговое окно Picture Editor (Редактор изображений). Щелкнуть на кнопке Load – откроется диалоговое окно Load Picture. Открыть папку C:\Windows и выбрать файл, щелкнуть по кнопке Открыть. Вернуться в окно Редактора изображений, щелкнуть на кнопке ОК.

5. Сохраните проект и протестируйте его

Контрольные вопросы:

1. Какой метод вызывает диалог?
2. Имеет ли значение расположение невизуального компонента на форме во время проектирования?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задач на компьютере;
- работа выполнена полностью и получен верный ответ или иное требуемое представление результата работы;

оценка «4» ставится, если:

- работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;
- правильно выполнена большая часть работы (свыше 85 %), допущено не более трех ошибок;
- работа выполнена полностью, но использованы наименее оптимальные подходы к решению поставленной задачи.

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 16 Программирование событий KeyPress, KeyDown, KeyUp

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки создания обработчиков событий, связанных с клавиатурой. Получить навыки использования объектов типа TImage, TShape и TTimer в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий:

В верхней части окна приложения непрерывно движется объект «Противник». В нижней части окна – объект «Охотник», который может перемещаться влево и вправо, а также стрелять в «Противника» вверх пулей. При попадании пули в «Противника» объект «Охотник» увеличивает свои баллы. Цель может быть опасна, т.к. сбрасывает объект «Бомбочки», которые при попадании в охотника уменьшают его баллы.

В начале игры у «Охотника» 10 баллов. Игра заканчивается победой, если «Охотник» удваивает баллы, и поражением, если баллы равны нулю.

1. Создать и сохранить новый проект.
2. Разместить на форме два компонента типа **TImage** (страница Additional), задайте для них имена, например: **Men** (охотник), **Samolet** (цель), и измените свойство **Picture** для каждого (выбрать графический файл в сетевой папке Барилова/МДК01.02/28).
3. Разместить компонент типа **Shape,** задать имя **Pula** (пуля), задать форму круга.
4. Установить два таймера (страница System). **Timer1** отвечает за непрерывное движение цели. **Timer2** за движение объекта пули. Рекомендуемый вид формы приведен на рисунке.

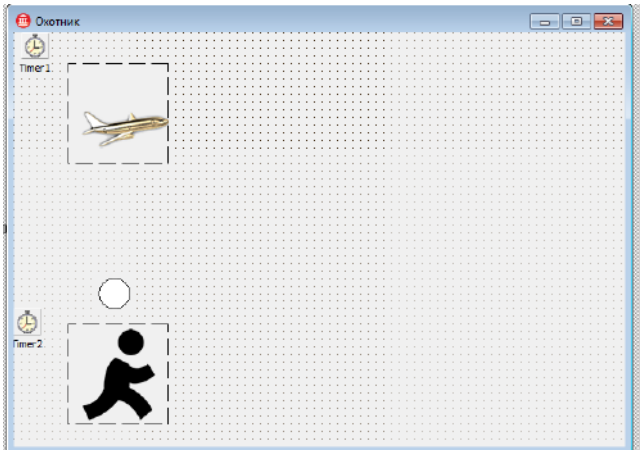


Рисунок 1 Рекомендуемый вид формы

5. Для объекта **Timer1** ввести следующий код:

```
Samolet->Left =Samolet->Left +1;
if (Samolet->Left >300)
    Samolet->Left =-30;
```

6. Проверить работу программы. Самостоятельно отрегулировать скорость движения объекта, изменяя значения свойства **Interval** для таймера.

7. Для формы выбрать событие **OnKeyDown** и введите код:

```
 if (Key==37)
    Men->Left =Men->Left-10;//нажата клавиша влево
 if (Key==39)
     Men->Left =Men->Left+10;//нажата клавиша вправо
 if (Key==32) //нажат пробел
 {
   Pula->Visible =true;
   Pula->Left=Men->Left ;
   Pula->Top=Men->Top  Pula->Height ;
   Timer2->Interval=10;
 }
```

8. Для объекта **Pula** изменить свойство **Visible** на **False**. Для объекта **Timer2** изменить свойства **Interval**=0.

9. Для объекта **Timer2** введите код:

```
Pula->Top=Pula->Top-5;
 if (Pula->Top<0)
 {
   Timer2->Interval=0;
   Pula->Visible=false;
 }
```

10. Проверить работу программы, отрегулировать скорость движения «пули».

11. Добавить метку (имя Result) для вывода количества баллов.

12. В обработчик для **Timer2** добавить код (выделен полужирным шрифтом) для проверки на попадание «пули» в «самолет»:

```
int x,y,h,w;// координаты и размеры пули
int x0,y0,h0,w0; // координаты и размеры самолета
 Pula->Top=Pula->Top-5;
 if (Pula->Top<0)
 {
   Timer2->Interval =0;
   Pula->Visible =false;
 };
 x=Pula->Left;
  y=Pula->Top;
 w=Pula->Width ;
 h=Pula->Height ;
 x0=Samolet->Left ;
 y0=Samolet->Top;
 w0=Samolet->Width ;
 h0=Samolet->Height ;
 if ((x+w>x0) && (x<x0+w0) && (y+h>y0) && (y<y0+h0))
    Result->Caption =IntToStr(StrToInt(Result->Caption)+1) ;
```

13. Проверить работы программы. **Определить, почему баллы начисляются неверно. Внести изменения в обработчик для того, чтобы за одно попадание добавлялся один балл.**

14. Добавить на форму еще один компонент типа **TShape** (имя **Bomba**, **Visible** =**False**, измените размер и цвет **Brush/Color**) и еще один таймер **Timer3** (Interval =10).

15. Для таймера ввести следующий код:

```
if (Bomba->Visible ==false)
 {
     Bomba->Visible =true;
    Bomba->Left =Samolet->Left+10 ;
    Bomba->Top=Samolet->Top+10;  }
```

```
else
{
    Bomba->Top=Bomba->Top+5;
    if (Bomba->Top>Men->Top+30)
        Bomba->Visible=false;
}
```

16. Проверить работу граммы. Самостоятельно внести изменения в обработчик для того, чтобы баллы уменьшались при попадании в «охотника».
17. Самостоятельно добавить проверку на окончание игры.
18. Проект сохранить. Результат показать преподавателю.

Контрольные вопросы

1. В каких случаях используется таймер? Что определяет свойство Interval?
2. Как остановить работу таймера?
3. Как определить код нажатой клавиши?
4. На какие события реагирует компонент Shape?

Задание для самостоятельного выполнения:

Модифицировать разработанное приложение: добавить еще один «летательный» объект, скорость и направление движения объектов должна быть различна. Программа должна добавлять баллы при попадании в любой из объектов. Оба объекта могут сбрасывать «бомбы».

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая контрольные вопросы и задание для самостоятельного выполнения;

оценка «4» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая контрольные вопросы ;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 17 Массивы

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить навыки использования массива объектов в приложении в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий:

Разработать приложение «Игра в пятнашки».

Алгоритм игры следующий: в прямоугольной коробке находится 15 фишек, на которых написаны числа от 1 до 15. Размер коробки - 4x4, таким образом, в коробке одна пустая ячейка. В начале игры фишки перемешаны. Задача игрока состоит в том, чтобы, не вынимая фишки из коробки, выстроить фишки в правильном порядке (по возрастанию).

1. Создать новый проект. Установить на форме командную кнопку, задать имя для нее **Pole.**

2. Объявить массив объектов и переменные:

```
TButton *game[15];
int d,i;
```

3. Для формы изменить свойство **AutoSize**=true, выбрать событие **OnCreate** и ввести код:

```
int row,col;
 d=50;// размер кнопки
for(i=1;i<15;i++)
 {
  game[i] = new TButton (Application)
  game[i]->Parent = Form1;
  game[i]->Width =d ;   game[i]->Height :=d ;
  game[i]->Font->Style=Pole->Font->Style ;
  game[i]->Font->Size =pole->Font->Size ;
  game[i]->Caption =intToStr(i);
  col=i%4;       //номер столбца для кнопки
  row=i/4;       //номер строки для кнопки
  game[i]->Top  =d*row;
  game[i]->Left =d*col;
  game[i]->OnClick=GameClick;
  }
     game[0]->Caption :=''; }
```

4. Объявить процедуру (добавить в описание класса формы, в раздел public)

```
void __fastcall TForm1::GameClick (TObject *Sender);
```

5. Ввести код для этой процедуры:

```
void __fastcall TForm1::GameClick (TObject *Sender)
{
  int x0,y0,x,y,index;
{
//определение номера нажатой кнопки
for (i=0;i<15;i++)
if (game[i]->Focused) index=i;
// координаты пустышки
x0=game[0]->Left;
y0=game[0]->Top;
//координаты нажатой кнопки
```

```
x=game[index]->Left;
y=game[index]->Top;
//если соседи по столбцу, то обмен
//подключить math.hpp
if ((x0==x) && (abs(y-y0)==d) )
{
game[0]->Left =x; game[0]->Top=y;
game[index]->Left =x0; game[index]->Top:=y0;
};
//если соседи по строке, то обмен
if ((y0==y) && (abs(x-x0)==d))
{
game[0]=>Left =x; game[0]->Top=y;
game[index]->Left =x0; game[index]->Top=y0;
};
};
```

6. Проверить работу программы.
7. Создать главное меню на форме из двух пунктов: **Перемешать** и **Выход**. Для пункта **Перемешать** ввести следующий код:

```
Randomize();
for (int i=1;i<=15;i++)
  {
   row=rand()%15+1;
   game[row]->SetFocus() ;
   GameClick(game[row]);
  };
```

Предложить способ, как можно определить что задача решена и вывести соответствующее сообщение.

Контрольные вопросы

1. Из каких объектов можно создать массив объектов?
2. Какова последовательность создания массива объектов?
3. Если в приложении используется массив из 10 объектов, каждый из которых должен реагировать на двойной щелчок мыши, сколько процедур необходимо при этом создать?
4. Могут ли объекты, входящие в массив, реагировать на несколько событий?

Задание для самостоятельного выполнения:

Доработайте созданное приложение «Пятнашки»:

- добавить пункты в главное меню, позволяющие:
  - выполнять настройку игрового поля (3x2, 4x4, 5x5);
  - менять уровень сложности игры (степень перемешивания)
- вести счет количества результативных ходов пользователя.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая контрольные вопросы и задание для самостоятельного выполнения;

оценка «4» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая контрольные вопросы;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 18 Программа для тестирования

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Закрепить навыки программирования в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий:

Текстовый файл содержит несколько вопросов и 4 варианта ответа, из которых только один ответ верный(помечен символом +). Файл имеет следующую структуру:

Тема теста (дисциплина)
/вопрос 1
- ответ 1
+ ответ 2
- ответ 3
- ответ 4
/вопрос 2
+ вопрос 1
- вопрос 2
- вопрос 3
- вопрос 4

Создать текстовый файл с тестом из 5 вопросов в соответствии с заданной структурой.

Разработать приложение, которое позволяет выбирать файл с тестовыми заданиями и проводить последовательное тестирование.

Предусмотреть вывод результат тестирования на экран

1. Загрузить C++ Builder
2. Установить на форму следующие объекты и изменить им свойства (см. таблицу), объявить используемые переменные как глобальные.

объект	свойство	значение
Label	Name	Label1
	WordWrap	True

RadioButton	Name	RadioButton1
Button	Name	Button1
	Caption	Далее
OpenDialog	Name	OpenFile

3. Для формы выбрать событие **OnCreate** и ввести код для создания массива радиокнопок и объявить необходимые переменные:

```
 for (int i=0; i<=3;i++)
 {
    otv[i] = TRadioButton->Create(RadioButton1) ;
    otv [i]->Parent = Form1;
    otv[i]->Left = RadioButton1->Left;
    otv[i]->Top= RadioButton1->Top+i*40;
    otv [i]->Visible =true;
    otv [i]->Width = RadioButton1->Width ;
    otv [i]->Height = RadioButton1->Height ;
    otv [i]->Color = RadioButton1->Color ;
    otv[i]->Checked =false;
    otv[i]->Font =  RadioButton1->Font;
  };
   RadioButton1->Visible =false;
```

4. Добавить на форму меню из двух пунктов **Файл / Открыть** и **Выход**

5. Для пункта **Открыть** ввести код:

```
if (OpenFile->Execute)
  {
  assignfile(f1,OpenFile->FileName);
  reset(f1);
  readln(f1,s);
  Form1->Caption =s;  //тема теста
  verno=0;//кол-во верных ответов
  n=0;  //номер текущего вопроса
  Button1->Enabled =true;
  Label1->Caption ="";
  for (int i=0; i<=3;i++)
  {
  otv[i]->Caption ="";
  otv[i]->Checked =false;
  };
 };
```

6. Для кнопки **Далее** ввести код:

```
if (n>0) {проверим ответ на предыдущий вопрос}
 for (int i=0; i<=3;i++)
   if ((otv[i]->Checked) && (otv[i]->Tag==1))  verno=verno+1;
if (Eof(f1)) //если тест закончен
{
 ShowMessage("кол-во верных ответов="+IntToStr(verno));
 CloseFile(f1);
 Button1->Enabled =false;
 exit;
};
n=n+1;
//÷читаем следующий вопрос
   readln(f1,s);
   delete(s,1,1);
```

```
   Label1->Caption =s;
for (int i=0; i<=3;i++) //варианты ответов
    {
    readln(f1,s);
    otv[i]->Checked =false;
    otv[i]->Tag=0;
    if (s[1]=="+") otv[i]->Tag=1;//запомним верный ответ
    delete(s,1,1);
    otv[i]->Caption =s;
   };;
```

7. Проверить работу приложения.

Задание для самостоятельного выполнения:

Модернизировать приложение: вывод каждого вопроса должен сопровождаться отображением картинки, имя графического файла для каждого вопроса записать в строке, следующей за вопросом.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая задание для самостоятельного выполнения;

оценка «4» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 19 Программа «Просмотр иллюстраций»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: создать программу, которая позволяет просматривать иллюстрации, расположенные в определённой директории в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Прядок выполнения заданий:

Программа Просмотр иллюстраций (рисунок 1) демонстрирует использование компонентов ListBox и OpenDiaiog. Выбор каталога (папки) выполняется в стандартном окне Открыть файл, которое становится доступным в результате щелчка по кнопке Выбор. Отображение диалога осуществляет компонент OpenDiaiog.

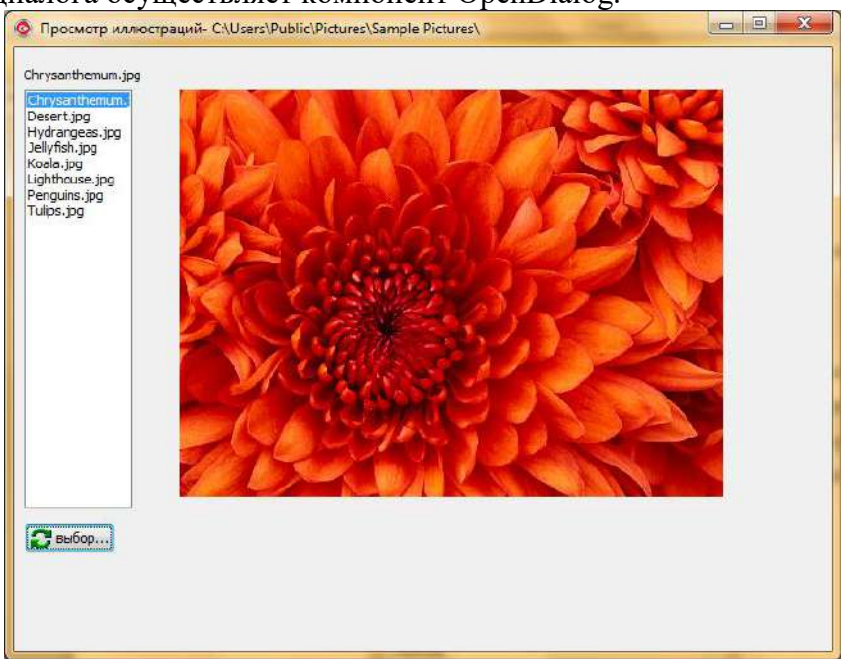


Рисунок 1 – Выбор фотографии выполняется в списке компонента ListBox

Листинг программы:

```
#include <jpeg.hpp> // обеспечивает работу // с иллюстрациями в формате JPEG
AnsiString aPath; // каталог, в котором находится // иллюстрация
TSearchRec aSearchRec; // рез-т поиска файла
// конструктор
fastcall TForml::TForml(TComponent* Owner):TForm(Owner)
Imagel->Proportional = true;
if (FindFirst(aPath+ "*.jpg"/ faAnyFile, aSearchRec) == 0)
ListBoxl->Items->Add(aSearchRec.Name);
while ( FindNext(aSearchRec) == 0 )// найти след. илл.
{
ListBoxl->Items->Add(aSearchRec.Name);
}
// отобразить первую иллюстрацию
ListBoxl->ItemIndex = 0;
Labell->Caption = ListBoxl->Items->Strings[0];
Imagel->Picture->LoadFromFile(aPath +
ListBoxl->Items->Strings[0]);
// щелчок в строке компонента ListBox
void fastcall TForml::ListBoxlClick(TObject *Sender)
{
int n = ListBoxl->ItemIndex; // номер выбранного эл-та списка
Labell->Caption = ListBoxl->Items->Strings[n];
Imagel->Picture->LoadFrornFile(aPath + ListBoxl->
Items->Strings[n]);
// щелчок на кнопке Выбор
void fastcall TForml::BitBtnlClick(TObject *Sender)
{
if ( OpenDialogl->Execute() )
{
```

```
// пользователь выбрал файл
ListBoxl->Clear(); // очистить список
aPath = ExtractFilePath(OpenDialogl->FileName);
Forml->Caption = "Просмотр иллюстраций - " + aPath;
if ( FindFirst(aPath+ "*.jpg", faAnyFile, aSearchRec) == 0)
{
ListBoxl->Items->Add(aSearchRec.Name);
while (FindNext(aSearchRec) == 0)//найти след.илл.
{
ListBoxl->Items->Add(aSearchRec.Name);
}
// определим позицию выбранного пользователем файла в списке ListBox и отобразим его
int n = ListBoxl->Items-> IndexOf(ExtractFileName(OpenDialogl->FileName));
ListBoxl->ItemIndex = n;
Labell->Caption = ListBoxl->Items->Strings[n];
Imagel->Picture->LoadFromFile(aPath + ListBoxl->Items->Strings[n]);
                }
        }
}
```

Контрольные вопросы:

1. Какой компонент отвечает за отображение графики?
2. Какую библиотеку надо подключить для корректной работы с файлами .jpeg?
3. Основной метод диалогов?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, включая контрольные вопросы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 20 Строковый калькулятор

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

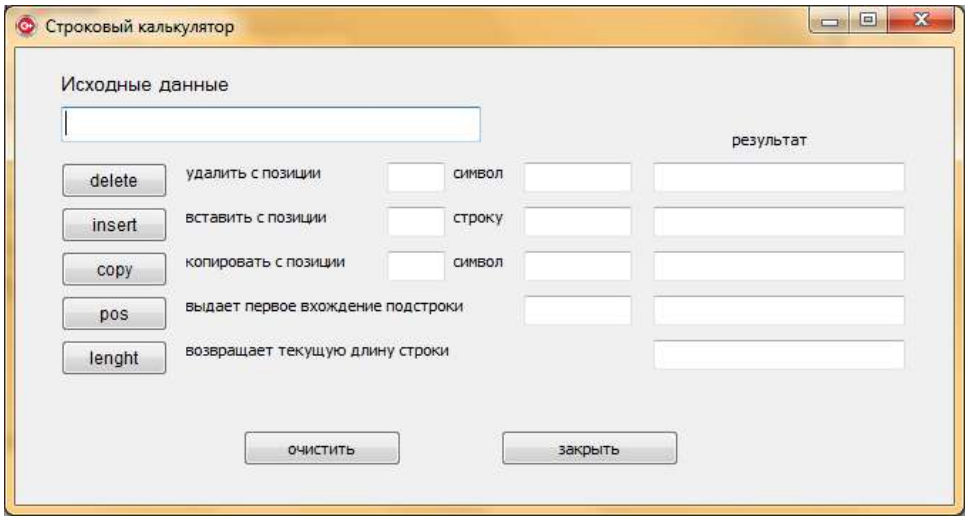
Цель: создать программу, выполняющую действия строкового калькуляторав C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Разработать строковый калькулятор, имеющий вид:



Задания для самостоятельной работы:

1. Модифицировать приложение: добавить функции преобразования в верхний и нижний регистры.
2. Отладить и протестировать программу «Строковый калькулятор»

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 21 Элементы пользовательского интерфейса

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки разработки пользовательского интерфейса в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Разработать приложение, реализующее основные функции текстового редактора:

- форматирование шрифта для выделенного контекста;
- копирование и перемещение выделенного контекста;
- выравнивание абзацев;
- поиск и замена в тексте;
- открытие и сохранение текстового файла;

Все функции приложения должны быть доступны через главное меню и панель инструментов.

Порядок выполнения задания:

Создать новый проект и сохранить его под именем MainEditor.

1. Создать на форме меню:

- установить на форме компонент MainMenu (Standard);
- присвоить ему имя mmMain. Каждому пункту меню задать информативное имя:

Файл (miFile) Новый(miNewFile) Открыть(miOpenFile) Сохранить(miSaveFile)	Правка(miEdit) Копировать (miCopy) Вырезать(miCut) Вставить(miPaste) Найти(miFind) Заменить(miReplace)
Формат (miFormat) Шрифт (miFont) Абзац (miPar)	Вставка (miIns) Дата (miData) Время (miTime)

- Выход – (miExit)

1. Создать панель инструментов:

1.1 расположить на форме компонент ToolBar (Win32), присвоить имя tlbMain, для свойства EdgeBorders добавить значение ebBotton True;

1.2 щелкнуть на созданной панели правой кнопкой мыши и выполнить NewButton, установить кнопке следующие свойства:

- Hint создать файл
- MenuItem miNewFile
- Name btnNewFile
- ShowHint True

1.3 аналогичным образом создать еще кнопки и изменить им свойства: Открыть, Сохранить, Копировать, Вырезать, Вставить, Найти, Заменить;

1.4 для добавления изображений командам и кнопкам добавить на форму компонент ImageList (Win32);

1.5 дважды щелкнуть по компоненту, появится редактор, щелкните по кнопке Add и указать размещение требуемых графических файлов Program Files\Common Files\Borland Shared\Images\Buttons, добавить соответствующий файл, в момент добавления на экране появится запрос о разбиении одной пиктограммы на две, ответьте утвердительно.

- . Связать набор пиктограмм с панелью инструментов и основным меню: для компонентов mmMain и tlbMain для свойства Images из списка выбрать ImageList1.
- Изменить значения свойства ImageIndex для соответствующих пунктов меню и кнопок панели инструментов.
- Добавить на панель инструментов компонент ColorBox (Additional) для выбора цвета шрифта, изменить значения свойств:
  - Name cobFontColor, Selected clBlack,
  - Style cbExtendedColors False
  - cbSystemColors False

Добавить на панель инструментов компонент SpinEditor (Samples) для ввода размеров шрифта, изменить значения свойств:

- Name - sdFontSize, MaxValue 70, MinValue 8)

1.6 Добавить на панель инструментов три компонента CheckBox (Standard) для изменения начертания шрифта, изменить значения свойств для каждого:

- Name – chBold, chItalic, chUnderLine
- Caption –Ж К Ч

1.7 На форме установить компонент RichEdit (Win32), изменить значения свойств для этого объекта:

- Name Document, ScrollBars ssBoth

1.8 Для объекта sdFontSize выбрать событие OnChange и ввести следующий код:

- document.Font.Size=sdFontSize->Value ;

1.9 Для объекта chBold выберите событие OnClick и ввести следующий код:

```
if (chBold->Checked){
document->Font->Style = document->Font->Style>>[fsBold]
}
```

1.10 Сохранить проект и проверить его работу:

- ввести произвольный текст;
- изменить размер шрифта и начертание Полужирный.

1.11 Самостоятельно ввести код для объектов chItalic и chUnderline (использовать константы fsItalic, fsUnderline), проверить работу приложения.

1.12 Для объекта cobFontColor выбрать событие OnChange и ввести следующий код:

1.13 Document->Font->Color = cobFontColor->Selected;

1.14 Сохранить и проверить работу программы.

Контрольные вопросы

1. Какие типы меню существуют и как они создаются в приложении?
2. Какие возможности имеются для настройки меню?
3. Какие возможности имеются для настройки формы?
4. Какая последовательность создания панели инструментов?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 22 Применение Drag&Drop для работы пользователя с интерфейсом программы

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки разработки пользовательского интерфейс получить навыки реализации технологии Drag&Drop в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Порядок выполнения задания:

1. Создать новый проект.

2. На форму добавить компонент **Image** с закладки **Additional.** В свойстве **Picture** компонента **Image** нажать на троеточие и выбрать нужный рисунок. При необходимости растянуть компонент, чтобы был виден весь рисунок.

3. Добавить нужное количество компонентов **StaticText.** Настроить свойства этого компонента: **Color = clBtnHighlight** *(белый фон)***, BorderStyle = sbsSingle** *(компонент в рамке),***Caption =" "***(нет текста в компоненте),* **AutoSize=false**, **Alignment= taCenter** (*текст в компоненте по центру*), **DragMode= dmAutomatic** *(свойство отвечающие за автоматическое перетаскивание компонента).* Все компоненты должны иметь одинаковые размеры, совпадающие с картинкой.

4. Добавить нужное количество компонентов **Label.** Настроить свойства этого компонента: **Transparent = true** *(прозрачный фон в компоненте)***, Caption =" "***(нет текста в компоненте),* **AutoSize=false**, **WordWrap =true, Alignment= taCenter и Layout= tlCenter** (*текст в компоненте по центру*). Все компоненты должны иметь одинаковые размеры, совпадающие с картинкой. Компоненты расположить в прямоугольниках на картинке. Имя компонента настроить так, чтобы было понятно, что этот компонент обозначает, т.е. если компонент отвечает за шрифт, то его имя **Name = labFont.**

5. Добавить кнопку **Далее**, по щелчку которой открывается вторая форма.

6. Добавить кнопку **Очистить**, щелчок по которой очищает поле с ответами. Для того в событиях этого компонента щелкнуть по событию **OnClick** и написать следующий код:

```
labFont->Caption="";
void __fastcall TfrmVopros1::BitBtn2Click(TObject *Sender)
{
// убрать надпись из компонента Label отвечающего за интервал
labInterval->Caption="";
labNachertanie->Caption="";
labSize->Caption="";
labSpisok->Caption="";
labStil->Caption="";
labViravnivanie->Caption="";
// сделать видимым компонент StaticText отвечающего за шрифт
stextFont->Visible=true;
stextInterval->Visible=true;
stextNachertanie->Visible=true;
stextSize->Visible=true;
stextSpisok->Visible=true;
stextStil->Visible=true;
stextViravnivanie->Visible=true;
}
```

7. В одном из компонентов **Label** в событие **DragDrop** написать следующий код**:**

```
void __fastcall TfrmVopros1::labFontDragDrop(TObject *Sender, TObject *Source, int X, int Y)
{
// компонент StaticText, который перетаскивается в данный момент
TLabel *S = (TLabel *)Source;
// компонент Label, в который помещается перетаскиваемый компонент
 ((TLabel*)Sender)->Caption=S->Caption;
      //  StaticText, принимает надпись компонента StaticText
     // компонент StaticText, который перетаскивается в данный момент,
 S->Visible=false;
   //становиться невидимым
}
```

Во всех остальных компонентах **Label** в событие **DragDrop** сослаться на компонент для которого написан код, т.е. в нашем случае на **labFont**

8. В одном из компонентов **Label** в событие **DragOver** написать следующий код**:**

*void __fastcall TfrmVopros1::labFontDragOver(TObject *Sender, TObject *Source, int X, int Y, TDragState State, bool &Accept)*

{

Accept=true ; // разрешить принимать компонент, который перетаскивается

}

Во всех остальных компонентах **Label** в событие **DragOver** сослаться на компонент для которого написан код, т.е. в нашем случае на **labFont**

9. Проверка на правильный ввод ответов осуществляется в последней форме путем накапливания правильных ответов и в зависимости от количества правильных ответов выводиться оценка.

Контрольные вопросы

1. Что означает параметр Sender?
2. Что означает параметр Source?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 23 Проектирование многооконных приложений

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки проектирования многооконных приложений в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Разработать приложение «Тригонометрические функции». Приложение должно работать следующим образом:

- после загрузки на экран выводится окно-заставка, которое отображается 5 сек;
- через 5 секунд заставка пропадает, на экран выводится основное окно;
- в основном окне отображается таблица со значениями тригонометрических функций с заданным шагом.

Порядок выполнения задания:

1. Загрузить C++ Builder, сохранить проект.
2. Изменить свойства главной формы в соответствии с таблицей:

BorderIcons	biMaximize=False
BorderStyle	bsSingle
Position	poScreenCenter

3. Добавить на форму компонент **StringGrid** (Additional)и изменить свойства в соответствии с таблицей:

Align	alClient
ColCount	5
RowCount	2
DefaultColWidth	120
DefaultRowHeight	18
ScrollBars	ssVertical
Options.goRangeSelect	False

4. Для события **OnCreate** ввести код:

```
int t, i, n;
double step,x,sx,cx;
step=0.01;
n=round(90/step)+1;
StringGrid1->Cells[0][0]='x';
StringGrid1->Cells[1][0]=”sin(x)”;
StringGrid1->Cells[2][0]=”cos(x)”;
StringGrid1->Cells[3][0]=”g(x)”;
StringGrid1->Cells[4[[0]=”ctg(x)”;
StringGrid1-> ColWidths[0]= StringGrid1->ColWidths[0] / 2;
StringGrid1-> RowCount=n+1;
for (i=1;i<n;i++)
{x=(i-1)*step;
 sx=sin(x*pi/180);
cx=cos(x*pi/180);
 StringGrid1->Cells[0][i]=FloatToStr(x);
StringGrid1->Cells[1][i]=FoatToStr(sx);
 StringGrid1->Cells[2][i]=FloatToStr(cx);
 if (cx != 0)
StringGrid1->Cells[3][i]=FloatToStr(sx/cx);
else
StringGrid1->Cells[3][i]=”не существует”;
 if (sx!=0)
StringGrid1->Cells[4][i]=FloatToStr(cx/sx);
else
StringGrid1->Cells[4][i]= “не существует”;
}
```

5. Проверить работу приложения.

Задание:

6. Добавить новую форму (заставка).
7. Изменить свойства **Form2** в соответствии с таблицей:

BorderIcons	biSystemMenu=False biMinimize=False biMaximize=False
BorderStyle	bsNone
Position	poScreenCenter
FormStyle	fsStayOnTop

8. На форму Form2 добавить метку, ввести для нее заголовок

ТРИГОНОМЕТРИЧЕСКИЕ ФУНКЦИИ

9. Проверить работу приложения.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 24 Файловый менеджер

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения, позволяющее пользователю работать с файловой системой используемого компьютера в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Разработать приложение, позволяющее пользователю работать с файловой системой компьютера:

1. просмотр содержимого внешних носителей информации;
2. просмотр дерева каталогов диска;
3. просмотр содержимого выбранного каталога;
4. удаление и переименование файла;
5. определение объема свободной памяти диска.

Контрольные вопросы:

1. Какими компонентами библиотеки VCL вы пользовались при разработке приложения?
2. Опишите компонент TListView.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи;

оценка «4» ставится, если:

-обучающийся реализовал четыре функции приложения;

оценка «3» ставится, если:

-обучающийся реализовал три функции приложения;

оценка «2» ставится, если:

-обучающийся реализовал менее трех функций приложения;

Практическое занятие 25 Программа «Редактор текста»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

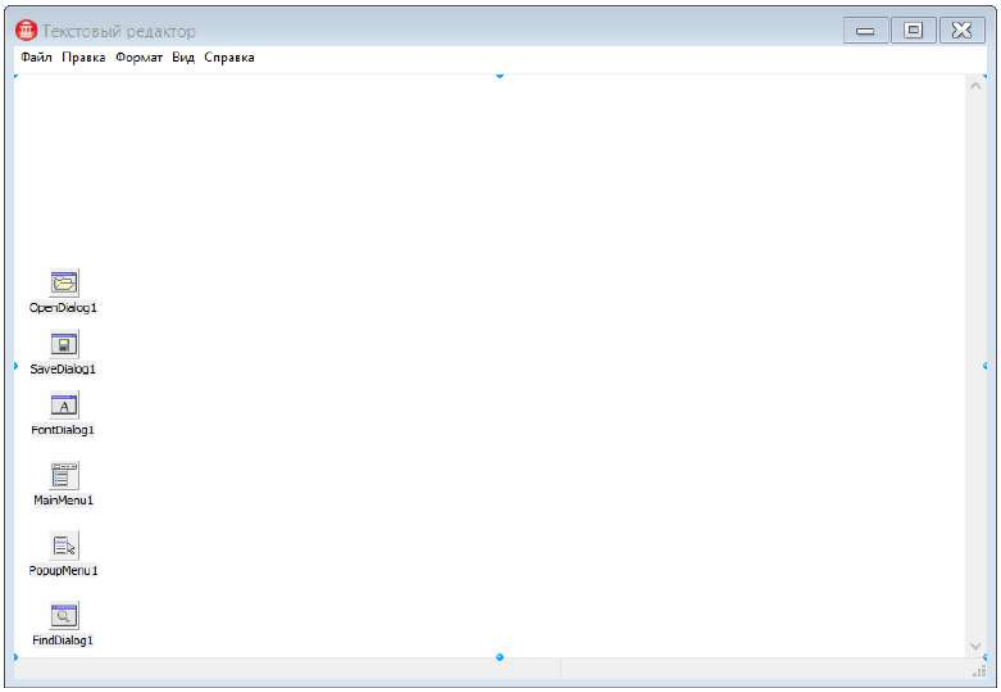
Цель: получить навыки создания приложение, позволяющее пользователю создать приложение для работы с текстами и текстовыми файлами в C++ Builder.

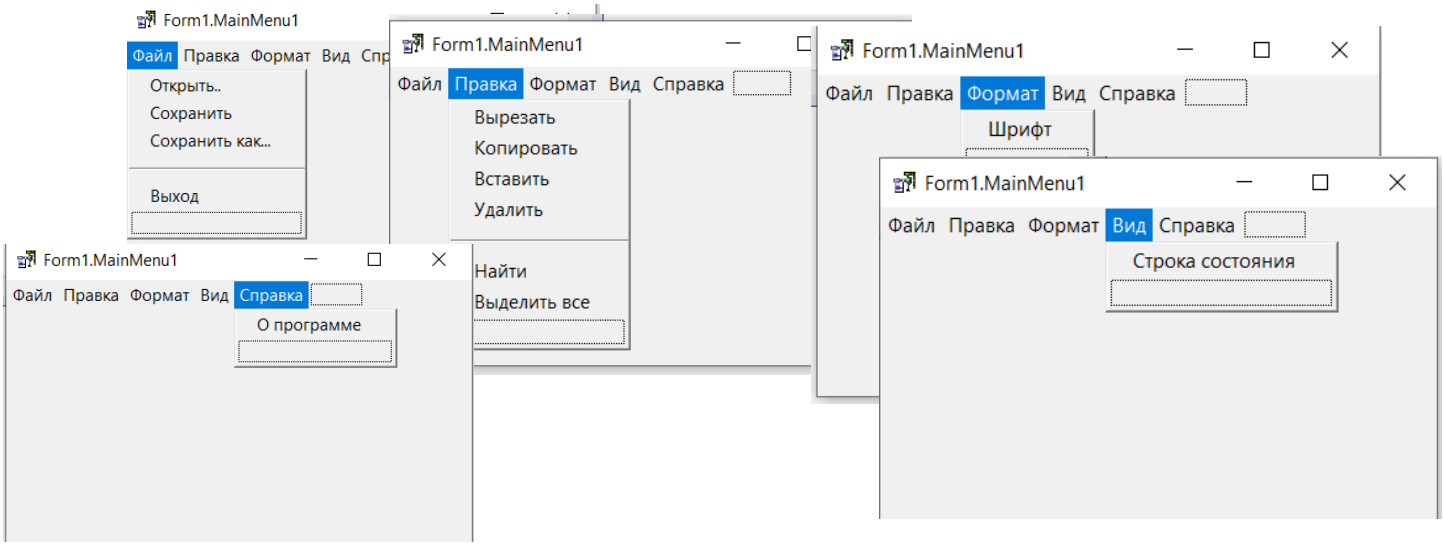
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Главная форма программы и структура меню приведены на рисунке:





Задание:

1. Создать контекстное меню, дублирующее основные команды (Вырезать, Копировать, Вставить)

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи;

оценка «4» ставится, если:

-обучающийся реализовал функции приложения без контекстного меню;

оценка «3» ставится, если:

-обучающийся реализовал все подпункты главного меню Файл и Справка;

оценка «2» ставится, если:

-обучающийся реализовал менее двух пунктов главного меню;

Практическое занятие 26 Рисование простейших геометрических объектов

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложений, позволяющих выводить графику на поверхность формы и в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

1: Разработать программу «Олимпийский флаг»

Окно программы Олимпийский флаг:


```
//конструктор формы
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
  Canvas->Font->Name = "Tahoma";
  Canvas->Font->Size = 12;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FormPaint(TObject *Sender)
{
   #define WB 140                          //ширина полотнища
   #define HB 80                           //высота полотнища
   #define D  36                           //диаметр колец
   int x,y;
   AnsiString st = "Быстрее, выше, сильнее!";
//вычислить координаты левого верхнего угла флага
```

```
  x =  (ClientWidth -  WB) / 2;
   y = (ClientHeight - HB ) / 2 - Canvas->Font->Size;
//полотнище
   Canvas->Brush->Color =  (TColor) RGB(255,255,255);
   Canvas->FillRect( Rect(x,y,x+WB,y+HB) );
   int x1=(ClientWidth-Canvas->TextWidth(st))/2;
/*Чтобы область вывода текста не была закрашена цветом фона, а также чтобы метод Ellipse рисовал окружность, а не круг, значение свойства Brush->Style должно быть равно bsClear*/
   Canvas->Brush->Style = bsClear;
//девиз
      Canvas->TextOut(x1,y+HB+6,st);
      Canvas->Pen->Width = 2;                 //ширина колец – два пикселя
//первый ряд колец
//3.2*D – ширина области, занимаемой кольцами 1-го ряда
      x=x+(WB-3.2*D)/2;
      y=y+(HB-1.6*D)/2;
      Canvas->Pen->Color = (TColor) RGB(0,0,225);          //синий
      Canvas->Ellipse(x,y,x+D,y+D);
      x= x + 1.1* D;
      Canvas->Pen->Color = clBlack;                        //черный
      Canvas->Ellipse(x,y,x+D,y+D);
      x= x + 1.1 * D;
      Canvas->Pen->Color = (TColor) RGB(255,0,0);          //красный
      Canvas->Ellipse(x,y,x+D,y+D);
      x= x - D * 0.55;
      y= y + 0.6 * D;
      Canvas->Pen->Color = (TColor) RGB(0,128,0);          //зеленый
      Canvas->Ellipse(x,y,x+D,y+D);
      x = x - 1.1 * D;
      Canvas->Pen->Color = (TColor) RGB(250,217,25);       //желтый
      Canvas->Ellipse(x,y,x+D,y+D);
}
//пользователь изменил размер окна
void __fastcall TForm1::FormResize(TObject *Sender)
{
 Form1->Refresh();
}
```

2: Программа «Светофор»

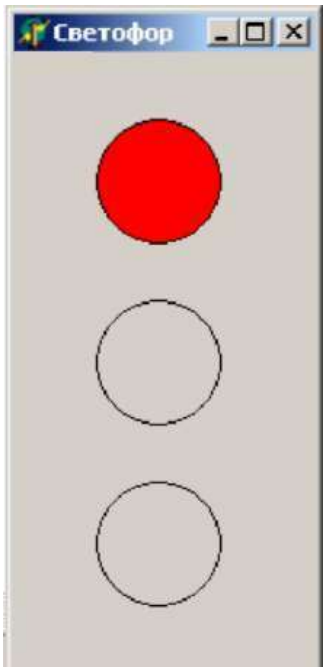
Создать программу, выполняющую следующие действия:

1. После запуска программы в окне изображается светофор с тремя лампочками, способными реагировать на наведение указателя мыши.

2. Когда указатель мыши наведен на лампочку, она меняет свой цвет.

3. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

Окно программы представлено на рисунке:



Описание плана разработки программы:

1. Открыть новый проект.
2. Разместить на форме экземпляры компонентов: фигура Shape.
3. Выполнить следующие действия:

Выделенный объект	Вкладка окна Object Inspector	Имя свойства/ имя события	Действие
Form1	Properties	Caption	Установка имени формы "Светофор"
		Height	Присвоить значение 300
		Width	Присвоить значение 120
		BorderIcons (Служебные кнопки)	Выбрать для подсвойства biMinimize (Сворачивание) и biMaximize (Разворачивание) значение False
		Color	Задать стандартный серый цвет
		BorderStyle (Стиль рамки)	Выбрать значение bsSingle
	Events	OnMouseMove	В процедуру передаются дополнительные параметры: Shift - указывает, не была ли при перемещении нажата клавиша SHIFT, CTRL или ALT; X -горизонтальная координата указателя мыши; Y -вертикальная координата указателя.
Shape1 (Вкладка Additional)	Properties	Height	Присвоить значение 61
		Width	Присвоить значение 61
		Shape (Форма)	Выбрать значение stCircle (Круг)
		Pen (Контур)	Выбрать для подсвойства Color (Цвет) значение clRed (красный цвет)
		Brush (Кисть)	Выбрать для подсвойства Style (стиль) значение bsClear (прозрачный)
		Enabled (Включен)	Выбрать значение False (Нет)
Shape2 (Вкладка Additional)	Properties	Height	Присвоить значение 61
		Width	Присвоить значение 61
		Shape	Выбрать значение stCircle (Круг)
		Pen	Выбрать для подсвойства Color (Цвет) значение clYellow (желтый цвет)
		Brush	Выбрать для подсвойства Style значение bsClear
		Enabled (Включен)	Выбрать значение False (Нет)

Shape3 (Вкладка Additional)	Properties	Height	Присвоить значение 61
		Width	Присвоить значение 61
		Shape	Выбрать значение stCircle (Круг)
		Pen	Выбрать для подсвойства Color (Цвет) значение clLime (ярко-зеленый цвет)
		Brush	Выбрать для подсвойства Style значение bsClear
		Enabled (Включен)	Выбрать значение False (Нет)

Контрольные вопросы:

1. Описать основные методы класса TCanvas
2. Опишите назначение объектов Pen и Brush.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 27 Разработка графического интерфейса

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки разработки графического интерфейса в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Краткие теоретические сведения

Создание меню

В приложениях, написанных на C++ Builder, могут быть реализованы меню двух основных типов:

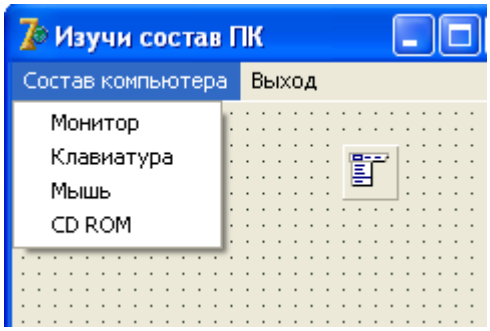
- главное меню (MainMenu),оно при-надлежит форме и отображается под ее панелью заголовка;

Постановка задачи:

Разработать приложение «Состав компьютера». При выборе соответствующего пункта меню или пиктограммы на панели инструментов на экране отображается графическое изображение устройства и информация о назначении устройства. При наведении курсора отображается всплывающая подсказка о названии устройства.

Порядок выполнения

1. Создать и сохранить новый проект.
2. Разместить на форме компонент **MainMenu** (Standard), выполните по нему двойной щелчок. Ввести подпись **Монитор** и имя **mnuMonitor** первому пункту меню и, по аналогии, всем остальным пунктам (см. рисунок).
3. Расположить на форме четыре компонента **Image**, задать для них информативные имена (например, picMinitor, picKey и т.п.), для свойства **Picture** выбрать соответствующий графический файл, сделать компоненты невидимыми, назначить всплывающие подсказки.



4. Расположить на форме метку (Name - Info).
5. Для пункта меню **Монитор** ввести следующий код:

```
picMonitor->Visible =true;
picKey->Visible=false;
picMouse->Visible =false;
picCd->Visible=false;
```

6. Самостоятельно добавить в программу вывод в метку информации о назначении выбранного устройства.
7. Ввести программный код для остальных пунктов меню.
8. Проверить работу приложения.
9. Создать панель инструментов:
   - расположить на форме компонент **ToolBar** (Win32), присвоить имя **MainPanel**, для свойства **EdgeBorders** добавить значение **ebBotton =True**
   - щелкнуть на созданной панели правой кнопкой мыши и выбрать **NewButton**, установить кнопке следующие свойства:

Hint	монитор
MenuItem	mnuMonitor
Name	btnMonitor
ShowHint	True

   - аналогичным образом создать еще кнопки для остальных устройств и изменить их свойства;
   - для добавления изображений командам и кнопкам добавить на форму компонент **ImageList** (Win32)**;**
   - дважды щелкнуть по компоненту, появится редактор, щелкните по кнопке **Add** и укажите размещение требуемых графических файлов \COMMON\GRAPHICS\ICONS\COMPUTER\... добавьте соответствующие файлы;
   - связать набор пиктограмм с панелью инструментов и основным меню: для компонентов **MainMenu1** и **MainPanel** для свойства **Images** из списка выбрать **ImageList1**;
   - для каждого пункта меню и для каждой кнопки панели инструментов изменить значения свойства **ImageIndex** в соответствии с их назначением.
10. Проверить работу приложения.
11. Добавить в заголовок формы бегущую строку, для этого:

- объявить глобальную переменную

AnsiString a;

- расположить на форме таймер (System) и ввести для него код:

```
int  i;
  Application->Title= a;
  Form1->Caption = a;
  for (i = 1; i< a.Length() – 1;i++)
{    a[i] = Application->Title[i + 1];
   a[a.Length()] = Application->Title[1];
}
```

- в процедуру **FormCreate** ввести код

a = 'Компьютер полезен равно настолько, насколько грамотен использующий его человек... ';

12. Проверить работу приложения.

13. Сделать форму прозрачной, для этого в процедуру **FormCreate** добавить код:

Form1->Brush->Style = bsClear;

Form1->BorderStyle = bsNone

14. Проверить работу приложения.

15. Отменить прозрачность формы.

16. Заполнить форму изображением, для этого:

- установить на форму еще один компонент Image, задать значение для свойства Picture (любой файл .bmp), сделать компонент невидимым;
- в процедуру **FormCreate** добавить код:

Form1->Brush->bitmap=Image1->Picture->Bitmap;

17.Проверить работу приложения.

18. Заблокировать введенный код.

19. Сделать форму полупрозрачной, для этого изменить значения следующих свойств:

AlphaBlend True, (включить полупрозрачность)

AlphaBlendValue 128 (степень прозрачности)

TransparentColorValue clBlack (какой цвет считать прозрачным)

TransparentColor True (включить прозрачность по цвету)

20. Проверить работу приложения.

21. Самостоятельно ввести код для пункта Выход , для того чтобы перед закрытием приложения форма постепенно «растворялась», т.е. от сплошного цвета переходила к прозрачному.

22. Результат показать преподавателю, проект сохранить.

Контрольные вопросы:

1. Какие типы меню существуют и как они создаются в приложении?
2. Какие возможности имеются для настройки меню?
3. Какие возможности имеются для настройки формы?
4. Какая последовательность создания панели инструментов?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 28 MDI-приложение

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки разработки MDI-приложений в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

1. Спроектировать и реализовать MDI-приложение «Обозреватель картинок». *Основная функция*: открывать и просматривать графические файлы. *Предусмотреть возможности*:
   - сохранения графического файла с другими именем;
   - переключения между открытыми картинками с помощью главного меню;
   - включения/выключения режима изменения масштаба и пропорций картинки под размеры окна;
   - упорядочения открытых дочерних окон;
   - вывода информации о количестве открытых в данный момент файлов.

   Реализовать вывод информации о программе в специальном диалоговом окне.

Контрольные вопросы:

1. Понятие формы в C++ Builder. Свойства Owner и Parent.
2. Главная форма приложения в C++ Builder.
3. Основные черты диалоговых окон. Модальные и немодальные окна.
4. Свойства формы BorderStyle и Position.
5. Смысл и назначение указателя this.
6. Понятие однодокументного и мультидокументного интерфейса.
7. Основные черты мультидокументного интерфейса.
8. Свойства формы FormStyle и WindowMenu.
9. Методы упорядочения и управления дочерними формами в MDI-приложениях.
10. SDI или MDI: какой подход, по вашему мнению, лучше?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 29 Графические возможности C++ Builder

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки работы с графическими возможностями в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Разработать приложение, реализующее движение битового образа (картинки) на фоне другой картинки. Примерный вид окна приложения представлен на рисунке.

Порядок выполнения

1. Создать новый проект.
2. Скопировать в папку, где будет сохранен создаваемый проект, файлы **factory.bmp** и **aplane.bmp**.
3. Установить на форме компонент **Image** и **таймер**.

   Примечание: Компонент Image используется для вывода фона, а компонент Timer — для организации задержки между циклами удаления и вывода на новом месте изображения самолета.
4. Объявить глобальные переменные:

```
TBitMap Back, bitmap, Buf; // фон, картинка, буфер
// область фона, которая должна быть восстановлена из буфера
TRect BackRet;
//область буфера, которая используется для восстановления фона
Trect BufRet;
```

```
int x,y; // текущее положение картинки
int W,H; // размеры картинки
```

5. Для формы выбрать событие **OnActivate** и ввести код:

```
// создать три объекта — битовых образа
Back = TBitmap->Create; // фон
bitmap = TBitmap->Create; // картинка
Buf = TBitmap->Create; // буфер
// загрузить и вывести фон
Back->LoadFromFile("factory.bmp");
Form1->Image1->Canvas->Draw(0,0,Back);
// загрузить картинку, которая будет двигаться
bitmap->LoadFromFile("aplane.bmp");
bitmap->Transparent = True; // определим "прозрачный" цвет
bitmap->TransParentColor = bitmap->Canvas->pixels[1,1];
// создать буфер для сохранения копии области фона,
// на которую накладывается картинка
W= bitmap->Width;          H= bitmap->Height;
Buf->Width= W;      Buf->Height=H;
Buf->Palette=Back->Palette;
// Чтобы обеспечить соответствие палитр
Buf->Canvas->CopyMode=cmSrcCopy;
// определим область буфера, которая будет использоваться
// для восстановления фона
BufRet=Bounds(0,0,W,H);
// начальное положение картинки
x = -W; y = 20;
// определим сохраняемую область фона // и сохраним ее
BackRet=Bounds(x,y,W,H);
Buf->Canvas->CopyRect(BufRet,Back.Canvas,BackRct);
```

6. Для таймера ввести код:

```
// восстановлением фона (из буфера) удалим рисунок
Form1->Image1->Canvas->Draw(x,y,Buf);
x=x+2;
if (x>Form1->Image1->Width) x=-W;
// определим сохраняемую область фона
BackRet=Bounds(x,y,W,H);
// сохраним ее копию
Buf->Canvas->CopyRect(BufRct,Back.Canvas,BackRct);
// выведем рисунок
Form1->Image1->Canvas->Draw(x,y,bitmap);
```

7. Для формы выбрать событие OnClose и ввести код:

```
// освободим память, выделенную  для хранения битовых образов
Back->Free;
bitmap->Free;
Buf->Free;
```

8. Проверить работу приложения.

Задание:

Самостоятельно модернизировать приложение, для того чтобы траектория движения соответствовала синусоиде.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 30 Рекурсивные графические построения

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

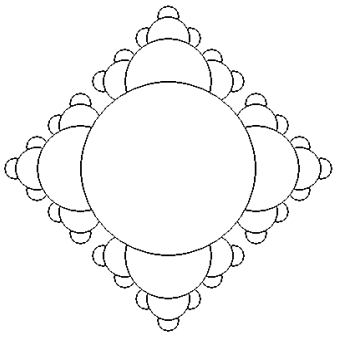
- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки использования рекурсии в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:



Разработать приложение для рекурсивного построения изображения, представленного на рис.

Порядок выполнения

1. Создать новый проект.
2. Объявить глобальные переменные

int k,gd,gm,mx,my;

char ch ;

3. Обработчик OnCreate

```
k=15;//минимальный размер окружности
```

4. Обработчик OnMouseDown

```
krug(X , Y , Y / 2);
```

5. Создать процедуру

```
void krug(int x,y,r);
{
  if (r>k)
    {
```

```
            krug(x+r,y,r / 2);
            krug(x,y+r,r / 2);
            krug(x-r,y,r / 2);
            krug(x,y-r,r / 2);
        }
      Form1->Canvas->Ellipse (x-r,y-r,x+r,y+r);
    }
```

6. Проверить работу программы.
7. Изменить процедуры **krug и**

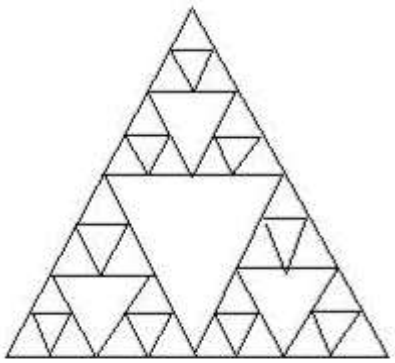
```
    if (r>k)
      {
      Form1->Canvas->Pen->Color =RGB(rand(255),rand(255),rand(255));
            krug(x+r+(r / 2),y,r / 2);
            krug(x,y+r+(r / 2),r / 2);
            krug(x-r-(r / 2),y,r / 2);
            krug(x,y-r-(r / 2),r / 2);
        }
       Form1->Canvas->Ellipse (x-r,y-r,x+r,y+r);
    }
```

OnMouseDown,проверить работу программы

```
    krug(X , Y , Y / 4)
```

Заменить рисование окружности на рисование прямоугольника и просмотреть результат.

8. Самым знаменитым примером площадного геометрического фрактала является треугольник Серпинского (см. рис.), строящийся путем разбиения треугольника, необязательно равностороннего – средними линиями на четыре подобных треугольника, исключением центрального и рекурсивного разбиения угловых треугольников до получения площадных элементов желаемого разрешения.



Добавить на форму командную кнопку и ввести для нее следующий код

```
    tri(150,150,225,0,300,150,5);
```

Создать и объявить следующую процедуру

```
    void TRI(int x1,y1,x2,y2,x3,y3, N);
    {
    int x12,y12,x23,y23,x31,y31;
    if (N!=0)
        begin
         x12=(x1+x2) / 2;    y12=(y1+y2) / 2;
         x23=(x2+x3) / 2;    y23=(y2+y3) / 2;
         x31=(x3+x1) / 2;    y31=(y3+y1) / 2;
    Form1->Canvas->MoveTo(x31,y31);
    Form1->Canvas->LineTo(x12,y12);
    Form1->Canvas->LineTo(x23,y23);
    Form1->Canvas->LineTo(x31,y31);
      TRI(x1,y1,x12,y12,x31,y31, N-1);
      TRI(x2,y2,x12,y12,x23,y23, N-1);
      TRI(x3,y3,x31,y31,x23,y23, N-1)
    }       }
```

9. Проверить работу программы.

Задание: Самостоятельно разработать программу, которая рисует множество Кантора.

Рисунок множество Кантора образован квадратами. Каждый следующий квадрат в четыре раза меньше предыдущего. Центр каждого следующего квадрата расположен в вершине предыдущего квадрата и т.д. Так как рисунок состоит из однотипных элементов, и есть явная зависимость, как размеров, так и положения, следовательно, при создании данного рисунка можно использовать в программе рекурсию.

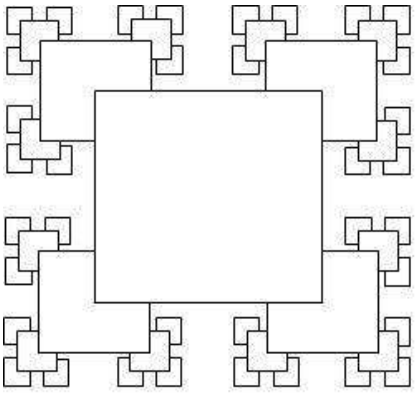


рис. 5

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 31 Построение графиков функций

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки использования компонента TChart в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи:

Написать программу, отображающую график функции y=sin(x).

Порядок выполнения

Листинг программы:

//Обработчик события OnActivate

```
void __fastcall TForm1::FormActivate(TObject *Sender)
{
 double x,y;
 Series1->Clear();
//цикл, на котором график будет строиться на промежутке от -10 до 10 с шагом 0.1
 for( int i=-10; x<=10; x=x+0.1)
 {
        y=sin(x);
        Series1->AddXY(x,y);
 }
 Chart1->Refresh();
 /*устанавливается шаг сетки по оси y. Шаг вычисляется как разница между max и min значением  на графике деленная на 5*/
 Chart1->LeftAxis->Increment=(Chart1->LeftAxis->Maximum-Chart1->LeftAxis->Minimum)/5;
}
```

Задание:

Написать программу, отображающую графики функций:

— $y = \sin(x) \cdot x$

— $y = cos^3(x)$

— $y = sin^3(x) + cos^3(x)$

— $y = \sin(10x) + cos\left(\frac{x}{2}\right)$

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 32 Использование звука и видео в приложениях

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки использования медиаплеера при проектировании приложений TChart в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Краткие теоретические сведения

Компонент MediaPlayer, значок которого находится на вкладке System, позволяет воспроизводить видеоролики, звук и сопровождаемую звуком анимацию.

Постановка задачи:

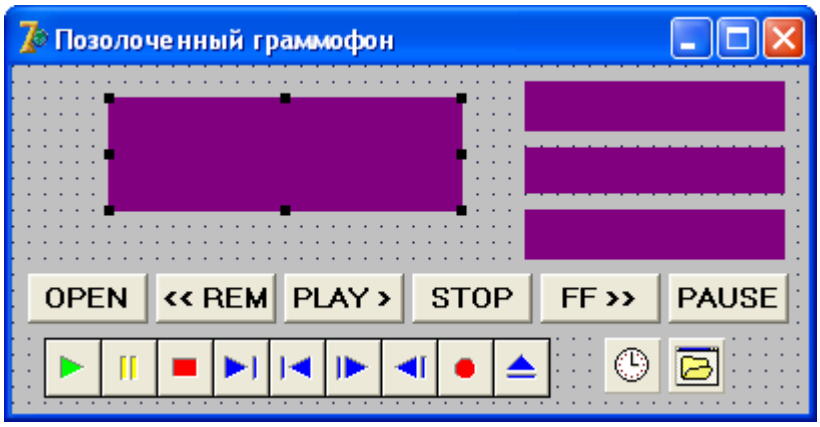
Разработать приложение, позволяющее воспроизводить звуковые файлы различных форматов, приложение должно выводить информацию о длине звукового файла, времени воспроизведения и времени, которое осталось до конца звучания файла.

Порядок выполнения

1. Создать новый проект.
2. Для формы установить следующие значения свойств:

Borderlcons / biMaximize	False
BorderStyle	bsSingle
Caption	Позолоченный граммофон
Color	ClGray
Position	poScreenCenter

3. Установить на форму компонент **MediaPlayer** (System), для свойства **Visible** установите значение **False**.
4. Установитm на форме следующие компоненты: шесть кнопок, четыре метки, таймер, компонент для открытия файлов/



Примечание:

в **Label1** (самый большой индикатор) будет показываться текущая позиция в файле.

в **Label2** (справа сверху) должно отображаться имя открытого файла

в **Label3** (справа в середине) - длина открытого файла в **Label4** - сколько осталось времени до конца файла.

5. Для всех меток установить значение свойства **AutoSize** = **False**
6. Привести форму в соответствие с образцом.
7. Для всех кнопок установить значение свойства **Cursor crHandPoint**
8. Самостоятельно изменить интерфейс проекта (цвет, шрифты) и сохранить.
9. Для инициализации объектов для **Form** выберите событие **OnActivate** и введите код:

```
label1->Font->Color = clGreen;          Label2->Font->Color = clGreen;
LabeL3->Font->Color := clGreen;         Label4->Font->Color = clGreen;
Label1->Caption = "00:00";              Label2->Caption = "no file...";
LabeL3->Caption :="'00:00";                    LabeL4->Caption = "00:00";
 //задание фильтра для OpenDialog
 OpenDialog1->Filter = "Wav files (*.wav)|*.WAV";
```

10. Определите процедуру, которая выполняется при открытии звукового файла **(**кнопка **OPEN)**

```
If (OpenDialog1.Execute)
{
MediaPlayer1->FileName = OpenDialog1->FileName;
MediaPlayer1->Open;                          // открываем файл
Label1->Font->Color = clLime;       // включаем индикаторы
Label2->Font->Color = clLime;
LabeL3->Font->Color = clLime;
Label4->Font->Color = clLime;
Output;                                      // присваиваем индикаторам значения
 Timer1->Enabled=True;
```

11. Создать новую функцию **MusicToStr** и процедуру **Output**, которая преобразует длину файла, текущую позицию или количество оставшегося до конца звучания времени в форму, легкую для восприятия.

```
char* TForm1::MusicToStr(int n)
{ char* s;
  int min, sec;
{
//получаем секунды и минуты из миллисекунд
sec = round(n/1000);
min = round(sec/60);
sec = sec - min*60;
s = IntToStr(min);
/* добавляем в строку 0, если секунд больше десяти // чтобы получилось 2:06, а не значение 2:6; */

if (sec<10)
   s = s + ":0" + IntToStr(sec);
 else
  s = s + ":" + IntToStr(sec);
 MusicToStr = s;
} ;
void TForml::Output()                    //вывод значений на индикаторы
{
int leng, posit, remain;
//  длина файла, позиция в файле и оставшееся время
{
//читаем свойства проигрывателя
leng = MediaPlayer1->Length;
posit = MediaPlayer1->Position;
remain = leng - posit;
Label1->Caption = MusicToStr(posit);
Label2->Caption = MediaPlayer1->FileName;
Label3->Caption = MusicToStr(leng);
Label4->Caption = MusicToStr(remain);
};
```

12. Функцию MusicToStr и процедуру Output добавить в класс.

13. Определите процедуру, которая выполняет проигрывание звукового файла(кнопка **PLAY**):

```
if (mpCanPlay in MediaPlayer.Capabilities) MediaPlayer1->Play;
```

13. Для кнопок **STOP** и **PAUSE** ввести соответствующий код:

```
if (mpCanPlay in MediaPlayer1.Capabilities)
```

```
{ MediaPlayer1->Stop();
MediaPlayer1->Position = 0;  };
```

```
if (mpCanPlay in MediaPlayer1.Capabilities) MediaPlayer->Pause();
```

14. Ввести код для кнопок перемотки вперед и назад:

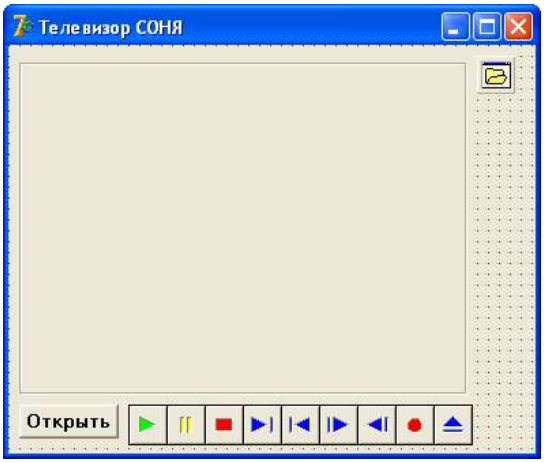
//кнопка перемотки вперед

```
if (mpCanPlay in MediaPlayer1.Capabilities)
{
if (MediaPlayer1->Position+10000==MediaPlayer1.Length)
      MediaPlayer1->Position = MediaPlayer1->Position + 10000
else
      MediaPlayer1->Position= MediaPlayer1.Length;
 MediaPlayer1->Play();
}
```

//кнопка перемотки назад

```
if (mpCanPlay in MediaPlayer1.Capabilities)
{
      if (MediaPlayer1->Position>=10000)
       MediaPlayer1->Position = MediaPlayer1->Position -10000
      else
           MediaPlayer1->Position = 0;
MediaPlayer1->Play();
}
```

12. Для объекта **таймер** установить значения свойства **Interval**= 1000, **Enabled** = **False**, выберить событие **OnTimer** и ввести вызов процедуры **OutPut.**
13. Добавить в конец процедуры **Button1Click** строку Timerl->Enabled=true;
14. Проверить работу приложения.
15. Самостоятельно модернизировать приложение для воспроизведения звуковых файлов различных форматов.
16. Проект сохранить. Результат показать преподавателю.



Задание:

1. Создать новый проект.
2. На форме установить компоненты: одну кнопку (для открытия файлов), один компонент **MediaPlayer**, одну рамку **GroupBox** (Standard), один диалог для открытия файлов
3. Всем объектам изменить свойств в соответствии с таблицей:

компонент	свойство	значение
Forml	BorderIcons/biMaximize bibiMaximize	False
	BorderStyle	bsSingle
	Caption	Телевизор «СОНЯ»
	Height	356
	Width	351
Button 1	Caption	Открыть
	Left	3
	Top	296
MediaPlayer1	Left	88
	Top	296
GroupBox	Caption	
	Height	289

	Width	337
	Left	3
	Top	0

4. Сравните свою форму с образцом
5. Для кнопки **Открыть** ввести код:

```
if (OpenDialog1->Execute==true)
{
MediaPlayer1->FileName = OpenDialog1->FileName;
      MediaPIayer1->Open();
}
```

6. Для формы выберите событие **OnActivate** и ввести код:

```
//назначаем область просмотра для видеофайла
MediaPlayer1_.Display = GroupBox1;
```

7. Проверьте работу приложения

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 33 Разработка элементов графического редактора

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки разработки приложений для работы с грфикой приложений TChart в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения

Задача '.1. Подготовить рисунок, содержащий три геометрические фигуры: круг, квадрат и прямоугольник. Причем требуется эти три фигуры разместить в окне `Image` таким образом, как это показано на рис. 17.1. Предусмотреть такую закраску фигур, чтобы каждая имела свой цвет и свою окантовку. Опробовать разные режимы рисования фигур на фоне имеющегося рисунка.

Решение

Анализ изображения фигур, представленных на рис.17.1, допускает выделение в этом рисунке двух составляющих: круг – это будет неизменная часть; прямоугольник и квадрат – это изменяющаяся часть. Такое распределение ролей позволит посмотреть возможности рисования в различных режимах.

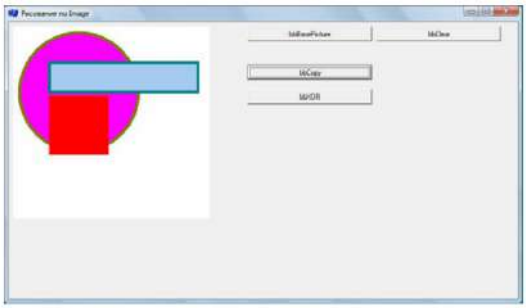


Рис. 17.1. Интерфейс приложения формирования простейшего изображения

Предлагается следующий порядок действий для построения изображения:
1) построить статическую часть – окружность;
2) построить динамическую часть – прямоугольник и квадрат.

Рассмотрим *два варианта* решения. В первом варианте будет использоваться режим `Mode – pmCopy`, который обеспечивает простое копирование. Во втором варианте будет использоваться режим `Mode – pmXOR`, который обеспечивает комбинацию цветов пера (текущего и нового).

Вариант 1.

Для каждого действия целесообразно предусмотреть отдельную процедуру (можно коды построения включить в обработчики событий под кнопки, как это показано в интерфейсе приложения на рис. 17.1):

- `bbBasePicture` (строит статическую часть изображения; здесь – только окружность);
- `bbCopy` (строит переменную часть изображения, имитирующую редактируемый элемент изображения).

Изображение окружности

Процедура (действия по построению) встроена в код обработчика события `OnClick` кнопки `Button` (с именем `bbBasePicture`). В теле процедуры устанавливается режим рисования, настраивается размер и положение элемента, цвета для отображения границы и области и вызывается подпрограмма `Ellipse`, необходимая для построения круга.

Предупреждение

На первый взгляд, необходимости в первой строке, устанавливающей режим рисования как копирование, у этой процедуры нет, так как этот режим установлен по умолчанию. Но предполагается, что процедура будет работать совместно с другими процедурами, которые будут изменять значение свойства `Mode`. Поэтому в будущем эта строка потребуется, чтобы установить именно режим `pmCopy` независимо от того, какое значение `Mode` было ранее.

Код процедуры обработчика события `OnClick` кнопки `bbBasePicture` представлен в листинге 17.1.

Листинг 17.1. Процедуры формирования изображения «Круг»

```
void __fastcall TfoMain::bbBasePictureClick(TObject *Sender)
{
  Image->Canvas->Pen->Mode = pmCopy; // режим рисования
        // параметры эллипса/круга
  int xLT_E = 10, yLT_E = 10, xRB_E = 210, yRB_E = 210;
        // цвет и ширина пера
  Image->Canvas->Pen->Color = clOlive;
  Image->Canvas->Pen->Width = 5;
        // цвет кисти – заливки круга
  Image->Canvas->Brush->Color = clFuchsia;
  Image->Canvas->Ellipse(xLT_E, yLT_E, xRB_E, yRB_E);
}
```

Изображение квадрата и прямоугольника

Процедура, обеспечивающая действия по построению квадрата и прямоугольника, встроена в код обработчика события `OnClick` кнопки `Button` с именем `bbCopy`. В теле процедуры

- устанавливается режим рисования;
- настраивается размер и положение прямоугольника;
- настраивается цвет отображения границы и области;
- вызывается подпрограмма `Rectangle`, необходимая для построения прямоугольника;
- строится квадрат.

Квадрат же рисуется просто: в цикле закрашиваются соответствующие пиксели, заданным цветом. В процедуре прямоугольник и квадрат строятся разными способами: первый – *посредством функции*; второй – *поПиксельно*. Описание процедуры-обработчика представлено в листинге 2.

Листинг .2. Процедуры формирования изображения «Прямоугольник и квадрат»

```
void __fastcall TfoMain::bbCopyClick(TObject *Sender)
{ Image->Canvas->Pen->Mode = pmCopy; // режим рисования
    // размер, положение
  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
    // цвета
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
    // построение прямоугольника
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
    // построение квадрата
  int xLT_S = 60, yLT_S = 115, xRB_S = 160, yRB_S = 215;
  for (int iy = yLT_S; iy < yRB_S; iy++)
    for (int ix = xLT_S; ix < xRB_S; ix++)
      Image->Canvas->Pixels [ix][iy] = clRed;
}
```

Вариант 2.

Оставим построение статической части (процедуру `BasePicture`) без изменения. А для отображения динамической части построим новую процедуру – обработчик события `OnClick` кнопки `Button` с именем `bbXOR`. Описание этой процедуры представлено в листинге 3. Отличие этой процедуры от текста процедуры `bbCopy`, представленной в листинге ' 2, лишь в значении свойства `Mode`. Здесь оно равно `pmXOR`. В таком режиме `Canvas` не просто копирует в пиксель заданный цвет, а выполняет для каждого пикселя операцию `XOR` над заданным цветом и текущим цветом пикселя.

Примечание

Важно отметить, что для того, чтобы добиться требуемого цвета изображения, процедуру `bbXOR` *нужно выполнить дважды*. Первый раз получится неправильная цветовая гамма. Второй раз – соответствует заданному цвету.

Листинг ' 3. Процедура формирования переменной части изображения

```
void __fastcall TfoMain::bbXORClick (TObject *Sender)
{ Image->Canvas->Pen->Mode = pmXor;  // режим рисования
    // размер, положение
  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
    // цвета
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
    // построение прямоугольника
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
        // построение квадрата
  int xLT_S = 60, yLT_S = 115, xRB_S = 160, yRB_S = 215;
  for (int iy = yLT_S; iy < yRB_S; iy++)
    for (int ix = xLT_S; ix < xRB_S; ix++)
      Image->Canvas->Pixels [ix][iy] = clRed;
}
```

Результат работы функций `bbBasePicture` и первого выполнения `bbXOR` приведен на рис. .2. Хорошо видно, что цвет изображения переменной части нигде не соответствует заданным цветам, но контуры переменной части изображения точно соответствуют заданным.

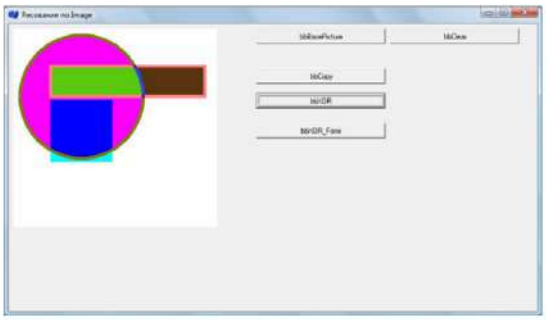


Рис. 17.2. Изображение, сформированное с использованием режима XOR с неправильным отображением цвета

Если теперь повторно выполнить процедуру `bbXOR`, то получим изображение базовой статичной части в «чистом виде», как это требуется.

ПОВЫСИМ ИНТЕЛЛЕКТ

Можно немного повысить «интеллект» процедуры рисования переменной части, например, заставив ее на фоновых участках изображения отображать цвета переменной части корректно. Для этого необходимо выполнять рисование *не заданным цветом*, а цветом, который *в сочетании* с фоновым даст *заданный цвет*.

Операция ИСКЛЮЧАЮЩЕГО ИЛИ

Операция «исключающего или» `XOR` переводит биты в новое состояние в соответствии с используемой маской. А повторное применение операции с той же маской приводит к «восстановлению» битовых значений.

Можно сделать «опережающий шаг» – сразу задать требуемый цвет. Для этого следует преобразовать требуемый цвет в некую переменную `aux`:

```
TColor aux = clRed ^ clWhite;
```

Здесь заданный цвет `clRed` (красный) выполняет маскирование цвета фона `clWhite` (белый). Теперь при построении изображения, если задавать не цвет `clRed` а `aux`, то этот цвет после операции `XOR` с фоном отобразится на экране как `clRed`, а после второго рисования цвет фона будет восстановлен как `clWhite`. Функция-обработчик события `OnClick` кнопки `Button` с именем `bbXOR_Fone`, выполняющая такое рисование, приведена в листинге 4, а результат ее работы – на рис. .3.

Листинг 4. Процедура формирования переменной части изображения с «правильным» отображением цвета на фоновой части

```
void __fastcall TfoMain::bbXOR_FoneClick(TObject *Sender)
{ Image->Canvas->Pen->Mode = pmXor; // режим рисования
```

```
TColor auxTeal = clWhite ^ clTeal; // предварительная
TColor auxSky = clWhite ^ clSkyBlue; // подготовка цвета
    // размер, положение
int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
Image->Canvas->Pen->Color = auxTeal;
Image->Canvas->Brush->Color = auxSky;
    // построение прямоугольника
Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
    // подготовка требуемого цвета
TColor aux = clRed ^ clWhite;
    // построение квадрата
int xLT_S = 60, yLT_S = 115, xRB_S = 160, yRB_S = 215;
for (int iy = yLT_S; iy < yRB_S; iy++)
  for (int ix = xLT_S; ix < xRB_S; ix++)
    Image->Canvas->Pixels [ix][iy] = aux;
}
```

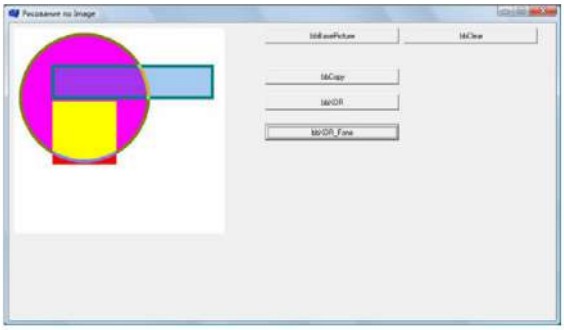


Рис. 3. Изображение с «правильным» отображением цвета на фоновой части

ВЫВОД

Нет необходимости рисовать статичную часть, достаточно рисовать только переменную часть дважды в режиме `pmXOR`:

- первое рисование отображает переменную часть в неправильной цветовой гамме;
- второе рисование «стирает» переменную часть, оставляя неискаженное изображение базовой части.

Задача .2. В окне `Image` помещено изображение некоторых графических объектов. Требуется стереть/очистить это изображение.

Решение

Стирание изображения или его фрагмента является важной операцией графического редактора. Единственный простой и универсальный метод решения – это нарисовать на всем поле изображения прямоугольник, цвет границы и области которого совпадает с цветом фона.

Если же цвет фона белый, что бывает довольно часто, то цвета пера `Pen` и кисти `Brush` можно не перенастраивать, а воспользоваться опять же свойством `Mode` пера, но на этот раз установить значение `pmWhite`, что обозначает *рисовать только белым цветом, несмотря на заданный цвет*. Заметим, что функция очистки используется редко, поэтому, вероятно, было бы правильно, чтобы после своей работы функция очистки `bbClear` приводила бы все параметры в прежнее состояние, здесь это только режим рисования – свойство `Mode`. Код процедуры обработки события `OnClick` кнопки `Button` с именем `bbClear`, выполняющую очистку изображения, предетавлен в листинге 5.

Листинг 5. Процедура очистки изображения белым фоном

```
void __fastcall TfoMain::bbClearClick(TObject *Sender)
{
          // запомнить прежний режим
  TPenMode tmp = Image->Canvas->Pen->Mode;
  Image->Canvas->Pen->Mode = pmWhite; // установить новый
        // установить границы области стирания
  int xLT_R = 0, yLT_R = 0, xRB_R = Image->Width, yRB_R = Image->Height;
        // нарисовать «стирающий» прямоугольник
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);

  Image->Canvas->Pen->Mode = tmp; // восстановить режим
}
```

Задача .3. Имеется некая картинка, состоящая из изображения неизменной части и изображения отрезка прямой линии. Требуется отредактировать изображение этого отрезка – изменить его положение.

Решение

Пусть задача содержит простое изображение с отрезком прямой линии (рис. 4). Для общности случая полагаем, что часть отрезка прямой рисуется *над некоторыми* элементами изображения (круг), а другие элементы (прямоугольник) *закрывают* часть отрезка.

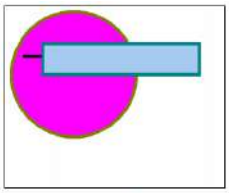


Рис. 4. Изображение, содержащее отрезок прямой

Предлагается следующая стратегия решения:

1. Определяется, какие элементы/фигуры изображения будут подвергаться изменению.
2. На основании такой классификации элементов определяются статическая и динамическая области изображения.
3. Определяется порядок действий для пользователя, которому будет предоставлена возможность корректировать основное изображение.
4. Подготавливаются специальные процедуры для обеспечения действий пользователя, а также для редактирования каждого элемента из динамической области.

Подготовка решения задачи в соответствии с описанной стратегией.

1. На основании условия задачи можно сделать вывод, что изменению подвергается один элемент – отрезок прямой.
2. Область изображения условно разбивается на две составные части: статичное изображение (круг и прямоугольник) и динамическое изображение (прямая, которая должна подвергаться редактированию).
3. Предлагается обеспечить следующий порядок работы с приложением:
   а) на старте программа отображает полное изображение;
   б) при необходимости пользователь с помощью кнопок (назовем их `bbBasePicture` и `bbClear`) может вновь строить изображение или удалять его;
   в) посредством соответствующей кнопки (назовем ее `sbEditLine`) пользователь может включать или выключать режим редактирования отрезка;
   г) в режиме редактирования программа поверх картинки строит изображение редактируемого отрезка с квадратиками на концах. В этом режиме пользователь *может перемещать концевые квадратики* отрезка и тем самым выполнять редактирование положения этого отрезка.

Интерфейс приложения может иметь вид, как на рис. 5.

Подготовка процедур

Самой важной составляющей при разработке процедур является решение задачи редактирования элемента (в данном случае отрезка). При редактировании какого-либо элемента можно воспользоваться одним из двух способов:

1) рисовать неизменную часть изображения, а редактируемый элемент рисовать только в текущем состоянии, то есть в том положении, какое он занимает в данный момент процесса редактирования;
2) рисовать все изображение полностью, а поверх этого изображения – редактируемый элемент повторно. При таком способе пользователь может видеть и прежнее, и текущее положение объекта.

В данном решении будем реализовывать второй вариант. В любом случае РЕДАКТИРУЕМЫЙ ОБЪЕКТ ОТРАЖАЕТСЯ С ДОПОЛНИТЕЛЬНЫМИ ГРАФИЧЕСКИМИ ЭЛЕМЕНТАМИ, которые можно перемещать манипулятором «мышь».

Дополнительные графические элементы

Дополнительные графические элементы – это небольшие квадратики с небольшой площадью, размещенные по контуру редактируемого изображения.

Поскольку в данной задаче редактируется только отрезок прямой, то достаточно иметь два квадратика на концах отрезка. При этом надо понимать, что в момент редактирования пользователь может выполнить щелчок по любому квадратику и начать его перемещение. Так как каждый квадратик имеет небольшую площадь, то щелчок обычно выполняется не в центр этого квадратика, а рядом с ним. Этот факт необходимо учитывать при реализации процедур редактирования.

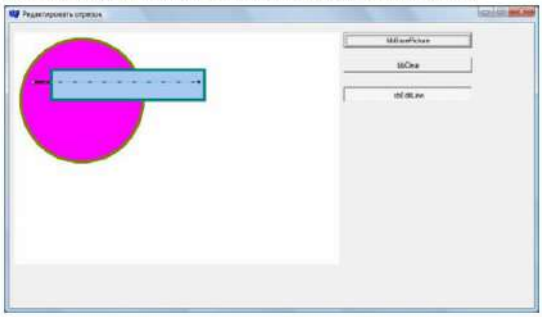


Рис. 5. Интерфейс программы в режиме редактирования отрезка

Для того чтобы предоставить пользователю возможность выполнения действий в соответствии с описанным порядком и особенностями решения задачи редактирования, целесообразно реализовать следующие подпрограммы:

- `DrawFullPict();` – подпрограмма отображения полного изображения;
- `DrawEditLine();` – подпрограмма отображения редактируемого отрезка в режиме «исключающего или»;
- `bbBasePictureClick(TObject *Sender);` – обработчик события `OnClick` кнопки `bbBasePicture`, предназначенной для построения полного изображения;
- `bbClearClick(TObject *Sender);` – обработчик события `OnClick` кнопки `bbClear`, предназначенной для очистки изображения;
- `FormCreate(TObject *Sender);` – обработчик события «на создание формы»;
- `sbEditLineClick(TObject *Sender);` – обработчик события `OnClick` кнопки `SpeedButton` с именем `sbEditLine`, предназначенной для переключения приложения в режим редактирования отрезка и обратно;
- `ImageMouseDown(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y); ImageMouseMove(TObject *Sender, TShiftState Shift, int X, int Y); ImageMouseUp(TObject *Sender, TMouseButton Button, TShiftState Shift, int X, int Y);` – обработчики событий «мыши», предназначенные для управления изображением во время редактирования отрезка:
  - нажатие левой кнопки «мыши» над компонентом `Image`;
  - отпускание левой кнопки «мыши» над компонентом `Image`;
  - перемещение нажатой «мыши» над компонентом `Image`.

Кнопка `sbEditLine` играет роль и обычной кнопки, и переключателя. Свои действия она определяет по своему состоянию – нажата или нет. Напомним, чтобы

заставить эту кнопку «залипать», необходимо ее свойству AllowAllUp при значение true, а свойству GroupIndex значение, отличное от нуля.

Примечание

В силу простоты данного примера для хранения инфор о местоположения отрезка введем глобальные целоч ные переменные: FxLT_L, FyLT_L, FxRB_L, FyRB_L, ко определим в приватной области класса формы. Для оперативных координат ных точек отрезка введем переменные: gxLT_L, gyLT_L, gxRB_L, gyRB_L – опис в заголовочном файле. То есть эти координаты будут использоваться во вре дактирования. Там же описана переменная grectSize, хранящая размеры э тов, отображающих точки редактирования. Значение ей можно задать там же головочном файле.

Отображение полного изображения

Процедура DrawFullPict не имеет параметров. В процедуре в режиме ко вания отображаются круг, отрезок прямой линии и прямоугольник. Каждая из закрашивается своим цветом. В начале программы производится очистка об рисунка. Код процедуры представлен в листинге .6.

Листинг 6. Процедура построения полного изображения

```
void __fastcall TfoMain::DrawFullPict()
{
  bbClearClick (NULL); // очистить область рисунка
  Image->Canvas->Pen->Mode = pmCopy;  // режим копирования
     // установить координаты для эллипса
  int xLT_E = 10, yLT_E = 10, xRB_E = 210, yRB_E = 210;
  Image->Canvas->Pen->Color = clOlive; // цвет пера
  Image->Canvas->Pen->Width = 5; // ширина пера
  Image->Canvas->Brush->Color = clFuchsia; // цвет кисти
        // нарисовать круг
  Image->Canvas->Ellipse(xLT_E, yLT_E, xRB_E, yRB_E);
        // построить отрезок
  Image->Canvas->Pen->Color = clBlack; // цвет пера
  Image->Canvas->MoveTo (FxLT_L, FyLT_L); // начало
  Image->Canvas->LineTo (FxRB_L, FyRB_L); // отрезок
        // построить прямоугольник
  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
}
```

Отображения редактируемого отрезка в режиме «исключающего или»

Процедура DrawEditLine не имеет параметров. Редактируемый элемент – отрезок прямой – отображается в прямоугольную область (прямоугольник), который «охватывает» отрезок. Координаты этой области определяются на основании значения оперативных координат редактируемого отрезка и размеров дополнительных графических элементов, необходимых для редактирования.

Для оперативного управления положением редактируемого отрезка необходимо иметь четыре переменные, которые будут хранить оперативные координаты отрезка во время редактирования: в данном случае gxLT_L, gyLT_L, gxRB_L, gyRB_L. Значения этих переменных определяются из текущего положения конечных точек отрезка при старте режима редактирования, то есть при нажатии кнопки sbEditLine.

Как было отмечено выше, по контуру редактируемого изображения располагаются графические элементы – небольшие квадратики. Каждый из таких квадратиков имеет фиксированный размер стороны, значение которого хранится в переменной grectSize, определенной и инициализированной в заголовочном файле. Для выделения области редактирования необходимо выполнить щелчок «мышью» по этой области. Этот щелчок в случае рисования линии может оказаться где-то близко от нее, но при этом будет считаться, что линия «захвачена». Для учета этого факта необходимо несколько расширить редактируемую область, нежели просто ограничиться областью отрезка. Для этого следует воспользоваться значением переменной grectSize.

Алгоритм процедуры отображения прямоугольной области можно описать следующими действиями:

1) устанавливаются параметры пера;
2) рисуется отрезок прямой;
3) пересчитываются координаты прямоугольника отображения на основании оперативных координат и величины grectSize;
4) отображается прямоугольник с изображением.

Описание процедуры DrawEditLine приведено в листинге 7.

Листинг .7. Процедура отображения редактируемого отрезка изображения

```
void __fastcall TfoMain::DrawEditLine()
{
     // устанавливаются параметры пера
  Image->Canvas->Pen->Mode = pmXor;
  Image->Canvas->Pen->Color = clRed;
  Image->Canvas->Pen->Style = psDash;
  Image->Canvas->Pen->Width = 1;
     // рисуется отрезок прямой
  Image->Canvas->MoveTo (gxLT_L, gyLT_L);
  Image->Canvas->LineTo (gxRB_L, gyRB_L);
     // пересчитываются координаты
  int xLT_R = gxLT_L - grectSize, yLT_R = gyLT_L - grectSize,
     xRB_R = gxLT_L + grectSize, yRB_R = gyLT_L + grectSize;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);

     xLT_R = gxRB_L - grectSize, yLT_R = gyRB_L - grectSize,
     xRB_R = gxRB_L + grectSize, yRB_R = gyRB_L + grectSize;
     // построение прямоугольника
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
}
```

Обработчик события кнопки, предназначенной для построения полного изображения

В действия обработчика события OnClick кнопки bbBasePicture входит только одно – вызов процедуры полного построения изображения. Описание обработчика события OnClick кнопки bbBasePicture приведено в листинге 8.

Листинг 8. Текст обработчика события «Построить полное изображение»

```
void __fastcall TfoMain::bbBasePictureClick(TObject *Sender)
{
  DrawFullPict();
}
```

Обработчик события OnClick кнопки bbClear, предназначенной для очистки изображения

Алгоритм работы этого обработчика сводится к следующим действиям:

1) запоминается текущий режим пера;
2) устанавливается режим «Рисовать белым цветом»;
3) устанавливаются границы для прямоугольной области стирания;
4) отображается прямоугольник на область стирания;
5) восстанавливается сохраненный режим.

Описание обработчика события OnClick кнопки bbClear приведено в листинге 9.

Листинг 9. Процедура очистки области изображения

```
void __fastcall TfoMain::bbClearClick(TObject *Sender)
{
     // запомнить текущий режим
  TPenMode tmp = Image->Canvas->Pen->Mode;
     // режим «рисовать белым цветом»
  Image->Canvas->Pen->Mode = pmWhite;
     // установить границы
  int xLT_R = 0, yLT_R = 0, xRB_R = Image->Width, yRB_R = Image->Height;
     // отобразить прямоугольник
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
     // восстановить сохраненный режим
  Image->Canvas->Pen->Mode = tmp;
}
```

Переключение приложения в режим редактирования отрезка и обратно

Действия обработчика события OnClick кнопки sbEditLine, предназначенной для переключения режима редактирования, основаны на оценке того факта, произошло или нет событие «Включен режим редактирования».

1. Если режим включен, то выполняется следующее:
   а) формируются оперативные координаты редактируемого отрезка;
   б) вызывается процедура отображения области редактирования отрезка.
2. Если режим не включен, то выполняется следующее:
   а) формируются параметры области полного изображения;
   б) рисуется полное изображение.

Описание текста процедуры обработчика приведено в листинге 10.

Листинг 10. Процедура отображения редактируемого отрезка изображения

```
void __fastcall TfoMain::sbEditLineClick(TObject *Sender)
{
  if (sbEditLine->Down) // если кнопка нажата
    {
          // сформировать оперативные координаты
      gxLT_L = FxLT_L, gyLT_L = FyLT_L,
      gxRB_L = FxRB_L, gyRB_L = FyRB_L;
    DrawEditLine(); // отобразить область редактирования
    }
  else // если кнопка не нажата
    {
          // сформировать параметры области отображения
      FxLT_L = gxLT_L, FyLT_L = gyLT_L,
      FxRB_L = gxRB_L, FyRB_L = gyRB_L;
    DrawFullPict(); // полное изображение
    }
}
```

Обработчик события «Нажатие левой кнопки 'мыши' над компонентом Image»

Событие OnMouseDown

Обработчик события OnMouseDown содержит два параметра X и Y. Эти параметры определяют координаты указателя «мыши». Параметр Button передает код нажатой кнопки «мыши». Два оставшихся параметра этой процедуры в данном приложении не используются.

В задачу обработчика события входит следующая проверка: включен или не включен режим редактирования. Когда режим редактирования включен, то необходимо выполнить:

1. Если нажата левая кнопка «мыши» и ее указатель находится в области квадратика редактирования левой точки отрезка, то запомнить величину смещения точки нажатия «мыши», относительно левого конца отрезка.
2. Если нажата левая кнопка «мыши» и ее указатель находится в области квадратика редактирования правой точки отрезка, то запомнить величину смещения точки нажатия «мыши», относительно правого конца отрезка.

Чтобы запомнить, на сколько пикселей пользователь «промахнулся» мимо конца отрезка, используются переменные gshX, gshY. Описание обработчика события OnMouseDown приведено в листинге 11.

Листинг .11. Процедура отображения редактируемого отрезка изображения

```
void __fastcall TfoMain::ImageMouseDown (TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{
  if (sbEditLine->Down) // включен режим редактирования
   {
     if (Button == mbLeft) // нажата левая кнопка
        if (abs (X - gxLT_L) <= grectSize &&
                    abs (Y - gyLT_L) <= grectSize)
     {
          // тепкнули по начальной точке
       gNumPnt = 0;  // указываем – первая точка
       gshX = X - gxLT_L;  // запоминаем смещение щелчка
       gshY = Y - gyLT_L;
     }
     if (abs (X - gxRB_L) <= grectSize &&
                    abs (Y - gyRB_L) <= grectSize)
     {
          // щелкнули по конечной точке
       gNumPnt = 1;  // указываем – вторая точка
       gshX = X - gxRB_L;  // запоминаем смещение щелчка
       gshY = Y - gyRB_L;
     }
   }
}
```

Обработчик события «Отпускание левой кнопки 'мыши' над компонентом Image»

Событие OnMouseUp

Обработчик события OnMouseUp содержит также пять параметров. Здесь тоже используются X и Y, которые определяют координаты указателя «мыши». Другие параметры аналогичны обработчику OnMouseDown.

Переменные gxLT_L, gyLT_L, gxRB_L, gyRB_L хранят оперативные координаты отрезка, которые при отпускании кнопки «мыши» требуется зафиксировать. Следовательно, алгоритм процедуры сводится к запоминанию новых значений в областях глобальных переменных FxLT_L, FyLT_L, FxRB_L, FyRB_L, которые должны теперь совпадать с gxLT_L, gyLT_L, gxRB_L, gyRB_L. И изображения самого отрезка, и его редактируемого двойника будут совпадать только после отпускания кнопки «мыши». Поэтому заново отображаются полный рисунок и отрезок прямой линии. Описание обработчика события OnMouseUp приведено в листинге 12.

Листинг 12. Описание действий на событие «Отпускание левой кнопки 'мыши' над компонентом Image»

```
void __fastcall TfoMain::ImageMouseUp(TObject *Sender,
                TMouseButton Button, TShiftState Shift, int X, int Y)
{
  if (sbEditLine->Down) // включен режим редактирования
    if (Button == mbLeft) // нажата левая кнопка
      {
        gNumPnt = -1;  // отменяем указание точки
        FxLT_L = gxLT_L, FyLT_L = gyLT_L, FxRB_L = gxRB_L,
          FyRB_L = gyRB_L;
     DrawFullPict(); // полное изображение
     DrawEditLine(); // редактируемая линия
    }
}
```

Обработчик события «Передвижение нажатой 'мыши' над компонентом Image»

Событие OnMouseMove

Обработчик события OnMouseMove содержит четыре параметра, среди которых X и Y определяют координаты указателя «мыши». А код кнопки «мыши» приходится узнавать из параметра Shift.

В процедуре используются координаты указателя «мыши» X и Y, на основании значений которых пересчитываются оперативные координаты отрезка – первой и второй точек. Глобальные переменные gshX и gshY, описанные в заголовочном файле и проинициализированные в обработчике нажатия кнопки «мыши», помогают правильно корректировать координаты элементов редактирования, чтобы складывалось впечатление, что эти элементы перемещаются за ту же точку, как и при захвате. Описание обработчика события OnMouseMove приведено в листинге 13.

Листинг 13. Описание действий на событие «Перемещение 'мыши' над компонентом Image»

```
void __fastcall TfoMain::ImageMouseMove(TObject *Sender,
  TShiftState Shift, int X, int Y)
{ if (Shift.Contains(ssLeft))  // нажата левая кнопка
   if (sbEditLine->Down)
    {
     DrawEditLine();
     switch (gNumPnt) // номер точки
      {
       case 0 : gxLT_L = X + gshX; gyLT_L = Y + gshY; break;
       case 1 : gxRB_L = X + gshX; gyRB_L = Y + gshY;
      }
     DrawEditLine();
    }
}
```

Обработчик события «На создание формы»

Обработчик события создания формы инициализирует поля класса и обращ ется к процедуре рисования всего изображения. Текст обработчика приведен в ли тинге 14.

Листинг 14. Описание действий на событие «Создать форму»

```
void __fastcall TfoMain::FormCreate(TObject *Sender)
{
  FxLT_L = 30, FyLT_L = 80, FxRB_L = 300, FyRB_L = 80;
  DrawFullPict();
}
```

Задача 4. Имеется некая картинка, состоящая из изображения неизменной ча ти и ломаной линии. Требуется отредактировать изображение ломаной линии – и менить положение ее узлов.

Решение

Задача сходна с предыдущей задачей. Структуру проекта можно сохранить. Принципиальное отличие состоит в структуре данных: редактированию будут подвергаться не две точки, а набор точек с произвольным количеством. Поэтому отличие реализации данного проекта заключается в том, что координаты вершин ломаной линии *необходимо хранить в массиве*, а процедуры отображения полного изображения и отображения редактируемой ломаной линии должны *включать циклы построения* отображения набора отрезков или вызов соответствующих методов холста. На рис. 17.6 приведен вид интерфейса программы, включенной в режим редактирования ломаной линии.

Для хранения координат вершин ломаной линии вместо четырех переменных будем использовать массив FPnts, состоящий из элементов типа TPoint. Переменная FCount хранит количество элементов в этом массиве. А для временного хранения координат ломаной линии, точнее ее дубликата, в момент редактирования будет использоваться массив gPnts. Этих изменений в интерфейсной части достаточно для перехода от редактирования отрезка к редактированию ломаной линии.

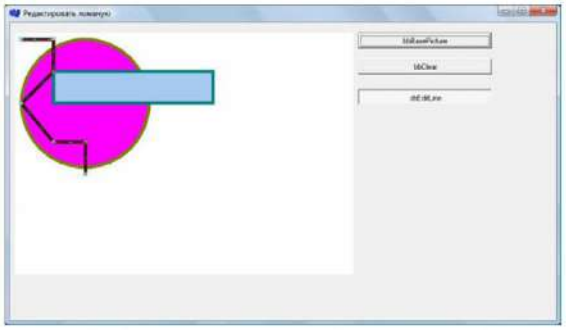


Рис. 17.6. Интерфейс программы в режиме редактирования ломаной линии

Изменения реализации методов проекта касается почти всех подпрограмм. Но эти изменения почти однотипные:

- присваивание значений отдельным переменным заменятся на циклы;
- отображения единственного отрезка – на циклы отображения набора отрезков, составляющих ломаную линию.

Тексты обновленных подпрограмм приведены в листингах 15– 21.

Листинг 15. Построение полного изображения

```
void __fastcall TfoMain::DrawFullPict()
{
  bbClearClick (NULL);
  Image->Canvas->Pen->Mode = pmCopy;  // режим копирования

  int xLT_E = 10, yLT_E = 10, xRB_E = 210, yRB_E = 210;
  Image->Canvas->Pen->Color = clOlive;
  Image->Canvas->Pen->Width = 5;
  Image->Canvas->Brush->Color = clFuchsia;
  Image->Canvas->Ellipse(xLT_E, yLT_E, xRB_E, yRB_E);

      // построение ломаной линии
  Image->Canvas->Pen->Color = clBlack;
  Image->Canvas->MoveTo (FPnts[0].x, FPnts[0].y);
  for (int i = 1; i < FCount; i++)
    Image->Canvas->LineTo (FPnts[i].x, FPnts[i].y);

  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
}
```

Описание обработчика события «На создание формы»

Листинг 16. Реализация процедуры инициализации ломаной линии

```
void __fastcall TfoMain::FormCreate(TObject *Sender)
{ FCount = cCount;
  FPnts[0].x = 10; FPnts[0].y = 10;
  FPnts[1].x = 60; FPnts[1].y = 10;
  FPnts[2].x = 60; FPnts[2].y = 60;
  FPnts[3].x = 10; FPnts[3].y = 110;
  FPnts[4].x = 60; FPnts[4].y = 170;
  FPnts[5].x = 110; FPnts[5].y = 170;
  FPnts[6].x = 110; FPnts[6].y = 220;
  DrawFullPict();
}
```

Отображения редактируемой ломаной линии в режиме «исключающего или»

Листинг 17. Реализация процедуры рисования редактируемой ломаной линии

```
void __fastcall TfoMain::DrawEditLine()
{ Image->Canvas->Pen->Mode = pmXor;
  Image->Canvas->Pen->Color = clRed;
  Image->Canvas->Pen->Style = psDash; // psInsideFrame;
  Image->Canvas->Pen->Width = 1;

      // построение ломаной линии
  Image->Canvas->MoveTo (gPnts[0].x, gPnts[0].y);
  for (int i = 1; i < FCount; i++)
    Image->Canvas->LineTo (gPnts[i].x, gPnts[i].y);

      // построение квадратиков на вершинах линии
  for (int i = 0; i < FCount; i++)
   {
    int xLT_R = gPnts[i].x - grectSize, yLT_R = gPnts[i].y - grectSize,
        xRB_R = gPnts[i].x + grectSize, yRB_R = gPnts[i].y + grectSize;
    Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
   }
}
```

Обработчик события кнопки, предназначенной для переключения приложения в режим редактирования отрезка и обратно

Листинг 18. Реализация процедуры переключения режима редактирования

```
void __fastcall TfoMain::sbEditLineClick(TObject *Sender)
{ if (sbEditLine->Down)
   {for (int i = 0; i < FCount; i++)
     {gPnts[i].x = FPnts[i].x; gPnts[i].y = FPnts[i].y;}
    DrawEditLine();
   }
  else
   {for (int i = 0; i < FCount; i++)
     {FPnts[i].x = gPnts[i].x; FPnts[i].y = gPnts[i].y;}
    DrawFullPict();
   }
}
```

Обработчик события «Нажатие левой кнопки 'мыши' над компонентом

Листинг 17.19. Реализация процедур проекта редактирования ломаной линии

```
void __fastcall TfoMain::ImageMouseDown (TObject *Sender,
      TMouseButton Button, TShiftState Shift, int X, int Y)
{ if (sbEditLine->Down) // включен режим редактирования
   {if (Button == mbLeft) // нажата левая кнопка
     for (int i = 0; i < FCount; i++)
       if (abs (X - gPnts[i].x) <= grectSize &&
                                   abs (Y - gPnts[i].y) <= grect
        { // щелкнуть по точке редактирования
         gNumPnt = i;  // указать номер точки
         gshX = X - gPnts[i].x; // запомнить смещение щелчка
         gshY = Y - gPnts[i].y;
        }
   }
}
```

Обработчик события «Перемещение 'мыши' с нажатой кнопкой над компонентом Image»

Листинг 17.20. Реализация процедуры перемещения «мыши»

```
void __fastcall TfoMain::ImageMouseMove(TObject *Sender,
                                   TShiftState Shift, int X, int
{ if (Shift.Contains(ssLeft))  // нажата левая кнопка
   if (sbEditLine->Down)
    {
     DrawEditLine();
     gPnts[gNumPnt].x = X + gshX; gPnts[gNumPnt].y = Y + gshY;
     DrawEditLine();
    }
}
```

Обработчик события «Отпускание левой кнопки 'мыши' над компонентом Image»

Листинг .21. Реализация процедур проекта редактирования ломаной линии

```
void __fastcall TfoMain::ImageMouseUp(TObject *Sender,
            TMouseButton Button, TShiftState Shift, int X, int Y)
{ if (sbEditLine->Down) // включен режим редактирования
   if (Button == mbLeft) // нажата левая кнопка
    {gNumPnt = -1;  // отменяем указание точки
     for (int i = 0; i < FCount; i++)
      {FPnts[i].x = gPnts[i].x, FPnts[i].y = gPnts[i].y;}
     DrawFullPict();
     DrawEditLine();
    }
}
```

***Задача* .5.** Имеется некий рисунок, состоящий из изображения неизменной части и изображения кривой Безье. Требуется отредактировать изображение кривой Безье – изменить положение ее характеристических узлов, то есть вид и положение кривой.

Решение

В ходе развития математических аспектов и алгоритмов компьютерной графики оказалось, что полиномы третьей степени и сплайны, построенные из таких полиномов, являются очень гибкими и простыми инструментами при построении произвольных кривых.

СПЛАЙН

Сплайн – это функция, определенная на заданном отрезке, совпадающая на частичных отрезках с некоторыми алгебраическими многочленами степени, не выше заданной (например, m) и имеющая непрерывную $(m-1)$ производную.

Одной из форм описания сплайнов третьей степени является описание их в форме кривых Безье. Не вдаваясь в математические аспекты построения кривой Безье, укажем только, что форма дуги каждого звена, то есть кубического полинома определяется вершинами характеристической ломаной линии, состоящей из трех звеньев.

ХАРАКТЕРИСТИЧЕСКАЯ ЛОМАНАЯ

Характеристическая ломаная – это кусочно линейная линия, которая формирует некий каркас для кривой Безье. Каждые четыре вершины ломаной определяют параметры одного звена кривой Безье.

Пример кривой Безье и ее характеристической ломаной приведен на рис. .7. Первые четыре вершины ломаной определяют форму первого кубического звена. Следующая четверка вершин – форму второго кубического звена кривой Безье. При этом надо помнить, что *последняя* вершина первой четверки является *первой* вершиной второй четверки вершин. То есть в данном примере ломаная, имеющая *семь* вершин, управляет формой кривой Безье, составленной из *двух* кубических звеньев.

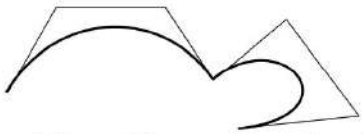


Рис. 7. Пример кривой Безье и ее характеристической ломаной

В общем случае для управления формой кривой Безье, составленной из N кубических звеньев, необходимо иметь ломаную линию с $(3N+1)$ вершинами. И этот факт необходимо точно соблюдать при построении кривой Безье с использованием методов холста.

Для построения кривой Безье в системе *C++ Builder* достаточно вызвать соответствующий метод *Canvas* – `PolyBezier`.

Функция PolyBezier

Метод *Canvas* – `PolyBezier` имеет прототип:

```
void __fastcall PolyBezier (const TPoint* Points,
                            const int Points_Size);
```

где первый параметр – указатель на массив из элементов типа `TPoint`, а второй – индекс последнего элемента. Обратите внимание, что второй параметр задает *не количество* элементов в массиве, а *индекс* последнего элемента.

Кроме этого метода, класс *Canvas* имеет метод `PolyBezierTo`. Единственное отличие этого метода состоит в том, что графический курсор системы после вывода кривой устанавливается на конечную точку, в то время как в методе `PolyBezier` курсор не меняет положения.

Во время редактирования кривой приложение должно отображать характеристическую ломаную линию и квадратики, подчеркивающие вершины редактирования. Это позволит пользователю управлять формой кривой с высокой точностью. При этом саму ломаную можно выводить не посредством оператора цикла, а с помощью метода `Polyline` холста. Этот метод имеет такие же параметры, как и метод `PolyBezier`, но строит ломаную линию с вершинами, координаты которых хранятся в массиве `Points`. Поэтому, несмотря на принципиально другой объект, подвергаемый редактированию, данная задача необычайно сходна с предыдущей. Структура проекта практически совпадает с предыдущим проектом. Изменению подвергаются лишь реализация методов `DrawFullPict` и `DrawEditLine`, да и то – только в части отображения ломаной. Точнее, в подпрограмме `DrawFullPict` вместо ломаной линии строится кривая Безье, а в подпрограмме `DrawEditLine` строится и кривая Безье, и ломаная линия с квадратиками в вершинах.

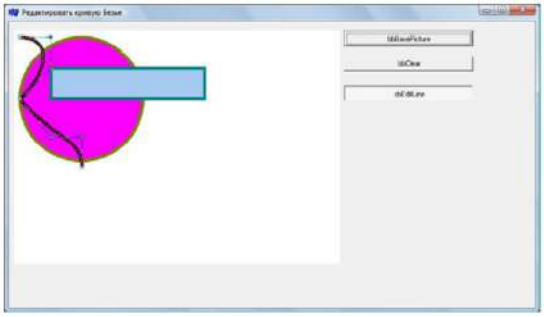


Рис. 17.8. Интерфейс программы в режиме редактирования кривой Безье

Вообще, решение данной задачи заключается лишь в корректировке текста процедур `DrawFullPict` и `DrawEditLine` с целью вывода кривой Безье и ее характеристической линии. Текст этих измененных процедур приведен в листингах .22 – .23, а пример интерфейса программы в режиме редактирования кривой – на р .8.

Отображение полного изображения рисунка

Листинг 17.22. Описание процедуры изображения рисунка с кривой Безье

```
void __fastcall TfoMain::DrawFullPict()
{ bbClearClick (NULL);
  Image->Canvas->Pen->Mode = pmCopy;  // режим копирования

  int xLT_E = 10, yLT_E = 10, xRB_E = 210, yRB_E = 210;
  Image->Canvas->Pen->Color = clOlive;
  Image->Canvas->Pen->Width = 5;
  Image->Canvas->Brush->Color = clFuchsia;
  Image->Canvas->Ellipse(xLT_E, yLT_E, xRB_E, yRB_E);

  Image->Canvas->Pen->Color = clBlack;

  Image->Canvas->PolyBezier(&FPnts[0], FCount-1);
                // вычерчивание кривой Безье

  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
}
```

Отображения редактируемой кривой Безье в режиме «исключающего или»

Листинг .23. Реализация процедур проекта редактирования кривой Безье

```
void __fastcall TfoMain::DrawEditLine()
{ Image->Canvas->Pen->Mode = pmXor;
  Image->Canvas->Pen->Color = clRed;
  Image->Canvas->Pen->Width = 1;

  Image->Canvas->Pen->Style = psDot; // psInsideFrame;
  Image->Canvas->PolyBezier(&gPnts[0], FCount-1);
              // вычерчивание кривой Безье

  Image->Canvas->Pen->Style = psDash;
  Image->Canvas->MoveTo (gPnts[0].x, gPnts[0].y);
  for (int i = 1; i < FCount; i++)
    Image->Canvas->LineTo (gPnts[i].x, gPnts[i].y);

  Image->Canvas->Pen->Style = psSolid;
  for (int i = 0; i < FCount; i++)
   {
    int xLT_R = gPnts[i].x - grectSize, yLT_R = gPnts[i].y - grectSize,
        xRB_R = gPnts[i].x + grectSize, yRB_R = gPnts[i].y + grectSize;
    Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
   }
}
```

***Задача* .6.** Заданное изображение требуется дополнить новыми ломаными. То есть обеспечить возможность построения новой ломаной линии на основании узлов, отмеченных «мышью». Предусмотреть возможность создания замкнутой/не замкнутой линии.

Решение

Предлагается следующая стратегия решения.

1. Определяется последовательность действий, которые необходимо выполнить пользователю для построения изображения ломаной линии.
2. Определяется интерфейс приложения, предназначенного для решения задачи.
3. Подготавливаются специальные процедуры, чтобы обеспечить функциональность приложения в соответствии с предложенным порядком действий.

Подготовка решения в соответствии с описанной стратегией.

1. Для задания ломаной линии необходимо обозначить узлы, ее образующие, а затем соединить эти узлы отрезками прямых линий. Определим правила построения:
   - пользователь находит на корректируемом изображении точку – *начало ломаной* – и фиксирует ее положение нажатием левой кнопки «мыши»;
   - затем находит положение следующего узла и также фиксирует его нажатием левой кнопки «мыши», и так далее до тех пор, пока не будет поставлена последняя точка в изображении;
   - нарисованное «закрепляется», например, с помощью правой кнопки «мыши».
2. В соответствии с поставленной задачей пользователю должна быть предоставлена возможность:
   - загрузить базовое изображение;
   - сообщить программе о своей готовности приступить к рисованию ломаной;
   - нарисовать эту ломаную;
   - очистить или убрать нарисованное изображение при необходимости.

Интерфейс приложения может быть таким, как на рис. .9.

3. Для того чтобы обеспечить требуемую функциональность приложения, целес образно подготовить следующие процедуры и обработчики событий:
   - `bbBasePictureClick (TObject *Sender);` – обработчик события `OnCli` кнопки `bbBasePicture`, предназначенной для построения базового изображ ния;
   - `bbClearClick (TObject *Sender);` – обработчик события `OnClick` кноп `bbClear`, предназначенной для очистки изображения;
   - `DrawFullPict();` – подпрограмма построения базового рисунка;
   - `DrawEditLine();` – подпрограмма отображения редактируемого отрезка пр мой линии в режиме «исключающего или»;
   - `ImageMouseDown (TObject *Sender, TMouseButton Button, TShiftSta Shift, int X, int Y);` – обработчик события «нажать кнопку 'мыши'» области изображения;
   - `ImageMouseMove(TObject *Sender, TShiftState Shift, int X, int Y);` обработчик события «передвинуть 'мышь'» в области изображения;
   - `sbCreateLineClick(TObject *Sender);` – обработчик события `OnCli` кнопки `sbCreateLine` – «создать линию» в области изображения;
   - `FormCreate (TObject *Sender);` – обработчик события `OnCreate` «на созд ние формы»;
   - `FormClose (TObject *Sender, TCloseAction &Action);` – обработчик с бытия `OnClose` «на закрытие формы»;
   - `sbEditLineClick(TObject *Sender);` – обработчик события `OnClick` кнопки `sbEditLine`, предназначенной для старта режима редактирования ломаной линии.

Примечание Для реализации описанных процедур и обработчиков событий в программу включены следующие *глобальные* константы и переменные:

```
const cCount = 100; // максимальная длина ломаной – число узлов
const cSizeCurve = 100; // максимальное количество создаваемых
                        // линий за сеанс
TPoint *FPnts [cSizeCurve];  // массив указателей на кривые
int    FCounts [cSizeCurve];  // массив длин каждой из кривых
int    FCount;  // текущее количество кривых
bool   FClosed [cSizeCurve];  // признак замкнутости кривой
```

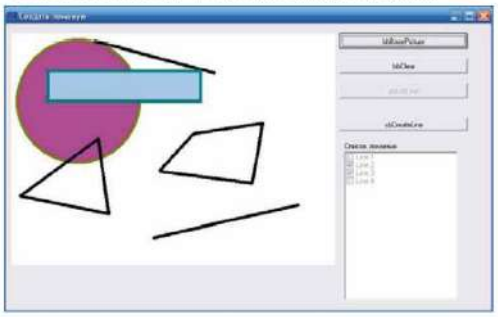


Рис. .9. Интерфейс приложения, предназначенного для построения ломаных линий в соответствии с условием задачи 5

Рис. 10. Диалоговое окно для подтверждения создания замкнутой кривой

Обработчик события `OnClick` кнопки «Создать базовый рисунок»

Алгоритм обработки этого события сводится к одному действию: вызову метода построения базового изображения. Код обработчика представлен в листинге 24.

Листинг 24. Действия под кнопку «Базовый рисунок»

```
void __fastcall TfoMain::bbBasePictureClick (TObject *Sender)
{
  DrawFullPict();
}
```

Обработчик события `OnClick` кнопки, предназначенной для очистки изображения

Алгоритм обработки этого события сводится к выполнению серии действий для очистки области изображения. Код обработчика представлен в листинге . .25.

Листинг .25. Реализация процедуры очистки полотна

```
void __fastcall TfoMain::bbClearClick (TObject *Sender)
{
  TPenMode tmp = Image->Canvas->Pen->Mode;
  Image->Canvas->Pen->Mode = pmWhite;

  int xLT_R = 0, yLT_R = 0,
      xRB_R = Image->Width, yRB_R = Image->Height;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
  Image->Canvas->Pen->Mode = tmp;
}
```

Построение базового рисунка

Листинг .26. Описание процедуры построения базового изображения

```
void __fastcall TfoMain::DrawFullPict()
{
  bbClearClick (NULL);
  Image->Canvas->Pen->Mode = pmCopy;  // режим рисования
  int xLT_E = 10, yLT_E = 10, xRB_E = 210, yRB_E = 210;
  Image->Canvas->Pen->Color = clOlive;
  Image->Canvas->Pen->Width = 5;
  Image->Canvas->Brush->Color = clFuchsia;
  Image->Canvas->Ellipse(xLT_E, yLT_E, xRB_E, yRB_E);
  Image->Canvas->Pen->Color = clBlack;
  for (int ind = 0; ind < FCount; ind++)
   {
    Image->Canvas->MoveTo (FPnts[ind][0].x, FPnts[ind][0].y);
    for (int i = 1; i < FCounts [ind]; i++)
      Image->Canvas->LineTo (FPnts[ind][i].x, FPnts[ind][i].y);
    if (FClosed [ind]) Image->Canvas->
                  LineTo (FPnts[ind][0].x, FPnts[ind][0].y);
   }
  int xLT_R = 60, yLT_R = 60, xRB_R = 310, yRB_R = 110;
  Image->Canvas->Pen->Color = clTeal;
  Image->Canvas->Brush->Color = clSkyBlue;
  Image->Canvas->Rectangle(xLT_R, yLT_R, xRB_R, yRB_R);
}
```

Отображение редактируемого отрезка прямой линии в режиме «исключающего или»

Листинг .27. Реализация процедуры отображения линии редактирования

```
void __fastcall TfoMain::DrawEditLine()
{ Image->Canvas->Pen->Mode = pmXor;
  Image->Canvas->Pen->Color = clRed;
  Image->Canvas->Pen->Style = psDash;
  Image->Canvas->Pen->Width = 1;

 Image->Canvas->MoveTo (gPnts[0].x, gPnts[0].y);
 for (int i = 1; i < gCount; i++)
   Image->Canvas->LineTo (gPnts[i].x, gPnts[i].y);

 for (int i = 0; i < gCount; i++)
  {int xLT_R = gPnts[i].x - grectSize, yLT_R = gPnts[i].y - grectSize,
       xRB_R = gPnts[i].x + grectSize, yRB_R = gPnts[i].y + grectSize;
   Image->Canvas->Rectangle (xLT_R, yLT_R, xRB_R, yRB_R);
  }
}
```

Обработчик события «Нажать кнопку ‘мыши’» в области изображения

Листинг .28. Реализация процедуры нажатия кнопки «мыши»

```
void __fastcall TfoMain::ImageMouseDown (TObject *Sender,
          TMouseButton Button, TShiftState Shift, int X, int Y)
{ if (sbCreateLine->Down) // включен режим создания
    if (Button == mbLeft) // нажата левая кнопка
     {if (gFirstPnt)  // нажатие первое, только порождаем
       {gPnts[0].x = X; gPnts[0].y = Y;
        gPnts[1].x = X; gPnts[1].y = Y;
        gCount = 2;
        gFirstPnt = false;
       }
      else
       {DrawEditLine ();  // стираем прежнее изображение
        gPnts[gCount].x = X; gPnts[gCount].y = Y;
        gCount++;
       }
      DrawEditLine();
     }
   else  // нажато, но не левая
     if (Button == mbRight && !gFirstPnt) // нажата правая кнопка
      {bool isClosed = false;
       if (abs (X - gPnts [0].x) <= grectSize &&
           abs (Y - gPnts [0].y) <= grectSize) // щелчок близок к п
         isClosed = mrYes == MessageDlg
                              ("Замкнуть кривую?", mtConfirmation,
                               TMsgDlgButtons()<<mbYes<<mbNo, 0),
       if (isClosed) {gCount--; FClosed [FCount] = true;}
       FPnts [FCount] = new TPoint [gCount];
                                // заведение нового массива точе
       for (int ind = 0; ind < gCount; ind++)
        {FPnts[FCount][ind].x = gPnts[ind].x;
         FPnts[FCount][ind].y = gPnts[ind].y;
        }
       FCounts [FCount] = gCount;
                          // фиксируем количество точек в этой ломано
       FCount++;  // количество ломаных увеличилось
       DrawFullPict();
       sbCreateLine->Down = false;  // отключить режим создания
         // отобразить в списке
       clbLineList->Items->Add ("Line " + IntToStr (FCount));
       clbLineList->Checked [clbLineList->Count - 1] = isClosed;
       clbLineList->ItemEnabled [clbLineList->Count - 1] = false;
      }
}
```

Обработчик события «Передвинуть ‘мышь’» в области изображения

Листинг 29. Реализация процедуры перемещения «мыши»

```
void __fastcall TfoMain::ImageMouseMove(TObject *Sender,
                                  TShiftState Shift, int X, int Y)
{ if (sbCreateLine->Down)  // если включен режим редактирования
    if (!gFirstPnt)
     {DrawEditLine();
      gPnts[gCount - 1].x = X + gshX;
      gPnts[gCount - 1].y = Y + gshY;
      DrawEditLine();
     }
}
```

Обработчик события «Создать ломаную линию»

Листинг .30. Реализация процедуры включения режима создания ломаной

```
void __fastcall TfoMain::sbCreateLineClick(TObject *Sender)
{ if (sbCreateLine->Down)
   {gCount = 0;
    gFirstPnt = true;
   }
}
```

Обработчик события «На создание формы»

Листинг .31. Реализация процедуры старта программы

```
void __fastcall TfoMain::FormCreate (TObject *Sender)
{ FCount = 0;  // еще нет кривых
  for (int ind = 0; ind < cSizeCurve; ind++)
   {FPnts [ind] = NULL;
    FCounts [ind] = 0;
    FClosed [ind] = false;
   }
}
```

Обработчик события «На закрытие формы»

Листинг .32. Реализация процедуры завершения программы

```
void __fastcall TfoMain::FormClose (TObject *Sender,
                                    TCloseAction &Action)
{ for (int ind = 0; ind < FCount; ind++)
    delete[] FPnts[ind];
}
```

Следует обратить внимание на то, что обработчик события нажатия кнопки «мыши» анализирует кнопку «мыши», которая нажата. При нажатии левой кнопки «мыши» создается очередная вершина ломаной и фиксируется ее положение, а при нажатии правой кнопки фиксируется конечная вершина и пополняется список линий. При этом если линия является замкнутой, то строка с названием линии помечается галочкой перед именем. Так, после создания набора линий всегда можно посмотреть какая линия является замкнутой, а какая нет.

ВЫПОЛНИТЬ САМОСТОЯТЕЛЬНО

1. Дополнить реализацию проекта к задаче `.6 кнопками и обработчиками, для реализации возможности сохранения и чтения набора построенных ломаных.
2. Дополнить реализацию проекта возможностью редактирования созданных ломаных линий.
3. Дополнить реализацию проекта возможностью создания и редактирования кривых Безье.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 34 Игра «Сапер»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

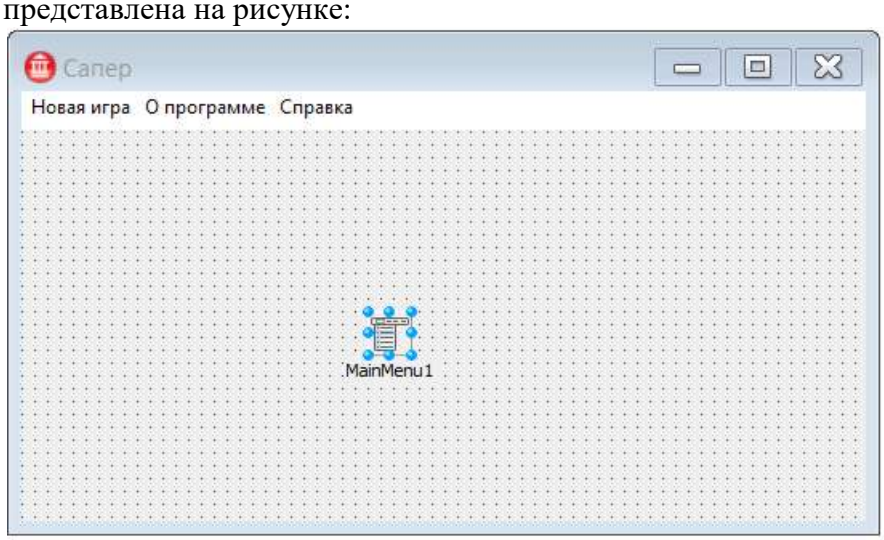
Цель: создать компьютерную версию игры «Сапер» в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Форма 1 представлена на рисунке:



Структура Unit1.h

```
#ifndef Unit1H
#define Unit1H
//---------------------------------------------------------------------------
#include <System.Classes.hpp>
#include <Vcl.Controls.hpp>
#include <Vcl.StdCtrls.hpp>
#include <Vcl.Forms.hpp>
#include <Vcl.Menus.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published: // IDE-managed Components
        TMainMenu *MainMenu1;
        TMenuItem *N1;
        TMenuItem *N2;
        TMenuItem *N3;
```

```
        void __fastcall FormMouseDown(TObject *Sender, TMouseButton Button,
TShiftState Shift,
        int X, int Y);
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall FormPaint(TObject *Sender);
        void __fastcall N1Click(TObject *Sender);
        void __fastcall N2Click(TObject *Sender);
        void __fastcall N3Click(TObject *Sender);
    private:    // User declarations
    public:             // User declarations
        __fastcall TForm1(TComponent* Owner);
        void __fastcall NewGame();
        void __fastcall Open(int row, int col);
        void __fastcall ShowPole(int status);
        void __fastcall Kletka(int row, int col, int status);
        void __fastcall Mina(int x, int y);
        void __fastcall Flag(int x, int y);
    };
```

Структура Unit1.cpp

```
    #include <stdlib.h>
    #include <time.h>  // для доступа к
    // генератору случайных чисел
    #include "Unit2.h"
    TForm1 *Form1;
    #define MR 10  // кол-во клеток по вертикали
    #define MC 10  // кол-во клеток по горизонтали
    #define NM 10  // кол-во мин
    int Pole [MR+2][MC+2];  // минное поле
    // 0..8 - количество мин в соседних клетках
    // 9 - в клетке мина
    // 100..109 - клетка открыта
    // 200..209 - в клетку поставлен флаг
    int nMin;  // кол-во найденных мин
    int nFlag; // кол-во поставленных флагов
    int status=0; // 0 - начало игры; 1 - игра; 2 - результат
    // смещение игрового поля относительно левого верхнего угла
    // поверхности формы
    #define LEFT 0 // по X
    #define TOP 1  // по Y
    #define W 40  // ширина клетки поля
    #define H 40    // высота клетки поля
    //---------------------------------------------------------------------------
    __fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
    {
    }
    // нажатие кнопки мыши на игровом поле
    //---------------------------------------------------------------------------
    void __fastcall TForm1::FormMouseDown(TObject *Sender, TMouseButton Button,
TShiftState Shift,
        int X, int Y)
    {
      if(status==2) return;
      if(status==0) status=1;
```

```
  X-=LEFT;
  Y-=TOP;
  if(X>0 && Y>0)
  {
  // преобразуем координаты мыши в индексы
// клетки поля
        int row=Y/H+1;
        int col=X/W+1;
        if (Button==mbLeft)
        {
              if(Pole[row][col]==9)
              {
                    Pole[row][col]+=100;
                    status=2;  // игра  закончена
                    ShowPole(status);
              }
              else if (Pole[row][col]<9)
                    {
                      Open(row,col);
                      ShowPole(status);
                    }
        }
        else if(Button==mbRight)
        {
               nFlag++;
               if(Pole[row][col]==9)
                     nMin++;
                     Pole[row][col]+=200;  // поставили флаг
               if(nMin==NM && nFlag==NM)
               {
                      status=2;   //игра закончена
                      ShowPole(status);
               }
               else Kletka(row, col,status);
     }
  }
}
//---------------------------------------------------------------------------
// Функция обработки- события OnCreate обычно используется
// для инициализации глобальных переменных
void __fastcall TForm1::FormCreate(TObject *Sender)
{
// В неотображаемые эл-ты массива, которые соответствуют
// клеткам по границе игрового поля, запишем число -3.
// Это значение используется функцией Open для завершения
// рекурсивного процесса открытия соседних пустых клеток.
 for(int row=0; row<=MR+1; row++)
        for(int col=0; col<=MC+1; col++)
        Pole[row][col]=-3;
        NewGame(); // "разбросать" мины
        ClientWidth=W*MC;
        ClientHeight=H*MR+TOP+1;
}
//---------------------------------------------------------------------------
```

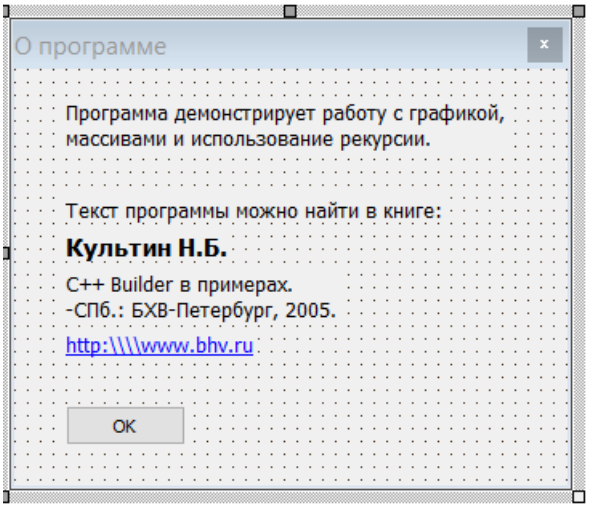
```
// Вывод поля как результат обработки события Paint
// позволяет проводить любые манипуляции с формой во время
// работы программы
void __fastcall TForm1::FormPaint(TObject *Sender)
{
 ShowPole(status);
}
//---------------------------------------------------------------------------
 // Показывает поле
void __fastcall TForm1::ShowPole(int status)
{
        for(int row=0; row<=MR; row++)
         for(int col=0; col<=MC; col++)
         Kletka(row, col,status);
}
//---------------------------------------------------------------------------
// рисует на экране клетку
void __fastcall TForm1::Kletka(int row, int col, int status)
{
 int x=LEFT+(col-1)*W;
 int y=TOP+(row-1)*H;
 if(status==0) // начало игры
 {
 // клетка - серый квадрат
         Canvas->Brush->Color=clBtnFace;
         Canvas->Rectangle(x-1, y-1,x+W, y+H);
         return;
 }
 // во время (status = 1) и в конце (status = 2) игры
 if(Pole[row][col]<100)
 {
 // клетка не открыта
         Canvas->Brush->Color=clBtnFace; // не открытые
                                                                  // серые
         Canvas->Rectangle(x-1, y-1,x+W, y+H);
         if(status==2 && Pole[row][col]==9)
                 Mina(x,y); // игра закончена, показать мину
                 return;
 }
 // клетка открыта
 Canvas->Brush->Color=clWhite;   // открытые — белые
 Canvas->Rectangle(x-1, y-1,x+W, y+H);
 if(Pole[row][col]==100)   // клетка открыта, но она
                                                         // пустая
                 return;
 if(Pole[row][col]>=101 && Pole[row][col]<=108)
 {
         Canvas->Font->Size=11;
         Canvas->Font->Color=clBlue;
         Canvas->TextOutW(x+3, y+3, IntToStr(Pole[row][col]-100));
                 return;
 }
  if(Pole[row][col]>=200)
         Flag(x,y);
```

```
 if(Pole[row][col]==109) // на этой мине подорвались!
 {
        Canvas->Brush->Color=clRed;
        Canvas->Rectangle(x-1, y-1,x+W, y+H);
 }
 if((Pole[row][col]%10==9) && (status==2))
        Mina(x,y);
}
//---------------------------------------------------------------------------
// рекурсивная функция открывает текущую и все соседние
// клетки, в которых нет мин
void __fastcall TForm1::Open(int row, int col)
{
 if(Pole[row][col]==0)
 {
        Pole[row][col]=100;
        // открываем клетки слева/ справа, снизу и сверху
        Open(row,col-1);
        Open(row-1,col);
        Open(row,col+1);
        Open(row+1,col);
        // открываем примыкающее диагонально
        Open(row-1,col-1);
        Open(row-1,col+1);
        Open(row+1,col-1);
        Open(row+1,col+1);
 }
 else
       if(Pole[row][col]<100 && Pole[row][col]!=-3)
              Pole[row][col]+=100;
}
//---------------------------------------------------------------------------
// новая игра - генерирует новое поле
void __fastcall TForm1::NewGame()
{
// Очистим эл-ты массива, соответствующие отображаемым
// клеткам, а в неотображаемые (по границе игрового поля)
// запишем число -3. Уникальное значение клеток границы
// используется функцией Open для завершения рекурсивного
// процесса открытия соседних пустых клеток.
       int row, col;
       for(row=0; row<=MR+1; row++)
        for(col=0; col<=MC+1; col++)
              Pole[row][col]=-3;
        for(row=1; row<=MR; row++)
         for(col=1; col<=MC; col++)
              Pole[row][col]=0;
       // расставим мины
       time_t t; // используется ГСЧ
       srand((unsigned)time(&t));  // инициализация ГСЧ
       int n=0;
       do
       {
              row=rand()%MR+1;
```

```
                col=rand()%MC+1;
                if(Pole[row][col]!=9)
                {
                        Pole[row][col]=9;
                        n++;
                }
        }
        while(n<10);
        // вычисление кол-ва мин в соседних клетках
        int k;
        for(row=1; row<=MR; row++)
          for(col=1; col<=MC; col++)
          if(Pole[row][col]!=9)
          {
                  k=0;
                  if( Pole[row-1][col-1]==9)k++;
                  if( Pole[row-1][col]==9)k++;
                  if( Pole[row-1][col+1]==9)k++;
                  if( Pole[row][col-1]==9)k++;
                  if( Pole[row][col+1]==9)k++;
                  if( Pole[row+1][col-1]==9)k++;
                  if( Pole[row+1][col]==9)k++;
                  if( Pole[row+1][col+1]==9)k++;
                  Pole[row][col]=k;
          }
          status=0;
          nMin=0;
          nFlag=0;
}
//---------------------------------------------------------------------------
// рисует мину
void __fastcall TForm1::Mina(int x, int y)
{
  Canvas->Brush->Color=clGreen;
  Canvas->Brush->Color=clBlack;
  Canvas->Rectangle(x+16, y+26,x+24, y+30);
  // корпус
  Canvas->Rectangle(x+8, y+30,x+32, y+34);
  Canvas->Pie(x+6, y+28,x+34, y+44,x+34, y+36,x+6, y+36);
  // полоса на корпусе
  Canvas->MoveTo(x+8, y+32);
  Canvas->LineTo(x+28, y+32);
  // основание
  Canvas->MoveTo(x+8, y+36);
  Canvas->LineTo(x+32, y+36);
  // вертикальный "ус
  Canvas->MoveTo(x+20, y+22);
  Canvas->LineTo(x+20, y+26);
       // боковые "усы"
  Canvas->MoveTo(x+8, y+30);
  Canvas->LineTo(x+6, y+28);
  Canvas->MoveTo(x+32, y+30);
  Canvas->LineTo(x+34, y+28);
}
```

```
//---------------------------------------------------------------------------
// Рисует флаг
void __fastcall TForm1::Flag(int x, int y)
{
        TPoint p[4]; ; // координаты флажка и нижней точки древка
// точки флажка
        p[0].X=x+4;     p[0].X=x+4;
        p[1].X=x+30;     p[1].X=x+12;
        p[2].X=x+4;     p[2].X=x+20;
        // установим цвет кисти и карандаша
        Canvas->Brush->Color=clRed;
        Canvas->Pen->Color=clRed;
        Canvas->Polygon(p,2); //флажек
        Canvas->Pen->Color=clBlack;
        Canvas->MoveTo(p[0].x, p[0].y);
        Canvas->LineTo(x+4, y+36);
        TPoint m[5];  // буква M
        m[0].X=x+8;    m[0].X=x+14;
        m[1].X=x+8;    m[1].X=x+8;
        m[2].X=x+10;   m[2].X=x+10;
        m[3].X=x+12;    m[3].X=x+8;
        m[4].X=x+12;    m[4].X=x+14;
        Canvas->Pen->Color=clWhite;
        Canvas->Polygon(m,4);
        Canvas->Pen->Color=clBlack;
}
// команда главного меню Новая игра
void __fastcall TForm1::N1Click(TObject *Sender)
{
  NewGame();
  ShowPole(status);
}
//---------------------------------------------------------------------------
// выбор в меню "?" команды Справка
void __fastcall TForm1::N2Click(TObject *Sender)
{
/* Отображение справочной информации
обеспечивает утилита hh.exe,  входящая
в состав Windows. Ключ mappid задает
отображаемый раздел справочной информации.  */
  WinExec("hh.exe -mapid 1 saper.chm", SW_RESTORE);
}
//---------------------------------------------------------------------------
 // выбор в меню "?" команды О программе
void __fastcall TForm1::N3Click(TObject *Sender)
{
  Form2->ShowModal();
}
//---------------------------------------------------------------------------
```

Форма 2 представлена на рисунке:



Структура Unit2.cpp

```
#include <vcl.h>
#pragma hdrstop
#include "Unit2.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Label5Click(TObject *Sender)
{
  ShellExecuteA(Form2->Handle, "open", "http:\\\\www.bhv.ru", NULL, NULL, SW_RESTORE);
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
  ModalResult=mrOk;
}
//---------------------------------------------------------------------------
```

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 35 Связь предложения MS Excel и Word по технологии OLE

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: научиться использовать технологию OLE в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

.1. Связь с MS Excel по технологии OLE

Использование технологий объектно-ориентированного программирования при создании приложений изменило восприятие самого приложения разработчиками. Программы стали восприниматься как изделия, состоящие из блоков, работающих согласованно для достижения общего результата. Это резко упростило разработку сложных приложений. При этом описание коллекции однотипных объектов в программе, то есть описание класса, состоит из двух частей – интерфейсная часть и описание реализаций методов. Интерфейсная часть описывает правила использования методов: как называется функция, входные и выходные параметры.

Технология OLE позволяет надстроить интерфейс и к приложению. В этом случае само приложение превращается в объект, доступ к полям и методам которого определяется его интерфейсом.

В рамках этого учебника не будет рассматриваться порядок и технология разработки интерфейса серверов. А вот примеры обращения к приложениям *Excel* и *Word* и использования их ресурсов будут приведены.

Структура приложения *Excel* с точки зрения технологии OLE

Программа *Excel*, с точки зрения OLE, представляет собой сложный составной объект. По-другому его же можно воспринимать как набор взаимосвязанных объектов, некоторые из которых являются элементами других. Для лучшего понимания взаимодействия с *Excel* средствами OLE необходимо вспомнить основные моменты работы в этом приложении. Рассмотрим ключевые объекты *Excel* и основные операции с ними.

Основные объекты программы *Excel* следующие:

- в первую очередь это само *приложение*. Программа *Excel* уже может быть запущена на момент его использования нашим приложением, а возможно, и нет. Следовательно, приложение должно проверять текущее состояние *Excel* и устанавливать с ним связь. Но простое приложение может и просто запустить новый экземпляр *Excel*, не проверяя состояния. При необходимости приложение может дать команду на завершение работы *Excel*. Как правило, это происходит на этапе завершения программы;
- запущенная программа *Excel* может содержать одну или несколько открытых *книг*, то есть файлов;
- каждая книга состоит из нескольких *страниц*;
- страница книги состоит из *ячеек*, которым можно присваивать какие-либо данные или даже формулы и считывать значения из этих ячеек;
- другие дополнительные элементы, например, *диаграммы*.

Основные операции над этими объектами:

- запускать и останавливать работу программы *Excel*;
- открывать и закрывать книги, сохранять их содержимое на диске;
- добавлять в книге новые страницы, удалять их или переименовывать;
- заполнять ячейки значениями, считывать значения из них;
- задавать в ячейки формулы и считывать результаты вычислений;
- управлять форматами и другими параметрами ячеек;
- управлять построением диаграмм и регулировать их параметры.

Попросту говоря, средствами автоматизации OLE можно выполнять в программе *Excel* те же действия, что и с помощью клавиатуры и «мыши».

Помимо объектов, которые представлены видимыми элементами в *Excel*, при работе приложения средствами автоматизации OLE приходится создавать временные указатели на различные дополнительные элементы, как то активная страница или выделенный регион ячеек.

Отметим Необходимо понимать, что непосредственная работа с *Excel* посредством клавиатуры и «мыши», несколько отличается от работы с ней средствами технологии OLE-интерфейса. В последнем случае приходится иметь дело с большим количеством объектов. Но при проектировании и реализации подпрограмм работы с *Excel*, конечно, следует руководствоваться навыками работы в программе *Excel*, ведь общая логика выполнения действий такая же.

Управление серверами OLE из программы *C++ Builder*

Помимо общеизвестных базовых и производных типов переменных, в языке *C++* определен и тип *Variant*.

Variant Переменная типа `Variant` может хранить любой другой тип данных. Это зависит от фактически сохраняемых данных. Удобством этого типа является его универсальность. Ему можно присваивать любые значения.

Но в данной главе интересно то, что переменной такого типа можно также присваивать и значение указателя на OLE-объект, а затем с этим OLE-объектом работать как с объектом класса: переменная типа *Variant* в таком случае будет представителем класса, который состоит из множества *методов* и *свойств*, подобно классу.

Напомним, что в ООП методы класса являются подпрограммами. При этом различают подпрограммы двух типов:

- процедура – подпрограмма, которая выполняет некие действия;
- функция – подпрограмма, которая выполняет действия, а результат возвращает через имя функции.

Также в состав интерфейса классов в ООП входят и СВОЙСТВА. То есть, это как бы поля, но обладающие, куда большими возможностями. Обращение к таким полям выполняется посредством процедур, и при этом со свойствами выполняют два действия:

- *присвоить* значение свойству;
- *считать* значение из свойства.

В общем случае с некоторыми свойствами можно выполнять обе операции, а с некоторыми – только одну.

При создании приложения автоматизации OLE нас интересуют только четыре метода, которые, собственно, и соответствуют четырем указанным операциям:

- `OleFunction` – вызов функции объекта OLE;
- `OleProcedure` – вызов процедуры объекта OLE;
- `OlePropertyGet` – чтение значения свойства объекта OLE;
- `OlePropertySet` – запись значения в свойство объекта OLE.

Все разнообразие наименований методов и свойств различных OLE-объектов указывается в параметрах этих четырех методов переменной типа `Variant`.

Следует отметить также, что методы переменной типа `Variant`, которые потребуются в этой главе для работы с OLE-объектами – это `CreateObject`, `GetActiveObject` и `IsEmpty`:

- метод `CreateObject` предназначен для старта OLE-приложения и установления связи с переменной типа `Variant`;
- метод `CreateObject` – для установления связи переменной типа `Variant` с уже запущенным приложением;
- метод `IsEmpty` можно использовать для проверки: присвоено ли этой переменной значение или еще нет. Такую проверку можно выполнять и функцией `VarIsEmpty`.

В объектно-ориентированной программе объект, то есть экземпляр класса, порождается динамически. Связь с ним программа осуществляет через переменную типа указатель, специально для этого выделенную. В программе же автоматизации OLE после старта объекта автоматизации, например *Excel*, связь с ним осуществляет

через переменную типа *Variant*. Общий порядок работы с объектами OLE, как бы экземплярами класса, такой же, как и с обычными объектами:

- создать объект автоматизации – соединить переменную типа с новым экземпляром приложения *Excel* или ранее запущенным;
- выполнить работу – активизировать методы приложения, используя методы переменной типа *Variant*;
- уничтожить объект – отсоединить переменную типа *Variant* и, возможно, выполнить команду завершения работы приложения *Excel*.

Порядок работы программы с сервером автоматизации OLE во многом схож с порядком работы с классическими объектами (экземплярами класса) в объектно-ориентированной программе. Отличие лишь в том, что этим объектом взаимодействия является целое приложение и иногда с невероятно большими возможностями, как, например, *Excel*.

Примечание

Еще одно отличие, которое следует отметить – это то, что приложение *Excel* во время работы приложения автоматизации может быть невидимым или видимым. Во втором случае пользователь может наблюдать, как выполняются действия в *Excel*. Но есть опасность, что пользователь может вмешаться и нарушить работу программы, даже по неосторожности.

***Задача* .1.** Разработать приложение, предназначенное для открытия книг программой *Excel* и чтения данных со страниц этих книг.

Решение

Предлагается следующая стратегия решения задачи.

1. Определяется, какую информацию следует предоставить пользователю.
2. Определяется последовательность действий, которые должен выполнить пользователь для того, чтобы обратиться к *Excel*, выбрать необходимую книгу *Excel* и просмотреть в ней нужные страницы и данные.
3. Разрабатывается интерфейс приложения.
4. Подготавливаются специальные процедуры для обеспечения действий пользователя.

Подготовка решения в соответствии с описанной стратегией.

1) Пользователю нужно предоставить информацию следующего вида:
   а) перечень книг *Excel*, с возможностью выбора книги;
   б) наименование выбранной книги;
   в) перечень страниц выбранной книги *Excel* с возможностью выбрать страницу;
   г) окно, содержащее прочитанные данные с выбранной страницы.
2) Предлагается обеспечить следующую последовательность действий с приложением:
   а) запустить *Excel* с помощью кнопки;
   б) открыть книгу *Excel* – пользователь нажимает кнопку, после чего производит выбор в стандартном диалоге. На экране отображается наименование выбранной книги и список наименований страниц;
   в) выбрать страницу из списка и выполнить команду «Прочитать страницу». На экране отражается содержимое страницы.
3) Для решения задачи может быть предложен интерфейс программы, как на рис. .1. Интерфейс данного приложения будет содержать следующие элементы, представленные в таблице .1.

Таблица .1

Основные компоненты приложения чтения документов Excel

№	Тип	Наименование	Назначение
1	TListBox	lbExcelBooks	Список открытых книг
2	TComboBox	cbExcelSheets	Список страниц текущей книги
3	TListBox	lbContent	Отображение содержимого текущей страницы
4	TBitBtn	bbInitExcel	Инициализация *Excel*
5	TBitBtn	bbOpenBook	Открытие книги
6	TCheckBox	cbVisible	Управляет видимостью *Excel*
7	TActionList	ActionList	Набор подпрограмм работы
8	TOpenDialog	dlgOpenExcelBook	Диалог открытия книги

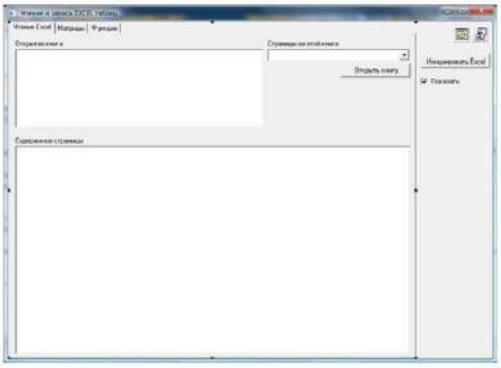


Рис. 1. Интерфейс программы чтения данных из документов Excel

4) Для того чтобы пользователь имел возможность выполнить предусмотренные действия и получить требуемую информацию, целесообразно разработать следующие подпрограммы (так как в этом приложении использован компонент `TActionList`, то основные методы обработки реализованы в нем):

- `actCreateExcelExecute(TObject *Sender);` – подпрограмма для запуска *Excel*;
- `cbVisibleClick(TObject *Sender);` – подпрограмма переключения видимости программы *Excel*;
- `actOpenWorksheetsExecute(TObject *Sender);` – подпрограмма для открытия книги;
- `lbExcelBooksClick(TObject *Sender);` – обработчик события выбора книги и страницы;
- `cbExcelSheetsChange(TObject *Sender);` – обработчик события «Выбор страницы *Excel* документа»;
- `FormCloseQuery(TObject *Sender, bool &CanClose);` – обработчик события «Запрос на закрытие главной формы программы».

Для связи программы на различных этапах с одним и тем же экземпляром программы *Excel* и некоторой выбранной страницей в текущей книге используются следующие поля типа `Variant`:

- `FExcel` – хранит указатель на запущенный экземпляр приложения *Excel*;
- `FSheet` – хранит указатель на текущую страницу.

Эти переменные являются полями главной формы приложения. По сути это могут быть и глобальные переменные.

Запуск программы *Excel*

Работа этой подпрограммы довольно прозрачна: если к переменной `FExcel` еще не привязан объект автоматизации OLE, то пытаемся установить связь с активным экземпляром программы *Excel*. Если нет активной сессии *Excel*, то стартуем новый экземпляр. После старта устанавливаем видимость окна *Excel* в соответствии с состоянием флага «Показать», а затем запретить работу этой подпрограммы, чтобы во время работы программы пользователь не имел возможности стартовать другой экземпляр *Excel*. Код программы представлен в листинге .1.

Листинг .1. Подпрограмма старта программы *Excel*

```
void __fastcall TfoMainForm::actCreateExcelExecute(TObject *Sender)
{
  if (FExcel.IsEmpty())  // если Excel еще не стартовали
  {
try {
    FExcel = Variant::GetActiveObject ("Excel.Application");
  }
catch (...) {
    FExcel = Variant::CreateObject ("Excel.Application");
  }
    FExcel.OlePropertySet ("Visible", cbVisible->Checked);
    actCreateExcel->Enabled = false;
  }
}
```

Обработчик запроса на закрытие формы программы

Завершение программы достаточно сложный процесс. По щелчку «мыши» по кнопке в заголовочной строке формы или пункту главного меню программа должна проанализировать свое состояние, проделать соответствующие действия и принять решение выполнить ли приказ пользователя или проигнорировать. Поэтому после щелчка пользователя приложение получает от *Windows* несколько событий, в том числе и запрос `CloseQuery`. Эта процедура – суть обработчик этого запроса и имеет два параметра `Sender` и `CanClose`. О первом из них речь уже шла раньше (см. предыдущие главы). Второй параметр `CanClose` – булевского типа. Он является выходным и передается по ссылке. Такой запрос позволяет системе спросить у формы приложения: а можно ли ее закрывать? В данной подпрограмме выполняется проверка: есть ли активная сессия программы той *Excel*, которая привязана к полю `FExcel`. Выдается предупреждение пользователю и в зависимости от ответа выполняется завершение сессии *Excel* и приложения, или происходит возврат к работе. В последнем случае параметру `CanClose` присваивается значение `false`. Это извещает систему, что процесс закрытия формы не надо выполнять. А так как это главная форма, то не завершается и приложение.

СЕССИЯ

Сессией называется период активного состояния приложения *Excel*, при котором программа пользователя находится во взаимодействии с программой *Excel*.

В программе вызывается стандартная функция `MessageDlg`, которая выдает сообщение пользователю и предоставляет ему возможность выбора Да/Нет/Отменить (Ok/No/Cancel) и в качестве результата своей работы возвращает выбранное значение. В отличие от подпрограммы старта, здесь проверка связи с *Excel* выполняется внешней функцией `VarIsEmpty`, а не методом переменной. Код подпрограммы представлен в листинге .2.

Листинг 2. Обработчик запроса на закрытие формы программы

```
void __fastcall TfoMainForm::FormCloseQuery
                          (TObject *Sender, bool &CanClose)
{
  CanClose = true;  // установка начального значения

  if (!VarIsEmpty (FExcel)) // если есть сессия
   {
          // сообщить об этом и получить ответ
    int answ = MessageDlg
               ("Excel не закрыт! Завершить его работу?",
                mtConfirmation,
        TMsgDlgButtons() << mbYes << mbNo << mbCancel, 0);
    switch (answ)  // действия в зависимости от выбора
     {case mrYes : FExcel.OleProcedure ("Quit");
      case mrNo : FExcel = NULL; break;
      case mrCancel : CanClose = false;
     }
   }
}
```

Открытие книги

Для демонстрации методов OLE-объектов, в частности при работе с *Excel*, предлагаемый алгоритм реализован далеко не оптимально. Выбор в пользу такого решения обеспечен простотой алгоритма. Порядок ее работы следующий.

- Проверяется, открыта ли программа *Excel*. Если нет, то выдается сообщение и процедура завершает работу.

- Открывается стандартное диалоговое окно для выбора файла.
- Визуализируется программа *Excel* в зависимости от наличия галочки в переключателе `cbVisible`.
- Опрашивается *Excel* и запоминается объект: список книг `workBooks`.
- Через объект-список книг открывается выбранный файл.
- Очищается список имен файлов книг в компоненте `lbExcelBooks`. Опрашивается количество книг, открытых на текущий момент в программе *Excel*.
- В цикле перебираются открытые книги, опрашиваются их имена и количество страниц. Заносятся эти данные в список книг `lbExcelBooks`.
- Для каждой книги опрашиваются имена всех страниц и загружают их в выпадающий список `cbExcelSheets`, предварительно его очистив.

Код программы представлен в листинге .3.

Листинг 3. Процедура открытия книги

```
void __fastcall
   TfoMainForm::actOpenWorksheetsExecute(TObject *Sender)
{
      // если Excel подключен к программе
  if (!VarIsEmpty(FExcel))
   {      // открыть диалог для выбора файла
    if (dlgOpenExcelSheet->Execute ())
     {
           // сделать Excel видимым, если так указано
      FExcel.OlePropertySet ("Visible", cbVisible->Checked);
         // получить указатель на список книг
      Variant workBooks = FExcel.OlePropertyGet ("WorkBooks");
         // открываем выбранный файл-книгу
      workBooks.OleProcedure
            ("Open", dlgOpenExcelSheet->FileName.c_str());

      lbExcelBooks->Clear ();  // чистим список книг

           // узнаем количество открытых книг
      int couWB = workBooks.OlePropertyGet ("Count");
      for (int indW = 1; indW <= couWB; indW++)
                // цикл по книгам
       {  // выбираем очередную книгу из Excel
        Variant wb = workBooks.OlePropertyGet ("Item", indW);
          // спрашиваем ее имя
        AnsiString name = wb.OlePropertyGet ("Name");
          // указатель на список страниц книги
        Variant sheets = wb.OlePropertyGet ("Sheets");
          // узнаем количество страниц в книге
        int couSh = sheets.OlePropertyGet ("Count");
          // пополняем список книг
        lbExcelBooks->Items->Add ("В книге \"" + name + "\" "
                                  + IntToStr (couSh) + " страниц.");
          // чистим список страниц
        cbExcelSheets->Items->Clear();
        for (int indS = 1; indS <= couSh; indS++)
                  // по всем страницам
         {
          Variant sheet = wb.OlePropertyGet
                                        ("Sheets", indS);

        AnsiString name = sheet.OlePropertyGet ("Name");
                              // имя очередной страницы
          // добавляем имена страниц в ComboBox
        cbExcelSheets->Items->Add (name);
       } // for indS
      cbExcelSheets->ItemIndex = 0;
     }
   } // if dlgOpenExcelSheet
  } // if !VarIsEmpty
else MessageDlg ("Приложение MS Excel не активизировано!",
                 mtError, TMsgDlgButtons() << mbOK, 0);
```

Выбор книги

Для просмотра содержимого книг и их страниц средствами данного приложения в компонент `lbExcelBooks` загружается список книг по мере их открытия. Для просмотра же списка страниц в текущей книге и содержимого конкретной страницы к компонентам типа список `lbExcelBooks` и `cbExcelSheets` достаточно присоединить обработчики событий по выбору в них строк. Текст реализации обработчика по выбору книги в списке `lbExcelBooks` приведены в листинге .4.

Листинг 4. Обработчик события – выбор книги *Excel*

```
void __fastcall TfoMainForm::lbExcelBooksClick (TObject *Sender)
{
  if(lbExcelBooks->Count>0 && lbExcelBooks->ItemIndex>=0)
   {
    Variant workBooks = FExcel.OlePropertyGet ("WorkBooks");
      // выбираем очередную книгу из Excel
    Variant wb = workBooks.OlePropertyGet
                  ("Item", lbExcelBooks->ItemIndex + 1);
      // спрашиваем ее имя
    AnsiString name = wb.OlePropertyGet ("Name");
      // указатель на страницы книги
    Variant sheets = wb.OlePropertyGet ("Sheets");
      // количество страниц
    int couSh = sheets.OlePropertyGet ("Count");

    cbExcelSheets->Items->Clear();
      // проходим по всем страницам
    for (int indS = 1; indS <= couSh; indS++)
     {  // выбираем очередную страницу из книги
      Variant sheet = wb.OlePropertyGet ("Sheets", indS);
        // спрашиваем имя очередной страницы
      AnsiString name = sheet.OlePropertyGet ("Name");
        // заносим это имя в ComboBox
      cbExcelSheets->Items->Add (name);
     } // for indS
    cbExcelSheets->ItemIndex = 0;
   }
}
```

Логика работы подпрограммы проста: по индексу выбранной строки выбираем книгу в *Excel* с тем же индексом и заполняем выпадающий список `cbExcelSheets` названиями страниц в этой книге.

Выбор страницы

Структура и порядок работы этой процедуры аналогичен предыдущим и даже частично совпадает. Дополнительного пояснения, вероятно, требует один фрагмент из обработчика выбора страниц. Строка –

```
// определение занятых ячеек
Variant aux = sheet.OlePropertyGet ("Cells").
        OlePropertyGet ("SpecialCells", xlCellTypeLastCell);
```

– несколько отличается от предыдущих операторов обращения к OLE-объекту. Но только на первый взгляд. Здесь, как и при работе с привычными объектами, происходит обращение к свойству `Cells` объекта `sheet` и к свойству `SpecialCells` этого свойства `Cells`. Как и в сложных классах, свойство может быть ссылкой на объект, являющийся составной частью сложного.

Другой момент, который здесь следует оговорить, это именно обращение к свойству `SpecialCells` с параметром `xlCellTypeLastCell`. Эта специфичная команда позволяет создать ссылку на внутренний объект `SpecialCells`. Значения свойств `Row` и `Column` этого объекта `SpecialCells` соответствуют нижнему правому углу заполнения данных. Напомним, что страница *Excel* имеет огромные размеры – 65536×256 ячеек. Но в практической работе используется только малая часть из этого. При работе в программе *Excel* пользователь может просмотреть страницу и определить, в какой части имеются данные. При эксплуатации же программы как средства автоматизации OLE-приложения *Excel*, границы данных можно определить, как показано в листинге . 5, если это требуется, конечно.

Завершающий цикл в листинге 5 построчно перебирает ячейки активной страницы в пределах области заполнения данных и отображает их в виде текстовых строк в компоненте `lbContent` приложения. Индекс ячейки страницы книги *Excel* указывается привычным в программах образом – номер строки и номер колонки, задаваемые целыми константами и переменными.

Листинг 5. Обработчики события выбора страницы

```
void __fastcall TfoMainForm::cbExcelSheetsChange(TObject *Sender)
{  // Выбор страницы Excel документа
  if (lbExcelBooks->Count>0 && lbExcelBooks->ItemIndex>=0)
   {
    if(cbExcelSheets->Items->Count > 0 &&
      cbExcelSheets->ItemIndex >= 0)
     {
      Variant workBooks = FExcel.OlePropertyGet ("WorkBooks");
        // выбираем очередную книгу из Excel
      Variant wb = workBooks.OlePropertyGet
                    ("Item", lbExcelBooks->ItemIndex + 1);
        // указатель на страницы книги
      Variant sheets = wb.OlePropertyGet ("Sheets");
      Variant sheet = wb.OlePropertyGet
                ("Sheets", cbExcelSheets->ItemIndex + 1);
        // активизировать выбранную страницу
      sheet.OleProcedure ("Activate");
        // определение занятых ячеек
      Variant aux = sheet.OlePropertyGet ("Cells").
              OlePropertyGet ("SpecialCells", xlCellTypeLastCell);
      int rowLast = aux.OlePropertyGet ("Row");
      int colLast = aux.OlePropertyGet ("Column");

       lbContent->Clear();
       for (int iR = 1; iR <= rowLast; iR++)
        {AnsiString auxStr = "";
         for (int iC = 1; iC <= colLast; iC++)
          {
           auxStr += sheet.OlePropertyGet ("Cells", iR, iC);
           auxStr += " | ";
          }
         lbContent->Items->Add (auxStr);
        }
     }
   }
}
```

***Задача* .2.** Разработать приложение табулирования функций с использованием программы *Excel*.

Решение задачи

Решение задачи 1 предыдущего раздела предполагает чтение данных из различных книг и страниц *Excel*. Конечно, писать в ячейки страницы *Excel* так же легко. Пример оператора заполнения ячейки может выглядеть так:

```
FSheet.OlePropertySet ("Cells", row, col, val);
```

где индексированному свойству `Cells` объекта `FSheet` присваивается значение переменной `val`. Индекс тоже указывается целочисленными номерами строки и колонки. При этом переменная `val` может быть различного типа – или строковая, или числовая вещественного и целого типа.

Как известно, основное назначение программы *Excel* – это создание электронных таблиц. Естественно, что все эти возможности программы *Excel* доступны и при обращении к ней по технологии автоматизации OLE. Так, например, ячейкам можно присваивать формулы, которые будут вычисляться.

Предлагается следующая стратегия решения задачи.

1. Определяется, какую информацию следует предоставить пользователю и получить от пользователя.
2. Определяется список и последовательность действий, которые должен выполнить пользователь для того, чтобы передать необходимую информацию и получить требуемые результаты.
3. Разрабатывается интерфейс приложения.
4. Подготавливаются специальные процедуры для обеспечения действий пользователя.

Подготовка решения в соответствии с описанной стратегией.

1. Требуется получить информацию (входные данные) от пользователя следующего вида:
   а) границы интервала и шаг табулирования;
   б) формулу для функции, таблицу значений которой требуется получить.

   Для пользователя следует передать таблицу значений функции в виде графика функции, для лучшего восприятия ее поведения.
2. Предлагается обеспечить следующую последовательность действий с приложением:

а) запустить *Excel* с помощью кнопки;

б) задать значения входных данных. Для этого нужно предоставить соответствующие поля;

в) выполнить команду ввода данных;

г) выполнить команду «Произвести табулирование».

3. Может быть предложен интерфейс программы для решения задачи, как на рис. .2. Тогда интерфейс данного приложения будет содержать следующие элементы, представленные в таблице 2.

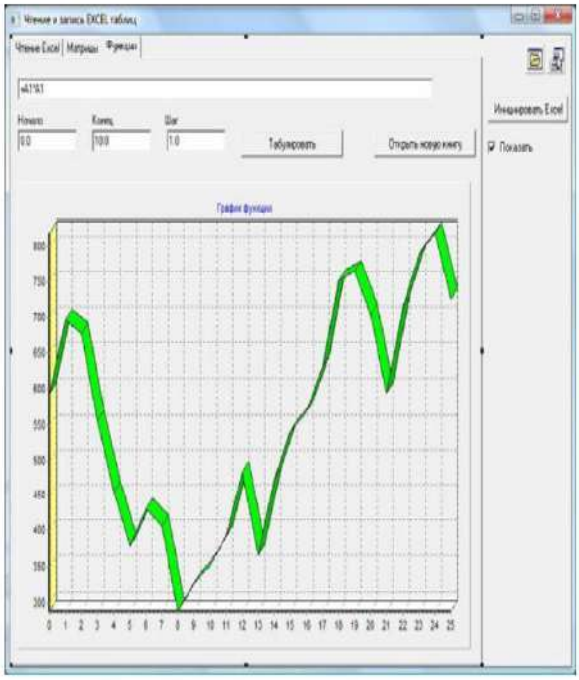


Рис. . 2. Интерфейс программы табулирования функции

Таблица .2

Основные компоненты приложения табулирования функции

№	Тип	Наименование	Назначение
1	TEdit	edFormula	Строка для ввода формулы
2	TEdit	edStart	Строка для ввода начала интервала табулирования
3	TEdit	edFinish	Строка для ввода конца интервала табулирования
4	TEdit	edStep	Строка для ввода шага табулирования
5	TBitBtn	bbInitExcel	Инициализация *Excel*
6	TBitBtn	bbInitBookFunc	Открытие книги
7	TBitBtn	bbTabulate	Запуск процесса табулирования
8	TChart	chGraphic	Отображение графика функции
9	TCheckBox	cbVisible	Управление видимостью *Excel*
10	TActionList	ActionList	Набор подпрограмм работы

4. Целесообразно разработать следующие подпрограммы для того, чтобы пользователь имел возможность выполнить предусмотренные действия, ввести и получить требуемую информацию:

- `actCreateExcelExecute(TObject *Sender);` – подпрограмма для запуска *Excel*;
- `FormCloseQuery(TObject *Sender, bool &CanClose);` – обработчик события «На закрытие формы»;
- `actOpenFuncBookExecute(TObject *Sender);` – создание нового документа в программе *Excel*;
- `actTabulateExecute(TObject *Sender);` – табулирование произвольной функции;

Примечание

В работе этого приложения используются те же процедуры инициализации работы программы *Excel* и обработки запроса на завершения работы, что и в предыдущей задаче. Поэтому описание их не приводится.

Создание новой книги

Алгоритм создания новой книги можно описать следующими шагами.

1. Выполняется проверка, выполнено ли уже подключение к *Excel*. Если подключение есть, то необходимо:
   1.1. Прочитать свойство «Рабочая книга».
   1.2. Выполнить процедуру «Добавить книгу».
   1.3. Установить свойство «Видимый» в соответствии с отметкой «Показать».
   1.4. Прочитать в книге свойство «количество страниц».
   1.5. Активизировать первую станицу.
2. Если подключения нет, то необходимо:
   2.1. Сообщить об этом.
   2.2. Завершить работу функции.

Описание функции приведено в листинге 6.

Листинг 6. Подпрограмма создания нового документа в программе *Excel*

```
void __fastcall TfoMainForm::actOpenFuncBookExecute
                                              (TObject *Sender)
{// открытие файла Excel для работы с функциями и матрицами
    // есть Excel, подключенный к программе,
  if (!VarIsEmpty(FExcel))
   {  // добавление нового документа
    FExcel.OlePropertyGet ("WorkBooks")
          .OleProcedure ("Add");
    FExcel.OlePropertySet ("Visible", cbVisible->Checked);
      // установка на первую страницу
      // только что загруженной книги
    int couWB = FExcel.OlePropertyGet ("WorkBooks")
                      .OlePropertyGet ("Count");
    FSheet = FExcel.OlePropertyGet ("WorkBooks")
                   .OlePropertyGet ("Item", couWB)
                   .OlePropertyGet ("Sheets", 1);
   }
  else MessageDlg ("Приложение MS Excel не активизировано!",
                   mtError, TMsgDlgButtons() << mbOK, 0);
}
```

Табулирование функции

Процедура табулирования функции с использование программы *Excel* почти такая же, как и стандартная процедура, приведенная в главе 3. Отличия лишь в том, что данная процедура загружает выражение функции в ячейку страницы *Excel*, и задает абсциссу и считывает ординату из соответствующих ячеек *Excel*.

Примечание Функцию необходимо задавать по правилам *Excel*, так как текст функции передается в ячейку страницы *Excel* без искажения. Так, например, для построения графика функции $F(x) = x^2$ можно задать строку `=A1*A1` или `=A1^2`.

В реализации данного алгоритма используются локальные переменные: `start`, `finish`, `step` – для описания границ и шага табулирования; `curr` – текущее значение аргумента функции. Тип `double` для переменных принят из-за того, что в программе используется функция проверки на ошибки `TryStrToFloat`, выполняющая преобразование именно в этот тип.

Алгоритм табулирования можно описать следующими шагами.

1. Читаются входные значения из окон строковых редакторов и выполняется проверка на правильность написания чисел. Если нет ошибок при чтении данных, то
   - очищается область отображения графика;
   - заносится формула в ячейку страницы *Excel*;
   - инициализируется начальное значение аргумента;
   - выполняется цикл до тех пор, пока значение аргумента в области. В теле этого цикла:
     - считывается значение функции;
     - это значение добавляется к графику;
     - вычисляется новое значение аргумента.
2. Выдается сообщение пользователю, если есть ошибки при написании. Описание функции приведено в листинге 7.

Листинг 7. Процедура табулирования функции

```
void __fastcall TfoMainForm::actTabulateExecute (TObject *Sende
{  // процесс табулирования произвольной функции
   // посредством Excel
   // тип double из-за того,
   // что TryStrToFloat преобразует в него
  double start, finish, step;
    // есть ли ошибки в написании чисел
  if (TryStrToFloat (edStart->Text, start) &&
      TryStrToFloat (edFinish->Text, finish) &&
      TryStrToFloat (edStep->Text, step))
   {serGraphicOfFunction->Clear();  // чистим график

    FSheet.OlePropertySet ("Cells", 2, 1,
              edFormula->Text.c_str()); // заносим формулу

    double aux;
    double curr = start;
    while (curr <= finish)
     {FSheet.OlePropertySet ("Cells", 1, 1, curr);
                              // заносим текущее значение
      aux = FSheet.OlePropertyGet ("Cells", 2, 1);
                              // считываем значение функции
      serGraphicOfFunction->AddXY (curr, aux);
                                    // значение в график

      curr += step;  // и т.д. и т.п.
     }
   }
  else MessageDlg ("Одно из чисел не правильно написано!",
                   mtError, TMsgDlgButtons() << mbOK, 0);
}
```

Такое решение задачи, то есть применение автоматизации OLE-программы *Excel*, позволяет строить простое приложение, в котором используются произвольные формулы.

Задача .3. Разработать приложение по выполнению основных операций программой *Excel* с матрицами. Реализовать операции: вычисление определителя, нахождением суммы элементов матрицы, вычисление обратной матрицы и их перемножение.

Решение

Как известно, возможности программы *Excel* не ограничиваются вычислениями простейших формул. Она позволяет решать довольно широкий круг стандартных задач. Например, выполнять операции над матрицами. И тогда, естественно, имея шаблоны обращения к программе *Excel*, можно быстро разработать приложение, которое будет решать такие задачи посредством *Excel*. В последующем, конечно, можно разрабатывать и тщательно тестировать свои процедуры выполнения таких операций.

Прежде чем приводить примеры выполнения операции с матрицами, отметим такой существенный аспект. В предыдущем примере формула задавалась как текстовое значение ячейки страницы *Excel*. Но не всегда такая простая конструкция позволяет задавать формулы. Другой способ задавать формулы в ячейку страницы программы *Excel* – это воспользоваться свойством `FormulaR1C1` ячейки. В качестве значения этому свойству передается такая же строка формулы. Отличительной особенностью строки-формулы является то, что адреса ячеек задаются не в привычном для *Excel* формате, например `"A1"`, а в виде `"R1C1"`. То есть адрес строки задается в виде: номер строки – номер столбца, предваряемые литерами `R` (row) и `C` (column).

Предлагается следующая стратегия решения задачи.

1. Определяется, какую информацию следует предоставить пользователю и получить от него.
2. Определяется список и последовательность действий, которые должен выполнить пользователь для того, чтобы передать необходимую информацию и получить требуемые результаты.
3. Разрабатывается интерфейс приложения.
4. Подготавливаются специальные процедуры для обеспечения действий пользователя.

Подготовка решения в соответствии с описанной стратегией.

1. Требуется от пользователя получить информацию (входные данные) следующего вида:
   а) размеры матрицы;
   б) значения элементов матрицы.

Пользователю следует передать результаты вычислений в виде числовых значений.

2. Предлагается обеспечить следующую последовательность действий с приложением:
   - запустить *Excel* с помощью кнопки;
   - задать значения входных данных. Для этого нужно предоставить соответствующие поля;
   - выполнить команду ввода данных;
   - выполнить команду «Вычислить/произвести операцию»;
   - для решения задачи может быть предложен интерфейс программы, как на рис. 3. Интерфейс данного приложения будет содержать следующие элементы, представленные в табл. .3.

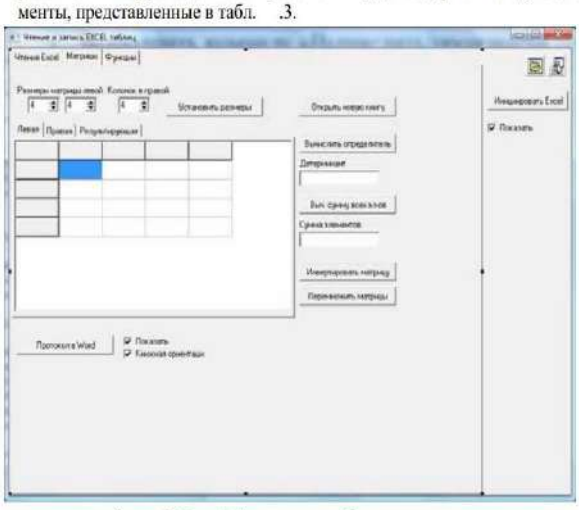


Рис. 3. Интерфейс программы работы с матрицами

3. Для того чтобы пользователь имел возможность выполнить предусмотренные действия, ввести и получить требуемую информацию, целесообразно разработать следующие подпрограммы:

- `actCreateExcelExecute(TObject *Sender);` – подпрограмма для запуска *Excel*;
- `FormCloseQuery(TObject *Sender, bool &CanClose);` – обработчик события «На закрытие формы»;
- `actOpenFuncBookExecute(TObject *Sender);` – создание нового документа в программе *Excel*;
- `actSetSizeExecute(TObject *Sender);` – ввод размеров матрицы и значений элементов матрицы в отведенные для них области;

- `actSummOfElemExecute(TObject *Sender);` – процедура суммирования элементов матрицы;
- `actMatrixInverse(TObject *Sender);` – процедура вычисления обратной матрицы;
- `actMatrixMultiple(TObject *Sender);` – процедура вычисления произведения матриц.

Таблица .3

Основные компоненты приложения работы с матрицами

№	Тип	Наименование	Назначение
1	TCSpinEdit	cseRowCount	Количество строк в матрице
2	TCSpinEdit	cseColCount	Количество столбцов в матрице
3	TCSpinEdit	cseColCountR	Количество столбцов в правой матрице
4	TStringGrid	sgLeftMatr	Таблица для задания и отображения значений исходной матрицы, левого сомножителя
5	TStringGrid	sgRightMatr	Таблица для задания и отображения значений исходной матрицы, правого сомножителя
6	TStringGrid	sgProductMatr	Таблица для отображения значений матрицы результатов
7	TBitBtn	bbInitExcel	Инициализация *Excel*
8	TBitBtn	bbInitBookMatrix	Открытие книги
9	TBitBtn	bbSetSize	Установка размеров и заполнение матриц случайными числами
10	TBitBtn	bbDeterminate	Определение детерминанта
11	TBitBtn	bbSumm	Определение суммы элементов
12	TBitBtn	bbInverse	Вычисление обратной матрицы
13	TBitBtn	bbProd	Вычисление произведения матриц
14	TEdit	edDeterminate	Окно отображения результата вычисления детерминанта
15	TEdit	edSummOfElem	Окно отображения результата вычисления суммы элементов
16	TCheckBox	cbVisible	Управляет видимостью *Excel*
17	TActionList	ActionList	Набор подпрограмм работы

Кнопки `bbInitExcel` и `bbInitBookMatrix` аналогичны по своему назначению подобным кнопкам из предыдущих примеров: открытие сессии связи с программой *Excel* и открытие нового документа.

Ввод размеров матрицы и значений элементов матрицы

Процедура задания размеров компонента `StringGrid` и заполнение его случайными данными довольно проста:

- из окон редакторов `TCSpinEdit` читаются значения, определяющие размеры матрицы, и на основании этих размеров определяется число строк и число столбцов в сетках `StringGrid` для левой, правой и результирующей матриц;
- выполняются двойные циклы (по строкам и столбцам сетки), в теле которых генерируются случайные числа; и из них формируются значения ячеек сетки. Для результирующей сетки ячейки очищаются.

Текст реализации процедуры приведен в листинге .8.

Листинг 8. Процедура заполнения таблицы значениями матрицы

```
void __fastcall TfoMainForm::actSetSizeExecute (TObject *Sender)
{  // установить размеры матрицы
  sgLeftMatr->RowCount = cseRowCount->Value + 1;
  sgLeftMatr->ColCount = cseColCount->Value + 1;

  for (int iRow = 1; iRow < sgLeftMatr->RowCount; iRow++)
    for (int iCol = 1; iCol < sgLeftMatr->ColCount; iCol++)
      sgLeftMatr->Cells[iCol][iRow] =
                          RandomRange (-100, 100) * 0.1;

  sgRightMatr->RowCount = cseColCount->Value + 1;
  sgRightMatr->ColCount = cseColCountR->Value + 1;

  for (int iRow = 1; iRow < sgRightMatr->RowCount; iRow++)
    for (int iCol = 1; iCol < sgRightMatr->ColCount; iCol++)
      sgRightMatr->Cells[iCol][iRow] =
                          RandomRange (-100, 100) * 0.1;

  sgProductMatr->RowCount = cseRowCount->Value + 1;
  sgProductMatr->ColCount = cseColCountR->Value + 1;

  for (int iRow = 1; iRow < sgProductMatr->RowCount; iRow++)
    for (int iCol = 1; iCol <
                          sgProductMatr->ColCount; iCol++)
      sgProductMatr->Cells[iCol][iRow] = "";
}
```

Примечание Результаты работы следующих двух подпрограмм (значения определителя и суммы) заносятся в текстовую строку редактирования. Поэтому из ячейки *Excel* результат переносится как текст. Если необходимо получить результат в числовом формате, то нужно считывать его непосредственно в вещественную переменную или потом правильно доставать и преобразовать из компонента `edSummOfElem`.

Вычисление определителя матрицы

Поскольку задача решается из предположения, что используется чистая страница *Excel*, то значения элементов матрицы заносятся в страницу, начиная с ячейки `A1`. Тогда при формировании строки формулы основная трудность – это написать завершающую часть, в частности `RnCm`, где `n` – номер последней строки, а `m` – номер последнего столбца.

Так как для формирования строки формулы (переменная `aux`) в процедуре используется вспомогательная переменная типа `AnsiString`, то удалось обойтись без явного вызова функций преобразований из численного формата в текстовый, а использовать только операторы, определенные для этого класса. В качестве параметра задается не вся переменная, а ее содержимое в виде стандартного массива типа `char`.

Логика работы этой процедуры заключается в следующем:

- переписать значения из таблицы `StringGrid` в ячейки страницы *Excel*;
- сформировать строку команды вычисления определителя;
- записать формулу в свободную ячейку страницы *Excel*;
- считать результаты вычисления – значение определителя данной матрицы.

Свойство `FormulaR1C1` предназначено для присвоения ячейке страницы *Excel* текста формулы. При этом адреса ячеек в формуле должны быть записаны в формате `R1C1`. Здесь текст формулы формируется в локальной переменной `aux` типа строка. В параметрах функции МОПРЕД формируется интервал размещения коэффициентов матрицы в ячейках страницы. Например, для матрицы 4×4 текст формулы будет сформирован в виде `"МОПРЕД(R1C1:R4C4)"`. Код процедуры вычисления детерминанта приведен в листинге .9.

Листинг 9. Процедура вычисления определителя матрицы

```
void __fastcall TfoMainForm::actDeterminateExecute (TObject *Sender)
{  // процедура вычисления определителя
  for (int iRow = 1; iRow < sgMatrix->RowCount; iRow++)
    for (int iCol = 1; iCol < sgMatrix->ColCount; iCol++)
      {  // заносим значения из StringGrid в Excel
       FSheet.OlePropertySet ("Cells", iRow, iCol,
                          sgMatrix->Cells[iCol][iRow].c_str());
      }
    // формирование строки с формулой
  AnsiString aux = "=МОПРЕД(R1C1:R";
  aux += sgMatrix->RowCount - 1; aux += "C";
  aux += sgMatrix->ColCount - 1; aux += ")";
    // выполнение команды
  FExcel.OlePropertyGet ("Cells", sgMatrix->RowCount + 1, 1)
                        .OlePropertySet ("FormulaR1C1", aux.c_str());
    // чтение результата
  edDeterminate->Text = FSheet.OlePropertyGet ("Cells",
                                sgMatrix->RowCount + 1, 1);
}
```

Вычисление суммы элементов матрицы

Текст этой подпрограммы почти полностью совпадает с подпрограммой вычисления определителя. Отличие, естественно, в имени вызываемой функции. Также выбрана другая ячейка страницы для занесения формулы и считывания результата. Помимо этого результат сохраняется в другой компонент приложения – в `edSummOfElem`. Код процедуры вычисления суммы элементов приведен в листинге . .10.

Листинг 10. Процедура вычисления суммы элементов матрицы

```
void __fastcall TfoMainForm::actSummOfElemExecute (TObject *Sender)
{  // процедура суммирования элементов матрицы
  for (int iRow = 1; iRow < sgMatrix->RowCount; iRow++)
    for (int iCol = 1; iCol < sgMatrix->ColCount; iCol++)
      {  // заносим значения из StringGrid в Excel

       FSheet.OlePropertySet ("Cells", iRow, iCol,
                          sgMatrix->Cells[iCol][iRow].c_str());
      }
    // формирование строки с формулой
  AnsiString aux = "=СУММ(R1C1:R";
  aux += sgMatrix->RowCount - 1; aux += "C";
  aux += sgMatrix->ColCount - 1; aux += ")";
    // выполнение команды
  FExcel.OlePropertyGet ("Cells", sgMatrix->RowCount + 1, 3)
              .OlePropertySet ("FormulaR1C1", aux.c_str());
    // чтение результата
  edSummOfElem->Text = FSheet.OlePropertyGet
                        ("Cells", sgMatrix->RowCount + 1, 3);
}
```

Примечание Все процедуры, приведенные в этой главе, сильно упрощены. Точнее приведены только необходимые действия. Вполне понятно, что здесь нет, например, проверки на допустимость операции – некоторые операции могут выполняться только над квадратными матрицами, нет проверки правильности результата. Помимо этого, конечно, эти подпрограммы должны быть дополнены проверками: установлена ли связь с программой *Excel*, со страницей *Excel* и тому подобное.

Вычисление обратной матрицы

Задача решается аналогично предыдущим. Последовательность выполнения действий следующая:

- заносятся значения из ячеек `StringGrid` в ячейки страницы *Excel*;
- формируется текст формулы и заносится в ячейку *Excel*;
- вычисляется формула с заполнением результатов по соответствующему интервалу ячеек страницы *Excel*;
- результаты переписываются в сетку `StringGrid`, предназначенную для этого.

Код процедуры вычисления суммы элементов приведен в листинге 11.

Листинг 11. Процедура вычисления обратной матрицы

```
void __fastcall TfoMainForm::actMatrixInverseExecute
                                              (TObject *Sender)
{ /* Вычисление инвертированной матрицы */
    // вычисление адресов региона ниже будущей матрицы
  AnsiString rgnLT = "=АДРЕС(";
  rgnLT += sgLeftMatr->RowCount + 1; rgnLT += ";1";
  FSheet.OlePropertySet ("Cells", 1, 1, rgnLT.c_str());
                                          // заносим формулу
  rgnLT = FSheet.OlePropertyGet ("Cells", 1, 1);
                        // читаем адрес "$A$6", например

  AnsiString rgnRB = "=АДРЕС(";
  rgnRB += 2 * sgLeftMatr->RowCount - 1; rgnRB += ";";
  rgnRB += sgLeftMatr->ColCount - 1; rgnRB += ")";
  FSheet.OlePropertySet ("Cells", 1, 1, rgnRB.c_str());
                                          // заносим формулу
  rgnRB = FSheet.OlePropertyGet ("Cells", 1, 1);
                        // читаем адрес "$G$15", например
```

```
  // процесс вычисления обратной матрицы
 for (int iRow = 1; iRow < sgLeftMatr->RowCount; iRow++)
   for (int iCol = 1; iCol < sgLeftMatr->ColCount; iCol++)
    {  // заносим значения из StringGrid в Excel
     FSheet.OlePropertySet ("Cells", iRow, iCol,
                  sgLeftMatr->Cells[iCol][iRow].c_str());
    }

  // формирование строки с формулой
 AnsiString aux = "=МОБР(R1C1:R";
 aux += sgLeftMatr->RowCount - 1; aux += "C";
 aux += sgLeftMatr->ColCount - 1; aux += ")";
  // выполнение команды
 FExcel.OlePropertyGet ("Cells",
                         sgLeftMatr->RowCount + 1, 1).
            OlePropertySet ("FormulaR1C1", aux.c_str());

 FSheet.OlePropertyGet ("Range", rgnLT.c_str(),
                  rgnRB.c_str()).OleFunction ("Select");
 FExcel.OlePropertyGet ("Selection").OlePropertySet
                        ("FormulaArray", aux.c_str());

  // копирование результатов
 for (int iRow = 1; iRow < sgProductMatr->RowCount; iRow++)
   for (int iCol = 1; iCol <
                      sgProductMatr->ColCount; iCol++)
    {  // заносим значения из Excel
       // в StringGrid матрица результатов
     sgProductMatr->Cells[iCol][iRow] =
              FSheet.OlePropertyGet ("Cells",
                    iRow + sgLeftMatr->RowCount, iCol);
    }
}
```

Как видно из листинга, общий ход решения стандартный, но есть две особенности. Первая особенность заключается в том, что решение необходимо «распространить» на область. Для этого используется свойство `FormulaArray` выделенной области `Selection`. Для выделения достаточно обратиться к функции `Select` свойства `Range`. Параметрами этой функции являются адреса левой верхней и правой нижней ячеек области, требующей выделения.

Вторая особенность заключается в том, что адреса ячеек в функции `Select` задаются в стандартном виде, принятом в электронных таблицах *Excel*: например, `"A1"` для самой левой верхней ячейки таблицы, поэтому номер столбца необходимо выразить в буквенном виде. Есть два пути – или разработать свою функцию для перевода чисел в соответствующие буквы или буквосочетания, или воспользоваться функциями того же *Excel*. В данной процедуре реализован второй вариант. Так как страница *Excel* перед выполнением операции обращения матрицы считается полностью свободной, то ячейка `A1` используется для пересчета необходимых адресов из формата `R1C1` в формат `A1`. В начале процедуры записываются формулы с функцией `АДРЕС`, например, `"АДРЕС(6, 1)"` и считывается результат `"A6"` в локальную переменную `rgnLT`. Аналогично вычисляется адрес правой нижней ячейки и сохраняется в переменной `rgnRB`. Позже эти переменные используются при указании области требующей выделения.

Вычисление произведения матриц

Решение этой задачи вполне аналогично решению предыдущей. Отличие лишь в том, что на страницу *Excel* приходится копировать две исходные матрицы и пользоваться функцией `МУМНОЖ` с указанием двух регионов расположения коэффициентов матриц. Код процедуры вычисления суммы элементов приведен в листинге .12.

Листинг 12. Процедура вычисления произведения матриц

```
void __fastcall TfoMainForm::actMatrixMultipleExecute(TObject *Sender)
{ /* Умножение матриц */
   // вычисление адресов региона НИЖЕ будущей матрицы
  AnsiString rgnLT = "=АДРЕС(";  // левая верхняя ячейка
  rgnLT += sgLeftMatr->RowCount + 1; rgnLT += ";1";
  FSheet.OlePropertySet ("Cells", 1, 1, rgnLT.c_str());
                                      // заносим формулу
  rgnLT = FSheet.OlePropertyGet ("Cells", 1, 1);
                          // читаем адрес "$A$6", например

  AnsiString rgnRB = "=АДРЕС(";  // правая нижняя ячейка
  rgnRB += 2 * sgLeftMatr->RowCount - 1; rgnRB += ";";
  rgnRB += sgLeftMatr->ColCount - 1; rgnRB += ")";
  FSheet.OlePropertySet ("Cells", 1, 1, rgnRB.c_str());
                                      // заносим формулу
  rgnRB = FSheet.OlePropertyGet ("Cells", 1, 1);
                         // читаем адрес "$G$15", например

   // процесс вычисления произведения матриц
  for (int iRow = 1; iRow < sgLeftMatr->RowCount; iRow++)
    for (int iCol = 1; iCol < sgLeftMatr->ColCount; iCol++)
     {  // заносим значения из StringGrid
        // в Excel левая матрица
      FSheet.OlePropertySet ("Cells", iRow, iCol,
                             sgLeftMatr->Cells[iCol][iRow].c_str());
     }

for (int iRow = 1; iRow < sgRightMatr->RowCount; iRow++)
  for (int iCol = 1; iCol < sgRightMatr->ColCount; iCol++)
   {  // заносим значения из StringGrid
      // в Excel правая матрица
    FSheet.OlePropertySet ("Cells", iRow,
                  iCol + sgRightMatr->ColCount,
                  sgRightMatr->Cells[iCol][iRow].c_str());
   }

  // формирование строки с формулой
AnsiString aux = "=МУМНОЖ(R1C1:R";
aux += sgLeftMatr->RowCount - 1;
aux += "C"; aux += sgLeftMatr->ColCount - 1;
aux += ";R1C";
aux += sgLeftMatr->ColCount + 1; aux += ":R";
aux += sgRightMatr->RowCount - 1; aux += "C";
aux += sgLeftMatr->ColCount +
                  sgRightMatr->ColCount - 1; aux += ")";

  // выполнение команды
FExcel.OlePropertyGet ("Cells",
            sgLeftMatr->RowCount + 1, 1).
```

.2. Связь с MS Word for Windows по технологии OLE

Эксплуатация приложений, выполняющих обработку данных и проводящих анализ, как правило, сопровождается составлением отчетов по результатам обработки. Обычно такие отчеты содержат таблицы и графики. И если текстовая часть отчета создается составителем, то таблицы и отдельные фрагменты текста могут быть стандартизованы и их составление можно поручить, например, тому же приложению, которое проводит обработку. Так как сегодня фактически стандартным средством составления текстовых документов является текстовый редактор *Word for Windows*, то здесь будут показаны основные приемы формирования фрагментов текста средствами OLE.

Для примера работы программой *Word* средствами OLE будет решена следующая задача.

Задача .4. Разработать процедуру формирования отчета в текстовом редакторе *Word* по результатам обработки матриц.

Решение задачи

Логика и структура процедуры формирования отчета о работе должна быть проста – переносить различные данные из приложения в виде текстов в документ в формате *Word*. Несмотря на то, что текст данной процедуры достаточно длинен, тем не менее, его структура проста и понятна по комментариям. Последовательность работы этой подпрограммы следующая:

- открывается новая сессия программы *Word*, если она еще не запущена;
- создается новый документ. Если указано в интерфейсе, то устанавливается ориентация листа как альбомная;
- вставляется произвольный текст и, как пример, дата и время создания документа;
- вставляется текст, соответствующий номеру и заголовку таблицы, с необходимыми параметрами выравнивания;
- определяется требуемое количество строк и колонок таблицы;
- создается таблица требуемого размера;
- прописываются названия колонок и строк;
- переписываются значения из интерфейса программы (компонент `StringGrid`) в созданную таблицу;
- выделяется таблица и задается требование отображать границы таблицы и ячеек и параметры толщины внешней линии и линий, разделяющих ячейки;
- добавляется текст из строк редактирования и размещается после таблицы;
- задается вопрос пользователю и, если требуется, производится завершение сессии программы *Word*.

Код, содержащий описание процедуры, приведен в листинге .13.

Листинг 13. Процедура формирования документа *Word*

```
void __fastcall TfoMainForm::actResultToWordExecute (TObject *Sender)
{ // сохранение результатов в документе Word for Windows
    // открытие программы Word
  if (FWord.IsEmpty())  // если Word еще не стартовали
   {
    FWord = Variant::CreateObject ("Word.Application");
    FWord.OlePropertySet ("Visible",
                          cbVisibleWord->Checked);
   }
    // создание нового документа
  FWord.OlePropertyGet ("Documents").OleProcedure ("Add");

    // установка на только что созданный документ
  Variant newDoc = FWord.OlePropertyGet ("ActiveDocument");

    // установка ориентации страниц
  if (chOrientation->Checked)
    newDoc.OlePropertyGet ("PageSetup")
                         .OlePropertySet ("Orientation", 0);
  else
    newDoc.OlePropertyGet ("PageSetup")
                         .OlePropertySet ("Orientation", 1);
    // занесение текста
  FWord.OlePropertyGet ("Selection")
  .OleProcedure ("TypeText", "Данные текст был создан ...");
    // перевод строки -- параграф
  FWord.OlePropertyGet ("Selection")
            .OleProcedure ("TypeText", WideString('\012'));
    // вывод даты ...
  AnsiString datetime = "в день " + DateToStr(Date ())
                        + " в " + TimeToStr(Time ()) + ";";
  FWord.OlePropertyGet ("Selection")
              .OleProcedure ("TypeText", datetime.c_str());
    // создание таблицы
  Variant parL = newDoc.OlePropertyGet
                 ("Paragraphs").OlePropertyGet ("Last");
  parL.OlePropertyGet ("Format")
                       .OlePropertySet ("Alignment", 2);

  FWord.OlePropertyGet ("Selection")
               .OleProcedure ("TypeText", "Таблица\012");

   parL.OlePropertyGet ("Format")
                        .OlePropertySet ("Alignment", 1);
   FWord.OlePropertyGet ("Selection")
         .OleProcedure ("TypeText", "Исходная матрица\012");
    // настройка региона на последнюю позицию
   Variant range = newDoc.OlePropertyGet ("Content");
   range.OlePropertySet ("Start", range.OlePropertyGet ("End"));

   int nRow = sgMatrix->RowCount; // сколько строк
   int nCol = sgMatrix->ColCount; // сколько колонок
    // добавляем таблицу
   Variant wordTable = newDoc.OlePropertyGet ("Tables")
                             .OleFunction ("Add", range, nRow, nCol);
    // подписываем колонки и строки
   for (int iCol = 1; iCol < nCol; iCol++)
    {AnsiString aux = "K"; aux += iCol;
     wordTable.OleFunction ("Cell", 1, iCol + 1)
              .OlePropertyGet ("Range")
              .OlePropertySet ("Text", aux.c_str());
    }
   for (int iRow = 1; iRow < nRow; iRow++)
    {AnsiString aux = "C"; aux += iRow;
     wordTable.OleFunction ("Cell", iRow + 1, 1)
              .OlePropertyGet ("Range")
              .OlePropertySet ("Text", aux.c_str());
    }
    // заносим данные
   for (int iRow = 1; iRow < nRow; iRow++)
     for (int iCol = 1; iCol < nCol; iCol++)
       wordTable.OleFunction ("Cell", iRow + 1, iCol + 1)
              .OlePropertyGet ("Range")
              .OlePropertySet ("Text",
                       sgMatrix->Cells [iCol][iRow].c_str());
    // очерчиваем границы
   wordTable.OleProcedure ("Select"); // выделить таблицу
   wordTable.OlePropertyGet ("Borders")
                             .OlePropertySet ("Enable", true);
   wordTable.OlePropertyGet ("Borders")
                    .OlePropertySet ("OutsideLineWidth", 8);
   wordTable.OlePropertyGet ("Borders")
                    .OlePropertySet ("InsideLineWidth", 4);

   parL.OlePropertyGet ("Range").OleProcedure ("Select");
    // пустая строка
   FWord.OlePropertyGet ("Selection")
              .OleProcedure ("TypeText", WideString('\012'));


    // занесение текста -- значение определителя
  AnsiString aux = "Определитель = "
                   + edDeterminate->Text + "\012";
  parL.OlePropertyGet ("Format")
                       .OlePropertySet ("Alignment", 0);
  FWord.OlePropertyGet ("Selection")
                 .OleProcedure ("TypeText", aux.c_str());

    // занесение текста -- сумма элементов
  aux = "Сумма элементов = " + edSummOfElem->Text + "\012";

     FWord.OlePropertyGet ("Selection")
                  .OleProcedure ("TypeText", aux.c_str());

      // закрытие документа Word
     if (!FWord.IsEmpty())
      {
       int answ = MessageDlg ("Завершить его работу Word?",
        mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0);
       switch (answ)
        {case mrYes : FWord.OleProcedure ("Quit");
        }
      }
}
```

Примечание Конечно, данные могут переноситься не только из переменных и компонентов приложения, но и из ячеек таблиц *Excel* напрямую. В данном примере все операции по работе с *Word for Windows* помещены в единую процедуру.

Для реализации более сложного приложения, необходимо тщательно планировать какие действия должны выполняться в одной процедуре, а какие разделены по разным процедурам.

Задания для самостоятельной работы:

1. Исправить приложение к задаче 3, так чтобы оно работало только для вещественных данных, записанных через точку.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы; *оценка «4» ставится, если:*

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи; *оценка «3» ставится, если:*

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 36 Вывод табличных данных

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: создать программу, рассчитывающую суммы по столбцам и строкам вводимых чисел в C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

1. Добавить на форму сетку – компонент StringGrid. Этот компонент служит для ввода и вывода табличных данных.

2. Необходимо ввести данные и описать событие реагирования на их ввод. В свойстве Options установите пункт Editing, иначе сетка будет доступной только для чтения. Теперь при запуске программы можно вводить текст.

3. Щелкнув мышкой в Инспекторе Объектов, описать обработчик события OnSelectCell, которое возникает, когда пользователь выбирает какую-либо ячейку для редактирования.

4. Теперь при запуске программы и выборе ячейки в заголовке формы отображается информация о столбце и строке ввода.

5. Обработать событие, возникающее, когда пользователь пытается выделить какую-нибудь ячейку (OnSelectCell):

Caption=«Выделена клетка (»+ IntToStr(Acol)+«:»+ IntToStr(Arow)+«)»;

В качестве параметров обработчик получает (кроме Sender) номер столбца, номер строки и переменную CanSelect, которую можно изменить внутри обработчика.

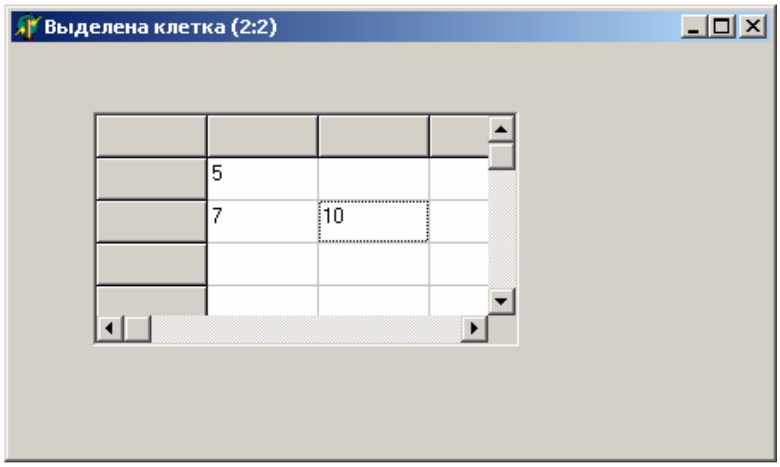


Рисунок 1

Можно, например, запретить выделение ячеек третьего столбца. Для этого вставьте вэтот обработчик еще строчку:

CanSelect = (Acol!= 3);

В этом случае CanSelect будет равен false, если Acol = 3.

6. Настроить сетку в зависимости от значения констант. Теперь при запуске программы сетка имеет размер, заданный в константе, и выглядит значительно аккуратнее. Если изменить значение константы, то при запуске и размер сетки будет соответствующим.

7. Установить размер сетки согласно значению констант Num и cSize.

```
const int Num=4, int cSize=30;
Обработчик события OnCreate формы:
{
MyGrid->DefaultColWidth = cSize;
MyGrid->DefaultRowHeight = cSize;
MyGrid->ColCount = Num;
MyGrid->RowCount = Num;
MyGrid->Width = Num * (cSize + 1) + 3;
MyGrid->Height = Num * (cSize + 1) + 3;
MyGrid->Font->Size = cSize / 2;
}
```

Рисунок 2

В приведенном участке кода изменяются ширина и высота сетки, установленные по умолчанию, а также количество строк и столбцов сетки. Добавление единицы к ширине каждой ячейки связано с наличием линий между ячейками, а добавление тройки ко всей сумме – наличием бордюра вокруг сетки. В последней строчке устанавливается соответствующий величине клетки размер шрифта.

8. Для расчета суммы по столбцам и строкам вводимых чисел написать собственную процедуру и две функции.

Функции будут рассчитывать сумму в строке и столбце, получая их номер в качестве параметра. Процедура будет в цикле вызывать эти функции и соответствующим образом заполнять клетки.

Теперь надо создать обработчик для нажатия на саму форму и описать в нем вызов расчета. Собственная процедура стоит в коде раньше, чем обработчик, из которого она вызывается. Когда описывается собственная процедура, нужно обращаться к компонентам через форму.

При запуске программы требуется ввести числа в сетку и кликнуть на саму форму.

```
int ColSum (int n)
{ int i, Result = 0;
for (int i=1; i<Num – 1: i++)
        Result+= StrToInt (Form1->MyGrid->Cells[n][i]);
return Result;
}
int RowSum (int n)
{
int i, Result = 0;
for (int i=1; i<Num – 1: i++)
        Result += StrToInt(Form1->MyGrid->Cells[i][ n]);
return Result;
}
void Calculate ()
{
int i;
for (int i=1; i<Num – 1: i++)
{
        Form1->MyGrid->Cells[i][0] = IntToStr(ColSum(i));
        Form1->MyGrid->Cells[0][i] = IntToStr(RowSum(i));
}
}
```

В данной программе используется свойство Cells, компонента сетки. Это свойство имеет тип двумерного массива строк. Счет в этом массиве начинается с нуля.

Фиксированные (серые) клетки, с точки зрения индексирования массива, ничем не отличаются от прочих. В нашем случае фиксированы первая строка и первый столбец (в массиве они имеют соответствующие нулевые координаты). Поэтому при расчете суммы проходят от 1 (а не от 0) до Num – 1 (при индексировании с нуля последний столбец/строка имеют, понятно, номер Num – 1).

Однако, если произойдет ввод какой-нибудь буквы, сразу возникнет исключительная ситуация. Она, очевидно, возникает из-за невозможности перевода буквы в число в функции StrToInt.

Задание: изучить и добавить обработку исключительной ситуации.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 37 База данных «Записная книжка»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

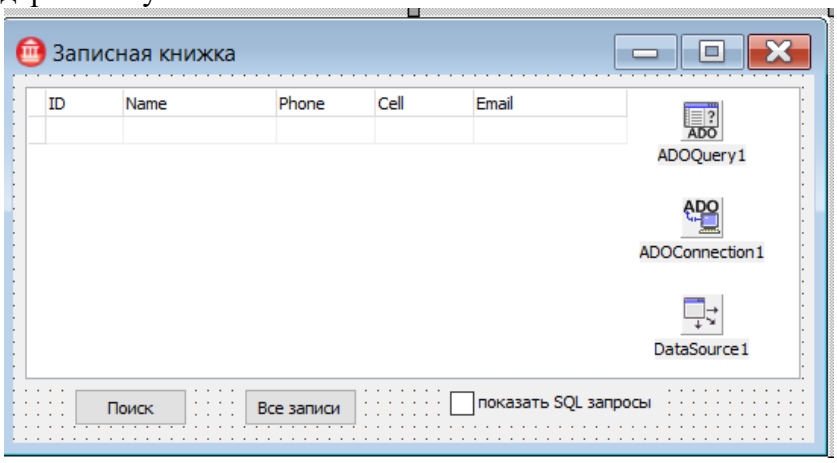
Цель: получить навыки создания приложения для базы даннных C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

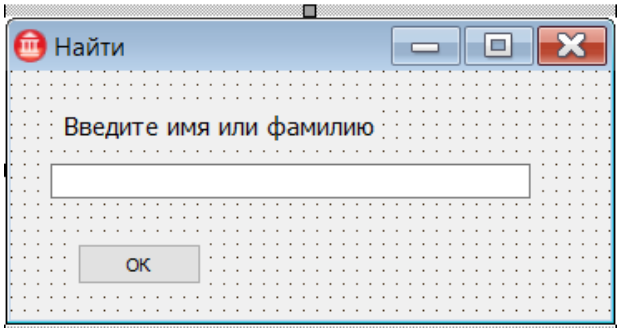
Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Программа Записная книжка демонстрирует использование компонентов ADO. База данных состоит из одной таблицы adrbk.accdb. Программа позволяет просматривать, редактировать, добавлять и удалять записи, а также обеспечивает выборку (поиск) информации по содержимому поля Имя.



Критерий запроса (имя, фамилия или фрагмент имени) вводятся в окне Найти, которое появляется в результате щелчка на кнопке Поиск.



Задание: добавить фильтрацию

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 38 Разработка приложения для работы с базой данных

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

В СУБД Access создана база данных «Телефонный справочник» (файл db_Telefon.accdb). База данных содержит одну таблицу, структура таблицы представлена на рисунке.

Используя технологию ADO, разработать приложение, которое будет работать с этой базой.

Порядок выполнения

1. Создать новый проект, с вкладки **dbGo** поместить на форму компонент **TADOCnnection**;
2. Настроить соединение с сервером:

- Дважды щелкнуть по компоненту **ADOConnection**
- В окне выбрать переключатель **Use Connection String** и щелкнуть **Build** (вызов мастера)
- На вкладке **Поставщик данных** выбрать драйвер доступа к данным **Microsoft Office 12.0 Access Database Engine OLE DB Provider** и нажать **Далее**;
- На вкладке **Подключение** в строке **Выберите или введите имя базы данных** введите имя **db_Telefon.accdb**

Примечание : *если база и исполняемый файл будут хранится в одной папке, путь указывать необязательно*

- Щелкнуть по кнопке **Проверить подключение**. Если все указано верно, то выводится сообщение. Нажмите **ОК**.
- Закройте окно создания строки подключения **Ок**, закрыть редактор строки подключения **ОК**.

3. Продолжить настройку приложения:

- Для компонента **ADOConnection** отключить свойство **LoginPromt – false**(для того, чтобы при обращении к базе не запрашивался пароль)
- Для свойства **Connected** задать значение **True**, чтобы произошло соединение с базой

4. Для получения доступа к таблице базы:

- Установить на форме компонент **ADOTable**, изменить имя на **BookTable**
- Для свойства **Connection** из списка выбрать **ADOConnection1**
- В свойстве **TableName** выбрать имя таблицы
- Для свойства **Active** установить значение **True**

5. Для отображения данных из таблицы :

- Установить на форму компонент **DataSource** (вкладка Data Access),для свойства **DataSet** выбрать **BookTable** (указали какую таблицу отображать)
- Установит компонент **DBGrid** (вкладка Data Controls), для свойства **DataSource** Выбрать **DataSource1**.

6. Приложение готово. Запустите его, добавьте новые строки, отредактируйте существующие строки, удалите что-нибудь (**Ctrl/Del**). Для вставки строки используйте **Ins**.

Управление отображением данных

7. Модифицировать таблицу в базе данных, добавив еще два поля **Дата** и **Мобильник** (файл взять у преподавателя).
8. Перенести компоненты доступа к базе в специальное окно:

- Выделить компоненты **ADOConnection1, DataSource1 и BookTable** и вырезать в буфер обмена **(Edit /Cut)**
- Выбрать в меню **File/New/Other,** выбрать категорию **Delphi Files,** выбрать **Data Module,** выполнить **Edit/Paste**
- Сохранить новый модуль под именем **DataModuleUnit**

9. Отобразить на экране менеджер проектов **New / Project Manager**
10. Вернитесь на главную форму, обратите внимание, что данных в сетке нет, т.к. потеряна связь с компонентами доступа к базе. Восстановим связь:

- Выбрать в меню **File / Use Unit**, выбрать в окне **DataModuleUnit, Ок.**
- Проверьте теперь в редакторе кода после слова **Implementation** появилась запись **uses DataModuleUnit**
- Для компонента **DBGrid1** изменить свойство **DataSource**, выбрав **DataSource1.**

11. Перейти в модуль **DataModuleUnit** и настроить отображение данных:

- Дважды щелкнуть по **BookTable,** отобразиться окно для редактирования полей базы, пока оно пустое
- Щелкните в окне правой кнопкой мыши и выберите **Add All Field**
- Выделить поле **Key1** и убрать у него видимость, перейдите на главную форму, убедитесь, что поле не отображается
- Отредактировать ширину колонок: выделить поле **Фамилия**, свойству **DisplayWidth** дать значение 15, сократите ширину для поля **Имя**.
- Выделите поле **Дата**, для свойства **DisplayLabel** укажите значение **Дата рождения** (имя поля не меняется, только подпись), для свойства **DisplayFormat** указать **dddddd**, для свойства **EditMask** указать маску ввода **99/99/99**.
- Выделите поле Мобильник, для свойства **DisplayValues** указать значение **Да;Нет**

12. Просмотрите, как изменилось отображение данных**.**

Поисковые поля

13. В поле **Город** отображается числовое значение (номер города), а пользователю нужно показывать название города. Создадим отдельную таблицу **Справочник городов** в нашей базе, включив в нее поля: **Key1** (счетчик, ключевое поле) и **Название города** (текстовое поле размер 30). Базу данных взять у преподавателя.

14. В модуль **DataModuleUnit** добавитькомпонент **DataSource** (назвать **TownSource**) и **ADOTable** (назвать **TownTable**), у компонента **TownSource** в свойстве **DataSet** указать **TownTable**.

15. Настроить отображение справочника городов:

- В свойстве **Connection** указать компонент **ADOConnection1**
- В свойстве **TableName** указать таблицу **Справочник городов**
- Для свойства **Active** значение **True**
- Дважды щелкнуть по **TownTable,** добавить все поля**,** поле **Key1** сделать невидимым.

16. Создать новую форму (name – **TownBookForm**)для редактирования справочника, сохранить ее в модуле под именем **TownBookUntil**

17. Подключить к новой форме модуль **DataModuleUnit**, выполнив **File / Use Unit**, выбрать **DataModuleUnit**, **Ок**

18. Добавить на форму сетку **DBGrid**, в свойстве **DataSource** указать таблицу **Справочник городов** – **DataModule1.TownSource**.

19. Доработать интерфейс формы Справочник городов:

- Добавить меню

Редактирование | Назад

сохранить

добавить

удалить

- Для пункта **Добавить** ввести код

```
DataModule1->TownTable->Insert();
dbGrid1->SetFocus();
```

- Для пункта **Сохранить** ввести код

```
if (DataModule1->TownTable->Modified())
   DataModule1->TownTable->Post();
```

- Для пункта **Удалить** ввести код

```
DataModule->TownTable->Delete();
```

20. Перейти на главную форму, создать меню:

Файл | Редактирование | Справочники | Выход

Справочник городов

21. Для пункта Справочник городов ввести код:

```
TownBookForm->ShowModal();
```

22. Загрузить программу, вызвать **Справочник городов** и добавить несколько строк. Закрыть программу.

23. Для компонента **DBGrid1** на главной форме для свойства **Options / dgEditing** установить значение **False** (редактирование данных запрещено).

24. В пункт меню **Редактирование** добавить подпункты: **Добавить запись, Редактировать запись, Удалить запись**

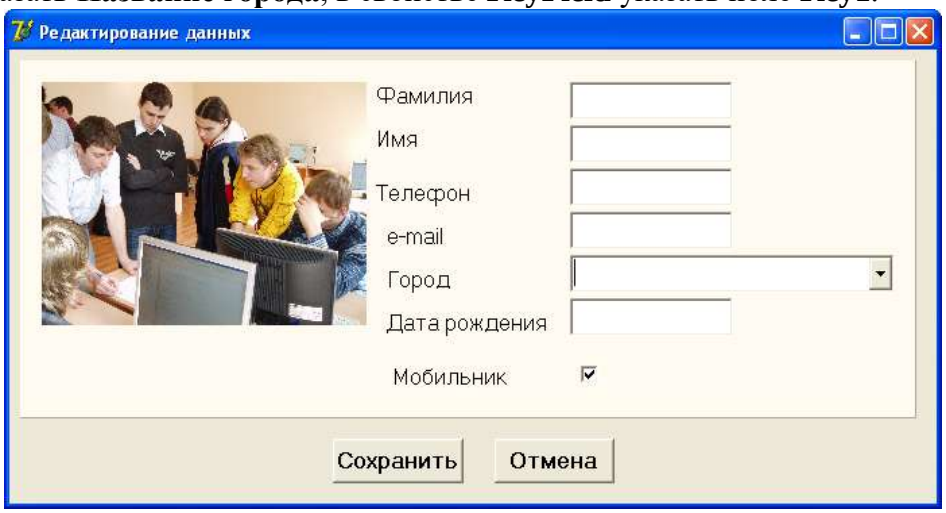
25. Создать панель с кнопками для быстрого доступа к пунктам меню

26. Создать новую форму для редактирования каждой записи, изменить свойства:
Name - EditFormUnit,
BorderStyle – bsSingle,
Position – poMainFormCenter
форму сохранить по именем **EditFormUnit**.

27. Подключить к новой форме модуль с данными

28. Привести форму в соответствие с образцом:

- Поля для ввода брать на вкладке **Data Control**.
- Чтобы компонент видел данные из нужного поля, указать у него в свойстве **DataSource** нужную таблицу (**DataModule1->DataSource1**, также как это делали с сеткой редактирования), в свойстве **DataField** указать поле, которое надо редактировать (самостоятельно).
- Для выбора города добавить компонент **DBLookupComboBox**, для свойства **DataSource** указать основную таблицу **DataModule1->DataSource1,** в свойстве **DataField** указать поле **Город**, в свойстве **ListSource** указать **DataModule1->TownSource**, свойстве **ListField** указать **Название города**, в свойстве **KeyField** указать поле **Key1**.



29. Для кнопки **Сохранить** ввести код:

```
if (DataModule1->BookTable->Modified())
        DataModule1->BookTable->Post();
```

30. Для кнопки **Отмена** ввести код:

```
DataModule1->BookTable->Cancel();
```

31. Перейти на главную форму и для пункта меню **Добавить запись** ввести код:

```
DataModule1->BookTable->Insert();
EditRecordForm->ShowModal();
```

32. Для пункта меню **Редактировать запись** ввести код:

```
EditRecordForm->ShowModal();
```

33. Загрузить программу, создать новую запись, в поле **Город** выбрав какое-нибудь значение из справочника, нажмите **Сохранить**. Просмотрите сетку.

34. Результат показать преподавателю

35. Для пункта меню **Удалить запись** ввести код:

```
if (Application->MessageBox((“Вы действительно хотите удалить”+DataModule1-
>BookTableDSDesigner->AsString), “Внимание!!!”, MB_OkCancel)=id_Ok )
```

```
DataModule1->BookTable->Delete();
```

Примечание:

DataModule1 имя модуля

BookTableDSDesigner имя поля с фамилией (*дважды щелкнуть по BookTable и смотреть свойство поля Фамилия, если имя отличается, скорректировать код*)

AsString метод, возвращающий значение в виде строки

36. В сетке на главной форме город отображается в виде индекса строки в справочнике городов, исправим недостаток, для этого:

- Перейти в модуль **DataModule1**, выделить компонент
- Сделать его неактивным (Active False)и дважды по нему щелкнуть
- Создать новое поле: щелкнуть в окне, выбрать **New Field**
- В окне заполнить поля следующим образом:
  Name – Town,
  Type – String,
  FieldType – Lookup (поисковое поле),
  KeyField – Город,
  DataSet – TownTable (таблица, где нужно искать)
  Lookup Keys - Key1 (по этому полю надо искать),
  Result Field – Название города, нажать **Ок**
- Появилось новое поле, перетащить его ближе к полю **Город**
- Сделать таблицу **BookTable** вновь активной загрузить программу, просмотреть результат.

37. Сделать поле **Город** невидимым, чтобы не видеть числа, а над полем **Town** написать надпись **Город** (самостоятельно)

Сортировка и фильтрация данных

Поиск данных и сортировка выполняются по индексным полям. В телефонном справочнике чаще ищут информацию по номеру телефона или фамилии. В таблице **Справочник** для полей **Фамилия и Телефон** свойство **Индексированное поле** имеет значение **Да(допускаются совпадения**).

38. Добавить в меню пункт **Сортировка** и подпункты **По телефону, По Фамилии**

39. для подпунктов ввести соответствующий код:

```
DataModule1->BookTable->IndexFieldMNames=”Фамилия”;
```

```
DataModule1->BookTable->IndexFieldMNames=”Телефон”;
```

40. Реализовать функцию поиска данных, для этого:

- добавить на форму панель, на нее метку **Найти** и **TEdit** с именем **FindEdit**
- ввести следующий код:

```
if (FindEdit->Text->Length()>0)
    DataModule1->BookTable->Filtered()=true;
Else
    DataModule1->BookTable->Filtered()=false;
DataModule1->BookTable->Filter=”Фамилия>’ “+FindEdit->Text+” ‘ ”;
```

41. Проверить работу программы.

Контрольные вопросы:

1. Назначение технологии ADO.
2. Назначение компонента TADOConnection.
3. Назначение компонента TDataSource.
4. Чем отличается TADOTable от TADOQuery.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 39 База данных «Ежедневник»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder.

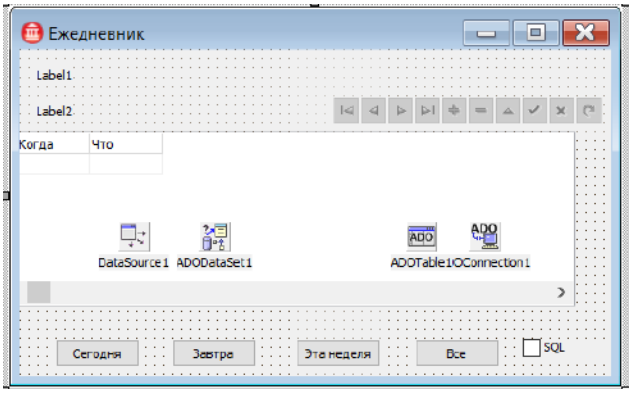
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Программа Ежедневник демонстрирует использованием компонентов ADO для доступа к базе данных формата Microsoft Access.

База данных содержит информацию о запланированных мероприятиях (дата, задача). Программа позволяет вносить в базу данных изменения (добавлять, удалять и редактировать записи), а также обеспечивает выбор информации по запросу – выводит список мероприятий, запланированных «на сегодня», «на завтра» и «на эту неделю». При запуске программа автоматически выводит список мероприятий, запланированных «на сегодня» или, если программа запущена в пятницу, субботу или воскресенье, «на сегодня и ближайшие дни».



База данных «Ежедневник» (Planner.accdb) состоит и з одной таблицы schedule:
aDate (Дата/Время) – дата
aTask (Строковый, 50 символов) – запланированное мероприятие (задача).
Значение свойств компонента ADODataSet1:
Connection=ADOConnection1
CommandText SELECT * FROM schedule ORDER BY aDate;
Значение свойств компонента DataGrid1:
DataSource=DataSource1
Columns[0].FieldName aDate
Columns[0].Title.Caption Когда
Columns[1].FieldName aTask
Columns[1].Title.Caption Что

Листинг кода:

```
#include <DateUtils.hpp>
#include <ComObj.hpp>                    //для доступа к EOleException
AnsiString stDay[7] = {"воскресенье","понедельник","вторник","среда","четверг","пятница","суббота"} ;
AnsiString stMonth[12] = {"января","февраля","марта","апреля","мая","июня","июля","августа","сентября",
                          "октября", "ноября","декабря"};
void __fastcall TForm1::FormShow(TObject *Sender)
{
    TDateTime Today, NextDay; // следующий день (не обязательно завтра)
    Word  Year, Month, Day;     // год, месяц, день
    Today = Now();
    DecodeDate(Today, Year, Month, Day);
    Label1->Caption = "Сегодня " + IntToStr(Day) + " " + stMonth[Month-1] + " " + IntToStr(Year) + " года, " + stDay[DayOfWeek(Today) -1];
    Label2->Caption = "Сегодня и ближайшие дни";
/* вычислим следующий день, если сегодня пятница, то,
чтобы не забыть, что запланировано на понедельник,
считаем, что следующий день - понедельник */
    switch  ( DayOfWeek(Today) ) {
    case 6  : NextDay = Today + 3; break; // сегодня пятница
    case 7  : NextDay = Today + 2; break; // сегодня суббота
    default : NextDay = Today + 1; break;
    }
    ADODataSet1->CommandText="SELECT * FROM Planner WHERE Когда BETWEEN DateValue('" + FormatDateTime("dd/mm/yyyy", Today) +
    "')AND DateValue('"+ FormatDateTime("dd/mm/yyyy", NextDay)+ "') ORDER BY Когда";
    //если надо отразить SQL-команду
    if(CheckBox1->Checked) ShowSQL();
    try
    {
        ADODataSet1->Open();
    }
    catch(EOleException &e)
    {
        ShowMessage("Ошибка обращения к БД. БД PLanner.accdb должна"
                    "быть зарегистрирована\nв системе как источник данных ODBC"
                    "под именем dplaner");
```

```
                Button1->Enabled=false;
                Button2->Enabled=false;
                Button3->Enabled=false;
                Button4->Enabled=false;
                return;
            }
            if(!ADODataSet1->RecordCount)
            ShowMessage("На сегодня и ближайшие дни ни каких дел не запланировано.");
        }
        //щелчок на кнопке Сегодня
        void __fastcall TForm1::Button1Click(TObject *Sender)
        {
          AnsiString today=FormatDateTime("dd/mm/yyyy",Now());
          Form1->Label2->Caption="Сегодня";
          ADODataSet1->Close();
          ADODataSet1->CommandText="SELECT * FROM Planner WHERE Когда = DataValue('" + today+"')";
          ADODataSet1->Open();
        }
        //щелчок на кнопке Завтра
        void __fastcall TForm1::Button2Click(TObject *Sender)
        {
           AnsiString tomorrow=FormatDateTime("dd/mm/yyyy",Now()+1);
          Form1->Label2->Caption="Завтра";
          ADODataSet1->Close();
          ADODataSet1->CommandText="SELECT * FROM Planner WHERE Когда = DataValue('" + tomorrow +"')";
          ADODataSet1->Open();
          if(!ADODataSet1->RecordCount)
                {
                ShowMessage("На сегодня и ближайшие дни ни каких дел не запланировано.");
                }
        }
        //щелчок на кнопке На этой неделе
        void __fastcall TForm1::Button3Click(TObject *Sender)
        {
        TDateTime Present, eWeek;
        Label2->Caption = "На этой неделе";
        Present=Now();  // Now - возвращает текущую дату
        eWeek = EndOfAWeek(YearOf(Present),WeekOf(Present));
        ADODataSet1->Close();
        ADODataSet1->CommandText =
        "SELECT * FROM Planner WHERE aDate BETWEEN DateValue('" +
        FormatDateTime("dd/mm/yyyy", Present) + "') AND DateValuef (' " +
        FormatDateTime("dd/mm/yyyy",eWeek)+"') ORDER BY aDate";
        if  ( CheckBox1->Checked) ShowSQL();
        ADODataSet1->Open();
        if(!ADODataSet1->RecordCount)
                {
                ShowMessage("На сегодня и ближайшие дни ни каких дел не запланировано.");
                }}
```

```
//Щелчок на кнопке Все
void __fastcall TForm1::Button4Click(TObject *Sender)
{
  ADODataSet1->Close();
ADODataSet1->CommandText =
"SELECT * FROM Planner ORDER BY aDate";
if  ( CheckBox1->Checked) ShowSQL();
ADODataSet1->Open();
Label2->Caption - "Все, что намечено сделать";
}
//отображает SQL-rjvfyle
void __fastcall TForm1::ShowSQL(void)
{
        ShowMessage(ADODataSet1->CommandText);
}
```

Контрольные вопросы:

1. Назначение технологии ADO.
2. Назначение компонента TADOConnection.
3. Назначение компонента TDataSource.
4. Чем отличается TADOTable от TADOQuery.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, допущено более трех ошибок, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 40 Книга «Доходов и расходов»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Программа Ежедневник демонстрирует использованием компонентов ADO для доступа к базе данных формата Microsoft Access.

Задания для самостоятельной работы:

1. Создать базу данных «Книга доходов и расходов» в Microsoft Access.
2. Подключить созданную базу данных к приложению C++ Builder по технологии ADO.
3. Спроектировать графический интерфейс (GUI) для работы с базой данных «Книга доходов и расходов».
4. Организовать навигацию по базе данных.
5. Организовать поиск и фильтрацию информации в базе данных.
6. Отладить и протестировать приложение «Книга доходов и расходов».

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 41 Программа подключения к базе данных с проверкой логина и пароля, база данных «Отдел кадров»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Программа Отдел кадров демонстрирует использованием компонентов ADO для доступа к базе данных формата Microsoft Access с возможностью авторизации пользователей.

Задания для самостоятельной работы:

1. Создать базу данных «Отдел кадров» в Microsoft Access (не меньше двух таблиц).
2. Подключить созданную базу данных к приложению C++ Builder по технологии ADO.
3. Спроектировать графический интерфейс (GUI) для работы с базой данных «Отдел кадров».
4. Добавить авторизацию пользователей.
5. Организовать корректное отображение и работу связанных таблиц (master-detail).
6. Организовать навигацию по базе данных.
7. Организовать поиск и фильтрацию информации в базе данных.
8. Отладить и протестировать приложение «Отдел кадров».

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 42 База данных «Библиотека»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

орядок выполнения:

Программа Библиотека демонстрирует использованием компонентов ADO для доступа к базе данных формата Microsoft Access с возможностью запросов на языке SQL.

Задания для самостоятельной работы:

1. Создать базу данных «Библиотека» в Microsoft Access (не меньше трех таблиц).
2. Подключить созданную базу данных к приложению C++ Builder по технологии ADO.
3. Для организации работы с наборами данных использовать компоненты-запросы (ADOQuery).
4. Спроектировать графический интерфейс (GUI) для работы с базой данных «Библиотека».
5. Организовать корректное отображение и работу связанных таблиц на основе запросов SQL.
6. Организовать навигацию по базе данных с использованием запросов SQL.
7. Организовать поиск и фильтрацию информации в базе данных с использованием запросов SQL.
8. Отладить и протестировать приложение «Библиотека».

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 43 База данных «Клиент-сервер»

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания клиент-серверного приложения для базы данных C++ Builder с использованием технологии InterBase

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Порядок выполнения:

Программа демонстрирует использование компонентов InterBase для доступа к базе данных формата FireBird с возможностью запросов на языке SQL.

Задания для самостоятельной работы:

1. Создать базу данных «Клиент-сервер» с использованием утилиты IBExpert (СУБД Firebird) из двух таблиц Клиент и Сервер.

2. Создать два приложения: для сервера и клиента.

3. Подключить созданную базу данных к приложениям C++ Builder по технологии InterBase.

4. Спроектировать графический интерфейс (GUI) для приложений «Сервер» и «Клиент».

5. Отладить и протестировать приложение.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 44 Выполнение комплексного задания заключительного этапа олимпиады профессионального мастерства обучающихся по специальности 09.02.03 Программирование в компьютерных системах за 2018 год: Разработка модуля демо-версии настольного приложения банка для открытия вкладов в автоматическом режиме под операционную систему Windows

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения для базы данных C++ Builder

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Задания для самостоятельной работы:

1. Создать базу данных (не меньше трех таблиц).

2. Подключить созданную базу данных к приложению C++ Builder.

4. Спроектировать графический интерфейс (GUI) для работы с базой данных.
5. Организовать корректное отображение и работу связанных таблиц.
6. Организовать навигацию по базе данных.
7. Организовать поиск и фильтрацию информации в базе данных.
8. Отладить и протестировать приложение.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 45 Программирование под Интернет

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения под Интернет в C++ Builder

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи

Разработать приложение, реализующее основные функции WEB-браузера.

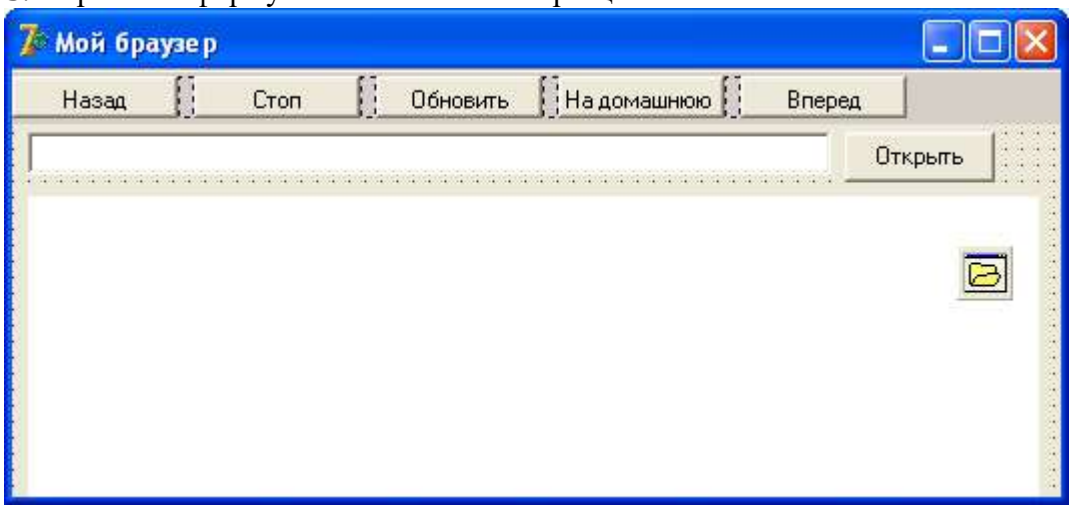
Порядок выполнения

1. Создать новый проект и сохранить его.
2. Добавить на форму компонент **ToolBar** (Win32), изменить значение свойств: **Height**= 28, **ShowCaptions**=True, **Flat**= True
3. Добавить на **ToolBar** 5 кнопок: Назад, Обновить, Стоп, Домашняя страница, Вперед, для этого повторить 5 раз следующую последовательность:

- щелкнуть правой кнопкой мыши на компоненте **ToolBar;**
- выбрать **New Button**;
- изменить размеры кнопки **Width** = 67,**Height** =21;
- вставить разделитель кнопок, щелкнув правой кнопкой на **ToolBar** и выбрав **New Separator;**

- изменить свойства **Caption** для кнопки.

4. Добавить на форму компоненты Edit, командную кнопку, OpenDialogs и WebBrowser.
5. Привести форму в соответствие с образцом.



6. Для того, чтобы компоненты размещались по местам после изменения размера формы, для процедуры **FormResize** ввести код:

```
Edit1->Left := 0;
Edit1->Top = ToolBar1->Height + 2;
Button1.Top = Edit1->Top;
Button1->Left = Form1->ClientWidth - Button1->Width - 2;
WebBrowser1->Left = 0;
WebBrowser1->Top = Edit1->Top + Edit1->Height + 2;
//меняем размеры некоторых компонентов
Edit1->Width = Form1->ClientWidth - Button1->Width - 4;
Button1->Height = Edit1->Height;
WebBrowser1->Width = Form1->ClientWidth;
WebBrowser1->Height = Form1->ClientHeight -  (Edit1->Top + Edit1->Height + 2);
```

7. Для кнопки **Открыть** ввести код:

```
if (OpenDialog1->Execute()) {
  WebBrowser1->Navigate(OpenDialog1->FileName);
  Edit1->Text = OpenDialog1->FileName;
  WebBrowser1->Navigate(Edit1->Text); //загрузка документа
  Edit1->SetFocus();   }
```

8. Загрузить приложение, проверить работу кнопки Открыть.
9. Для кнопок панели ввести соответствующие коды:

```
WebBrowser1->GoBack(); //назад
WebBrowser1->Stop(); //остановить загрузку
WebBrowser1->Refresh() ; //обновить страницу
WebBrowser1->GoHome(); //на домашнюю страницу
WebBrowser1->GoForward(); //вперед
```

10. Проверить работу приложения.
11. Добавить на форму компонент **StatusBar** (Win32)и изменить значение его свойства **SimplePanel**= True.
12. Для компонента **WebBrowser1** выбрать событие **OnStatusTextChange** и ввести код:

```
StatusBar1->SimpleText = Text;
```

13. Добавить на форму компонент **ProgressBar** (Win32)и изменить значение его свойства **Align**=alButtom.
14. Для компонента **WebBrowser1** выбрать событие **OnProgressChange** и ввести код:

ProgressBar1->Max = ProgressMax;
ProgressBar1->Position = Progress;

15. Проверить работу приложения.
16. Модернизировать приложение для реализации следующих функций:

- при изменении размера формы компоненты **StatusBar и ProgressBar** должны всегда отображаться на экране.
- кнопки **Назад** и **Вперед** после загрузки приложения должны быть недоступны. Кнопка **Назад** получает доступ только после выбора пользователем гиперссылки в окне браузера. Кнопка **Вперед** получает доступ после щелчка по кнопке **Назад**

Контрольные вопросы:

1. Как называется компонент для создания браузера?
2. Перечислить основные свойства и методы компонента из п.1.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 46 Разработка простейшего мобильного приложения

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания мобильных приложений в C++ Builder

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: C++ Builder (Embarcadero RAD Studio XE2)

Постановка задачи

Разработать простейшее мобильное приложение.

Задания для самостоятельной работы:

С помощью Embarcadero FireMonkey создать простое мобильное приложение под Андроид.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 47 Исследование возможностей среды разработки программ Microsoft Visual Studio

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

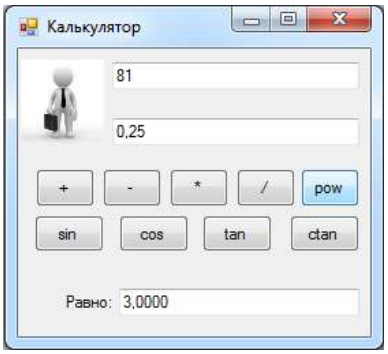
Программное обеспечение: VisualStudio 2019

Постановка задачи

Приобретение практических навыков работы с приложениями Windows Form на языке C++ в Visual Studio 2019

Прядок выполнения заданий

Создать приложение:



Контрольные вопросы

1. Что общеязыковая среда CLR.
2. Назовите элементы управления, использованные при создании приложения.
3. Использовали ли вы библиотеку math.h

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 48 Реализация простейшей программы на VC++, с использованием графического интерфейса

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

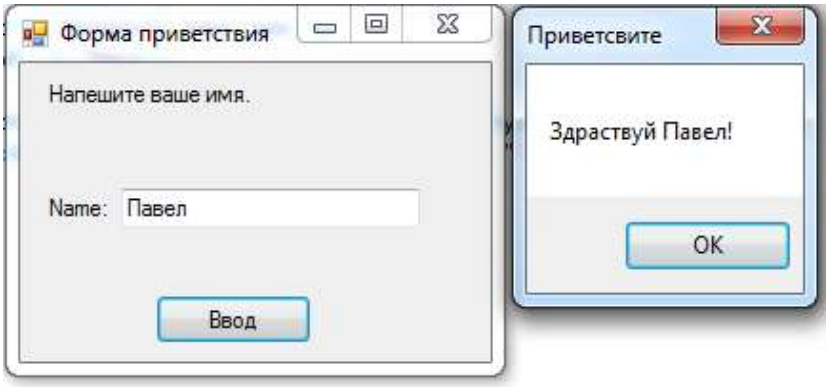
Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать приложение:



Контрольные вопросы

1. Назовите элементы управления, использованные при создании приложения.
2. MessageBox

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 49 Создание графического интерфейса с использованием переключателей, флажков. Реализация разветвляющих алгоритмов

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

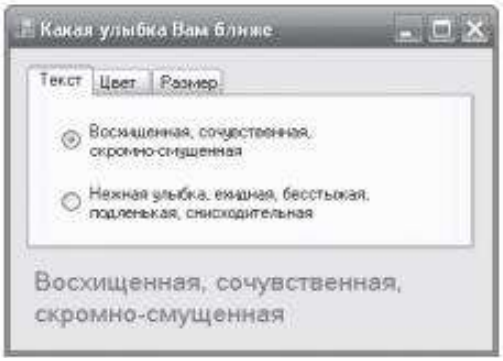
Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Разработать программу, позволяющую выбрать текст из двух вариантов, задать цвет и размер шрифта этого текста на трех вкладках TabControl с использованием переключателей RadioButton.



Контрольные вопросы

1. Опишите, как добавить в программу вкладки.
2. По какому принципу работают переключатели RadioButton.
3. Опишите, как организовать работу группы переключателей.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 50 Графический интерфейс с использованием списков, комбинированныйх полей. Решение задач, использующих изучаемые элементыграфического элемента, массивы и структуры данных

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать приложение:

В гараже имеется 4 различных автомобиля (ВАЗ, Газель, ГАЗ-66, Мерседес), каждый автомобиль имеет свой расход топлива на 100 км пути, а также свою стоимость топлива.

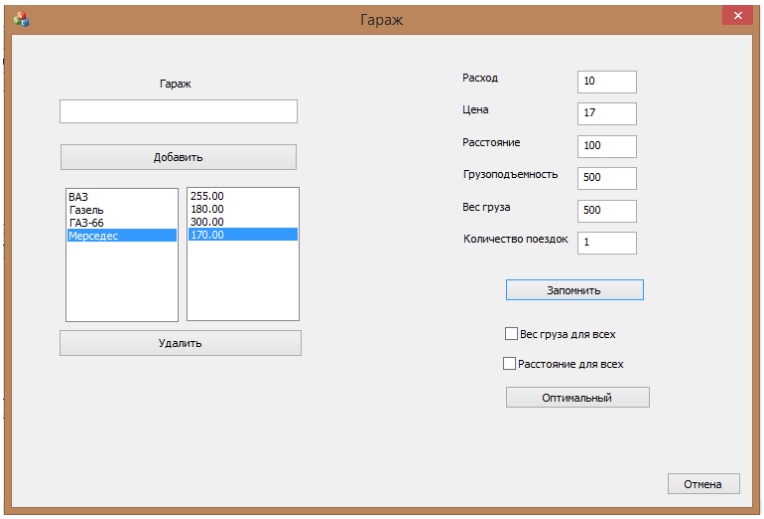


Рисунок 1 – Приложение

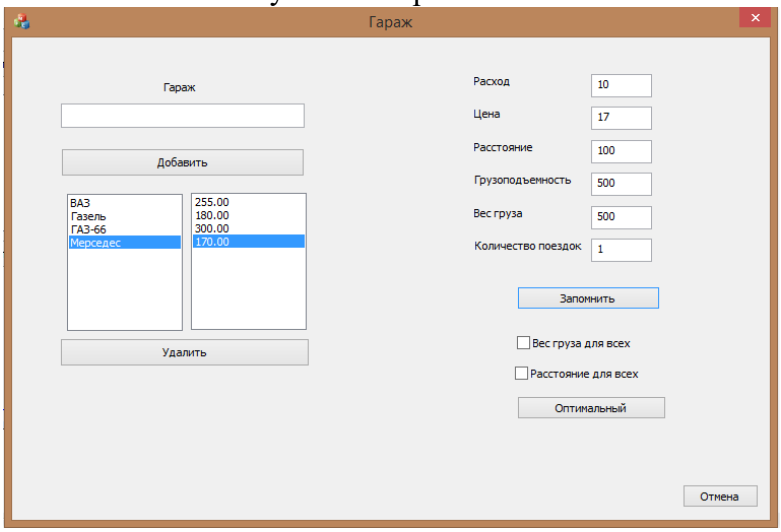


Рисунок 2 – Добавление записи

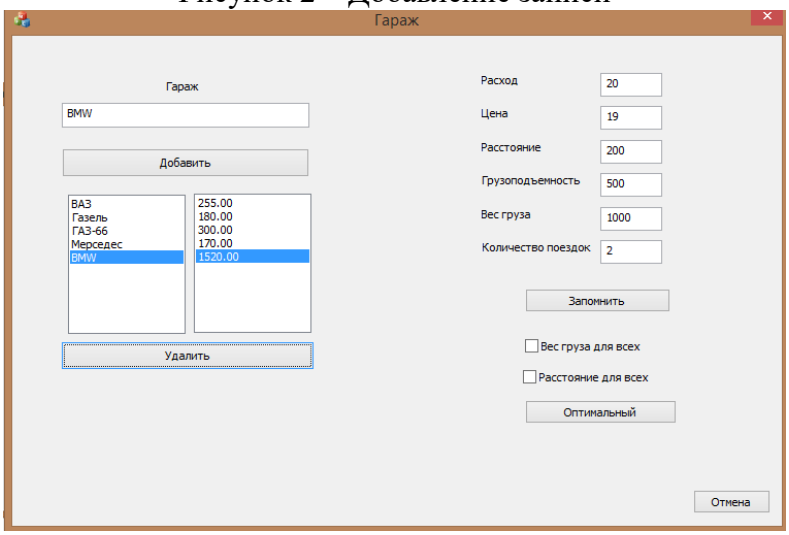


Рисунок 3 – Удаление записи

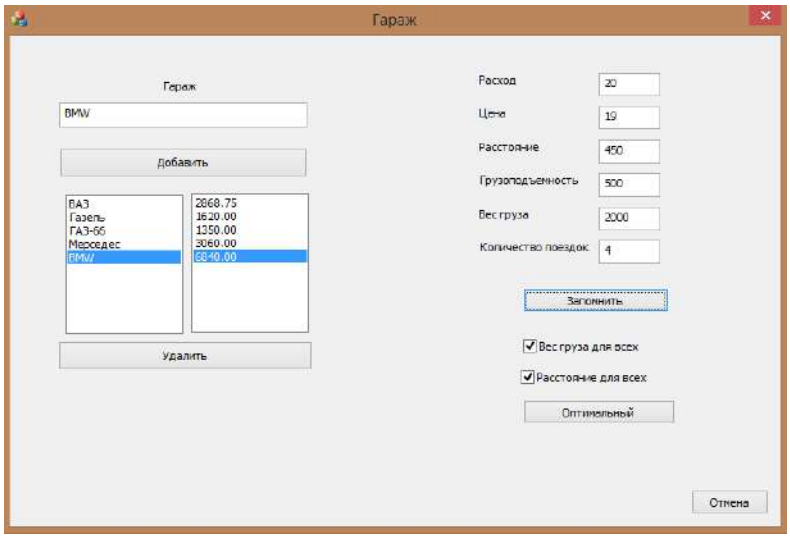


Рисунок 4 – Параметр для всех

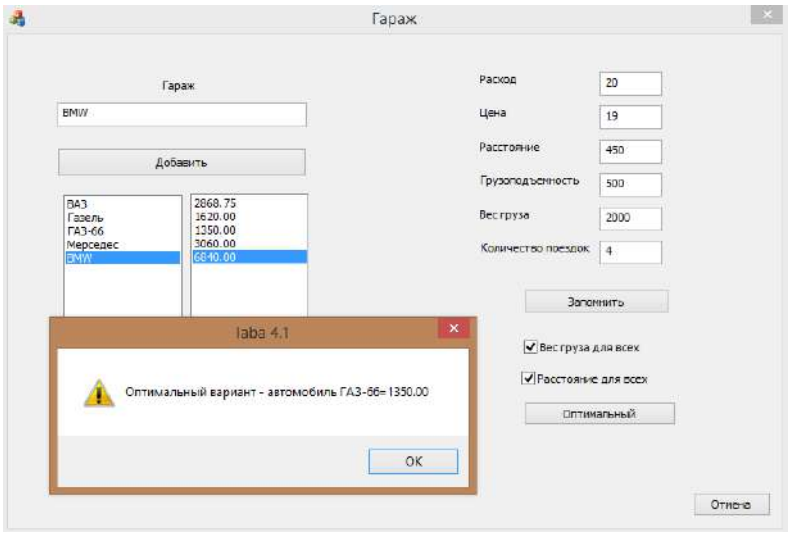


Рисунок 5 – Оптимальный вариант

Листинг программы:

```
struct Garage
{
float Litres;
float Price;
float Way;
int Tonnage;
int Weight;
};

Garage car[10];
float Result1;
int NumCars;

BOOL Claba41Dlg::OnInitDialog()
{
CDialogEx::OnInitDialog();

Garage.AddString(L"ВАЗ");
Garage.AddString(L"Газель");
Garage.AddString(L"ГАЗ-66");
Garage.AddString(L"Мерседес");
```

```
NumCars = 4;
Garage.SetCurSel(0);
Result.SetCurSel(0);
Lit = car[0].Litres;
Price = car[0].Price;
ras = car[0].Way;
gruz= car[0].Tonnage;
ves_gruz = car[0].Weight;
kol = 2;
UpdateData(0);
Result1 = (ras / 100)*Lit*Price*kol;
CString s;
s.Format(L"%0.2f",Result1);
Result.AddString(s);
Result.AddString(L"?");
Result.AddString(L"?");
Result.AddString(L"?");

void Claba41Dlg::OnLbnSelchangeList2()
{
int x = Garage.GetCurSel();
Lit = car[x].Litres;
Price = car[x].Price;
ras = car[x].Way;
gruz = car[x].Tonnage;
ves_gruz = car[x].Weight;
float z;
if (gruz 0)
{
kol = int(ves_gruz / gruz);
z = float(ves_gruz) / float(gruz);
if (kol
}
else kol = 1;
Result1 = (ras / 100)*Lit*Price*kol;
CString s;
s.Format(L"%0.2f", Result1);
Result.DeleteString(x);
Result.InsertString(x, s);
Result.SetCurSel(x);
UpdateData(0);
}

void Claba41Dlg::OnBnClickedButton1()
{
UpdateData(1);
if (Add != "" && NumCars
{
Garage.AddString(Add);
car[NumCars].Litres=Lit;
car[NumCars].Price = Price;
car[NumCars].Way = ras;
car[NumCars].Tonnage = gruz;
car[NumCars].Weight = ves_gruz;
```

```
float z;
if (gruz 0)
{
kol= int(ves_gruz / gruz);
z = float(ves_gruz) / float(gruz);
if (kol
}
else kol = 1;
Result1 = (ras / 100)*Lit*Price*kol;
CString s;
s.Format(L"%0.2f", Result1);
Result.AddString(s);
Garage.SetCurSel(NumCars);
Result.SetCurSel(NumCars);
NumCars++;
}
}

void Claba41Dlg::OnBnClickedButton2()
{
int x = Garage.GetCurSel();
for (int i = 0; i = car[i];
if (NumCars 0) NumCars--;
for (int i = 0; i
if (i = Mas[i];
else car[i] = Mas[i + 1];
Garage.DeleteString(x);
Result.DeleteString(x);
if (NumCars 0)
{
Garage.SetCurSel(NumCars - 1);
Result.SetCurSel(NumCars - 1);
Claba41Dlg::OnLbnSelchangeList2();
};
}

void Claba41Dlg::OnBnClickedOk()
{
int x =Garage.GetCurSel();
UpdateData(1);
car[x].Litres = Lit;
car[x].Price = Price; car[x].Tonnage = gruz;
if (Ways == 0) car[x].Way = ras; else
for (int i = 0; i if (Weights == 0) car[x].Weight = ves_gruz;
else
for (int i = 0; i

for (int i = 0; i
{
Garage.SetCurSel(i);
Claba41Dlg::OnLbnSelchangeList2();
UpdateData(0);
}
Garage.SetCurSel(x);
```

```
Claba41Dlg::OnLbnSelchangeList2();
}

void Claba41Dlg::OnBnClickedButton3()
{
for (int i = 0; i
{
Garage.SetCurSel(i);
//Claba41Dlg::OnLbnSelchangeList2();
UpdateData(0);
}
if (NumCars 0)
{
CString st;
CString st2;
Result.GetText(0, st);
Result1 = _wtof(st);
int j = 0;
float n;
for (int i = 0; i
{
Result.GetText(i, st);
n = _wtof(st);
if (n
{
Result1 = n;
j = i;
}
}
st2.Format(L"%0.2f", Result1);
Garage.GetText(j, st);
st =L"Оптимальный вариант - автомобиль " + st +"=" + st2;
AfxMessageBox(st);
```

Контрольные вопросы:

1. Опишите работу со структурами на языке С++
2. Опишите работу со массивами на языке С++

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 51 Ипользование файлов для хранения данных

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Напишем программу, содержащую на экранной фор текстовое поле и две командные кнопки. При щелчке мышью первой кнопке происходит чтение текстового файла в текстов поле в кодировке Unicode. При щелчке на второй кноп отредактированный пользователем текст в текстовом по сохраняется в файл на диске.

Листинг:

```
// ..............
// Программный код, расположенный выше, создан средой V
Studio
// автоматически, поэтому автором не приводится
this->ResumeLayout(false);
this->PerformLayout();
}
#pragma endregion
// Программа для чтения/записи текстового файла в коди
Unicode
String ^ filename;
// Объявляем filename здесь, чтобы эта переменная была "видн
// в процедурах обработки событий.
private: System::Void Form1_Load(System::Object^ sender,
System::EventArgs^ e)
{
// Установка начальных значений:
textBox1->Multiline = true; textBox1->Clear();
textBox1->Size = Drawing::Size(268, 112);
button1->Text = "Открыть"; button1->TabIndex = 0;
button2->Text = "Сохранить";



Form1::Text = "Здесь кодировка Unicode";
filename = "C:\\Text1.txt";
}
private: System::Void button1_Click(System::Object^ sende
System::EventArgs^ e)
{
// Щелчок на кнопке Открыть.
// Русские буквы будут корректно читаться,
// если открыть файл в кодировке UNICODE:
try
{
// Создание объекта StreamReader для чтения из файла:
auto Читатель = gcnew IO::StreamReader(filename);
// Непосредственное чтение всего файла в текстовое поле:
textBox1->Text = Читатель->ReadToEnd();
Читатель->Close(); // закрытие файла
// Читать текстовый файл в кодировке UNICODE в массив стр
// можно также таким образом (без Open и Close):
// array <String^>^ МассивСтрок =
// IO::File::ReadAllLines("C:\\Text1.txt");
}
catch (IO::FileNotFoundException^ Ситуация)
{ // Обработка исключительной ситуации:
MessageBox::Show(Ситуация->Message + «\nНет такого файла»,
"Ошибка", MessageBoxButtons::OK,
MessageBoxIcon::Exclamation);
}
```

```
catch (Exception^ Ситуация)
{
// Отчет о других ошибках:
MessageBox::Show(Ситуация->Message, "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
}
}
private: System::Void button2_Click(System::Object^ sender,
System::EventArgs^ e)
{
// Щелчок на кнопке Сохранить:
try
{
// Создание объекта StreamWriter для записи в файл:
auto Писатель = gcnew
IO::StreamWriter(filename, false);
Писатель->Write(textBox1->Text);
Писатель->Close();
// Сохранить текстовый файл можно также таким образом
// (без Close), причем, если файл уже существует,
// то он будет заменен:
// IO::File::WriteAllText("C:\\tmp.tmp", textBox1->Text);
}
catch (Exception^ Ситуация)
{
// Отчет обо всех возможных ошибках:



MessageBox::Show(Ситуация->Message, "Ошибка",
MessageBoxButtons::OK, MessageBoxIcon::Exclamation);
}
}
};
}
```

Блоки try, которые, как мы видим, используются в данном программном коде. Логика использования try следующая: *попытаться* (try) выполнить некоторую задачу, например прочитать файл. Если задача решена некорректно (например, файл не найден), то *«перехватить»* (catch) управление и *обработать* возникшую (*исключительную*, Exception) ситуацию.

При обработке события «щелчок на кнопке Открыть» организован ввод файла C:\Text1.txt. Обычно в этой ситуации пользуются элементом управления OpenFileDialog для выбора файла. Мы не стали использовать этот элемент управления для того, чтобы *не «заговорить»* проблему, а также свести к минимуму программный код.

Далее создаем объект (поток) Читатель для чтения из файла. Для большей выразительности операций с данным объектом мы назвали его русскими буквами.

При обработке события «щелчок на кнопке Сохранить» организована запись файла на диск аналогично через объект Писатель. При создании объекта Писатель первым аргументом является filename, а второй аргумент false указывает, что данные следует *не добавить* (append) к содержимому файла (если он уже существует), а *перезаписать* (overwrite). Запись на диск производится с помощью метода Write() из свойства Text элемента управления textBox1. На рисунке 7.1 приведен фрагмент работы программы.

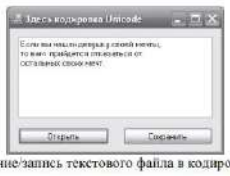


Рис.7.1 Чтение/запись текстового файла в кодировке Unicode

Запись текстового файла с помощью данной программы будет происходить *в формате (кодировке) Unicode*, как и чтение из файла. То есть вы сможете читать эти файлы Блокнотом, редактировать их, но каждый раз при сохранении файлов следить, чтобы кодировка была (оставалась) Unicode.

Контрольные вопросы

К чему свелась обработка исключительной ситуации ?
Каким методом происходит чтение файла filename?
Каким методом происходит закрытие файла?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 52 Использование ползунка и обработка сообщений

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

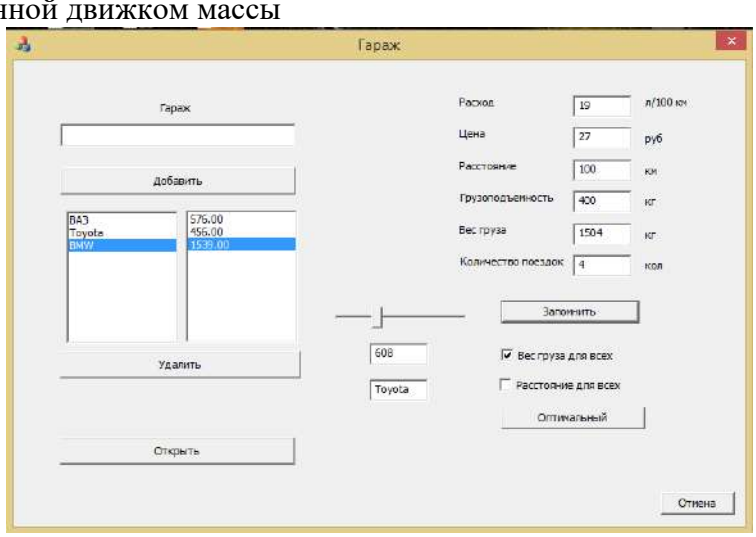
Оборудование: ноутбук Dell Inspiron 15 5000 Series

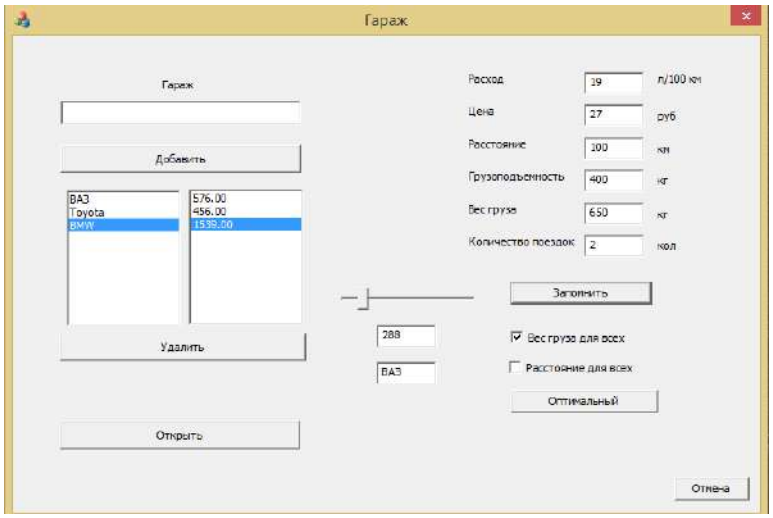
Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

1. Использовать решенную в практическом занятии 50 задачу, добавив в нее возможность изменять массу перевозимого груза с помощью элемента графического интерфейса Slider (Ползунок).

2. Усовершенствовать полученную программу, добавив к методу, переделяющему положение движка вычисление оптимального варианта использования автомобиля для перевозки груза заданной движком массы





Контрольные вопросы:

1. Элемент управления Slider.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 53 Создание меню и подключение команд меню к коду программы

Практическое занятие направлено на формирование профессиональных компетенций:

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Для создания меню на панели инструментов выберите *Menu*. Дважды кликните на появившемся в нижней области окна об а затем перейдите на форму и в области *(Вводить здесь)* вв меню верхнего уровня с текстом *Цвет*.

Переместитесь на нижнюю область и вв текст *Черный*. Заполните элемент MenuStrip следующим образ

Цвет
Черный
Красный
Синий
Зеленый

Запустите программу и поэкспериментируйте: выбирай разные элементы созданного объекта.

Запрограммируйте событие *Click* для каждого пункта; наприме для элемента *Черный* необходимо написать следующий ко (дважды щелкнув на пункте, чтобы открыть код):

```
private: System::Void черныйToolStripMenuItem_Click(System::Object^ sender,
System::EventArgs^ e) {
 this->txt->BackColor=System::Drawing::Color::Black;
}
```

Запрограммируйте аналогично остальные пункты *Цвет*. Запустите и отладьте приложение. Сохраните проект.

Контрольные вопросы

1. Каково основное назначение объекта MenuStrip?
2. Как запрограммировать необходимый пункт меню формы в Visual c++?
3. Какое свойство служит для изменения фона объекта?
4. С помощью какого свойства menu можно сделать недоступным какой-либо пункт?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 54 Вызов диалоговых окон из меню. Решение задач с использованиием различных элементов графического интерфейса

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

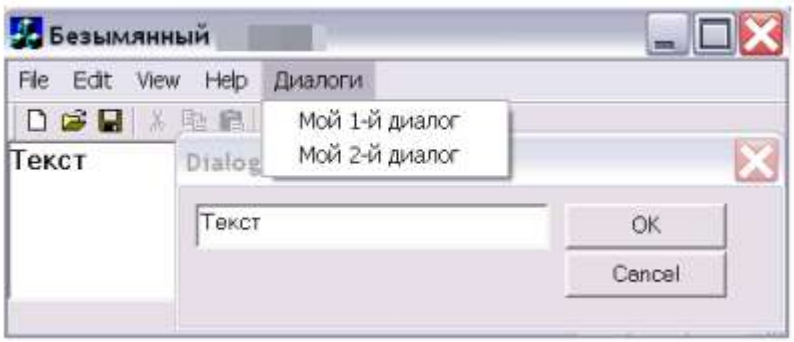
Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать приложение:



Контрольные вопросы

1. Описать разницу между модальными и немодальными окнами.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 55 Создание SDI-приложений

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать SDI простой текстовый редактор.

Контрольные вопросы

1. Что такое SDI-приложение и чем отличается от MDI.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 56 Создание MDI-приложений

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Доработать текстовый редактор из практического занятия 55, сделав его MDI-приложением.

Контрольные вопросы

1. Назвать примеры MDI-приложений.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 57 Программирование приложения с базой данных

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать приложение для работы с базой данных «Телефонный справочник» (1-2 таблицы).

Контрольные вопросы

1. С исползовнием какой технологии осуществлено подключение базы данных.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 58 Программирование приложения с базой данных MySQL

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: получить навыки создания приложения с графическим интерфейсом в VisualStudio 2019 на языке C++

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Прядок выполнения заданий

Создать приложение для работы с базой данных «Складской учет» (1-2 таблицы).

Контрольные вопросы

1. Чем отличается подключение базы данных MySQL от MS Access.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся самостоятельно выполнил все этапы решения задачи, задания для самостоятельной работы;

оценка «4» ставится, если:

-работа выполнена полностью, но при выполнении обнаружилось недостаточное владение навыками работы с программным обеспечением в рамках поставленной задачи;

оценка «3» ставится, если:

- работа выполнена не полностью, но обучающийся владеет основными навыками работы со средой разработки, требуемыми для решения поставленной задачи.

оценка «2» ставится, если:

- допущены существенные ошибки, показавшие, что обучающийся не владеет обязательными знаниями, умениями и навыками работы с программным обеспечением и средой разработки или значительная часть работы выполнена не самостоятельно.

Практическое занятие 59 Работа со строками

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить принципы разработки программ с использованием строк. Изучить стандартные свойства и методы работы со строками типа string языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

1. Дана строка, в которой слова разделены одним или несколькими пробелами. Выделить слова из этой строки и отсортировать их в алфавитном порядке.

Внешний вид главной формы программы показан на рисунке 1.

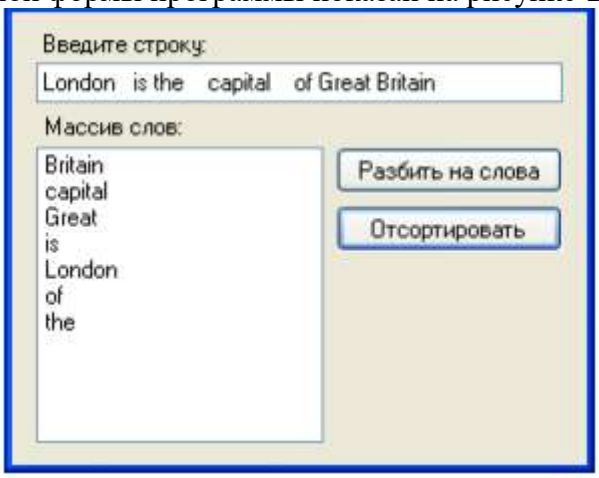


Рисунок 1 – Главное окно программы

Листинг программы

```
//массив описываем вне обработчиков
string[] mas;
//Обработчик кнопки "Разбить на слова"
private void button1_Click(object sender, EventArgs e)
{
   //исходная строка
   string s = textBox1.Text;
   //массив разделителей состоит из одного элемента - пробела
   char[] sep = new char[] { ' ' };
   //разбиваем строку на слова и удаляем пустые строки
   mas = s.Split(sep, StringSplitOptions.RemoveEmptyEntries);
   //выводим слова в список на форме
   listBox1.Items.Clear();
   for (int i = 0; i < mas.Length; i++)
      listBox1.Items.Add(mas[i]);
}
//Обработчик кнопки "Отсортировать"
private void button2_Click(object sender, EventArgs e)
{
   string t = "";
   //сортировка слов методом "пузырька"
   for (int i = 0; i < mas.Length; i++)
      for (int j = mas.Length - 1; j > i; j--)
      if (mas[j].CompareTo(mas[j - 1]) < 0)
```

```
{
   t = mas[j];
   mas[j] = mas[j - 1];
   mas[j - 1] = t;
}
//выводим отсортированные слова в список на форме
listBox1.Items.Clear();
for (int i = 0; i < mas.Length; i++)
   listBox1.Items.Add(mas[i]);
```

2 Дана текстовая строка. Сформировать новую строку из тех символов, которые стоят на нечетных позициях в исходной строке и не являются цифрами.

Внешний вид приложения приведен на рисунке 2

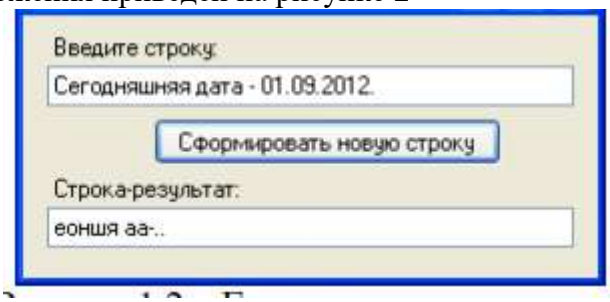


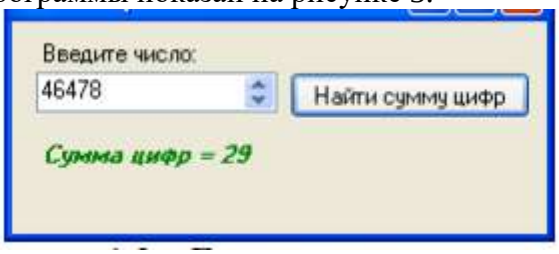
Рисунок 2 – Главное окно программы

Листинг программы

```
private void button1_Click(object sender, EventArgs e)
{
   //исходная строка
   string s1 = textBox1.Text;
   string s2 = "";
   //просмотр символов исходной строки
   for (int i = 0; i < s1.Length; i++)
      //если выполняются заданные условия,
      if (i % 2 != 0 && !char.IsDigit(s1[i]))
         //то добавляем символ в новую строку
         s2 += s1[i];
   textBox2.Text = s2;
}
```

3 Найти сумму цифр введенного числа.

Внешний вид окна программы показан на рисунке 3.



Листинг программы

```
private void button1_Click(object sender, EventArgs e)
{
   decimal n = numericUpDown1.Value;
   string s = n.ToString();
int sum = 0;
   for (int i = 0; i < s.Length; i++)
      sum += Convert.ToInt32(s[i].ToString());
```

```
    label2.Text = "Сумма цифр = " + sum;
}
```

Задачи для самостоятельного решения:

1. Дана последовательность слов. Проверить правильность и исправить ошибки написания сочетаний "жи", "ши", "ча", "ща", "чу", "щу".

2. Дан массив фамилий. Составить новый массив, который будет содержать только женские фамилии. (Примечание: те фамилии, по которым пол трудно определить, не считать).

3. Найти произведение и минимальную четную цифру, составляющую некоторое число X.

4. Написать программу, которая запрашивает полное имя пользователя, затем пользователь вводит строку символов. Компьютер должен подсчитать, сколько раз в тексте встречается каждая буква из имени.

5. Перевести число из 10-ой системы в 16-тиричную и наоборот.

6. Дан текст, состоящий не менее чем из пяти слов. Сформировать строку, в которую попадут только те слова, где одинаковые буквы встречаются более двух раз. Например, молоко.

7. Из имеющегося набора слов выбрать наиболее длинное и записать его прописными буквами.

8. Дана строка цифр. Сформировать строку, в которую войдут все цифры из исходной строки, кроме той, которая встречается наибольшее количество раз. Ее вывести отдельно.

9. Дана последовательность, состоящая из цифр, букв и знаков пунктуации в произвольном порядке. Подсчитать чего больше и составить строки только из цифр, букв и знаков пунктуации.

10. Дана некоторая последовательность букв русского алфавита. Написать программу, которая запрашивает Ваше имя и определяет, можно ли из букв исходной строки составить его.

11. Дан текст, записанный в виде криптограммы (шифрограммы), в которой буквы истинного текста размещаются в позициях, кратных 3. Прочитать исходный текст.

12. Дана строка цифр. Составить из них 5-значные числа. Если на последнее число не хватит цифр, дополнить его первыми цифрами исходной строки.

13. Перевести число из 2-ой системы в 10-тичную и наоборот.

14. Дан текст. Найти самое короткое слово длиной не менее 3 букв.

15. Написать программу, которая переставит слова в строке по мере увеличения их длины.

16. Дан текст, состоящий не менее чем из пяти слов. Вывести на экран слова, в которых отсутствует буква "Е".

17. Дан текст, состоящий не менее чем из пяти слов. Определить процентное соотношение в нем коротких слов (длиной менее четырех символов) к длинным (все остальные).

18. Найти количество нечетных цифр и максимальную нечетную цифру, составляющую некоторое число X.

19. Дан текст, состоящий не менее чем из пяти слов. Определить, есть ли в нем слова, начинающиеся и заканчивающиеся с буквы "А", а также количество таких слов.

20. Дан текст, состоящий не менее чем из пяти слов. Написать программу, которая в каждом слове делает заглавной первую букву.

21. Написать программу, которая проверяет, является ли введенное слово палиндромом. Палиндромом называется слово, которое читается одинаково слева направо и справа налево, например, "КАЗАК".

22. Дана строка слов. Определить буквы, которые встречаются наибольшее и наименьшее количество раз.

23. Дан текст, состоящий не менее чем из пяти слов. Вывести на экран слова, которые имеют окончания "ИЯ","ИСТ","ИКА".

24. Найти минимальную и максимальную цифры среди четных и нечетных цифр, составляющих некоторое число X.

25. Дан текст. Преобразовать его по следующему правилу: если нет символа ‘*’, то оставить его без изменения, иначе заменить каждый символ, встречающийся после первого вхождения символа ‘*’, на символ ‘–’ .

26. Подсчитать сумму и количество всех цифр, входящих в некоторое предложение, вводимое с клавиатуры.

27. Дан текст, состоящий не менее чем из семи слов. Все слова из четырех букв записать наоборот.

28. Дана последовательность из латинских букв и цифр. Определить, чего больше.

29. Дан текст, оканчивающийся точкой. Найти количество слов, у которых первый и последний символы совпадают.

30. Найти среднее арифметическое значение между минимальной и максимальной цифрами некоторого числа X.

31. Дан текст, состоящий не менее чем из семи слов. Определить, есть ли в нем слова, начинающиеся с буквы "Ф", а также количество таких слов.

32. Дан текст. Заменить в нем символы, расположенные между круглыми скобками на символ ‘*’ (внутри каждой пары скобок нет других скобок).

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания и решил менее трех задач для самотоятельного решения.

Практическое занятие 60 Составные типы данных: структуры

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить практические навыки использования комбинированного типа данных СТРУКТУРА в разработке приложений языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Программа находит в массиве данных о людях самого младшего (по годам) человека.

Внешний вид окна приложения приведен на рисунке 1/

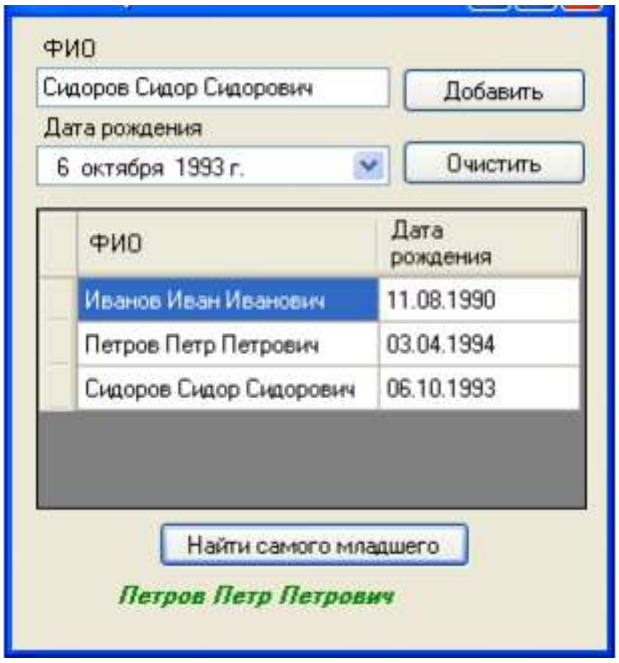


Рисунок 1 – Главное окно приложения

Для указания даты рождения на форме используется компонент DateTimePicker, а в программном коде – тип DateTime, представляющий значения типа «Дата/время». Для того чтобы получить из переменной типа DateTime значение даты в формате «чч.мм.гггг» (для последующего занесения его в таблицу), в программе использован метод ToShortDateString().

Обратите внимание, что описания структур и массива находятся вне обработчиков событий (например, перед ними).

```
//структура "Дата" struct Date
{
   public int number;
   public int month;
   public int year;
}
/структура "Человек"
struct Person
{
   public string FIO;
   public Date Birthday;
}

//описание массива людей
Person[] mas;

//обработчик кнопки "Добавить"
private void button1_Click(object sender, EventArgs e)
{
   //берем данные о человеке из компонентов на форме
   string fio = textBox1.Text;
   DateTime birthday = dateTimePicker1.Value;
   //и заносим их в таблицу dataGridView1
   dataGridView1.Rows.Add(fio, birthday.ToShortDateString());
}

//обработчик кнопки "Очистить"
private void button2_Click(object sender, EventArgs e)
```

```
{
    dataGridView1.Rows.Clear();
}

//обработчик кнопки "Найти самого младшего"
private void button3_Click(object sender, EventArgs e)
{
    //определяем кол-во людей в массиве
    int n = dataGridView1.RowCount;
    mas = new Person[n];
    //записываем данные из таблицы на форме в массив mas
    for (int i = 0; i < n; i++)
    {
        mas[i].FIO = dataGridView1.Rows[i].Cells[0].Value.ToString();
        DateTime dt = Convert.ToDateTime(dataGridView1.Rows[i].
Cells[1].Value.ToString());
        mas[i].Birthday.number = dt.Day;
        mas[i].Birthday.month = dt.Month;
        mas[i].Birthday.year = dt.Year;
    }
    //элемент массива, соответствующий самому младшему человеку
    Person min = mas[0];
    //его порядковый номер в массиве
    int ind = 0;
    //ищем самого младшего
    for (int i = 1; i < n; i++)
        if (mas[i].Birthday.year > min.Birthday.year)
        {
            min = mas[i];
ind = i;
        }
        //выводим его ФИО на форму
        label3.Text = mas[ind].FIO;
}
```

Задачи для самостоятельного решения:

1. Для каждого из N студентов группы известны ФИО и оценки (в баллах) по четырем дисциплинам. Найти среднюю оценку каждого студента и выбрать человека, имеющего максимальный средний бал.

2. Дан массив студентов ВУЗа: ФИО, возраст, регион, факультет. Вывести на экран результирующую таблицу: регион, количество студентов из этого региона. Отсортировать данные по названию региона.

3. Багаж пассажира характеризуется количеством вещей и общим весом вещей. Дан массив, содержащий сведения о багаже нескольких пассажиров. Найти средний вес одной вещи в каждом багаже и по всем пассажирам.

4. Дан массив данных о клиентах пункта проката автомобилей: ФИО, адрес (улица, дом, квартира) и марка машины. Во второй массив записать данные только тех людей, кто ездит на «Audi».

5. Дан список английских глаголов: тип (правильный/неправильный), первая форма, вторая форма, третья форма. Вывести в алфавитном порядке неправильные глаголы, у которых все три формы совпадают.

6. Рациональное число можно представить записью с двумя полями: числитель и знаменатель. Дан массив из N рациональных чисел. Разработать функцию для нахождения максимального среди них.

7. Дан массив, в котором хранятся данные о расписании самолетов на неделю: пункт назначения, время вылета, количество свободных мест. В кассу аэропорта обращается пассажир, желающий купить n билетов на первую половину сегодняшнего дня до выбранного пункта назначения. Определить, есть ли возможность выполнить его заказ, предоставить ему возможные варианты на выбор.

8. Дан массив данных о работниках фирмы: ФИО и год поступления на работу. Во второй массив записать только данные тех из них, кто на сегодняшний день проработал уже не менее 5 лет.

9. Дан массив данных о работниках фирмы: фамилия, имя, отчество, адрес (улица, дом, квартира) и год поступления на работу. Определить, есть ли в списке Петровы (Петров, Петрова), если есть, то вывести их адрес (адреса).

10. Дан список иногородних студентов из n человек: ФИО, адрес (город, улица, дом-квартира), расстояние до Краснодара. Для них в общежитии выделено k мест. Вывести список студентов, которых необходимо селить в общежитие в первую очередь. Критерий отбора: расстояние до города. Под-сказка: отсортировать исходный массив по убыванию расстояний.

11. Дан массив данных о студентах некоторой группы: фамилия, имя, отчество и дата рождения (день, месяц, год). Вывести на экран фамилию и имя тех студентов, у кого сегодня день рождения (сегодняшнюю дату вводить с клавиатуры).

12. Дан массив, содержащий сведения о студентах некоторой группы: ФИО, оценки по пяти экзаменационным дисциплинам. Вывести студентов, получающих повышенную стипендию. Организовать поиск студента по фамилии с выводом информации о нем.

13. Дан массив книг: название, автор, количество страниц. Вывести все книги А.С. Пушкина в алфавитном порядке.

14. Дан массив, в котором хранятся данные о расписании поездов на сегодняшний день: номер поезда, название (т.е. откуда – куда, например, Новороссийск-Москва), время прибытия на станцию и время отправления (часы, минуты). По данному времени определить, какие из поездов стоят сейчас на станции.

15. Дан массив, содержащий сведения о студентах некоторой группы: ФИО, адрес (улица, дом, квартира) и телефон (если есть). Вывести на экран сведения о тех из них, до которых нельзя дозвониться.

16. Точка плоскости может быть представлена двумя координатами X и Y. Дан массив, содержащий N точек. Найти точку, которые максимально удалена от начала координат.

17. Багаж пассажира характеризуется количеством вещей и общим весом вещей. Дан массив, содержащий сведения о багаже нескольких пассажиров. Выяснить имеется ли пассажир, багаж которого состоит из одной вещи весом более 30 кг.

18. Комплексное число можно представить через реальную (Re) и мнимую (Im) часть. Разработать функции сложения и вычитания комплексных чисел. Пусть Z1=(Re1, Im1), Z2=(Re2, Im2), тогда: 1) сумма: Z = (Re1+Re2, Im1+Im2); 2) разность: Z = (Re1-Re2, Im1-Im2).

19. Дан перечень постельного белья: название (наволочка/одеяло/простыня), цвет, размер (односпальный/полуторный/двуспальный). Подсчитать, сколько комплектов двуспального белья белого цвета можно составить из данного перечня.

20. Дан массив, содержащий сведения о студентах группы: фамилия, имя, отчество, дата рождения (день, месяц, год). Найти самого младшего студента по полной дате рождения.

21. Время можно представить с помощью часов, минут и секунд. Написать функцию проверка(t1,t2), проверяющую, предшествует ли время t1 времени t2 (в рамках суток).

22. Для каждого из N студентов группы известны ФИО и оценки (в баллах) по пяти дисциплинам. Вывести на экран фамилию и имя тех из них, у которых количество положительных оценок больше, чем отрицательных (положительная оценка – 3 и больше).

23. Рациональное число можно представить записью с тремя полями: целая часть, числитель и знаменатель. Разработать функцию, позволяющую из неправильной дроби получить правильную. Неправильной называется дробь, у которой числитель больше знаменателя.

24. Рациональное число можно представить записью с тремя полями: целая часть, числитель и знаменатель. Дан массив рациональных чисел. Найти их сумму и по возможности сократить.

25. Багаж пассажира характеризуется количеством вещей и общим весом вещей. Дан массив, содержащий сведения о багаже нескольких пассажиров. Найти число пассажиров, имеющих более двух вещей.

26. Имеется список членов коллектива с указанием принадлежности каждого к различным общественным организациям (профком, ученый совет, общество книголюбов и т.п.). Напечатать приглашение всем членам на очередное заседание указанной организации. Задается только вид организации, место и время сбора.

27. Дан массив, содержащий информацию об учениках некоторой школы. Вывести на экран сведения об учениках только десятых классов. На сколько человек в девятых классах больше, чем в десятых.

28. Дан массив, содержащий сведения о книгах: название, жанр, автор. Вывести книги только классического жанра, отсортировав их по фамилии автора. 29. Время можно представить с помощью часов, минут и секунд. Написать функцию перевод(t1,t2), присваивающую параметру t2 время на 1 секунду большее времени t1 (учесть смену суток).

30. Дан массив книг: название, тип, автор, количество страниц, странародина автора. Организовать поиск по автору, по типу. Вывести все книги зарубежных авторов.

31. Рациональное число можно представить записью с двумя полями: числитель и знаменатель. Разработать функцию сократить(m) приведения рационального числа m к несократимому виду.

32. Дан массив данных о работниках фирмы: ФИО и адрес (улица, дом, квартира). Во второй массив записать только тех из них, которые живут на улице Красной. Вывести их на экран в алфавитном порядке.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания и решил менее трех задач для самотоятельного решения.

Практическое занятие 61 Основы построения классов

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: Получить практические навыки использования классов в разработке приложений языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

1) Нажать Файл -> Создать -> Проект, в появившемся окне выбрать “Приложение Windows Forms”, ввести имя проекта.

2) В форме Form1 разместить четыре элемента TextBox, один элемент GroupBox, Один элемент Button, три элемента Label.

3) Сгруппировать элементы так, как показано на рисунке 1, изменить значение поля Text элемента GroupBox1 на «Информация», сменить значение поля Name элементов TextBox1, TextBox2, TextBox3 на NameBox, AgeBox, ProfessionBox и ResultBox соответственно. Сменить значение поля Name элемента Button1 на StartButton.

4) Поместить Элементы NameBox, AgeBox, ProfessionBox в GroupBox1, подписать их с помощью элементов Label1, Label2 и Label3(сменив значение поля Text элементов Label). Элемент ResultBox перевести в режим MultiLine (рисунок 2).

5) Значение поля Text элемента StartButton сменить на «Добавить».

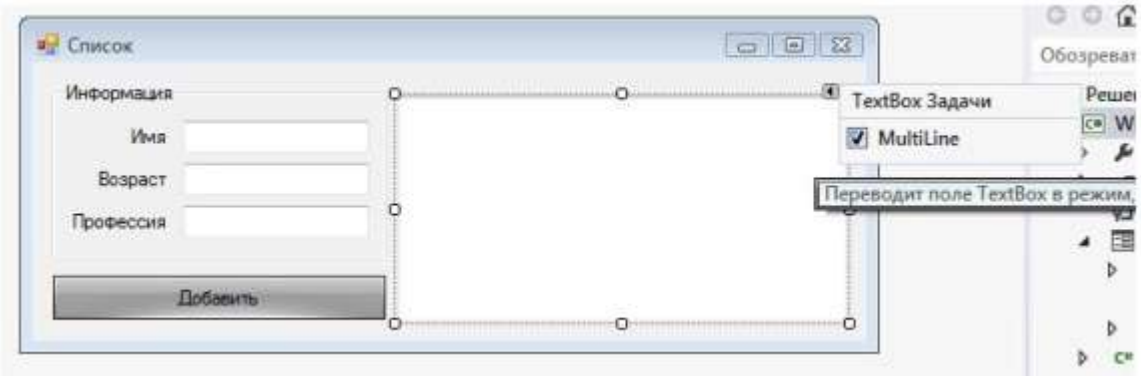


Рисунок 1 – Форма списка

6) В окне Обозревателя Решений щелкнуть правой кнопкой по имени проекта, затем Добавить -> Класс. В появившемся окне ввести имя класса (Person) и нажать кнопку Добавить .

7) Объявить переменные name и profession типа string и переменную age типа integer:

```
string name;
int age;
string profession;
```

3) Объявить конструктор класса Person, который будет принимать значение поля name:

```
public Person(string name)
{
this.name = name;
}
```

8) Перегрузить этот конструктор для различных параметров, которые могут передаваться при создании экземпляра класса.

Пример конструктора, принимающего значения полей name и age: public Person(string name, int age)

```
{
        this.name = name;
this.age = age;
}
```

Пример конструктора, принимающего значения полей name и profession:

```
public Person(string name, string profession)}
```

```
{
this.name = name;
 this.profession = profession;
}
```
Пример конструктора, принимающего значения полей name, age и profession^
```
public Person(string name, int age, string profession)
{
this.name = name;
this.age = age;
this.profession = profession;
}
```
9) Дважды щелкнуть по элементу StartButton, откроется код обработчика события нажатия кнопки
```
private void button1_Click(object sender, EventArgs e)
```
10) Объявить экземпляр класса Person, не вызывая при этом конструктор (необходимо, чтобы работать с объектом впоследствии, вызывая различные перегрузки конструктора для конкретной ситуации):
```
Person new_person;
```
11) Реализовать проверку на наличие текста в NameBox, если текста нет, то выводим сообщение пользователю путем создания диалогового окна Mes-sageBox .
```
if (NameBox.Text == "")
{
MessageBox.Show("Введите имя");
}
```
12) Реализовать конструкцию из условий, проверяющих либо наличие, либо отсутствие текста в элементах управления TextBox, в результате выполнения условий, явно вызывать одну из перегрузок конструктора класса Person:
```
if (AgeBox.Text != "")
{
if (ProfessionBox.Text != "")
{
new_person = new Person(NameBox.Text, Con-vert.ToInt32(AgeBox.Text), ProfessionBox.Text);
}
else
{
new_person = new Person(NameBox.Text, Con-vert.ToInt32(AgeBox.Text));
}
}
else
{
if (ProfessionBox.Text != "")
{
new_person = new Person(NameBox.Text, 0,ProfessionBox.Text);
}
else
{
new_person = new Person(NameBox.Text);
}
}
```
13) Для вывода результатов работы программы, необходимо использовать метод GetInformation() класса Person, задача которого будет выводить строку, в которой будет содержаться текущая информация об экземпляре класса: поля name, age, profession (см. рисунок 2).

```
public string GetInformation()
{
string information;
information = "Имя: " + this.name + "; Возраст: " + this.age.ToString() + "; Профессия: " + this.profession;
return information;
}
```

Синтаксис в коде:

```
ResultBox += new_person.GetInformation();
```

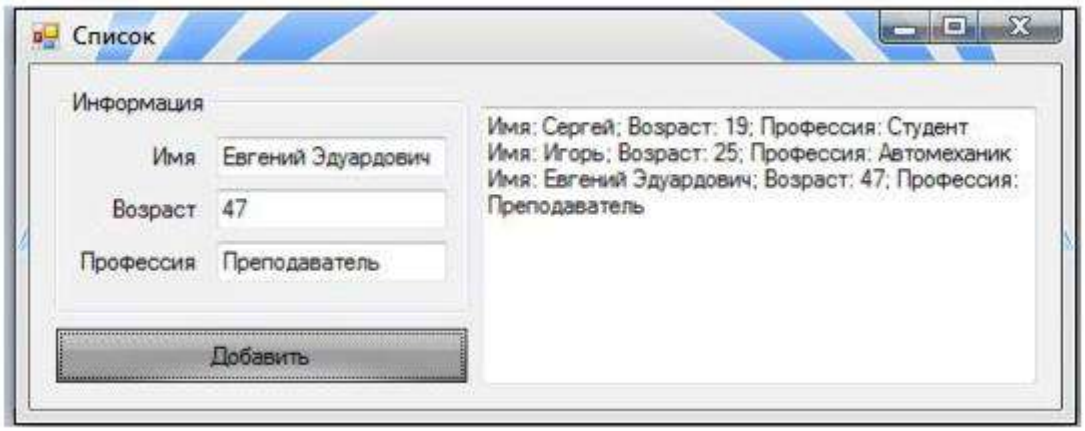


Рисунок 4 – Результат в форме

Задачи для самостоятельного решения:

Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы, свойства. Методы и свойства должны обеспечивать непротиворечивый, полный, минимальный и удобный интерфейс класса. При возникновении ошибок должны выбрасываться исключения.

В программе должна выполняться проверка всех разработанных элементов класса.

1.Описать класс, реализующий десятичный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство, позволяющее получить его текущее состояние. При выходе за границы диапазона выбрасываются исключения.

Написать программу, демонстрирующую все разработанные элементы класса.

2.Описать класс, реализующий шестнадцатеричный счетчик, который может увеличивать или уменьшать свое значение на единицу в заданном диапазоне. Предусмотреть инициализацию счетчика значениями по умолчанию и произвольными значениями. Счетчик имеет два метода: увеличения и уменьшения, — и свойство,

позволяющее получить его текущее состояние. При выходе за границы диа-пазона выбрасываются исключения.

Написать программу, демонстрирующую все разработанные элементы класса.

3.Описать класс, представляющий треугольник. Предусмотреть методы для создания объектов, перемещения на плоскости, изменения размеров и вращения на заданный угол. Описать свойства для получения состояния объекта. При невозможности построения треугольника выбрасывается исключение. Написать программу, демонстрирующую все разработанные элементы класса.

4.Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса и проверки допустимости вводимых значений. В случае недопустимых значений полей выбрасываются исключения.

Написать программу, демонстрирующую все разработанные элементы класса.

5.Составить описание класса для представления комплексных чисел. Обеспечить выполнение операций сложения, вычитания и умножения комплексных чисел.

Написать программу, демонстрирующую все разработанные элементы класса.

6.Составить описание класса для вектора, заданного координатами его концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

Написать программу, демонстрирующую все разработанные элементы класса.

7.Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

Написать программу, демонстрирующую все разработанные элементы класса.

8.Составить описание класса для представления даты. Предусмотреть возможности установки даты и изменения ее отдельных полей (год, месяц, день) с проверкой допустимости вводимых значений. В случае недопустимых значений полей выбрасываются исключения. Создать методы изменения даты на заданное количество дней, месяцев и лет.

Написать программу, демонстрирующую все разработанные элементы класса.

9.Составить описание класса для представления времени. Предусмотреть возможности установки времени и изменения его отдельных полей (час, минута, секунда) с проверкой допустимости вводимых значений. В случае не-допустимых значений полей выбрасываются исключения. Создать методы изменения времени на заданное количество часов, минут и секунд.

Написать программу, демонстрирующую все разработанные элементы класса.

10.Составить описание класса многочлена вида ax2 + bx + c. Предусмотреть методы, реализующие:

- вычисление значения многочлена для заданного аргумента;
- операцию сложения, вычитания и умножения многочленов с получением нового объекта-многочлена;
- вывод на экран описания многочлена.

Написать программу, демонстрирующую все разработанные элементы класса.

11.Описать класс, представляющий треугольник. Предусмотреть методы для создания объектов, вычисления площади, периметра и точки пересечения медиан. Описать свойства для получения состояния объекта. При невозможности построения треугольника выбрасывается исключение.

Написать программу, демонстрирующую все разработанные элементы класса.

12.Описать класс, представляющий круг. Предусмотреть методы для создания объектов, вычисления площади круга, длины окружности и проверки попадания заданной точки внутрь круга. Описать свойства для получения состояния объекта.

Написать программу, демонстрирующую все разработанные элементы класса.

13.Описать класс для работы со строкой, позволяющей хранить только дво-ичное число и выполнять с ним арифметические операции. Предусмотреть инициализацию с проверкой допустимости значений. В случае недопустимых значений выбрасываются исключения.

Написать программу, демонстрирующую все разработанные элементы класса.

14.Описать класс дробей — рациональных чисел, являющихся отношением двух целых чисел. Предусмотреть методы сложения, вычитания, умножения и деления дробей.

Написать программу, демонстрирующую все разработанные элементы класса.

15 .Описать класс «файл», содержащий сведения об имени, дате создания и длине файла. Предусмотреть инициализацию с проверкой допустимости значений полей. В случае недопустимых значений полей выбрасываются исключения. Описать метод добавления информации в конец файла и свойства для получения состояния файла.

Написать программу, демонстрирующую все разработанные элементы класса.

16.Описать класс «комната», содержащий сведения о метраже, высоте потол-ков и количестве окон. Предусмотреть инициализацию с проверкой допустимости значений полей. В случае недопустимых значений полей выбрасываются исключения. Описать методы вычисления площади и объема комнаты и свойства для получения состояния объекта.

Написать программу, демонстрирующую все разработанные элементы класса.

17.Описать класс, представляющий нелинейное уравнение вида ax - cos(x) = 0. Описать метод, вычисляющий решение этого уравнения на заданном интервале методом деления пополам (см. раздел «Цикл с параметром for») и выбрасывающий исключение в случае отсутствия корня. Описать свойства для получения состояния объекта.

Написать программу, демонстрирующую все разработанные элементы класса.

18.Описать класс, представляющий квадратное уравнение вида ax2 + bx + c = 0. Описать метод, вычисляющий решение этого уравнения и выбрасывающий исключение в случае отсутствия корней. Описать свойства для получения состояния объекта.

Написать программу, демонстрирующую все разработанные элементы класса.

19.Описать класс «процессор», содержащий сведения о марке, тактовой частоте, объеме кэша и стоимости. Предусмотреть инициализацию с проверкой допустимости значений полей. В случае недопустимых значений полей выбрасываются исключения. Описать свойства для получения состояния объекта. Описать класс «материнская плата», включающий класс «процессор» и объем установленной оперативной памяти. Предусмотреть инициализацию с проверкой допустимости значений поля объема памяти. В случае недопустимых значений поля выбрасывается исключение. Описать свойства для получения состояния объекта.

Написать программу, демонстрирующую все разработанные элементы классов.

20.Описать класс «цветная точка». Для точки задаются координаты и цвет. Цвет описывается с помощью трех составляющих (красный, зеленый, синий). Предусмотреть различные методы инициализации объекта с проверкой допустимости значений. Допустимым диапазоном для каждой составляющей является [0, 255]. В случае недопустимых значений полей выбрасываются исключения. Описать свойства для получения состояния объекта и метод изменения цвета.

Написать программу, демонстрирующую все разработанные элементы класса.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания и решил менее трех задач для самотоятельного решения.

Практическое занятие 62 Разработка приложений с использованием коллекций

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить интерфейсы и классы коллекций библиотеки .NET Framework, основные свойства и методы этих классов, применяемые при работе с коллекциями в разработке приложений языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Дан список, содержащий точки с координатами (x,y). Создать новый список, в который попадут точки, лежащие левее оси ординат. Соответственно, в старом списке останутся точки, которые расположены справа от оси ординат. Предусмотреть возможность сортировки нового списка по возрастанию координаты x.

Внешний вид работающего приложения с описанием использованных компонентов приведен на рисунке 1.

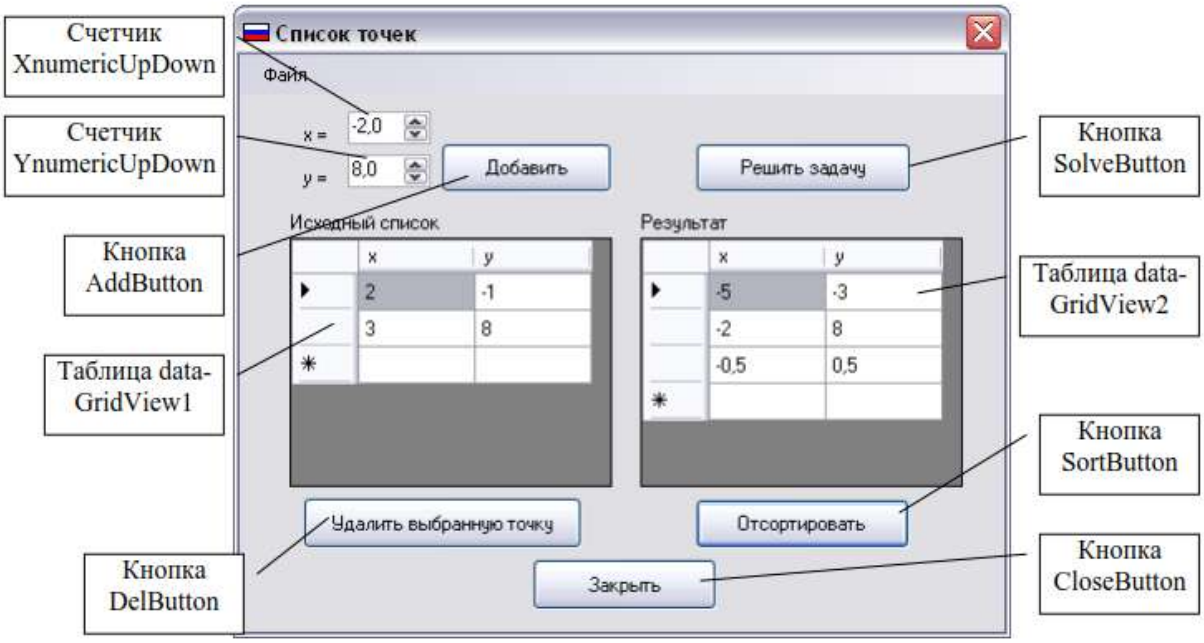


Рис. 1. Пример работающего приложения

Для счетчиков XnumericUpDown и YnumericUpDown устанавливаются следующие дополнительные свойства:

Свойство	Значение	Описание
`Minimum`	-100	Минимальное значение компонента
`Maximum`	100	Максимальное значение компонента
`Increment`	0,1	Шаг увеличения или уменьшения
`DecimalPlaces`	1	Количество знаков после запятой при отображении

Для таблиц данных dataGridView1 и dataGridView2 устанавливаются следующие дополнительные свойства:

Свойство	Значение	Описание
`ColumnHeadersHeight-SizeMode`	`AutoSize`	Регулирует высоту заголовков столбцов
`EditMode`	`EditProgrammatically`	Режим редактирования данных (здесь – только программно)
`Columns`	редактируется	Столбцы таблицы

Для добавления столбцов в таблицу следует нажать кнопку свойства Columns. Откроется диалоговое окно, где нужно добавлять и устанавливать свойства столбцов таблицы (рис. 2)

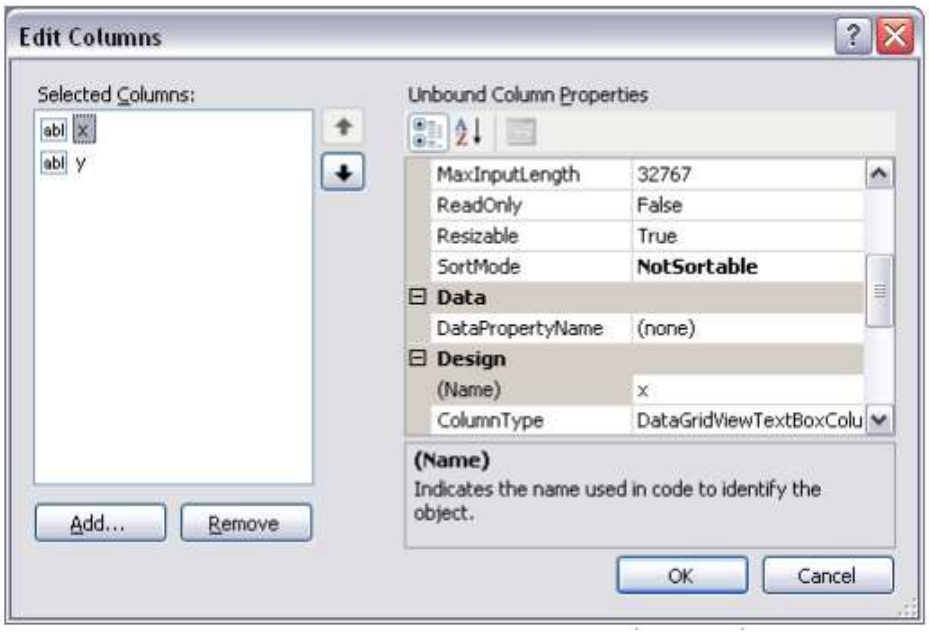


Рис. 2. Окно редактирования столбцов таблицы

В нашем случае добавляются 2 столбца с именами x и y (свойства Name и HeaderText) и для них отменяется режим автоматической сортировки строк (свойство SortMode устанавливается равным NotSortable).

Проектирование кода начинается с разработки класса, представляющего

собой структуру точки. Чтобы добавить класс, нужно выбрать пункт меню Project->Add Class…, в появившемся диалоговом окне снова выбрать «Class» и указать имя создаваемого файла (и класса). В нашем случае класс называется «Point».

Так как в задании необходимо реализовать сортировку списка точек по одной из координат, программе (а именно методу Sort) необходимо явно указать, по какому именно признаку (по какой координате) следует производить сравнение точек при сортировке. Для этого нужно, чтобы класс реализовывал интерфейс IComparable.

Созданный класс будет содержать следующие компоненты:

 два закрытых поля типа float, задающих координаты точки;

 конструктор с двумя параметрами, инициализирующий эти поля;

 два открытых свойства для доступа к полям (чтения и установки им значений);

 реализацию открытого метода CompareTo, описанного в интерфейсе

IComparable и задающего способ сравнения объектов класса. Этот метод возвращает целочисленное значение: положительное, если вызывающий объ-ект больше объекта параметра, отрицательное, если меньше, и ноль, если объекты равны. Именно в этом методе мы указываем, что сравнение точек должно производиться по координате x.

Ниже приведен листинг описания класса Point:

```
//класс реализует интерфейс IComparable
class Point : IComparable
{
   float X;    //координаты точки
   float Y;
   //конструктор класса
   public Point(float X, float Y)
   {
      this.X = X;
      this.Y = Y;
   }
```

```
    //свойство для доступа к координате x
    public float x
    {
      get { return X; }   //получить x
      set { X = value; }  //установить x
    }
    //свойство для доступа к координате y
    public float y
    {
      get { return Y; }  //получить y
      set { Y = value; } //установить y
    }
    //установить способ сравнения объектов-точек
    public int CompareTo(object obj)
    {
      //преобразуем параметр obj к типу точки
      Point p = (Point)obj;
      if (x > p.x)   //сравниваем координаты x
        return 1;  //и возвращаем либо положительное,
      if (x == p.x)
        return 0;  //либо нулевое,
      return -1;     //либо отрицательное значение
    }
  }
```

Прежде всего для работы с коллекциями в начале файла главной формы нужно подключить пространство имен System.Collections:

```
using System.Collections;
```

Далее необходимо в классе формы объявить два динамических массива: исходный массив точек points и массив-результат res. Обоим массивам следует выделить память либо в конструкторе класса, либо в обработчике события Load:

```
ArrayList points, res;   //объявление динамических массивов
private void MainForm_Load(object sender, EventArgs e)
{
  points = new ArrayList(); //выделение памяти
  res = new ArrayList();
}
```

При нажатии кнопки «Добавить» необходимо взять текущие значения компонентов-счетчиков, создать объект класса точки с этими значениями в качестве координат, добавить этот объект в коллекцию и отобразить координаты новой точки в таблице на экране:

```
private void AddButton_Click(object sender, EventArgs e)
{
  //берем значения из компонентов-счетчиков
  float x = (float)XnumericUpDown.Value;
  float y = (float)YnumericUpDown.Value;
  //создаем новую точку
  Point p = new Point(x,y);
  //добавляем ее в коллекцию
  points.Add(p);
  //добавляем в таблицу на форме
  dataGridView1.Rows.Add(p.x,p.y);
}
```

При удалении точки из списка следует сначала проверить, выбрана ли строка таблицы для удаления. Реализуется это стандартными средствами обработки исключений

(операторы try… catch). Если точка выбрана, то надо получить номер соответствующей строки из таблицы, удалить точку с этим номером из коллекции и из таблицы на форме:

```
private void DelButton_Click(object sender, EventArgs e)
{
   try
   {
      //получаем номер выбранной строки
      int num = dataGridView1.SelectedRows[0].Index;
      //удаляем точку с данным номером из коллекции
      points.RemoveAt(num);
      //удаляем выбранную строку из таблицы
     dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);
   }
   catch
   {
      MessageBox.Show("Выберите строку!!!");
   }
}
```

При нажатии на кнопку «Решить задачу» должны выполняться следующие действия. Прежде всего необходимо очистить массив-коллекцию результатов res, а также обе таблицы на форме, т.к. данные в них будут обнов-ляться. Далее, путем использования методов Add и Remove классов коллекций добавляем и удаляем элементы массивов в соответствии с условиями задачи и отображаем эти массивы в таблицах на форме:

```
private void SolveButton_Click(object sender, EventArgs e)
{
   //очистить коллекцию res
   res.Clear();
   //очистить содержимое таблиц на форме
   dataGridView1.Rows.Clear();
   dataGridView2.Rows.Clear();
   //просматриваем исходный массив
   foreach (Point p in points)
      if (p.x < 0)    //при выполнении условия
         res.Add(p); //добавляем точку в массив-результат
   foreach (Point p in res)
   {
      //удаляем найденные точки из исходного массива
      points.Remove(p);
      //отображаем 2-й массив в таблице на форме
      dataGridView2.Rows.Add(p.x, p.y);
   }
   foreach (Point p in points)
      //отображаем 1-й массив в таблице на форме
      dataGridView1.Rows.Add(p.x, p.y);
}
```

Метод-обработчик кнопки «Отсортировать» достаточно простой. Реализация классом Point интерфейса IComparable позволяет применять к коллекции из объектов этого класса метод Sort(), после чего нужно только обновить содержимое отсортированного массива в таблице на форме:

```
private void Sortbutton_Click(object sender, EventArgs e)
{
   //сортируем
   res.Sort();
   //очищаем содержимое таблицы на форме
```

```
            dataGridView2.Rows.Clear();
            foreach (Point p in res)
                //обновляем таблицу на форме
                dataGridView2.Rows.Add(p.x, p.y);
        }
```

Задачи для самостоятельного решения:

1. Дан стек, содержащий фамилии студентов и средний балл сессии каждого студента. Создать новый список, в который войдут студенты, средний балл у которых не меньше "4", а в стеке останутся все остальные. Отсортировать новый список по убыванию среднего балла.

2. Дана очередь данных о клиентах пункта проката автомобилей: ФИО, адрес (улица, дом, квартира) и марка машины. Во второй массив записать отсортированные по алфавиту данные только тех людей, кто ездит на "Audi".

3. Дан список английских глаголов: тип (правильный/неправильный), первая форма, вторая форма, третья форма. Вывести только те глаголы, у которых все три формы совпадают. Вывести только правильные (неправильные) глаголы в алфавитном порядке.

4. Рациональное число можно представить записью с двумя полями: числитель и знаменатель. Дан стек из N рациональных чисел. Создать новый список из дробей, обратных исходным (числитель и знаменатель меняются местами), отсортировать его по убыванию дробей. Удалить из этого списка максимальное и минимальное значения.

5. Дан массив данных о работниках фирмы: ФИО и дата поступления на работу (месяц, год). Во второй массив записать только данные тех из них, кто на сегодняшний день проработал уже не менее 5 лет. Отсортировать данные по алфавиту.

6. Дан список данных о работниках фирмы: фамилия, имя, отчество, адрес (улица, дом, квартира) и дата поступления на работу (месяц, год). Отсортировать этот список по году поступления на работу . Определить, есть ли в списке Петровы (Петров, Петрова), если есть, то вывести их адрес (адреса) и записать их в стек.

7. Дан список иногородних студентов из n человек: ФИО, адрес (город, улица, дом-квартира), приблизительное расстояние до Краснодара. Для них в общежитии выделено k мест. Вывести очередь студентов, которых необходимо селить в общежитие в первую очередь. Критерий отбора: расстояние до города.

8. Дан массив данных о студентах некоторой группы: фамилия, имя, отчество и дата рождения (день, месяц, год). Вывести на экран фамилию и имя тех студентов, у кого сегодня день рождения (сегодняшнюю дату вводить с клавиатуры). Удалить сведения об этих студентах из исходного списка.

9. Дан массив, содержащий сведения о студентах некоторой группы: ФИО, оценки по пяти экзаменационным дисциплинам. Вывести сначала студентов, получающих повышенную стипендию, затем обычную, и, наконец, без стипендии. Организовать поиск студента по фамилии с выводом информации о нем.

10. Дан стек книг: название, автор, количество страниц. Вывести все книги А.С. Пушкина в алфавитном порядке.

11. Дан массив, в котором хранятся данные о расписании поездов на се-годняшний день: номер поезда, название (т.е. откуда – куда, например, Новороссийск-Москва), время прибытия на станцию и время отправления (часы, минуты). По данному времени определить, какие из поездов стоят сейчас на станции.

12. Для каждого из N студентов группы известны ФИО и оценки (в баллах) по пяти дисциплинам. Сформировать очередь из фамилий и имен тех из них, у которых количество положительных оценок больше, чем отрицательных.

13. Рациональное число можно представить записью с двумя полями: числитель и знаменатель. Дан список рациональных чисел. Записать в другой

список все неправильные дроби, предварительно удалив их из исходного списка и преобразовав в правильные. Неправильной называется дробь, у которой числитель больше знаменателя. Определить количество элементов каждого списка. Отсортировать оба списка по убыванию числителя.

14. Дана хеш-таблица, содержащий сведения о книгах: название, жанр, автор (где название – это ключ хеш-таблицы, остальные поля - значение). Вывести названия книг только классического жанра, отсортировав их по фамилии автора. Найти количество таких книг, насколько их больше (меньше), чем остальных.

15. Рациональное число можно представить записью с двумя полями: числитель и знаменатель. Разработать функцию сократить(m) приведения рационального числа m к несократимому виду. Дан стек рациональных чисел. Сократить эти числа и записать их в список, отсортировав по убыванию.

16. Дана очередь данных о работниках фирмы: ФИО и адрес (улица, дом, квартира). Во второй массив записать только тех из них, которые живут на улице Красной. Вывести их на экран в алфавитном порядке.

17. Дан список, содержащий числовые данные. Отсортировать его по возрастанию и сформировать два новых списка таким образом, чтобы половина элементов исходного списка попала в первый новый список (1, 3, 5, …), а вторая половина – во второй новый (2, 4, 6, …).

18. Дана очередь, содержащая перечень товаров различных фирм. Из элементов этого списка создать новый список, который будет содержать товары, изготовленные фирмой SONY. Отсортировать их по алфавиту, определить количество таких товаров и их долю от общего количества товаров.

19. Дан список, содержащий информацию о клиентах пункта проката автомобилей: ФИО и марка машины. Отсортировать его марке машины. Записать в стек данные только тех людей, кто ездит на "Audi", предварительно удалив их из исходного списка. Организовать запрос на наличие в исходном списке требуемого человека.

20. Даны два списка, содержащие перечни товаров, производимых концернами SHARP и LG. Создать список товаров, выпускаемых как одной, так и другой фирмой. Отсортировать его по алфавиту. Организовать поиск в этом списке заданного товара.

21. Дан список четырехзначных номеров лотерейных билетов. Отсортировать его по возрастанию сумм цифр числа. Сформировать очередь из чисел, которые состоят из одинаковых цифр (1111, 2222, 3333…).

22. В магазине формируется список лиц, записавшихся на покупку товара повышенного спроса. Каждая запись этого списка содержит: порядковый номер, ФИО, домашний адрес покупателя и дату постановки на учет. Удалить из списка все повторные записи, проверяя ФИО и домашний адрес. Данные организовать в виде хеш-таблицы, где порядковый номер – это ключ, а все остальные данные – значение.

23. Информация о сотрудниках двух отделов предприятия записана в стеке и содержит: ФИО, название отдела, должность, дату начала работы. Вывести списки сотрудников по отделам в порядке убывания стажа.

24. Для книг, хранящихся в библиотеке, задаются: регистрационный номер книги, автор, название, год издания, издательство, количество страниц. Данные организованы в виде хеш-таблицы, где регистрационный номер – это ключ, а все остальные данные – значение. Создать список типа ArrayList книг с фамилиями авторов в алфавитном порядке, изданных после заданного года.

25. Различные цехи завода выпускают продукцию нескольких наименований. Сведения о выпущенной продукции включают: наименование, количество, номер цеха. Для заданного цеха необходимо вывести количество вы-пущенных изделий по каждому наименованию в порядке убывания количества. Исходные данные задаются в виде очереди, результат – в виде динамического массива ArrayList.

26. Стек данных о группе студентов содержит следующую информацию: ФИО, рост и вес. Сформировать и вывести список ФИО студентов, рост и вес которых являются в списке уникальными. Отсортировать список по алфавиту.

27. Информация об участниках спортивных соревнований двух команд содержит: наименование страны, название команды, ФИО игрока, игровой номер, возраст. Вывести и отсортировать по игровому номеру информацию о самой молодой команде.

28. Дана очередь данных о служащих предприятия: фамилия, отдел, зарплата. Сформировать список данных по алфавиту о работниках бухгалтерии и определить их суммарную зарплату.

29. Ведомость абитуриентов, сдавших вступительные экзамены в институт, содержит: ФИО, адрес, оценки. Определить количество абитуриентов, проживающих в Краснодаре, и сдавших экзамены со средним баллом не ниже 4,5, записать их в отдельный список, отсортировать по фамилии и удалить из исходного списка.

30. На междугородней АТС информация о разговорах содержит дату разговора, код и название города, время разговора, тариф, номер телефона в этом городе и номер телефона абонента. Сформировать и вывести хеш-таблицу, где ключом будет код и название города, а значением – общее время разговоров с ним и сумма.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания и решил менее трех задач для самотоятельного решения.

Практическое занятие 63 Наследование классов

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: ознакомиться с основными механизмами наследования классов и интерфейсов изучить понятия виртуальных функций и абстрактных классов в разработке приложений языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Создать класс двумерной фигуры. Используя наследование, создать классы треугольника и прямоугольника с методами вычисления площади, и определения их типов. Поскольку для не определенной заранее двумерной фигуры понятие площади не имеет смысла, в следующей версии предыдущей программы метод area() в классе TwoDShape объявляется как абстрактный, как, впрочем, и сам класс TwoDShape. Безусловно, это означает, что все классы, выведенные из TwoDShape, должны переопределить метод area().

```
using System;
abstract class TwoDShape {
double pri_width; // Закрытый член,
double pri_height; // Закрытый член,
```

```
string pri_name; // Закрытый  член.
public TwoDShape() {// Конструктор по умолчанию,
width = height = 0.0;
name = "null";
}
// Конструктор с параметрами.
public TwoDShape(double w, double h, string n) {
width = w;height = h;name = n;
}
// Создаем объект, у которого ширина равна высоте, public TwoDShape(double x,
string n) {
width = height = x;name = n;
}
// Создаем объект из объекта,
public TwoDShape(TwoDShape ob) {
width = ob.width;height = ob.height; name = ob.name;
}
// Свойства width, height и name,
public double width {
get { return pri_width; }set { pri_width = value; }
}
public double height {
get { return pri_height; }set { pri_height = value; }
}
public string name {
get { return pri__name; }set { pri_name = value; }
}
public void showDim() {
Console.WriteLine("Ширина и высота равны " +
width + " и " + height);
}
// Теперь метод area() абстрактный,
public abstract double area();
}
// Класс треугольников, производный от класса TwoDShape. class Triangle :
TwoDShape {
string style; // Закрытый член.
// Конструктор по умолчанию,
public Triangle() {
style = "null";
}
// Конструктор с параметрами.
public Triangle(string s, double w, double h) :
base(w, h, "triangle") {
style = s;
}
// Создаем равнобедренный треугольник.
public Triangle(double x) : base(x, "треугольник") { style = "равнобедренный";
}
// Создаем объект из объекта. ,
public Triangle(Triangle ob) : base(ob) {
style = ob.style;
}
// Переопределяем метод area() для класса Triangle,
```

```
public override double area() {return width * height / 2 ;}
// Отображаем тип треугольника,
public void showStyle() {
Console.WriteLine("Треугольник " + style);
}
}
```

//Класс прямоугольников, производный от класса TwoDShape. class Rectangle : TwoDShape {

```
// Конструктор с параметрами.
public Rectangle(double w, double h) :
base(w, h, "прямоугольник"){ }
// Создаем квадрат.
public Rectangle (double x) :base(x, "прямоугольник") { }
// Создаем объект из объекта.
public Rectangle(Rectangle ob) : base(ob) { }
// Метод возвращает значение true, если
// прямоугольник является квадратом,
public bool isSquare() {
if (width == height) return true;
return false;
}
// Переопределяем метод area() для класса Rectangle,
public override double area() {
return width * height;
}}
class AbsShape {
public static void Main() {
TwoDShape [] shapes == new TwoDShape[4];
shapes[0] = new Triangle("прямоугольный", 8.0, 12.0); shapes[1] = new Rectangle(10);
shapes[2] = new Rectangle(10, 4);
shapes[3] = new Triangle (7.0);
```

for(int i=0; i < shapes.Length; i++) { Console.WriteLine("Объектом является " +shapes[i].name); Console.WriteLine("Площадь равна " +shapes[i].area()); Console.WriteLine();

```
}}}
```

Задачи для самостоятельного решения:

Заданы названия базовых и производных классов. Необходимо разработать поля и методы, наследуемые из базового класса, и собственные компоненты производных классов. Базовый класс может быть абстрактным.

1-2. Первый базовый класс – средство передвижения. Поля в нем: вес, мощность мотора, скорость. Во втором базовом классе описать страныпроизводители. Производные классы – автомобиль (1 вариант), самолет (2 вариант); производные второго поколения – спортивный автомобиль, грузовой автомобиль (1 вариант), военный самолет, транспортный самолет (2 вариант).

3-4. Первый базовый класс – млекопитающие; поля – способ питания, вес, среда обитания. Во втором базовом классе описываются географические регионы. Производные классы – хищники (3 вариант) и травоядные (4 вариант).

5-6. Первый базовый класс – личность; поля – фамилия, пол, адрес. Во втором базовом классе задается структура университета – факультеты, кафедры, службы. Производные классы – студенты (5 вариант) и сотрудники университета (6 вариант).

7-8. Первый базовый класс – библиотека. Второй базовый класс – система УДК или ключевых слов. Производные классы – техническая (7 вариант) и художественная (8 вариант) литература.

9-10. Первый базовый класс – документ предприятия. Во втором базовом классе описываются корреспонденты предприятия. Производные классы – приказы (9 вариант), письма (10 вариант).

11-12. Первый базовый класс – точка. Производные классы – геометрические фигуры (11 вариант) и текст, который вписывается в фигуру (12 вариант).

13-14. Первый базовый класс – магазин; поля – номер, телефон, адрес, владелец, номер лицензии. Во втором базовом классе описываются поставщики товаров. Производные классы – отделы (13 вариант), продавцы (14 вариант).

15. Первый базовый класс – товар; поля – наименование, количество,

производитель, цена за единицу изделия, общая цена. Во втором базовом классе описываются качественные характеристике товара. Производный класс – товар на складе. Создать массив указателей на класс. Вывести все данные о товарах, отсортированные по возрастанию цены за единицу изделия.

16. Создать абстрактный базовый класс с виртуальной функцией – площадь. Создать производные классы: прямоугольник, круг, прямоугольный треугольник, трапеция со своими функциями площади. Для проверки определить массив ссылок на абстрактный класс, которым присваиваются адреса различных объектов. Площадь трапеции: $S = (a+b)\frac{h}{2}$.

17. Создать абстрактный класс с виртуальной функцией – норма. Создать производные классы: комплексные числа, вектор из 10 элементов, матрица 2x2. Определить функцию нормы: для комплексных чисел – модуль в квадрате, для вектора – корень квадратный из суммы элементов по модулю, для матрицы – максимальное значение по модулю.

18. Создать абстрактный класс «кривые» вычисления координаты y для некоторого x. Создать производные классы: прямая, эллипс, гипербола со своими функциями вычисления y в зависимости от входного параметра x.

Уравнение прямой: $y = ax + b$, эллипса: $\frac{x^2}{a^2} + \frac{y^2}{a^2} = 1$, гиперболы: $\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$

19. Создать абстрактный базовый класс с виртуальной функцией – сумма прогрессии. Создать производные классы: арифметическая прогрессия и геометрическая прогрессия. Каждый класс имеет два поля типа double. Первое – первый член прогрессии, второе – постоянная разность (для арифметической) и постоянное отношение (для геометрической). Определить функцию вычисления суммы, где параметром является количество элементов прогрессии.

Арифметическая прогрессия $a_j = a_0 + jd, j = 0,1,2,...$

Сумма арифметической прогрессии: $s_n = (n+1)\frac{a_0 + a_n}{2}$

Геометрическая прогрессия: $a_j = a_0 r^j, j = 0,1,2,...$

Сумма геометрической прогрессии: $s_n = \frac{a_0 - a_n r}{1-r}$

20. Создать базовый класс «список» с виртуальными функциями вставки и извлечения. Реализовать на базе списка производные классы стека и очереди.

21. Создать базовый класс – фигура, и производные: круг, прямоугольник, трапеция. Определить виртуальные функции: площадь, периметр и вывод на печать.

22. Создать базовый класс – работник, и производные классы – служащий с почасовой оплатой, служащий в штате и служащий с процентной ставкой. Определить функцию начисления зарплаты.

23. Создать абстрактный класс – млекопитающие. Определить производные классы – животные и люди. У животных определить производные классы собак и коров. Определить виртуальные функции описания человека, собаки и коровы.

24. Создать базовый класс – предок, у которого есть фамилия. Определить виртуальную функцию печати. Создать производный класс – ребенок, у которого функция

печати дополнительно выводит имя. Создать производный класс от последнего класса – внук, у которого есть отчество. Написать свою функцию печати.

25. Создать абстрактный базовый класс с виртуальной функцией – корни уравнения. Создать производные классы: класс линейных уравнений и класс квадратных уравнений. Определить функцию вычисления корней уравнений.

Контрольные вопросы:

1. В чем заключается механизм простого наследования? Как описать производный класс?
2. Как соблюдается принцип инкапсуляции (сокрытия данных) при простом наследовании?
3. Можно ли в классе-наследнике переопределять компонентные функции базового класса? Как это сделать?
4. Что такое полиморфизм? В каких случаях он применяется?
5. Нужно ли объявлять функцию виртуальной в производном классе, если она объявлена таковой в базовом классе?
6. Дайте определение чистой виртуальной функции.
7. В каких случаях целесообразно создание абстрактных классов?
8. Можно ли создать объект абстрактного класса? А указатель на него?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 64 Обработка исключений

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: ознакомиться с понятием исключительных ситуаций и способами их обработки, изучить методы создания классов исключений и их объектов, способов формирования исключений в программей языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

1 Известно, что попытка индексировать массив за пределами его границ вызывает ошибку нарушения диапазона. В этом случае C#-система динамического управления генерирует исключение типа IndexOutOfRangeException, которое представляет собой стандартное исключение, определенное языком C#.

```
using System; // Демонстрация обработки исключений,
class ExcDemo1 {
public static void Main() {
int[] nums = new int [4];
try {
Console.WriteLine(
"Перед генерированием исключения.");
// Генерируем исключение, связанное с попаданием
// индексавнедиапазона,
for(int i=0; i < 10; i++) {
nums[i] = i;
Console.WriteLine("nums[{0}]: {1}", i, nums[i]);
}
Console.WriteLine("Этот текст не отображается.");
}
catch (IndexOutOfRangeException) {
// Перехватываем исключение.
Console.WriteLine("Индекс вне диапазона!");
}
Console.WriteLine("После catch-инструкции.");
}}
```

При выполнении этой программы получаем такие результаты: Перед генерированием исключения.

```
nums[0]: 0
nums[1]: 1
nums[2]: 2
nums[3]: 3
Индекс вне диапазона!
```

2 Например, следующая программа перехватывает как ошибку нарушения границ массива, так и ошибку деления на нуль.

```
// Использование нескольких catch-инструкций.
usingSystem;
class ExcDemo3 {
public static void Main() {
// Здесь массив numer длинне массива denom.
int [] numer = { 4, 8, 16, 32, 64, 128, 256, 512 };
int[] denom = { 2, 0, 4, 4, 0 , 8 };
for(int i=0; i < numer.Length;i++){
try {
Console.WriteLine(numer[i] + " / " +denom[i] + " равно " +
numer[i]/denom[i]);
}
catch (DivideByZeroException) {// Перехватываем исключение
Console.WriteLine("Делитьнанульнельзя!");
}
catch (IndexOutOfRangeException) {
// Перехватываем исключение.
Console.WriteLine("Нет соответствующего элемента.")
}}}}
```

Эта программа генерирует следующие результаты:

```
4/2
равно 2
Делить на нуль нельзя!
16/4
равно 4
```

```
32/4
равно 8
Делить на нуль нельзя!
128/8 равно 16
Нет соответствующего элемента.
Нет соответствующего элемента.
3 Использование catch-инструкции для “глобального перехвата". using System;
class ExcDemo4 {
public static void Main() {
// Здесь массив numer длиннее массива denoiti.
int[] numer = { 4, 8, 16, 32, 64, 128, 256, 512 };
int[] denom = { 2 , 0, 4, 4, 0, 8 } ;
for(int i=0; i < numer.Length; i++){
try {
Console.WriteLine(numer[i] + " / " +
denom[i] + " равно " +
numer[i]/denom[i]);
}
catch {
Console.WriteLine(
"Произошло некоторое исключение.");
}}}}
```

Вот как выглядят результаты выполнения этой программы:

```
4/2
равно 2
Произошло некоторое исключение. 16/4 равно 4
32/4 равно 8
Произошло некоторое исключение. 128/8 равно 16
Произошло некоторое исключение. Произошло некоторое исключение.
```

Задачи для самостоятельного решения:

Для каждого варианта необходимо создать три массива a, b и c размерами соответственно n1, n2 и n3 (n1 n2 n3). В массив a занести значения функции f(x) согласно варианту (при возникновении исключения заносить нули). Массив b заполнить случайными числами (среди них должны быть и отрицательные числа и нули). Массив c формируется согласно варианту. Предусмотреть и обработать возникающие при этом исключительные ситуации (деление на ноль, корень из отрицательного числа, арифметическое переполнение, выход за пределы диапазона индексов массива и т.п.). Варианты заданий приведены в таблице.

Таблица – Варианты заданий

Вариант	Функция *f(x)*	Способ формирования массива *c*
1.	$\ln(x-1), x \in [0;10], \Delta x = 0{,}5$	$c_i = a_i + 1/b_i$
2.	$\ln(x^2-1), x \in [0;4], \Delta x = 0{,}2$	$c_i = a_i - 1/b_i$
3.	$\ln(x-1)^3, x \in [0;6], \Delta x = 0{,}3$	$c_i = a_i / b_i$
4.	$\ln(x+1)^3, x \in [-3;3], \Delta x = 0{,}3$	$c_i = a_{i-1} + 1/b_{i+1}$
5.	$\ln(1-x), x \in [-1;3], \Delta x = 0{,}2$	$c_i = a_{i+1} - 1/b_{i-1}$
6.	$\ln\frac{1}{x+1}, x \in [-2;4], \Delta x = 0{,}3$	$c_i = a_i / b_{i+1}$
7.	$\ln\frac{1}{x-1}, x \in [-1;9], \Delta x = 0{,}5$	$c_i = \sqrt{a_i \cdot b_i}$
8.	$\lg\frac{x-1}{x+1}, x \in [-3;7], \Delta x = 0{,}5$	$c_i = \sqrt{a_i + b_i}$
9.	$\lg((x-1)\cdot(x+1)), x \in [-3;3], \Delta x = 0{,}3$	$c_i = \sqrt{\frac{a_i}{b_i}}$
10.	$\lg\frac{1}{x-1}, x \in [-2;4], \Delta x = 0{,}3$	$c_i = \sqrt{a_i - b_{i-1}}$
11.	$\ln\frac{x+1}{x-1}, x \in [-2;8], \Delta x = 0{,}5$	$c_i = \sqrt{a_{i+1} \cdot b_{i+1}}$
12.	$\lg\frac{1}{x+1}, x \in [-2;2], \Delta x = 0{,}2$	$c_i = \sqrt{a_{i-1} + b_{i+1}}$

13.	$\ln((x-1)\cdot(x+1)), x\in[-3;5], \Delta x=0{,}4$	$c_i=\sqrt{\frac{a_{i+1}}{b_{i-1}}}$
14.	$\lg(1-x)^3, x\in[-3;3], \Delta x=0{,}3$	$c_i=\sqrt{a_{i+1}-b_i}$
15.	$\sqrt{x^2-1}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\ln(a_i+b_i)$
16.	$\sqrt{x^3+1}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\ln(\frac{a_i}{b_i})$
17.	$\sqrt{1-x^2}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\lg(a_i-b_i)$
18.	$\sqrt{1-x^3}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\ln(a_{i-1}+b_{i+1})$
19.	$\sqrt{\frac{1}{x^2}}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\ln(\frac{a_{i+1}}{b_{i-1}})$
20.	$\frac{1}{x-2}, x\in[-1;5], \Delta x=0{,}3$	$c_i=\lg(a_i-\frac{1}{b_i})$
21.	$\frac{1}{x+1}, x\in[-2;0], \Delta x=0{,}1$	$c_i=a_{i+1}+1/(b_{i+1}-1)$
22.	$\frac{1}{x^2-1}, x\in[-2;0], \Delta x=0{,}1$	$c_i=a_{i-!}-2/(b_{i-1}+1)$
23.	$\sqrt{\frac{1}{x^3-2}}, x\in[0;10], \Delta x=0{,}5$	$c_i=\sqrt{a_{i-1}+b_i-1}$
24.	$\frac{1}{\sqrt{1-x^2}}, x\in[-3;3], \Delta x=0{,}3$	$c_i=\sqrt{\frac{a_{i+1}}{2b_i}}$
25.	$\sqrt{x^2-1}, x\in[-2;2], \Delta x=0{,}2$	$c_i=\sqrt{1-a_{i+1}\cdot b_i}$
26.	$\arcsin(x+1), x\in[-3;1], \Delta x=0{,}2$	$c_i=\sqrt{2(a_{i-1}+b_i)}$
27.	$\arccos(\frac{x}{2}-1), x\in[-1;5], \Delta x=0{,}3$	$c_i=\ln\frac{2a_i}{b_{i-1}+1}$
28.	$\arcsin(\frac{2}{x}+1), x\in[-2;0], \Delta x=0{,}1$	$c_i=\lg(a_{i+1}+\frac{2}{b_i})$

Контрольные вопросы:

1. Что такое исключительная ситуация? Приведите примеры программных исключений.
2. Как выделить в программе контролируемый блок?
3. С помощью какого ключевого слова осуществляется генерация исключительных ситуаций? Укажите существующие формы этого оператора.
4. Перечислите известные Вам формы описания обработчиков исключений.
5. Каким образом осуществляется обработка исключений во вложенных контролируемых блоках?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 65 Разработка приложений с применением управляемых провайдеров ADO.NET

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить общую структуру и основные виды провайдеров технологии ADO.NET. Научиться применять классы и методы, используемые при работе с управляемым провайдером OLE DB языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

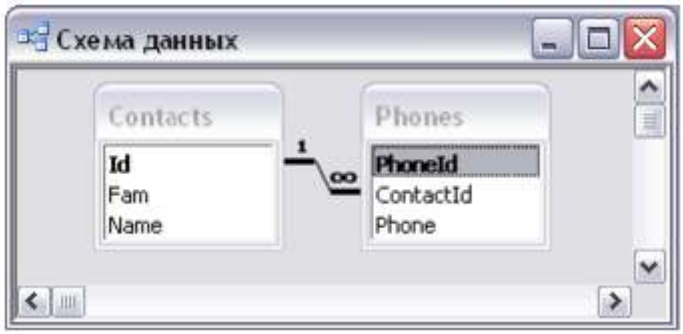
Порядок выполнения работы:

База данных из двух связанных таблиц «Телефонная книжка» создана в Microsoft Access. Реализовать с помощью управляемого провайдера OLE DB доступ к этой базе данных.

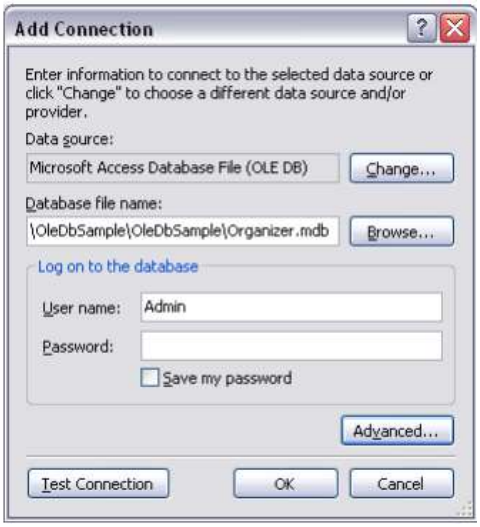
База данных Organizer состоит из двух связанных таблиц: Contacts и Phones. Структура таблиц приведена ниже:

Название поля	Тип поля	Размер поля	Примечание	Назначение поля
Таблица Contacts				
Id	Счетчик	Длинное целое	Ключевое поле	Номер записи
Fam	Текстовый	20		Фамилия
Name	Текстовый	20		Имя
Таблица Phones				
PhoneId	Счетчик	Длинное целое	Ключевое поле	Номер записи
ContactId	Числовой	Длинное целое		Идентифика-тор контакта
Phone	Текстовый	20		Телефон

Таблицы связываются между собой по номеру записи в таблице контактов. Данная связь показана на рисунке/

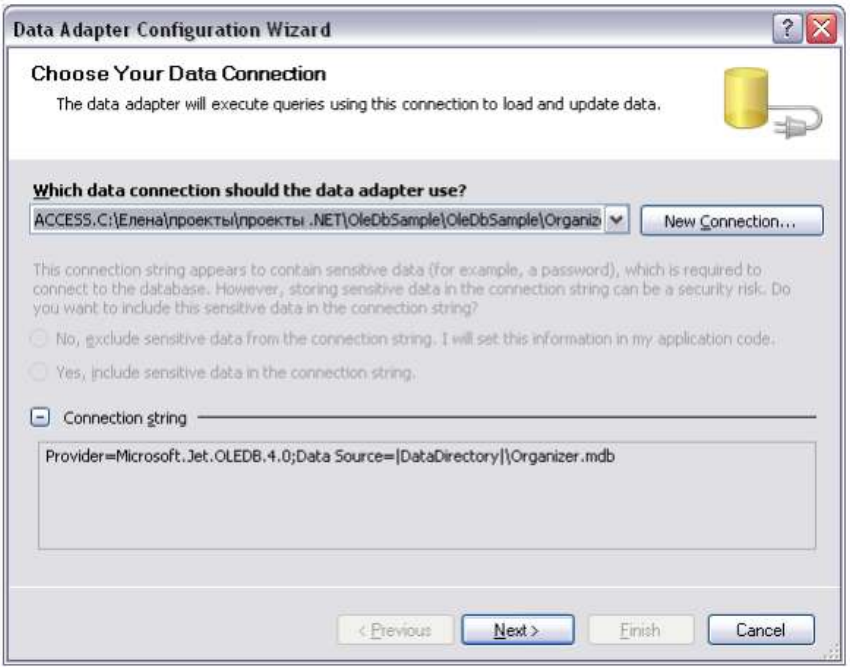


Прежде всего, следует поместить на форму компонент oleDbConnection и настроить его свойство ConnectionString, выбрав пункт <New connec-tion…>. Открывается диалоговое окно, представленное на рисунке 3.3. С помощью кнопки Change… в правом верхнем углу окна надо установить вид источника данных (Data source) как Microsoft Access Database file. Далее с помощью кнопки Browse… в строке Database file name нужно указать путь и имя файла с базой данных. Кнопка Test Connection позволяет проверить правильность установления соединения.



Следующим шагом является создание адаптера – компонента, выполняющего непосредственную передачу данным между приложением и базой данных. Для начала необходимо поместить на форму компонент OleDbDataAdapter. При этом на экране появляется окно мастера настройки адаптера. Здесь следует проверить правильность указания пути к файлу БД.

Примечание. После нажатия кнопки Next мастером настройки будет предложен вариант работы с базой данных, при котором при каждом запуске программе будет создаваться копия базы данных в папке приложения. На данном шаге следует нажать кнопку «NO»!

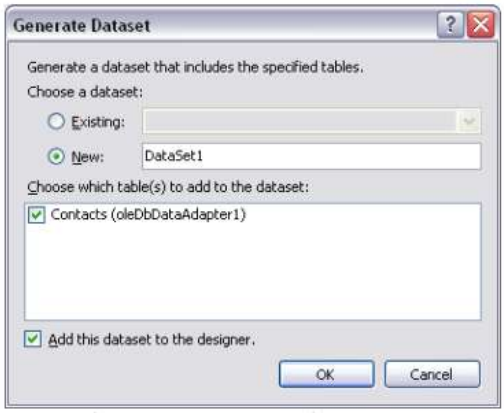


Далее мастер предлагает задать команды SQL для загрузки данных в приложение. Кнопка Query Builder… данного окна предлагает визуальный метод построения данной команды. В открывшемся окне Add Table следует добавить в Query Builder главную таблицу БД – таблицу Contacts, а затем отметить в этой таблице все ее столбцы (рис. 3.5).

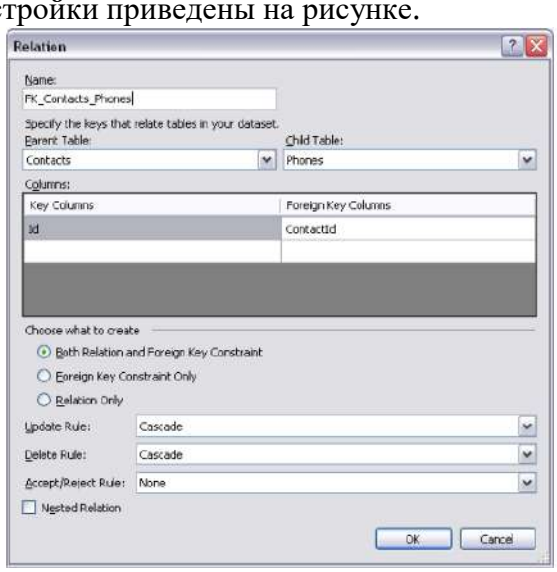
На этом создание адаптера таблицы Contacts завершается. Следующим шагом является формирование набора данных – объекта DataSet. Для этого нужно в режиме дизайнера формы вызвать контекстное меню и выбрать в нем пункт Generate DataSet… Открывается соответствующее диалоговое окно, с помощью которого создается новый набор данных.

Аналогичным образом добавляется адаптер и для второй таблицы БД – таблицы Phones. Процесс создания данного адаптера аналогичен адаптеру таблицы Contacts (в окно Query Builder, естественно, нужно добавлять таблицу Phones). Чтобы добавить таблицу

Phones в DataSet, следует щелкнуть правой кнопкой мыши на адаптере данной таблицы, в контекстном меню выбрать Generate DataSet… и в открывшемся диалоговом окне выбрать существующий объект DataSet (т.е. оставить настройки по умолчанию).



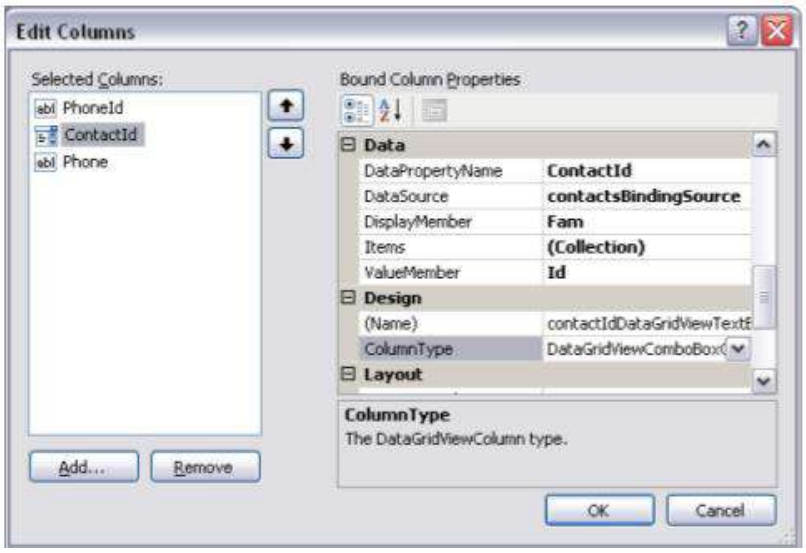
После этого в сгенерированный набор данных можно добавлять другие необходимые элементы: остальные таблицы БД, связи между ними, запросы и т.п. Для этого нужно в окне Solution Expolrer выбрать (дважды щелкнуть мышью) элемент DataSet1.xsd. Так, для нашего приложения в набор данных следует добавить отношение – связь между таблицами БД: в окне DataSet1.xsd в контекстном меню выбирается пункт Add->Relation… Открывается диалоговое окно редактирования отношения, где можно задать имя создаваемого отношения, родительскую и подчиненную таблицы, ключевые поля, а также режимы автоматических изменений дочерней таблицы при изменении родительской таблицы. Для разрабатываемого приложения создается отношение с именем FK_Contacts_Phones. Его настройки приведены на рисунке.



Внешний вид работающего приложения приведен на рисунке.

Для отображения данных, полученных из базы, в программе будут использоваться компоненты DataGridView, по одному для каждой таблицы базы данных. Имена их по умолчанию устанавливаются как dataGridView1 и dataGridView2.

Компонентам-таблицам dataGridView1 и dataGridView2 нужно установить свойство DataSource, задающее источник данных для отображения в таблице, в «contactsBindingSource» и «phonesBindingSource» соответственно. Кроме того, при необходимости можно отредактировать заголовки и другие настройки столбцов таблиц (свойство Columns). В данном приложении в столбце ContactId компонента dataGridView2 для наглядности отображается не числовой идентификатор человека, которому принадлежит телефон, а его фамилия, причем для отображения используется тип столбца «DataGridViewComboBoxColumn» (свойство ColumnType). Настройка остальных свойств данного столбца приведена на рисунке.



Для отображения в компонентах DatagridView данных из таблиц необходимо прежде всего открыть настроенное соединение с базой данных, а затем заполнить таблицы, созданные в объекте DataSet данными из таблиц в базе. В нашем случае делается это при загрузке главного окна программы, в обработчике события Load:

```
private void Form1_Load(object sender, EventArgs e)
{
   oleDbConnection1.Open(); //открыть соединение
   //заполнить таблицы в объекте DataSet
   oleDbDataAdapter1.Fill(dataSet11.Contacts);
   phonesTableAdapter.Fill(dataSet11.Phones);
}
```

При закрытии формы необходимо отключить соединение с базой данных:

```
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
   oleDbConnection1.Close();   //закрыть соединение
}
```

Задачи для самостоятельного решения:

В каждом варианте необходимо разработать базу данных минимум из двух связанных между собой таблиц. В каждой таблице – не менее трех полей. Реализовать доступ к созданной базе с помощью управляемого провайдера OLE DB.

№ варианта	БД	Таблицы
1.	Студенты университета	Студенты, факультеты, специальности и т.п.
2.	Склад магазина	Товары, поставщики, категории товаров и т.п.
3.	Персонал предприятия	Сотрудники, отделы, документы отдела кадров и т.п.
4.	Владельцы автомобилей	Автомобили, автовладельцы и т.д.
5.	Библиотека	Книги, читатели, книги на руках у читателей и т.п.
6.	Очередь на жилье	Список жилья, список очередников и т.д.
7.	Аптека	Лекарства, категории лекарств, виды болезней и т.п.
8.	Касса аэропорта	Рейсы, проданные билеты и т.д.
9.	Банковские кредиты	Заемщики, виды кредитов, поручители и т.п.
10.	Гостиница	Список номеров, категории номеров, постояльцы и т.п.
11.	Риэлтерская фирма	Квартиры, покупатели, сделки и т.д.
12.	Справка по языку C#	Пространства имен, классы, методы и т.п.
13.	Учет операций с акциями	Виды акций, владельцы, операции и т.д.
14.	Таксопарк	Транспортные средства, водители, рейсы и т.п.
15.	Кафе	Продукты, рецепты, поставщики и т.д.
16.	АЗС	Виды топлива, поставщики, продажи и т.п.
17.	Поликлиника	Врачи, пациенты, консультации и т.д.
18.	Семейный бюджет	Виды поступлений, виды затрат, покупки и т.д.
19.	Табель рабочего времени	Сотрудники, виды работ, табель и т.п.
20.	Туристическая фирма	Виды туров, клиенты, заказы и т.п.
21.	Справочник географа	Континенты, страны, реки, моря и т.д.
22.	ЖЭК	Дома, жильцы, виды обслуживания, заявки и т.п.
23.	Адвокатура	Адвокатские конторы, адвокатские услуги, адвокаты
24.	Пресса	Виды периодического издания, издания, авторы и т.д.
25.	Банкомат	Карточки, виды операций, совершенные операции и т.п.
26.	Касса стадиона	Матчи, категории билетов, проданные билеты и т.д.
27.	АТС	Абоненты, категории звонков, звонки и т.п.
28.	Web-форум	Посетители, темы, сообщения и т.д.
29.	Кинопрокат	Кинотеатры, фильмы в прокате, жанры и т.п.
30.	Учебные курсы	Области знаний курсов, преподаватели, курсы и т.д.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания и решил менее трех задач для самотоятельного решения.

Практическое занятие 66 Доступ к данным с помощью ADO.NET

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить принципы доступа к данным с помощью технологии ADO.NET. Получить навыки работы с пространствами имен, классами, методами, используемыми для работы с данными языка C#

Оборудование: ноутбук Dell Inspiron 15 5000 Series

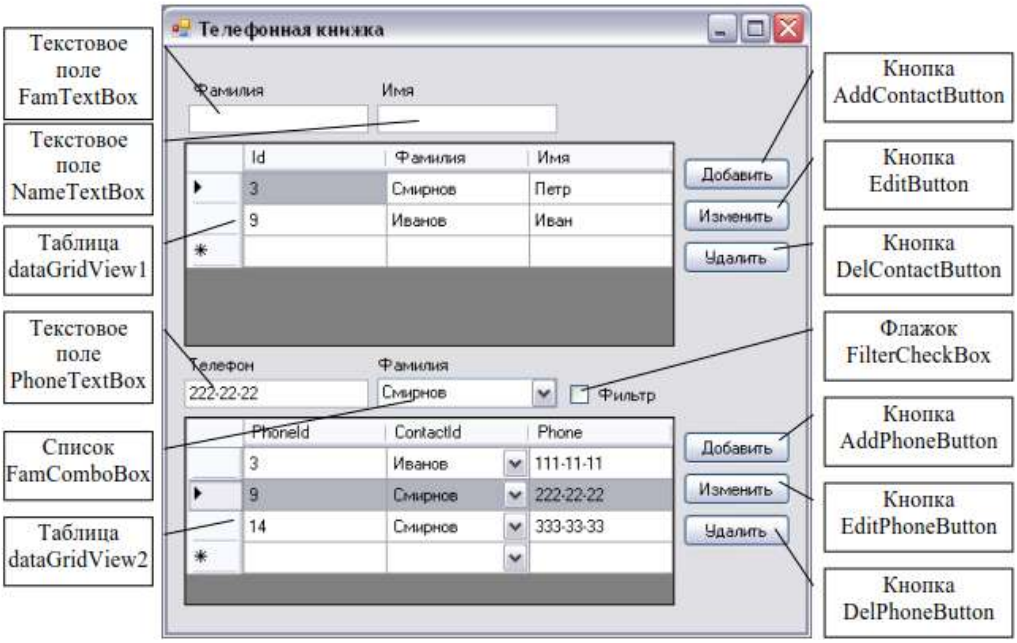
Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Реализовать с помощью технологии ADO.NET программный интерфейс управления базой данных, созданной в предыдущей лабораторной работе. Предусмотреть возможность добавления, редактирования, удаления, фильтрации записей. Внешний вид работающего приложения приведен на рисунке.

Настройки таблиц dataGridView1 и dataGridView1 приведены в описании предыдущей лабораторной работы.

Текстовые поля FamTextBox и NameTextBox используются для добавления новой записи или редактирования записи, выделенной в таблице. Для удобства редактирования при выделении строки в таблице данные об имени и фамилии человека копируются в соответствующие текстовые поля.



Обновлять содержимое главного окна необходимо, когда пользователь производит какие-либо изменения с записями базы данных: добавление, редактирование, удаление записей таблиц, а также при открытии соединения с базой данных в начале работы программы.

Добавить в класс новый метод можно либо вручную, либо с использованием визуальных средств разработки. Для этого выбрать в меню View пункт Class View. В результате откроется окно, где можно просматривать список всех классов, описанных в пространстве имен приложения. Щелкнув правой кнопкой на класс главной формы, следует выбрать в контекстном меню пункт View Class Diagram. После этого в рабочей области откроется файл диаграммы классов, где можно создавать новые классы, интерфейсы, делегаты, перечисления и т.п., а также добавлять в существующие классы методы, поля, свойства, события и т.д.

Чтобы добавить в класс метод, нужно щелкнуть правой кнопкой в заголовке класса формы и выбрать в контекстном меню пункт Add->Method. После этого в окне Properties можно будет установить свойства для добавля-емого метода: его имя, вид доступа, тип возврата и т.д.

Для обновления содержимого формы следует добавить в класс формы метод UpdateContacts(). Тип возврата метода – void, вид доступа – pri-vate, параметров нет. Программный код метода следующий:

```
private void UpdateContacts()
{
   //обновить содержимое таблиц базы данных
   phonesTableAdapter.Update(dataSet11);
   oleDbDataAdapter1.Update(dataSet11);
   //очистить DataSet
   dataSet11.Clear();
   //заполнить таблицы в объекте DataSet
   oleDbDataAdapter1.Fill(dataSet11.Contacts);
   phonesTableAdapter.Fill(dataSet11.Phones);
   //очистить содерижмое списка фамилий
   FamComboBox.Items.Clear();
   //заполнить список фамилий значениями из таблицы
   foreach (DataRow row in dataSet11.Contacts.Rows)
      FamComboBox.Items.Add(row["Fam"]);
   //в списке фамилий - ни одна фамилия не выделена
```

```
    FamComboBox.Text = "";
}
```

Прежде всего, необходимо сохранить сделанные в программе изменения непосредственно в базу данных. Делается это с помощью метода Update(), применяемого к адаптерам таблиц. Далее содержимое объекта DataSet очищается (Clear()) и заполняется заново методом Fill().

Выпадающий список FamComboBox содержит список фамилий людей. При изменении содержимого базы данных его также нужно обновить. Сначала его строки методом Clear() очищаются, а затем заполняются заново фамилиями из обновленной таблицы (метод Add()). Цикл foreach используется для просмотра всех записей таблицы.

Так как обновление данных должно выполняться и при запуске программы, то содержимое обработчика события FormLoad изменится следующим образом:

```
private void Form1_Load(object sender, EventArgs e)
{
    oleDbConnection1.Open(); //открыть соединение
    UpdateContacts();       //обновить главное окно
}
```

Добавление нового контакта происходит тогда, когда пользователь щелкает кнопку Добавить, расположенную в верхней части главного окна приложения. Предварительно пользователь должен ввести имя и фамилию человека в поля Имя и Фамилия соответственно.

Для этой кнопки необходимо подготовить следующий программный код:

```
private void AddContactButton_Click(object sender, EventArgs e)
{
//если текстовые поля не пусты,
    if (FamTextBox.Text != "" && NameTextBox.Text != "")
    {
        //то создать новую запись в таблице Contacts,
        DataRow row = dataSet11.Contacts.NewRow();
        //заполнить ее столбцы
        row["Fam"] = FamTextBox.Text;
        row["Name"] = NameTextBox.Text;
        //и добавить запись в таблицу
        dataSet11.Contacts.Rows.Add(row);
        //обновить содержимое главного окна
        UpdateContacts();
    }
}
```

Здесь нужно убедиться в том, что пользователь ввел как имя, так и фамилию. Если поле fnTextBox или lnTextBox пустое, обработчик события завершает свою работу без выполнения других дополнительных действий. В том случае, когда пользователь ввел все необходимые данные, обработчик события создает новую строку в таблице Contacts. Для этого используется метод NewRow(). На следующем этапе введенные строки имени и фамилии записываются в соответствующие столбцы строки. Подготовленная таким способом строка добавляется в таблицу.

Далее обработчик события последовательно вызывает метод с именами UpdateContacts(), который предназначен для обновления содержимого формы приложения. Этот метод описан далее.

Выделив в верхнем списке имя человека, пользователь может добавить

для него один или несколько телефонов. Эта операция выполняется при помощи кнопки Добавить, расположенной возле списка телефонов. Вот обработчик событий для этой кнопки:

```
private void AddPhoneButton_Click(object sender, EventArgs e)
{
```

```
    //если поле ввода телефона не пустое
    //и выбрана какая-либо фамилия
    if (PhoneTextBox.Text != "" && FamComboBox.Text != "")
    {
       try
       {
          //создать новую запись таблицы Phones
          DataRow row = dataSet11.Phones.NewRow();
          //заполнить столбец "номер телефона"
          row["Phone"] = PhoneTextBox.Text;
          //получить из выпадающего списка фамилию,
  //для которой добавляется телефон
          string fio = FamComboBox.SelectedItem.ToString();
          //составить условие для поиска этого человека
//в таблице Contacts
          string str = "Fam='" + fio + "'";
          //получить id этого человека в таблице Contacts
          DataRow[] contacts = dataSet11.Contacts.Select(str);
          //заполнить столбец "ContactId" добавляемой записи
          row["ContactId"] = contacts[0]["id"];
          //добавить запись в таблицу
          dataSet11.Phones.Rows.Add(row);
          //обновить форму
          UpdateContacts();
       }
       catch (Exception)
       {
       }
    }
}
```

Здесь вначале проверяется, что поле нового телефона phoneTextBox не пустоеи в выпадающем списке выбран человек, для которого будет добавляться телефон. Если это так, то путем вызова метода NewRow() в таблице Phones создается новая строка как объект класса DataRow.

Номер добавляемого телефона сохраняется в столбце Phone. Что же касается столбца ContactId, то в него нужно записать идентификатор строки таблицы Contacts (грубо говоря, порядковый номер человека, которому добавляется телефон). Для этого прежде всего нужно получить фамилию человека из списка FamComboBox с помощью свойства SelectedItem, затем найти этого человека по фамилии в таблице Contacts и получить его id. Поиск данных в таблице осуществляется с помощью метода Select(), где в качестве параметра выступает условие отбора данных. Это условие записывается в строку str и имеет вид: Fam = ‘<фамилия человека>’. Столбец Con-tactId новой записи заполняется найденным значением.

Заполненная строка добавляется в таблицу Phones методом Add().

И, наконец, после добавления строки обработчик событий обновляет содержимое базы данных в окне программы методом UpdateContacts().

Если требуется изменить или удалить данные из базы, то для этого необходимо указать запись, с которой будут производиться эти действия. В данном приложении запись выбирается путем выделения соответствующей строки компонента DataGridView. При этом желательно для обеих таблиц на форме установить свойство MultiSelect в false, чтобы запретить выбор более чем одной строки.

Номер выбранной записи будет сохраняться в переменных: RowId для таблицы Contacts и PhoneId для таблицы Phones. Эти переменные следует добавить в класс главной

формы либо визуальным способом (как описано в пункте 2.2.1), либо вручную следующим образом:

```
private int RowId;
private int PhoneId;
```

Выбор нужной записи осуществляется в обработчике события Row-HeaderMouseClick компонентов DataGridView. Это событие происходит, когда пользователь щелкает мышью в столбце заголовка строки, т.е. выделяет запись. Код данного обработчика для таблицы Contacts представлен ниже:

```
private void dataGridView1_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
{
   try
   {
      //получить номер выделенной строки
      RowId = e.RowIndex;
      //отобразить фамилию и имя выбранного человека
//в текстовых полях
FamTextBox.Text =
  dataSet11.Contacts.Rows[RowId]["Fam"].ToString();
      NameTextBox.Text =
dataSet11.Contacts.Rows[RowId]["Name"].ToString();
   }
   catch (Exception)
   {
   }
}
```

Когда пользователь выбирает строку в таблице, этот обработчик вначале определяет индекс строки, которая стала выделенной в результате выполнения этой операции, и записывает его в переменную RowId. Здесь параметр e обработчика содержит параметры, описывающие характеристики нажатия мышью заголовка строки, в том числе и номер строки (свойство RowIndex). После получения номера выбранной строки выполняется копирование полей фамилии и имени человека из таблицы в соответствующие текстовые поля.

Код аналогичного обработчика для таблицы Phones выглядит следующим образом:

```
private void dataGridView2_RowHeaderMouseClick(object sender,
DataGridViewCellMouseEventArgs e)
{
   try
   {
      //получить номер выделенной строки
      PhoneId = e.RowIndex;
      //найти запись с полученным номером в DataSet
      DataRow p = dataSet11.Phones.Rows[PhoneId];
      //отобразить номер телефона найденной записи в текстовом поле
      PhoneTextBox.Text = p["Phone"].ToString();
      //получить записи из таблицы Contacts, связанные с данной
DataRow[] fams = p.GetParentRows( dataSet11.Relations["FK_Contacts_Phones"]);
      //выбрать фамилию человека (из текущей записи) в списке
      FamComboBox.SelectedItem = fams[0]["Fam"].ToString();
   }
   catch (Exception)
   {
   }
}
```

Начало обработчика сходно с описанием предыдущего метода: получение номера выделенной строки в переменную PhoneId и копирование номера телефона найденной записи в текстовое поле. Но в данном случае необходимо также отобразить (сделать выбранным) в компоненте ComboBox, фамилию человека, к которому относится этот номер телефона. Т.к. фамилии находятся в таблице Contacts, то нужно получить список записей, связанных с выбранной через отношение FK_Contacts_Phones (см. предыдущую лабораторную работу). Делается это с помощью метода GetParentRows(). Далее из полученных «родительских» записей извлекается поле фамилии (“Fam”), и данная фамилия становится выбранной в выпадающем списке (свойство SelectedItem).

Для того чтобы отредактировать имя или фамилию человека, нужно выделить требуемую строку в таблице DataGridView1, содержащей записи таб-лицы Contacts. После этого только что рассмотренный обработчик события RowHeaderMouseClick запишет имя и фамилию в текстовые поля редактирования Имя и Фамилия соответственно.

Отредактировав имя или фамилию, пользователь должен щелкнуть кнопку Изменить. Ниже приведен исходный текст обработчика событий для этой кнопки, изменяющего содержимое ячеек соответствующей строки в таблице Contacts.

```
private void EditButton_Click(object sender, EventArgs e)
{
   //если выбрана строка в таблице
   if (dataGridView1.SelectedRows.Count != 0)
   {
      //получить содержимое выбранной строки
      DataRow row = dataSet11.Contacts.Rows[RowId];
      //изменить фамилию и имя на введенные значения
      row["Fam"] = FamTextBox.Text;
      row["Name"] = NameTextBox.Text;
      //сохранить изменения и обновить содержимое формы
      UpdateContacts();
   }
   else
      MessageBox.Show("Выберите строку для редактирования",
"Ошибка");
}
```

Здесь проверяется, выбрана ли строка для редактирования. Если нет, то выдается сообщение об ошибке и обработчик завершает работу. Иначе надо получть содержимое выбранной строки в объект типа DataRow. Далее значения полей Fam и Name этой строки заменяются на новые, введенные пользователем в текстовых полях, и вызывается метод UpdateContacts() для сохранения изменений в БД и обновления таблиц на форме.

Т.к. таблицы БД связаны между собой, и при связывании было указано каскадное изменение и удаление записей, то после сохранения изменений в БД эти изменения отобразятся и в таблице Phones.

Для изменения номера телефона используется кнопка Изменить, расположенная справа от списка телефонов. Исходный текст обработчика событий для этой кнопки представлен ниже:

```
private void EditPhoneButton_Click(object sender, EventArgs e)
{
   //если выбрана запись для редактирования
   if (dataGridView2.SelectedRows.Count != 0)
   {
      try
      {
         //получить содержимое выбранной строки
         DataRow row = dataSet11.Phones.Rows[PhoneId];
```

```
            //изменить номер телефона на введенное значение
            row["Phone"] = PhoneTextBox.Text;
            //получить из выпадающего списка фамилию,
  //для которой добавляется телефон
            string fio = FamComboBox.SelectedItem.ToString();
            //составить условие для поиска этого человека
  //в таблице Contacts
            string str = "Fam='" + fio + "'";
            //получить id этого человека в таблице Contacts
            DataRow[] contacts = dataSet11.Contacts.Select(str);
            //заполнить столбец "ContactId" редактируемой записи
            row["ContactId"] = contacts[0]["id"];
        }
        catch (Exception)
        {
        }
        //сохранить изменения и обновить содержимое формы
        UpdateContacts();
    }
    else
        MessageBox.Show("Выберите строку для редактирования",
"Ошибка");
}
```

Начало обработчика идентично предыдущему: если выбрана строка для редактирования, то получить ее содержимое и изменение значение номера телефона.

Далее, если нужно изменить и человека, к которому этот номер телефона относится, то используется технология, описанная ранее: поиск человека в таблице Contacts с помощью метода Select(), получение его порядкового номера (id) и заполнение этим значением поля ContactId таблицы Phones.

Для того чтобы удалить строку, выделенную пользователем в списке контактов, нужно воспользоваться кнопкой Удалить. Ниже приведен исходный текст метода, выполняющего обработку событий от этой кнопки:

```
private void DelContactButton_Click(object sender, EventArgs e)
{
    //если выбрана запись для удаления
    if (dataGridView1.SelectedRows.Count != 0)
    {
        if (MessageBox.Show("Вы действительно хотите удалить
запись?", "Подтверждение",
MessageBoxButtons.YesNo) == DialogResult.Yes)
        {
            try
            {
                //удалить выбранную строку
                dataSet11.Contacts.Rows[RowId].Delete();
            }
            catch (Exception)
            {
            }
            //обновить БД и ее содержимое на форме
            UpdateContacts();
        }
    }
    else
```

```
        MessageBox.Show("Выберите строку для удаления",
    "Ошибка");
    }
```

После проверки, выбрана ли в таблице строка для удаления и вывода подтверждающего сообщения, непосредственное удаление записи выполняется методом Delete(), после чего изменения сохраняются в базу данных и отображаются на форме.

При связывании таблиц Contacts и Phones в списке Delete rule было выбрано значение Cascade, задающее режим автоматического внесения изменений в дочернюю таблицу при удалении записей родительской таблицы.

Поэтому при удалении строк родительской таблицы происходит удаление соответствующих строк дочерней таблицы.

Для удаления номера телефона предназначена кнопка Удалить, расположенная возле списка телефонов. Далее приведен обработчик событий для этой кнопки:

```
    private void DelPhoneButton_Click(object sender, EventArgs e)
    {
       //если выбрана запись для удаления
       if (dataGridView2.SelectedRows.Count != 0)
       {
          if (MessageBox.Show("Вы действительно хотите удалить
    запись?", "Подтверждение",
    MessageBoxButtons.YesNo) == DialogResult.Yes)
          {
             try
             {
                //удалить выбранную строку
                dataSet11.Phones.Rows[PhoneId].Delete();
             }
             catch (Exception)
             {
             }
             //обновить БД и ее содержимое на форме
             UpdateContacts();
          }
       }
       else
          MessageBox.Show("Выберите строку для удаления",
    "Ошибка");
    }
```

Код обработчика идентичен предыдущему (естественно, работа идет с таблицей Phones и компонентом dataGridView2).

В данном приложении приведен пример фильтра, который позволяет просматривать номера телефона одного, выбранного человека. Фамилия этого человека выбирается в выпадающем списке FamComboBox. Режим филь-трации включается флажком FilterCheckBox. Код обработки щелчка по этому флажку приведен ниже:

```
    private void FilterCheckBox_CheckedChanged(object sender, EventArgs e)
    {
       //если установлен флажок и выбрана фамилия
       if (FilterCheckBox.Checked && FamComboBox.Text != "")
    {
          //получить выбранную фамилию
          string fio = FamComboBox.SelectedItem.ToString();
          //составить условие для поиска нужного человека
      //в таблице Contacts
          string str = "Fam='" + fio + "'";
```

```
            //найти нужного человека в таблице Contacts
            DataRow[] contacts = dataSet11.Contacts.Select(str);
            //составить условие для фильтра
            str = "ContactId=" + contacts[0]["id"];
            //применить фильтр
            phonesBindingSource.Filter = str;
        }
        else
            //отменить фильтрацию
            phonesBindingSource.Filter = "";
    }
```

В начале работы метода проверяется, установил ли пользователь флажок фильтрации и задал ли требуемую фамилию.

Так как фильтрация будет задаваться для таблицы Phones, а поле «фамилия» находится в таблице Contacts, необходимо получить идентификатор (порядковый номер) требуемой фамилии, чтобы потом найти его в таблице телефонов. Получить выбранную фамилию из выпадающего списка можно методом SelectedItem, а отобрать запись с этой фамилией из таблицы Contacts – методом Select().

Далее из полученной записи выделяется значение поля id, на основе которого задается условие для фильтрации: ContactId = <значение поля id>. Сформированный фильтр применяется к компоненту phonesBindingSource с помощью свойства Filter. Этот компонент генерируется автоматически при добавлении адаптеров доступа к базе данных.

Если флажок FilterCheckBox не отмечен или в списке не выбрана фамилия, свойству Filter присваивается значение пустой строки, т.е. фильтрация отменяется.

Задачи для самостоятельного решения:

Добавить в программу предыдущей лабораторной работы возможность добавления, редактирования и удаления записей базы данных, а также фильтрацию по двум (любым) полям. При этом хотя бы один фильтр должен использовать связи между таблицами (фильтрация осуществляется в одной таблице, а поле для фильтра берется из другой).

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания и задание для самостоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания и решил задание для самостоятельного решения с фильтрацией по одному полю.

оценка «3» ставится, если:

- обучающийся выполнил задания и решил задание для самостоятельного решения без фильтрации .

оценка «2» ставится, если:

- обучающийся выполнил задания из описания.

Практическое занятие 67 Разработка графических приложений с использованием GDI+. Построение графиков функций

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;

- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить пространства имен и классы интерфейса графических устройств .NET, основные свойства и методы этих классов, применяемые при сеансах вывода графики. Научиться использовать классы GDI+ для рисования графиков функций

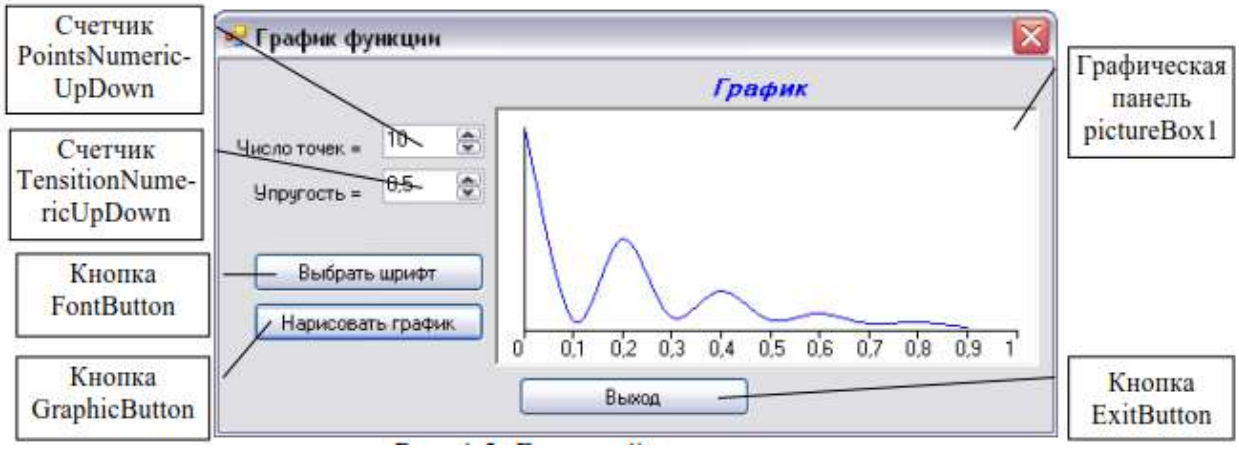
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Построить график функции $y = e^{-4x} \cdot |cos(15x)|$. Самостоятельно выбрать удобные параметры настройки. Пользователь задает количество точек графика, коэффициент упругости, а также шрифт для подписей осей координат.

Внешний вид окна приложения с описанием используемых компонентов приведен на рисунке.



Для графической панели pictureBox1 устанавливаются следующие дополнительные свойства:

Свойство	Значение	Описание
`BorderStyle`	`Fixed3D`	Стиль границы
`BackColor`	`White`	Цвет фона

Коэффициент упругости графика может принимать значение в диапазоне от 0 до 1. Поэтому для счетчика TensitionNumericUpDown устанавливаются следующие свойства:

Свойство	Значение	Описание
`Minimum`	0	Минимальное значение
`Maximum`	1	Максимальное значение
`Increment`	0,1	Шаг инкремента
`DecimalPlaces`	1	Количество знаков после запятой

Также на форму нужно поместить компонент FontDialog (диалоговое окно выбора шрифта), для которого следует установить в true свойство ShowColor. Благодаря этому свойству в окне можно будет выбирать не только характеристики шрифта, но и его цвет. При желании можно также изменить шрифт по умолчанию (свойство Font).

Для обеспечения графического вывода данному приложению понадобятся кисть и шрифт. Поэтому в класс формы необходимо добавить объекты классов Font (шрифт) и SolidBrush (сплошная кисть):

```
Font font;
SolidBrush brush;
```

Создание этих объектов может производиться в обработчике события Load или в конструкторе формы:

```
//шрифт берем установленный по умолчанию
font = fontDialog1.Font;
//создаем сплошную кисть черного цвета
```

```
brush = new SolidBrush(Color.Black);
```

При нажатии на кнопку «Выбрать шрифт» на экране должно появляться соответствующее диалоговое окно. Если после выбора настроек шрифта пользователь нажимает кнопку OK, то выбранные установки должны передаться объектам font и brush:

```
private void FontButton_Click(object sender, EventArgs e)
{
   //если выбор шрифта завершен нажатием кнопки OK
   if (fontDialog1.ShowDialog() == DialogResult.OK)
   {
//получить параметры шрифта из диалогового окна
      font = fontDialog1.Font;
      //получить цвет шрифта из того же окна
      brush.Color = fontDialog1.Color;
   }
}
```

Функцию, для которой будет рисоваться график, можно задать отдельно:

```
private double f(double x)
{
   return Math.Exp(-4 * x) * Math.Abs(Math.Cos(15 * x));
}
```

Обработчик нажатия кнопки «Нарисовать график» состоит из нескольких частей:

- описание и инициализация вспомогательных переменных, задающих начало координат, масштабы по осям x и y, а также число точек графика;
- создание и инициализация графического объекта;
- создание и заполнение массива точек, по которым будет строиться график, при этом производится преобразование физических координат точек в экранные с учетом значений масштабов;
- непосредственно рисование графика и осей координат;
- разметка оси x с выводом числовых значений.

Более подробное описание кода содержится в комментариях к листингу:

```
private void GraphicButton_Click(object sender, EventArgs e)
{
   //Начало координат графика
   int x0 = 15;
   int y0 = (int)(pictureBox1.Height * 0.85);
   //Масштаб по оси X
   int Mx = pictureBox1.Width - 2 * x0;
   //Масштаб по оси Y
   int My = -y0 + 10;
   //Число точек графика
   int M = (int)PointsNumericUpDown.Value;

   //Создание графического объекта
   Graphics G = pictureBox1.CreateGraphics();
   //Очистка PictureBox1
G.Clear(Color.White);

   //Описание и создание массива точек
   Point[] p = new Point[M];
   //Цикл по числу точек графика
   for (int n = 0; n < M; n++)
   {
      //Физические координаты
      double x = (double)n / M;
```

```
        double y = f(x);
            //Экранные координаты
        int xi = (int)(x0 + Mx * x);
        int yi = (int)(y0 + My * y);
            //заносим в массив вычисленные значения координат
        p[n] = new Point(xi, yi);
         }
         //коэффициент упругости графика
         float tension = (float)TensitionNumericUpDown.Value;

         //рисование графика
         G.DrawCurve(Pens.Blue, p, tension);
         //Рисование оси X
         G.DrawLine(Pens.Black, x0, y0, x0 + Mx, y0);
         //Рисование оси Y
         G.DrawLine(Pens.Black, x0, y0, x0, y0 + My);

         //Разметка оси X
         for (int n = 0; n <= 10; n++)
         {
            //физическая координата штриха
            double x = n / 10.0;
            //экранная координата штриха
            int xi = (int)(x0 + Mx * x);
            //Наносим штрих
            G.DrawLine(Pens.Black, xi, y0, xi, y0 + 4);
            //Наносим число
            G.DrawString(x.ToString(), font, brush, xi - 9, y0 + 4);

         }
      }
```

После создания данного обработчика при нажатии на кнопку «Нарисовать график» в компоненте PictureBox отображается график функции. Однако, если окно программы перекрыть другим окном или перетащить его (или только часть окна с графиком) за пределы экрана, а затем вернуть назад, то рисунок (или его часть) исчезнет. Происходит это из-за того, что каждый раз при появлении Windows-окна на экране его необходимо перерисовывать. Перерисовка самой формы и компонентов (кнопок, флажков и т.д.) производится автоматически операционной системой, а перерисовка всей выводимой в окне графики должна производиться программистом.

Если форма или какой-то компонент поврежден, генерируется событие

Paint, которое и нужно обрабатывать для перерисовки. Сгенерировать событие Paint вручную можно с помощью метода Invalidate().

Таким образом, для данного приложения весь код рисования нужно пе-ренести из обработчика щелчка кнопки «Нарисовать график» в обработчик события Paint компонента pictureBox1. При этом строка

Graphics G = pictureBox1.CreateGraphics();

заменяется на

Graphics G = e.Graphics;

т.е. графический объект для рисования мы получаем из параметра обработчика.

Обработчик кнопки «Нарисовать график» теперь будет содержать всего одну строчку – принудительный вызов перерисовки графической панели. private void GraphicButton_Click(object sender, EventArgs e)

{ pictureBox1.Invalidate();}

Теперь график будет выведен в панели постоянно: при запуске программы, при перекрытии окна, при нажатии кнопки (с новыми параметрами).

Задачи для самостоятельного решения:

Построить график функции, вывести, разметить и подписать оси координат (обе!). Предусмотреть возможность установки количества точек и коэффициента упругости графика, а также возможность выбора шрифта с помощью стандартного диалогового окна. Подобрать параметры осей координат, обеспечивающие наглядность (для этого сначала можно построить график в Excel, чтобы оценить его параметры).

1. $y=\frac{x^{\cos x}}{|x+e^x|+tgx}$ при $0{,}4 \le x \le 6$.
2. $y=\frac{2\cos(x-\pi/6)}{1/2+\sin^2(1+\frac{x^2}{3})}$ при $0 \le x \le 1$.
3. $y=x\sin x^3+x+\frac{x^2}{2!}+\frac{x^3}{3!}$ при $1 \le x \le 3$.
4. $y=|\cos^2 x|\frac{x-\frac{2}{x-2}}{1+(1-x)^2}$ при $-1 \le x \le 1$.
5. $y=e^{-2x}\sin(2x+1)-\sqrt{|x+2|}$ при $-0{,}5 \le x \le 4{,}5$.
6. $y=\sqrt{x^2+5}-10\sin^3(x+1)$ при $-2 \le x \le 8$.
7. $y=e^{-x}\frac{tgx^2+x^3}{x\cos^2 2x}$ при $0{,}05 \le x \le 0{,}55$.
8. $y=\frac{x^2(x+1)}{2}-\sin^2\sqrt{x+2}$ при $-1 \le x \le 1$.
9. $y=\frac{x^2+1}{\sin 3x}+\frac{\sqrt{x/2}}{\cos 3x}$ при $0{,}2 \le x \le 2{,}2$.
10. $y=\sin^3(x^2+\pi/3)^2-\sqrt{\frac{x}{2}}$ при $2 \le x \le 12$.
11. $y=10^{-x}\sqrt{|\sin 2x+\cos x^2|}$ при $0 \le x \le 3$.
12. $y=\frac{e^{-\frac{x}{2}}tg^2 3x}{1+|\cos x^3|}$ при $-2{,}3 \le x \le -1{,}8$.
13. $y=2tg^2 x-\frac{1}{\frac{1}{2}\sin^2\frac{x}{2}}$ при $2{,}4 \le x \le 3{,}4$.
14. $y=\ln(1+x^2)+\sin^2\left(\frac{x}{3}\right)$ при $-2 \le x \le 1$.
15. $y=\frac{2^{3x}+10^{-x}\cos(x+\pi/3)}{x\sin x}$ при $1{,}2 \le x \le 2{,}2$.
16. $y=\frac{x^{\sin x}}{|x+e^x|+\ln x}$ при $0{,}4 \le x \le 4$.
17. $y=\frac{2\cos(x-\pi/6)}{1/2-\cos^2(1+\frac{x^2}{3})}$ при $0 \le x \le 2{,}5$.
18. $y=x\cos x^3+x+\frac{x^2}{2!}-\frac{x^3}{3!}$ при $0 \le x \le 4$.
19. $y=|\sin^2 x|\frac{x-\frac{2}{x-2}}{1+(1+x)^2}$ при $-2 \le x \le 1{,}6$.
20. $y=e^{-2x}\sin(2x+1)+\sqrt{|x-2|}$ при $0 \le x \le 3$.
21. $y=\sqrt{x^2+5}-10\sin^2(x-1)$ при $2 \le x \le 10$.
22. $y=e^{-x}\frac{tgx^2-x^3}{x\sin^2 2x}$ при $0{,}2 \le x \le 0{,}5$.
23. $y=\frac{x^2(x-1)}{2}+\sin^2\sqrt{x-2}$ при $2 \le x \le 4$.
24. $y=\frac{x^2+1}{\sin 2x}-\frac{\sqrt{x/2}}{\cos 2x}$ при $0{,}5 \le x \le 2{,}5$.
25. $y=\cos^3(x^2+\pi/3)^2+\sqrt{\frac{x}{2}}$ при $0 \le x \le 4$.
26. $y=10^{-x}\sqrt{|\sin 2x-\cos x^2|}$ при $0{,}5 \le x \le 2{,}5$.
27. $y=\frac{e^{-\frac{x}{2}}tg^2 3x}{1+|\sin x^3|}$ при $-2 \le x \le 1$.
28. $y=2tg^2 x+\frac{1}{\frac{1}{2}\sin^2\frac{x}{3}}$ при $1{,}4 \le x \le 1{,}8$.
29. $y=\ln(1+x^2)-\cos^2\left(\frac{x}{3}\right)$ при $-1 \le x \le 2$.
30. $y=\frac{2^{3x}-10^{-x}\sin(x+\pi/3)}{x\sin x}$ при $0{,}2 \le x \le 2{,}2$.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 68 Разработка приложения, имитирующего движение графических объектов

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить теоретические принципы использования графических объектов GDI+ и получить практические навыки разработки программ, имитирующих движение графических объектов.

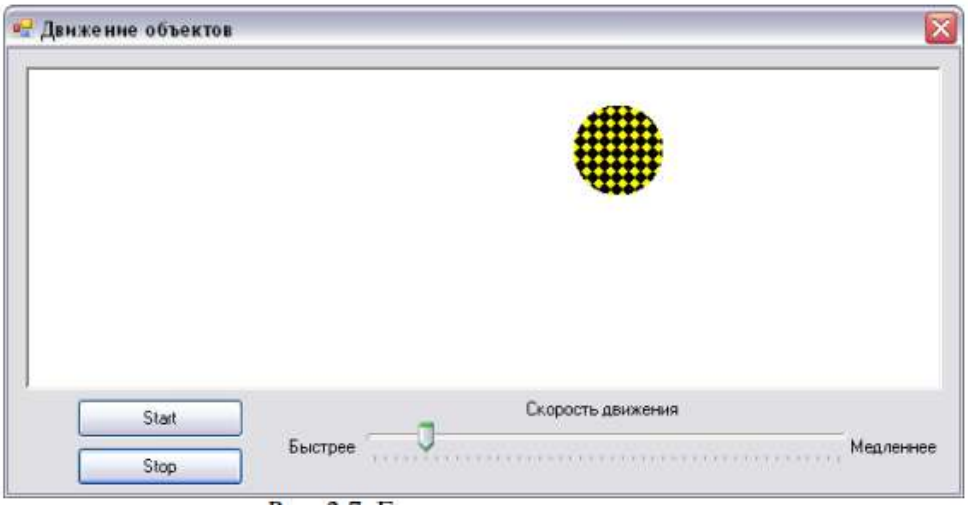
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Реализовать эффект движения простейшего геометрического объекта (круга) внутри области рисования с «отскакиванием от стенок». Предусмотреть возможность изменения его цвета и скорости движения. Изменение цвета круга происходит при щелчке в нем левой кнопкой мыши. Скорость регулируется с помощью компонента TrackBar.

Внешний вид главного окна приложения приведен на рисунке.



Для графической панели pictureBox1 устанавливаются свойства:

Свойство	Значение	Описание
BorderStyle	Fixed3D	Стиль границы
BackColor	White	Цвет фона

Настройки компонента trackBar1 будут рассматриваться далее.

Т.к. в данном приложении будет создаваться кисть со штриховой заливкой, прежде всего, необходимо в файл главной формы подключить пространство имен System.Drawing.Drawing2D.

Для рисования круга необходимо иметь координаты его центра и радиус. Для этого нужно объявить соответствующие переменные в классе формы:

```
int x0, y0;
int radius;
```

Начальные значения этим переменным можно установить в конструкторе формы или в обработчике события Load. Здесь начальное положение круга – левый нижний угол области рисования, радиус – 30 пикселей.

```
x0 = 30;
y0 =  pictureBox1.Height - 35;
radius = 30;
```

Рисование круга будет производиться в обработчике события Paint области рисования pictureBox1. Процесс рисования состоит из создания объекта Graphics, очистки области рисования, создания кисти и непосредственного рисования круга. Кисть создается как объект класса HatchBrush (штриховая кисть). При этом первым параметром задается стиль заливки (здесь – Hatch-Style.SolidDiamond – «шахматка»), вторым параметром – основной цвет заливки (желтый). Можно указать и третий параметр – «неосновной» цвет или цвет фона (по умолчанию – черный). При рисовании круга (метод Fil-lEllipse()) необходимо задать кисть для заливки, координаты левого верхнего угла прямоугольника, ограничивающего окружность, а также его стороны. Код обработчика приведен ниже:

```
private void pictureBox1_Paint(object sender, PaintEventArgs e)
{
   //создаем графический объект
   Graphics g = e.Graphics;
   //очищаем область рисования
   g.Clear(Color.White);
   //создаем штриховую кисть желтого цвета
   HatchBrush brush = new HatchBrush(HatchStyle.SolidDiamond, Color.Yellow);
   //рисуем круг
   g.FillEllipse(brush, x0 - radius, y0 - radius, 2 * radius, 2
* radius);
}
```

Реализация движения основана на использовании компонента-таймера (Tim-er). После помещения на форму его свойству Interval (интервал в миллисекундах, через который поступает сигнал от таймера) устанавливается значение 10.

По нажатию кнопки «Start» необходимо активизировать («включить») тай-мер, а кнопки «Stop» – остановить его.

```
//кнопка "Start"
private void button1_Click(object sender, EventArgs e)
{
   timer1.Start();
}
//кнопка "Stop"
private void button2_Click(object sender, EventArgs e)
{
   timer1.Stop();
}
```

Чтобы заставить фигуру «двигаться», достаточно изменять ее координаты по сигналу таймера, после чего перерисовать фигуру:

```
private void timer1_Tick(object sender, EventArgs e)
{
   //изменяем координаты круга
   x0++;
   y0--;
   //обновляем область рисования (перерисовываем круг)
   pictureBox1.Invalidate();
}
```

В данном случае после нажатия кнопки «Start» круг будет двигаться по диагонали вправо вверх (координата x увеличивается, y – уменьшается), пока не исчезнет за пределами области рисования или пока не будет нажата кнопка «Stop».

Для создания подобного эффекта нужно отслеживать координаты круга с учетом его радиуса. В этом случае удобно создать в классе формы две дополнительные переменные, задающие направление движения по каждой из осей координат:

```
int xDir, yDir;
```

Переменная xDir принимает значение 1, если круг двигается вправо, и –1, если влево. Переменная yDir равна 1, если идет движение вниз, и –1, если вверх. Поскольку изначально напрвление вижения задается вправо вверх, этим переменным нужно задать соответствующие исходные значения (в кон-структоре или в обработчике события Load):

```
xDir = 1;
yDir = -1;
```

«Отскакивание от стенок» реализуется за счет изменения значений этих переменных. Если круг «подлетел» к левой стенке, то после этого он должен начать двигаться вправо, т.е. переменной xDir нужно присвоить значение 1, и т.д. Приближение к стенкам отслеживается по значениям переменных x0 и y0 с учетом радиуса круга. При этом при «касании» кругом правой и нижней стенок необходимо вносить корректировку в 5 пикселей из-за наличия трехмерной рамки у области рисования.

Таким образом, обработчик события Tick для таймера модифицируется следующим образом:

```
private void timer1_Tick(object sender, EventArgs e)
{
   //если круг "подлетел" к левой стенке,
   if (x0 - radius < 0)
      xDir = 1;   //то начинаем двигаться вправо
   //если к правой стенке,
   if (x0 + radius + 5 > pictureBox1.Width)
      xDir = -1;  //то влево
   //если к верхней стенке,
   if (y0 - radius < 0)
      yDir = 1;   //то вниз
   //если к нижней стенке,
   if (y0 + radius + 5 > pictureBox1.Height)
      yDir = -1;  //то вверх
   //изменяем координаты круга
   x0 += xDir;
   y0 += yDir;
   //обновляем область рисования (перерисовываем круг)
   pictureBox1.Invalidate();
}
```

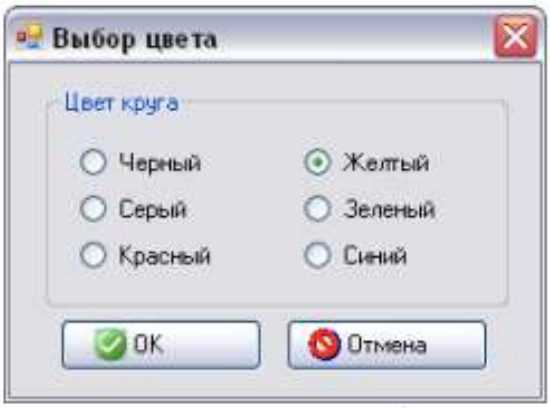
Для изменения скорости движения нужно изменять интервал поступления сигнала от таймера, что, в свою очередь, будет регулироваться компонентом TrackBar. Интервал сигнала от таймера в данном приложении задается диа-пазоном от 5 до 50 миллисекунд. Соответствующие настройки задаются и для компонента TrackBar:

Свойство	Значение	Описание
Minimum	5	Минимальное значение
Maximum	50	Максимальное значение
Value	10	Текущее значение

При прокрутке движка компонента TrackBar программе поступает сообщение Scroll, в обработчике которого и нужно изменять значение интервала таймера:

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
   //изменяем значение интервала таймера на выбранное
   timer1.Interval = trackBar1.Value;
}
```

Для выбора цвета заливки круга в данном приложении будет использоваться дополнительное диалоговое окно с группой радиокнопок. Внешний вид этого окна приведен на рисунке.



В этом окне в компоненте GroupBox расположены 6 радиокнопок (Radio-Button)с названиями цветов, а также 2 кнопки: «OK» и «Отмена». На кнопки помимо надписей добавлены соответствующие рисунки.

Рисунки можно добавить на кнопку либо через ее свойство Image, либо, как и сделано в данной программе, через компонент ImageList. Этот компонент представляет собой список изображений, которые потом можно назначать обычным кнопкам, пунктам меню, кнопкам панели инструментов и другим компонентам. После помещения этого компонента на форму нужно настро-ить его свойство Images, где с помощью отдельного диалогового окна в про-ект добавляются файлы с изображениями. Каждое добавленное изображение имеет свой индекс (порядковый номер), начиная с 0. В данном приложении рисунок для кнопки «OK» имеет индекс 0, а для кнопки «Отмена» – 1.

Настройки кнопок «OK» и «Отмена» приведены в таблице:

Свойство	Значение	Описание
`ImageList`	`imageList1`	Источник изображений
`ImageIndex`	0 для кнопки «OK», 1 для кнопки «Отмена»	Индекс изображения в источнике
`ImageAlign`	`MiddleCenter`	Выравнивание изображения
`TextAlign`	`MiddleCenter`	Выравнивание текста
`TextImageRelation`	`ImageBeforeText`	Взаимное расположение текста и изображения
`DialogResult`	`OK` для кнопки «OK», `Cancel` для кнопки «Отмена»	Возвращаемое значение для нажатия кнопки

Установка значения OK свойству DialogResult для кнопки «OK» означает, что при нажатии на эту кнопку форма закроется и вернет в главное окно значение DialogResult.OK (для кнопки «Cancel» – DialogResult.Cancel).

После отображения на экране диалогового окна значение выбранного цвета необходимо передать в главную форму программы. Непосредственно значение выбранной радиокнопки передать не удастся, т.к. переменные, описывающие компоненты формы, закрыты (private). Поэтому для решения этой проблемы необходимо создать в классе Form2 общедоступное свойство для получения выбранного цвета, а также для установки текущего цвета круга (соответствующая радиокнопка становится выбранной). Тип свойства – Col-or. Код свойства приведен далее:

```
public Color CircleColor
{
   get
   {
      //если выбрана первая радиокнопка,
      if (radioButton1.Checked)
         //то вернуть черный цвет
         return Color.Black;
      if (radioButton2.Checked)   //и т.д.
         return Color.Gray;
      if (radioButton3.Checked)
         return Color.Red;
      if (radioButton4.Checked)
         return Color.Yellow;
      if (radioButton5.Checked)
         return Color.Green;
      return Color.Blue;
   }
   set
   {
      //если текущий цвет круга - черный,
      if (value == Color.Black)
         //то сделать выбранной первую радиокнопку
         radioButton1.Checked = true;
      if (value == Color.Gray)    //и т.д.
         radioButton2.Checked = true;
      if (value == Color.Red)
         radioButton3.Checked = true;
      if (value == Color.Yellow)
         radioButton4.Checked = true;
      if (value == Color.Green)
         radioButton5.Checked = true;
      if (value == Color.Blue)
         radioButton6.Checked = true;
   }
}
```

Изменение цвета круга в программе должно происходить при щелчке левой кнопкой мыши внутри круга (событие MouseClick для компонента picture-Box1). Для этого необходимо отслеживать координаты курсора мыши: попал ли он в момент щелчка внутрь круга.

сли область изображения прямоугольная, то такая проверка осуществляется достаточно просто:

```
if (прямоугольник.Contains(точка_курсора))
   //выполнить нужные действия
```

Если же область изображения – не прямоугольная (круг, как в нашем случае, или более сложная фигура), нужно использовать компоненты класса GraphicsPath для формирования сложных фигур. Объект этого типа описывается в классе Form1

```
GraphicsPath path;
```

и в конструкторе класса формы ему выделяется память:

```
path = new GraphicsPath();
```

Когда на экране рисуется очередной новый круг, его изображение помещается в предварительно очищенный объект GraphicsPath. Таким образом, в конец метода pictureBox1_Paint надо добавить следующие строки:

```
//очищаем фигуру
```

```
path.Reset();
//начинаем формирование фигуры
path.StartFigure();
//добавляем круг в фигуру
path.AddEllipse(x0-radius, y0 - radius, 2 * radius, 2 * radius); //завершаем формирование фигуры
path.CloseFigure();
```

Если изображение состоит из нескольких элементов (не только из одного круга, как в нашем случае), их все нужно добавить в объект GraphicsPath.

Прежде всего, в класс формы необходимо добавить переменную, задающую цвет круга:

```
Color myColor;
```

Начальное значение (желтый цвет) присваивается ей в конструкторе формы: myColor = Color.Yellow;

Изменение цвета круга происходит при щелчке левой кнопки мыши внутри круга (возможно, движущегося) в обработчике события MouseClick (не просто Click!) компонента pictureBox1. Пояснения к коду приведены в комментариях:

```
private void pictureBox1_MouseClick(object sender, MouseEven-tArgs e)
{
   //проверяем, был ли щелчок именно левой кнопкой мыши
   if (e.Button == MouseButtons.Left)
   {
      //получаем точку с координатами места щелчка
      Point pt = new Point(e.X, e.Y);
      //если эта точка - внутри фигуры (круга)
      if (path.IsVisible(pt))
{
         //создаем объект формы выбора цвета
         Form2 form = new Form2();
         //передаем в форму текущий цвет круга
         form.CircleColor = myColor;
         //отображаем форму на экране
         if (form.ShowDialog() == DialogResult.OK)
         {
            //получаем значение выбранного цвета
            myColor = form.CircleColor;
            //перерисовываем круг
            pictureBox1.Invalidate();
         }
      }
   }
}
```

После этого остается лишь указание значение выбранного цвета при создании кисти: в методе pictureBox1_Paint строку

```
HatchBrush brush = new HatchBrush(HatchStyle.SolidDiamond, Col-or.Yellow);
```

заменить на строку

```
HatchBrush brush = new HatchBrush(HatchStyle.SolidDiamond, my-Color);
```

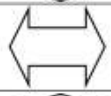
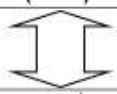
Задачи для самостоятельного решения:

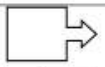
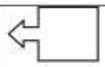
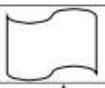
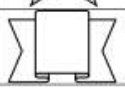
В каждом варианте необходимо реализовать движение требуемого изображения по заданной траектории, не допуская при этом выход фигуры за пределы области рисования. Каждое изображение состоит из нескольких (двух-трех) элементарных фигур. Предусмотреть:

 возможность изменения скорости движения с помощью компонента TrackBar;

 выбор цвета заливки фигуры нажатием левой кнопки мыши внутри изображения. Для выбора используется стандартное диалоговое окно выбора цвета (компонент ColorDialog);

 выбор стиля заливки фигуры нажатием правой кнопки мыши внутри изображения. Для выбора используется пользовательское диалоговое окно с радиокнопками.

№ варианта	Вид изображения	Траектория движения
1.		Вдоль границ области рисования по часовой стрелке
2.		Вдоль границ области рисования против часовой стрелки
3.		От центра области рисования: вверх, вниз, влево, вправо
4.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
5.		От центра области рисования поочередно к углам. От угла – назад в центр
6.		Вдоль границ области рисования по часовой стрелке
7.		Вдоль границ области рисования против часовой стрелки
8.		От центра области рисования: вверх, вниз, влево, вправо
9.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
10.		От центра области рисования поочередно к углам. От угла – назад в центр
11.		Вдоль границ области рисования по часовой стрелке
12.		Вдоль границ области рисования против часовой стрелки
13.		От центра области рисования: вверх, вниз, влево, вправо
14.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
15.		От центра области рисования поочередно к углам. От угла – назад в центр
16.		Вдоль границ области рисования по часовой стрелке
17.		Вдоль границ области рисования против часовой стрелки
18.		От центра области рисования: вверх, вниз, влево, вправо
19.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
20.		От центра области рисования поочередно к углам. От угла – назад в центр

21.		Вдоль границ области рисования по часовой стрелке
22.		Вдоль границ области рисования против часовой стрелки
23.		От центра области рисования: вверх, вниз, влево, вправо
24.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
25.		От центра области рисования поочередно к углам. От угла – назад в центр
26.		Вдоль границ области рисования по часовой стрелке
27.		Вдоль границ области рисования против часовой стрелки
28.		От центра области рисования: вверх, вниз, влево, вправо
29.		От центра области рисования: влево, вверх, вправо, вниз. От каждой стенки – к центру
30.		От центра области рисования поочередно к углам. От угла – назад в центр

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 69 Разработка приложений с многодокументным интерфейсом

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить теоретические принципы и получить практические навыки разработки программ на основе многооконного (многодокументного) интерфейса.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Для создания тестового примера необходимо выполнить следующие пошаговые инструкции:

1) Создайте новый проект.

2) Проект содержит только одну форму Form1. Сделайте ее главной (родительской).

3) Установите свойству WindowState формы Form1 значение Maximized. При запуске программы главное окно будет принимать максимальный размер (раскрываться на весь экран). Аналогичного эффекта можно добиться, если в конструкторе формы Form1 дописать следующий код:

this.WindowState=FormWindowState.Maximized;

4) Создайте главное меню.

5) Добавьте пункты меню «&Файл» и «&Окно». Измените их свойство name на menuItemFile и menuItemWindow соответственно.

6) Добавьте в меню «&Файл» пункт «&Создать». Свойство name установите в menuItemNew. Этот пункт меню будет предназначен для создания дочерних окон.

7) Добавьте в проект еще одну форму, которая будет шаблоном дочерней формы.

8) Создайте обработчик события Click для пункта меню «&Создать» и добавьте код из п.4.2. краткой теории лабораторной работы.

9) Добавьте в меню «&Окно» пункты «&Горизонтально», «&Вертикально» и «&Каскадом». Свойство name установите в menu-ItemTileHorizontal, menuItemTileVertical и menuItemCascade соответственно. Эти пункты меню предназначены для вариантов упорядочения дочерних окон.

10) Выберите элемент главного меню и установите его свойству MdiWindowListItem значение menuItemWindow. Это означает, что в пункте меню «&Окно» будет отображаться список всех открытых дочерних окон.

11) Создайте один общий обработчик для подпунктов меню пункта «&Окно». Для этого выделите все три пункта меню, удерживая клавишу Ctrl, и щелкните два раза указателем мыши по событию Click в окне свойств. Замените имя события (свойство name), созданное по умолчанию, на имя menuItemTool_Click. Функция menuItemTool_Click будет устанавливать вариант упорядочения дочерних окон.

12) Добавьте следующий код в функцию menuItemTool_Click:

```
ToolStripMenuItem item = sender as ToolStripMenuItem;
    switch (item.Text)
{
        case "&Горизонтально":
            this.LayoutMdi(MdiLayout.TileHorizontal);
            break;
    case "&Вертикально":
            this.LayoutMdi(MdiLayout.TileVertical);
            break;
    case "Каскадом":
             this.LayoutMdi(MdiLayout.Cascade);
             break;
}
```

13) Запустите программу. Выберите из меню пункт «Файл»/ «Создать». На экране появится дочернее окно с именем «Form2». Вы можете создать еще несколько дочерних окон, воспользовавшись все тем же пунктом меню. При этом пункт меню «Окно» будет содержать список всех дочерних окон, открытых на текущий момент.

14) Попробуйте различные варианты упорядочения дочерних окон, воспользовавшись пунктом меню «Окно».

15) Заголовки всех дочерних окон одинаковые – «Form2». Модифицируем приложение таким образом, чтобы при создании нового дочернего окна в заголовке появлялся порядковый

номер этого окна. Для этого необходимо изменить код обработчика события Click пункта меню «&Создать» следующим образом:

```
Form2 newMDIChild = new Form2();
newMDIChild.MdiParent = this;
int wnd_num = this.MdiChildren.Length-1;
newMDIChild.Text = "Doc"+wnd_num.ToString();
newMDIChild.Show();
```

Свойство MdiChildren содержит массив дочерних форм.

16) Запустите программу. Создайте несколько дочерних окон. Посмотрите как изменяются заголовки.

17) Добавьте на форму панель инструментов (компонент ToolStrip) с кнопками, соответствующими пунктам меню. Рисунки на кнопки можно добавить через свойство Image.

18) В качестве обработчика события Click каждой кнопки панели ин-струментов укажите функцию-обработчик соответствующего пункта меню. После этого реакция на выбор пункта меню или нажатие соответствующей кнопки панели инструментов будет одинаковой.

Задачи для самостоятельного решения:

В каждом варианте в главном окне необходимо создать меню и панель инструментов (компонент ToolStrip) с кнопками, дублирующими пункты меню.

1. Дан список фамилий (загружается из файла при создании дочернего окна). Организовать отбор во всех списках (во всех дочерних окнах) по первой букве фамилий. Результаты отображать в виде списка в отдельном окне (не дочернем).

2. Создать простейший текстовый редактор с возможностью открытия и сохранения файлов.

3. Создайте приложение «Секундомер». При создании дочернего окна должен запускаться секундомер. В окне должна быть кнопка останова, паузы и нового запуска.

4. Создать простейшее приложение для просмотра графических файлов с возможностью открытия и сохранения изображений.

5. Создать следующее приложение. В каждом дочернем окне – текстовая метка или поле ввода, которые работают как числовой счетчик. Скорость увеличения задается таймером. Скорость таймера можно менять одновременно для всех дочерних окон в отдельном диалоговом окне (не дочернем).

6. Создать приложение с двумя видами шаблонов дочерних окон. В первом случае должен осуществляться контроль ввода только цифровых данных, во втором – только текстовых. Заголовок должен содержать слово «Цифры» или «Символы».

7. Создайте приложение «Конверт». Шаблон дочернего окна должен представлять собой образец заполнения полей на почтовом конверте: в левом верхнем углу адрес отправителя, внизу справа адрес получателя. В заголовке окна должен отображаться получатель.

8. Создайте приложение «Статистика текста». При наборе текста с клавиатуры должен проводиться подсчет и осуществляться вывод количества знаков препинания, букв и цифр.

9. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета линии этого объекта.

10. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета линии этого объекта с помощью задания интенсивности красного, зеленого и синего цветов (модель RGB) через компоненты TrackBar.

11. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения стиля линии этого объекта (для этого у пера нужно изменять свойство DashStyle).

12. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения толщины линии этого объекта.

13. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета заливки этого объекта.

14. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения цвета заливки этого

объекта с помощью задания интенсивности красного, зеленого и синего цветов (модель RGB) через компоненты TrackBar.

15. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения стиля заливки этого объекта.

16. Создать приложение, где в каждом дочернем окне изображается движущийся графический объект (круг, прямоугольник и т.п.).

17. Разработать приложение для ввода результатов сессии. Организовать табличный ввод, используя компонент DataGridView. Предусмотреть запись результатов в текстовый файл.

18. Разработать приложение «Табулирование функций». Программа позволяет получить значения аргумента и функции в заданном интервале с заданным шагом. Вид функции можно задать выбором в компоненте ListBox или в компонентах RadioButton.

19. Разработать приложение, где в каждом дочернем окне должен вводиться пароль. Если пароль введен верно, то отображается какое-нибудь графическое изображение.

20. Разработать следующее приложение. В каждом дочернем окне – компонент-счетчик. В главном меню есть пункт, по нажатию которого открывается диалоговое окно, где отображается сумма чисел из счетчиков во всех открытых дочерних окнах.

21. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения координат этого объекта.

22. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность изменения размера этого объекта.

23. Создать приложение, где в каждом дочернем окне изображается какой-то графический объект (круг, прямоугольник и т.п.). Предусмотреть возможность поворота этого объекта (влево на 90□, вправо на 90□, на 180□).

24. Создать простейший текстовый редактор с возможностью изменения шрифта и цвета текста.

25. Разработать следующее приложение. В каждом дочернем окне выбирается дата с помощью компонента DateTimePicker. В главном меню есть пункт, по нажатию которого открывается диалоговое окно, где выводится самая ранняя из всех дат, заданных в открытых дочерних окнах.

26. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится текстовая строка. Программа должна разбить эту строку на слова и каждое слово передать в отдельное дочернее окно, например, в комонент Label или TextBox (должно открыться столько окон, сколько слов в строке).

27. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится матрица чисел (в компонент DataGridView). Программа должна разбить эту матрицу на строки и каждую строку отобразить в отдельном дочернем окне (должно открыться столько окон, сколько строк в матрице).

28. Разработать следующее приложение. В диалоговом окне (не дочернем) вводится матрица чисел (в компонент DataGridView). Программа должна разбить эту матрицу на столбцы и каждый столбец отобразить в от-дельном дочернем окне (должно открыться столько окон, сколько столбцов в матрице).

29. Разработать следующее приложение. В каждом дочернем окне вводится одно и то же количество чисел (например, в компонент DataGridView или ListBox). Необходимо все введенные числа из каждого дочернего окна объединить по строкам в матрицу, матрицу отобразить в отдельном диалоговом окне (не дочернем).

30. Разработать следующее приложение. В каждом дочернем окне вводится одно и то же количество чисел (например, в компонент DataGridView или ListBox). Необходимо все введенные числа из каждого дочернего окна объ-единить по столбцам в матрицу, матрицу отобразить в отдельном диалоговом окне (не дочернем).

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 70 Разработка многопоточных приложений

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить методы и средства, пространства имен и классы, применяемые при работе с потоками в среде .NET Framework. Получить практические навыки разработки многопоточных приложений на языке C#.

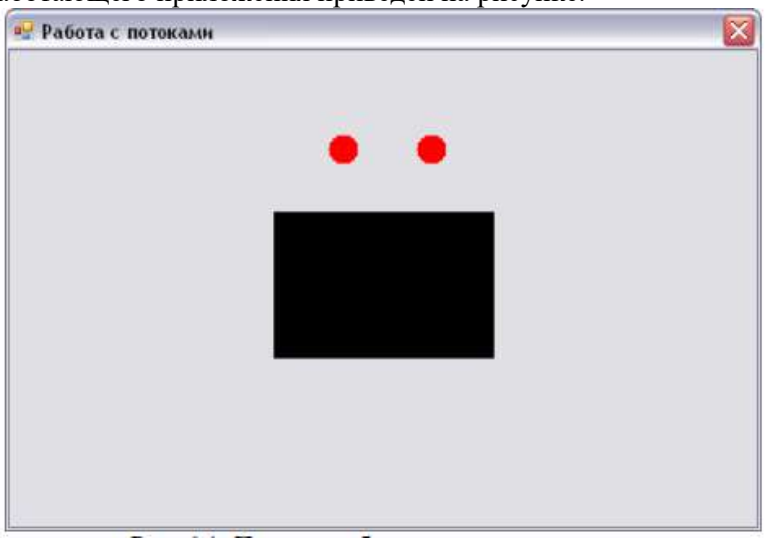
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Необходимо разработать многопоточное приложение с синхронизацией двух потоков и управлением доступом к общему ресурсу. В каждом потоке осуществляется движение графического объекта (в данном случае, круга) от верхнего края формы к нижнему, после чего потоки завершаются. В качестве общего ресурса выступает прямоугольная область в середине формы. Необходимо скоординировать потоки таким образом, чтобы в каждый момент времени внутри этой области находился только один круг.

Внешний вид работающего приложения приведен на рисунке.



Прежде всего, для работы с потоками необходимо подключить пространство имен System.Threading.

Для того чтобы создавать в приложении новые потоки, необходимо за-

дать метод, который будет являться точкой входа в поток. Этот метод должен быть типа void и без параметров. В нашем случае именно в этом методе (который здесь назван MoveCircle()) и будет осуществляться движение круга.

В этом методе сначала создается контекст графического устройства и две кисти: одна (красного цвета) для круга, вторая (цвета формы) – для стирания предыдущего изображения круга.

Чтобы два круга в разных потоках не сливались друг с другом, для них указываются разные координаты x. Значения координаты устанавливаются в зависимости от имени потока (свойство Name).

После этого в цикле, пока не достигнут нижний край формы, рисуется очередной круг, поток на некоторое время приостанавливается (иначе движение будет происходить настолько быстро, что увидеть его просто не получится), а затем круг стирается.

После окончания цикла для корректности выводится сообщение о завершении текущего потока. Листинг метода приведен ниже:

```
private void MoveCircle()
{
   //создаем контекст устройства
   Graphics g = this.CreateGraphics();
   //создаем красную кисть - для круга
   Brush b1 = Brushes.Red;
   //и кисть цвета формы - для стирания круга
Brush b2 = SystemBrushes.Control;
   int x;
   //координата x круга будет зависеть от имени потока
   if (Thread.CurrentThread.Name == "First")
      x = Width / 2 - 30;
   else
      x = Width / 2 + 30;
   //цикл от верхнего до нижнего края формы
   for (int y = 10; y < Height - 40; y++)
   {
      //рисуем круг
      g.FillEllipse(b1, x - 10, y - 10, 20, 20);
      //"усыпляем" поток на 30 миллисекунд
      Thread.Sleep(30);
      //стираем круг
      g.FillEllipse(b2, x - 10, y - 10, 20, 20);
   }
   //проверяем корректность завершения потока
   MessageBox.Show("Поток " + Thread.CurrentThread.Name + "
завершен!");
}
```

Непосредственное создание потоков можно выполнять в обработчике события Load главной формы. При этом для каждого потока нужно создать объект класса Thread, указав при этом имя метода, который будет точкой входа в поток. Далее следует присвоить потоку имя и запустить его с помощью метода Start(). Код обработчика приведен ниже:

```
private void Form1_Load(object sender, EventArgs e)
{
   //создаем первый поток
   //(точка входа в него - метод MoveCircle())
   Thread thread1 = new Thread(new ThreadStart(MoveCircle));
   //присваиваем потоку имя
   thread1.Name = "First";
   //аналогично для второго потока
   Thread thread2 = new Thread(new ThreadStart(MoveCircle));
   thread2.Name = "Second";
   //запускаем оба потока
   thread1.Start();
   thread2.Start();
}
```

Для задания координат и размера прямоугольника необходимо добавить в класс формы соответствующую переменную типа Rectangle:

```
Rectangle rect = new Rectangle(180, 110, 150, 100);
```

Рисование прямоугольника производится в обработчике события Paint формы:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
   //получаем контекст устройства
   Graphics g = e.Graphics;
   //создаем черную кисть
   Brush b = Brushes.Black;
   //рисуем прямоугольник
   g.FillRectangle(b, rect);
}
```

Если на этом этапе запустить программу, то можно увидеть, как круги, проходя сквозь прямоугольник, «стирают» на своем пути часть его. Это про-исходит из-за того, что при реализации движения круг «стирается» путем заливки его цветом формы. Устранить данный эффект можно путем перерисо-вки прямоугольника при попадании круга внутрь него. Таким образом, в методе MoveCircle() после строчки Thread.Sleep(30); следует добавить следующее:

```
//если круг или его часть - внутри прямоугольника,
if (y + 10 > rect.Y && y - 10 < rect.Y + rect.Height)
   //то перерисовываем прямоугольник
   Invalidate();
```

Так как в данной программе необходимо обеспечить управление досту-

пом потоков (кругов) к общему ресурсу (прямоугольнику), то, прежде всего, нужно обеспечить блокировку объекта с помощью метода lock(). Эта блокировка должна действовать, только если один из кругов находится внутри прямоугольника, иначе блокировку нужно снять (метод Wait() класса Monitor) и уведомить об этом второй поток (метод Pulse() класса Monitor).

Также для корректного завершения работы потоков необходимо вызвать метод Pulse() при завершении работы потока.

Таким образом, окончательный вариант метода MoveCircle() будет выглядеть следующим образом:

```
private void MoveCircle()
{
   //устанавливаем блокировку метода
   lock (this)
   {
      //создаем контекст устройства
      Graphics g = this.CreateGraphics();
      //создаем красную кисть - для круга
      Brush b1 = Brushes.Red;
      //и кисть цвета формы - для стирания круга
      Brush b2 = SystemBrushes.Control;
int x;
      //координата x круга будет зависеть от имени потока
      if (Thread.CurrentThread.Name == "First")
         x = Width / 2 - 30;
      else
         x = Width / 2 + 30;
      //цикл от верхнего до нижнего края формы
      for (int y = 10; y < Height - 40; y++)
      {
         //рисуем круг
         g.FillEllipse(b1, x - 10, y - 10, 20, 20);
         //"усыпляем" поток на 30 миллисекунд
         Thread.Sleep(30);
         //если круг или его часть - внутри прямоугольника,
         if (y+10 > rect.Y && y - 10 < rect.Y + rect.Height)
            //то перерисовываем прямоугольник
            Invalidate(rect);
```

```
                else
                {
                    //разрешаем выполнение другого потока
                    Monitor.Pulse(this);
                    //ждем приостановки/завершения другого потока
                    Monitor.Wait(this);
                }
                //стираем круг
                g.FillEllipse(b2, x - 10, y - 10, 20, 20);
            }
            //выводим поток из режима ожидания
            Monitor.Pulse(this);
        }
        //проверяем корректность завершения потока
        MessageBox.Show("Поток " + Thread.CurrentThread.Name + "
    завершен!");
    }
```

Задачи для самостоятельного решения:

Модифицировать данную программу следующим образом:

- приложение должно быть основано на многодокументном интерфейсе;
- вся работа с потоками (движение круга) должна производиться в дочернем окне;
- увеличить количество потоков в каждом дочернем окне до трех;
- в каждом дочернем окне должно быть два прямоугольника, круги (потоки) должны «проходить сквозь них» последовательно;
- круги должны двигаться горизонтально.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 71 Разработка сетевых приложений

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы работы компонентов, связанных с обработкой данных, полученных после использования HTTP-запросов и создать приложение,

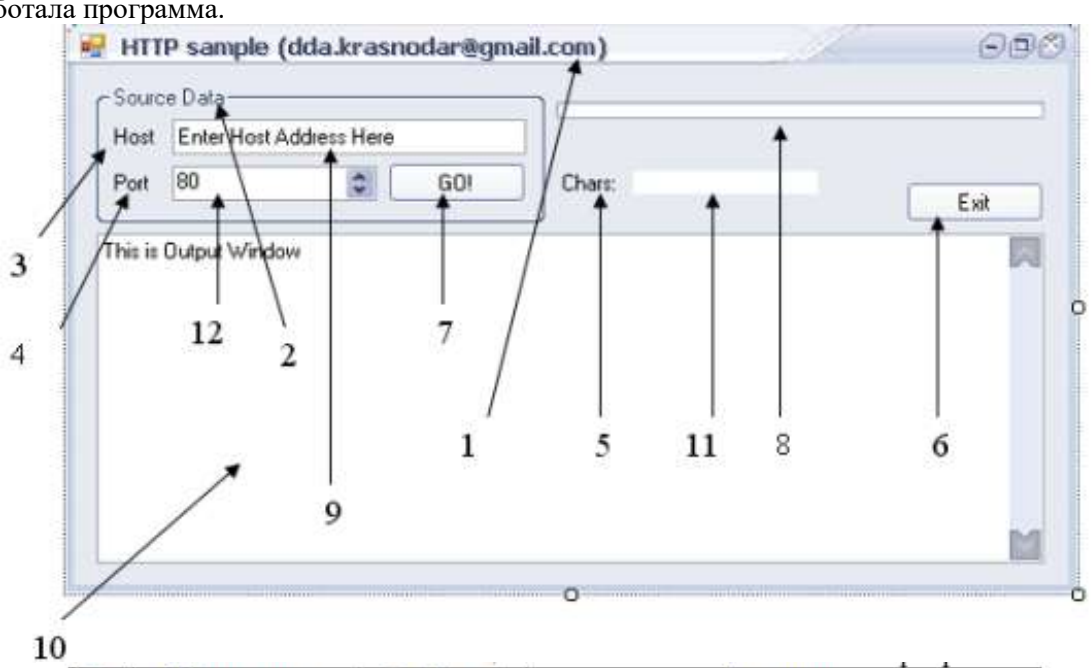
иллюстрирующее работу этих компонентов и возможности получения информации.на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Разработать Windows-приложение, основанное на форме. Функционально приложение будет таким же, как и предыдущий консольный пример. Обеспечить возможность ввода адреса, номера порта. Описать обработчики исключений и посчитать количество символов, которое обработала программа.



№	Компонент	Свойство	Значение
1	Form1	Text	HTTP sample (dda.krasnodar@gmail.com)
		StartPosition	CenterScreen
2	groupBox1	Text	Source Data
3	label1	Text	Host
4	label2	Text	Port
5	label3	Text	Chars:
		Visible	False
6	button1	Text	Exit
7	button2	Text	GO!
8	progressBar1		
9	textBox1	Text	Enter Host Address Here
10	textBox2	Multiline	True
		ScrollBars	Vertical
		Text	This is Output Window
		False	Enabled

окончание таблицы 7.10

11	textBox3	Enabled	False
12	numericUpDown1	Name	numericUpDown
		Maximum	65535
		Value	80

Если Вы запустите программу без внесения кода, то увидите что компоненты label3 и textBox3 не появляются на форме. Это сделано для того, чтобы можно было проверить правильность набора кода программы и проследить в каком месте происходит неверное выполнение кода.

Чтобы программа начала реагировать на действия пользователя надо добавить обработку на компоненты button1, button2. Первое действие – это выходи из программы. Ее код выглядит так:

```
private void button1_Click(object sender, EventArgs e)
    {
      Application.Exit();
    }
```

О действии второй кнопки будет описано позже. В программе описывается 2 реакции на события. Первое событие сгенерирует строку "http://" в компоненте textBox1 в том случае, если пользователь собирается вводить адрес хоста. Второе событие создано для того чтобы проследить за правильностью ввода данных в numericUpDown. Как Вы помните, каждое соединение может быть осуществлено с подключением к определенному порту, если только этот порт не является "портом по умолчанию" в интерпретации протокола HTTP. Согласно таблице 1, мы можем использовать порты из пространства 0-49151. Код следит за корректностью ввода этого числа и если оно выпадает из диапазона – ставит 80-й порт автоматически.

Код для реакции на событие Enter компонента textBox1

```
private void textBox1_Enter(object sender, EventArgs e)
    {
      textBox1.Text = "http://";
    }
```

Код для реакции на событие ValueChanged компонента numericUpDown private void numericUpDown_ValueChanged(object sender, EventArgs e)

```
    {
      int a = (int)numericUpDown.Value;
      if ((a < 0) || (a > 49151))
      numericUpDown.Value = 80;
    }
```

Теперь для корректной работы программы и выполнения поставленного задания нужно сделать обработку кнопки button2, которая как раз и будет инициировать соединение программы с хостом (внешним сервером) по указанному адресу (как в формате n.n.n.n, так и в формате имен), подсоединяться к указанному порту и "вытягивать" необходимые данные. В данном случае нам нужно прочитать содержимое корневого каталога сайта. Но как это сделать в компонент textBox2, если учесть что мы "достаем" по одному байту? Можно организовать массив, помещать все "байты" в него, а затем преобразовывать их в Char, затем в String и отправлять в textBox2. В этом примере был выбран другой способ: создавалась временная переменная, которая хранила текущее значение textBox2.text, затем "вытягивался" следующий байт и он прибавлялся к уже существующему значению. Затем значение снова отправлялось во временную переменную и к нему на следующем шаге прибавлялся новый знак (стоит отметить тот факт, что он сразу конвертировался из byte в char, а затем в string чтобы сразу получить символ, а не байтовое представление). В этом же коде помещена обработка исключительных ситуаций с ошибками протокола HTTP и подсчет количества символов, обработанных программой.

p.s. Конечно же работа через массив была бы намного быстрее, ведь можно сразу добавлять элемент в его конец, а потом отобразить его полностью. Но так как скорость работы программы не была очень важна, а важно было показать последовательность действий, которая производится при запросе данных с веб-узла, был выбран вариант с записью во временную переменную.

Код обработки кнопки button2

```
private void button2_Click(object sender, EventArgs e)
    {
      long ch;
      try
      {
        HttpWebRequest req = (HttpWebRequest)
          WebRequest.Create(textBox1.Text + ":" + numericUpDown.Value);
        HttpWebResponse resp = (HttpWebResponse)
          req.GetResponse();
  Stream istrm = resp.GetResponseStream();
        textBox2.Text = "";
        textBox2.Enabled = true;
        textBox3.Visible = true;
        label3.Visible = true;
        for (long i = 1; ; i++)
        {
          ch = istrm.ReadByte();
          if (ch == -1) break;
          //Создается временная переменная чтобы была возможность
```

```
                //поместить более одного символа в TextBox
                string temp = textBox2.Text;
                string data = Convert.ToString((char)ch);
                //Тут временная переменная складывается с текущим символом
                //и получается непрерывная последовательность символов
                textBox2.Text = temp + data;
                //Вывод количества символов, обработанных программой
                textBox3.Text = Convert.ToString(i);
                progressBar1.Increment(1);
              }
              resp.Close();
            }
            catch (WebException exc)
            {MessageBox.Show(exc.Message + "\n" + exc.Status);}
            catch (ProtocolViolationException exc)
            {MessageBox.Show(exc.Message); }
            catch (UriFormatException exc)
            {MessageBox.Show(exc.Message); }
            catch (NotSupportedException exc)
            {MessageBox.Show(exc.Message); }
            catch (IOException exc)
            {MessageBox.Show(exc.Message); }
        }
```

Задачи для самостоятельного решения:

1-2. Дается список адресов, который содержится в компоненте Data-GridView в одном столбце. Необходимо создать программу, которая во втором столбце отобразит версию программного обеспечения сервера. Использовать адреса "192.168.20.111" и "192.168.20.200" (порт 80).

3-4. Дается список адресов, который содержится в компоненте Data-GridView в одном столбце (адреса "192.168.20.111" и "192.168.20.200" [порт 80]). Обеспечить вывод информации о дате и времени на хостах в компонент TextBox.

5-6. Пользователь вводит адреса для проверки времени последней модификации ПО на сервере через TextBox. После обработки программой все даты записать парами "хост/дата" в две колонки DataGridView.

7-8. Отобразить в неформатированном виде все ссылки (строки, которые содержат значение "<a href=" в корневом каталоге сервера http://192.168.20.111.

9-10. Дается список адресов, который содержится в компоненте Data-GridView в одном столбце (адреса "192.168.20.111" и "192.168.20.200" [порт 80]). Обеспечить вывод информации о дате и времени на хостах во второй столбец компонента DataGridView.

11-12. Загрузить данные из указанного местоположения в файл. Местоположение задается через компонент TextBox. Предусмотреть возможность выбора альтернативного номера порта пользователем.

13-14.Загрузить данные из указанного местоположения в файл. Местоположение задается через столбец компонента DataGridView. Предусмотреть возможность выбора альтернативного номера порта пользователем. 15-17. Дается список серверов (192.168.20.111 и 192.168.20.200) через DataGridView. Осуществить просмотр всех доступных о них сведений из заголовков и сохранить их в файл.

18-21. Дается список серверов (192.168.20.111 и 192.168.20.200) через два поля TextBox. Осуществить просмотр всех доступных о них сведений из заголовков и отобразить их последовательно в TextBox.

22-24. Отобразить всё мета-содержимое (строки, начинающиеся с "<meta " заглавной страницы сервера 192.168.20.111:80. Обеспечить его ввод через TextBox, а вывод в файл.

25-29. Отобразить всё мета-содержимое (строки, начинающиеся с "<meta "заглавной страницы сервера 192.168.20.111:80. Обеспечить ввод адреса через TextBox (но предусмотреть чтобы он уже был в поле в момент запуска программы). Вывод осуществить в другой TextBox.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 72 Введение XAML и WPF

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

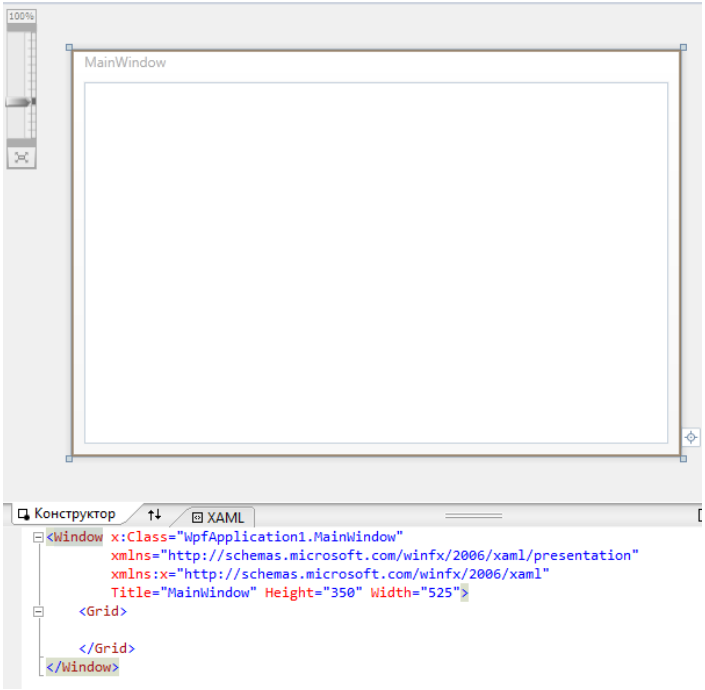
Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Задание1

Создайте проект «Приложение WPF». Среда разработки создаст заготовку, показанную на рисунке:



Среда разработки предоставляет возможность графического и дескрипторного способов разработки пользовательского интерфейса, которые являются равнозначными. Дескрипторный

файл MainWindow.xaml и кодовый файл MainWindow.xaml.cs дополняют друг друга при описании одного и того же содержимого -класса MainWindow в пространстве имен WpfApplication1, совпадающим с названием проекта.

Платформа WPF проектировалась в рамках концепции отделения дизайнерской части пользовательского интерфейса от кодовой части программирования функциональности. Дизайнерская часть проектируется декларативно, чаще всего - с помощью графического дизайнера формы, который в автоматическом режиме генерирует соответствующий синтаксически правильный дескрипторный код на языке XAML.

В заготовке дескрипторного XAML-кода видно, что корнем приложения является контейнер <Window>, в который в дальнейшем будут включены дочерние элементы. Все элементы WPF существуют в двух вариантах: дескрипторном и объектном. Объектное описание размещается в пространствах имен, подключаемых в кодовую часть проекта с помощью инструкции using для видимости компилятором. Дескрипторное описание находится в двух пространствах имен: обычном и расширенном. Эти пространства имен подключаются как значения атрибутов xmlns и xmlns:x в корневом дескрипторе проекта

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

Используемые URL-адреса не указывают на какой-либо документ или содержимое на веб-сервере, а используются лишь для определения уникальных пространств имен.

Разместите в коде XAML в содержимом элемента Grid следующий код:

```
<Button x:Name="Btn1"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        Width="150"
        Height="30"
        FontSize="17"
        Content="Обычная кнопка"
        Foreground="#006699"
        Background="#f0f0f0"
        BorderBrush="#303030" />
```

Запустите приложение и проверьте его поведение при изменении размеров окна.

В приведенном выше примере для элемента Button было задано простое значение атрибута Background. Для этого был использован синтаксис

<ЭЛЕМЕНТ АТРИБУТ=”ЗНАЧЕНИЕ” />

Для задания значения атрибуты может быть использован другой синтаксис:

```
<ЭЛЕМЕНТ>
    <ЭЛЕМЕНТ.АТРИБУТ>
        ЗНАЧЕНИЕ_АТРИБУТА
    </ЭЛЕМЕНТ. АТРИБУТ>
</ЭЛЕМЕНТ>
```

Например, для задания того же значения атрибута Background можно записать:

```
<Button>
    <Button.Background>
      #f0f0f0
    </Button.Background>
</Button>
```

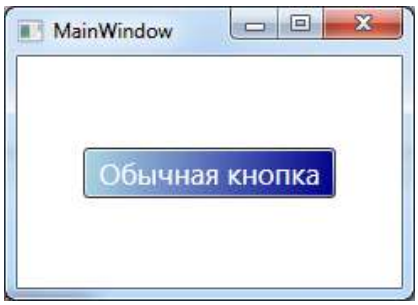
Данный синтаксис используется для задания сложных значений атрибутов в виде дерева элементов. Пример задания для фона кнопки линейной градиентной заливки:

```
<Button.Background>
        <LinearGradientBrush>
          <LinearGradientBrush.GradientStops>
            <GradientStop Color="LightBlue" Offset="0" />
            <GradientStop Color="DarkBlue" Offset="1" />
          </LinearGradientBrush.GradientStops>
 </LinearGradientBrush>
```

</Button.Background>

Данное дерево элементов задает градиентную заливку с использованием двух цветов: LightBlue и DarkBlue. В атрибуте Offset указывается относительное значение от 0 до 1, соответствующее положению цвета на отрезке от начальной точки до конечной.

Внешний вид данного приложения:



Задание 2

В XAML-коде для элемента Windows определите линейную градиентную заливку фона в соответствии с рисунком:

Используемые цвета: DarkBlue и LightBlue.

Необходимо указать четыре промежуточные точки со смещениями 0, 0.2, 0.8 и 1.

Для задания вертикальной заливки необходимо определить атрибуты StartPoint и EndPoint для элемента LinearGradientBrush. Значения этих атрибутов указываются в формате “X,Y”, где X – относительное значение (от 0 до 1) абсциссы точки, Y – относительное значение (от 0 до 1) ординаты точки. Начало координат находится в левом верхнем углу окна. По умолчанию значения атрибутов StartPoint и EndPoint следующие: StartPoint=”0,0” EndPoint=”1,1”.

Контрольные вопросы:

1. Что такое WPF?
2. Что такое XAML?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 73 Диспетчеры компоновки

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Диспетчер компоновки Canvas

Панель Canvas поддерживает абсолютное позиционирование содержимого пользовательского интерфейса. Если пользователь изменяет размер окна, делая его меньше, чем компоновка, обслуживаемая панелью Canvas, ее внутреннее содержимое становится невидимым до тех пор, пока контейнер вновь не увеличится до размера, равного или больше начального размера области Canvas.

Панель Canvas обладает следующим недостатком: элементы внутри Canvas не изменяются динамически при применении стилей или шаблонов.

Рассмотрим панель Canvas со следующим содержимым:

```
<Canvas>
    <Label Canvas.Left="10" Canvas.Top="10" Content="Регистрация пользователя" />
    <Label Canvas.Left="10" Canvas.Top="40" Content="ФИО" />
    <TextBox Canvas.Left="70" Canvas.Top="40" Width="200" />
    <Label Canvas.Left="10" Canvas.Top="70" Content="Email" />
    <TextBox Canvas.Left="70" Canvas.Top="70" Width="200" />
    <Button Canvas.Left="50" Canvas.Top="100" Content="Зарегистрироваться" />
</Canvas>
```

У элементов управления Label, TextBox, Button отсутствуют атрибуты Left и Top, поэтому для определения положения элементов на панели используется синтаксис присоединяемых свойств.

Присоединяемые свойства XAML (attached properties)

В XAML поддерживается специальный синтаксис, используемый для определения значения

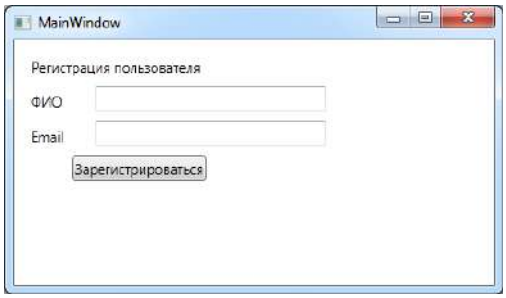
присоединяемого свойства. Присоединяемые свойства позволяют дочернему элементу устанавливать значение какого-то свойства, которое в действительности определено в родительском элементе. Общий шаблон:

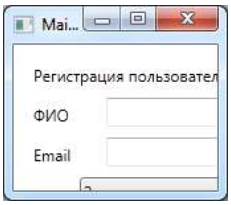
```
<РодительскийЭлемент>
        <ДочернийЭлемент РодительскийЭлемент.СвойствоРодительскогоЭлемента =
"Значение">
</РодительскийЭлемент>
```

С помощью присоединяемых свойств можно определить значения лишь ограниченного набора свойств родительских элементов, которые определены специальным образом в классе родительского элемента.

Дополнительное задание: определите, каким образом присоединяемые свойства объявляются в классе родительского элемента.

Пример работы приложения:

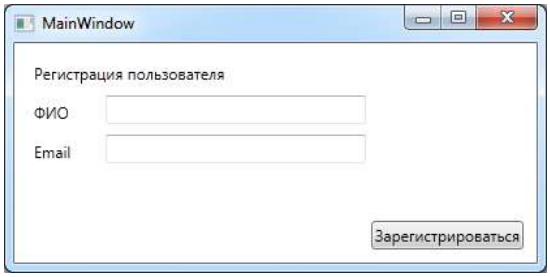


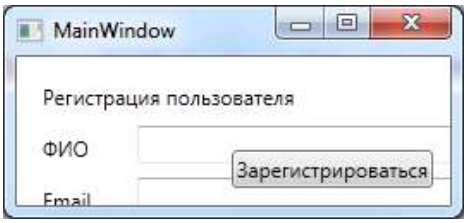


Для дочернего элемента необходимо указать привязку по вертикали (Canvas.Top или Canvas.Bottom) и привязку по горизонтали (Canvas.Left или Canvas.Right). Также можно (не обязательно) задать ширину и высоту элемента с помощью атрибутов Width и Height.

Таким образом, положение дочернего элемента управления можно задать относительно любого угла окна. Например, в следующем примере положение кнопки задано относительно нижнего правого угла окна: <Canvas>

```
...
    <Button Canvas.Right="10" Canvas.Bottom="10" Content="Зарегистрироваться" />
</Canvas>
```





Порядок объявления дочерних элементов управления определяет порядок их вывода на экран. В приведенном выше примере кнопка выводится перед текстовыми полями, т.к. она была объявлена в файле XAML последней.

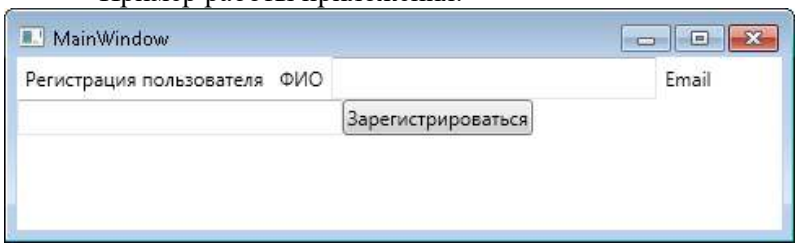
Диспетчер компоновки WrapPanel

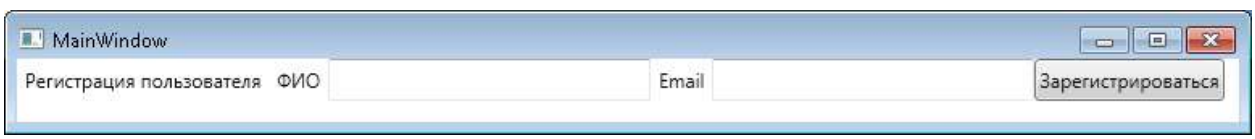
Панель WrapPanel выводит дочерние элементы последовательно слева направо (либо сверху вниз, если для атрибута Orientation установлено значение “Vertical”) и при достижении границы окна переходит на новую строку (столбец). При изменении размеров окна панель перераспределяет компоненты таким образом, чтобы они находились в окне.

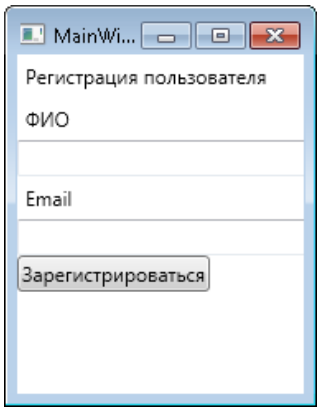
Рассмотрим панель WrapPanel со следующим содержимым:

```
<WrapPanel>
    <Label Content="Регистрация пользователя" />
    <Label Content="ФИО" />
    <TextBox Width="200" />
    <Label Content="Email" />
    <TextBox Width="200" />
    <Button Content="Зарегистрироваться" />
</WrapPanel>
```

Пример работы приложения:







Диспетчер компоновки StackPanel

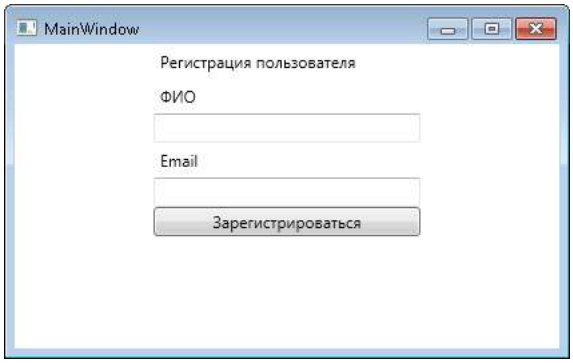
Панель StackPanel располагает содержащиеся в нем элементы управления либо в вертикальном столбце (по умолчанию), либо в горизонтальной строке (если в атрибут Orientation записано значение “Vertical”). Если в панель StackPanel добавлено больше элементов управления, чем может быть отображено по ширине/высоте StackPanel, лишние элементы обрезаются и не отображаются.

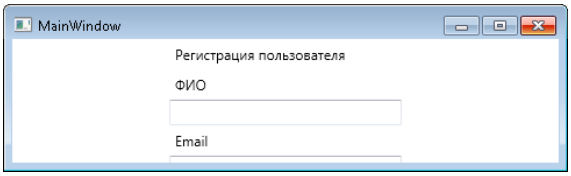
При выводе элементов сверху вниз элементы по умолчанию растягиваются по горизонтали. Это поведение можно изменить с помощью свойств HorizontalAlignment и VerticalAlignment.

Рассмотрим панель StackPanel со следующим содержимым:

```
<StackPanel HorizontalAlignment="Center">
    <Label Content="Регистрация пользователя" />
    <Label Content="ФИО" />
    <TextBox Width="200" />
    <Label Content="Email" />
    <TextBox Width="200" />
    <Button Content="Зарегистрироваться" />
 </StackPanel>
```

Пример работы приложения:





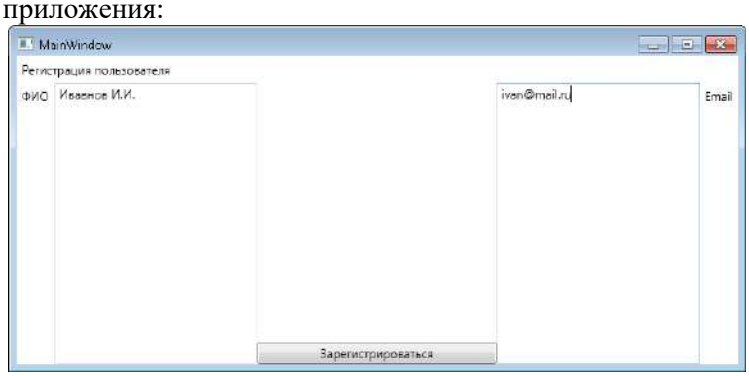
Диспетчер компоновки DockPanel

Панель DockPanel пристыковывает дочерние элементы к различным сторонам панели: Top, Bottom, Left, Right. Атрибут LastChildFill по умолчанию имеет значение True, что означает, что последний дочерний элемент управления будет занимать всё оставшееся пространство панели.

Рассмотрим панель DockPanel со следующим содержимым:

```
<DockPanel LastChildFill="False">
    <Label DockPanel.Dock="Top" Content="Регистрация пользователя" />
    <Label DockPanel.Dock="Left" Content="ФИО" />
    <TextBox DockPanel.Dock="Left" Width="200" />
    <Label DockPanel.Dock="Right" Content="Email" />
    <TextBox DockPanel.Dock="Right" Width="200" />
    <Button DockPanel.Dock="Bottom" Content="Зарегистрироваться" />
</DockPanel>
```

Пример работы приложения:



Диспетчер компоновки Grid

Подобно HTML-таблице, панель Grid может состоять из набора ячеек, каждая из которых имеет свое содержимое. При определении панели Grid выполняются следующие шаги:

1. Определение и конфигурирование каждого столбца.
2. Определение и конфигурирование каждой строки.
3. Назначение содержимого каждой ячейке сетки с использованием синтаксиса присоединяемых свойств.

Если не определить никаких строк и столбцов, то по умолчанию панель Grid будет состоять из одной ячейки, занимающей всю поверхность окна. Кроме того, если не указать ячейку для дочернего элемента, то он разместится в столбце 0 и строке 0.

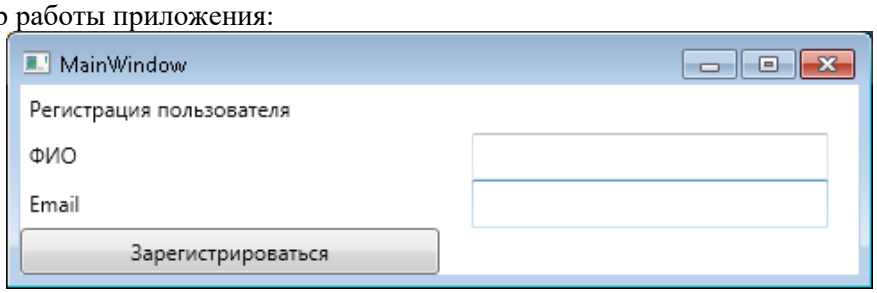
Определение столбцов и строк выполняются за счет использования элементов <Grid.ColumnDefinitions> и <Grid.RowDefinitions>, которые содержат коллекции элементов <ColumnDefinition> и <RowDefinition>, соответственно.

Каждый дочерний элемент прикрепляется к ячейке сетки, используя присоединяемые свойства Grid.Row и Grid.Column. Левая верхняя ячейка определяется с помощью Grid.Column="0" и Grid.Row="0".

Рассмотрим панель Grid со следующим содержимым:

```
<Grid>
      <Grid.RowDefinitions>
        <RowDefinition/>
<RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition/>
        <ColumnDefinition/>
      </Grid.ColumnDefinitions>
      <Label Grid.Row="0" Grid.Column="0" Content="Регистрация пользователя" />
      <Label Grid.Row="1" Grid.Column="0" Content="ФИО" />
      <TextBox Grid.Row="1" Grid.Column="1" Width="200" />
      <Label Grid.Row="2" Grid.Column="0" Content="Email" />
      <TextBox Grid.Row="2" Grid.Column="1" Width="200" />
      <Button Grid.Row="3" Grid.Column="0" Content="Зарегистрироваться" />
 </Grid>
```

Пример работы приложения:

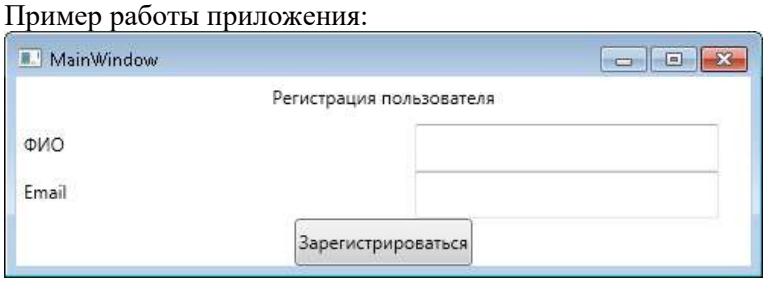


Объединение ячеек осуществляется с помощью присоединяемых свойств Grid.ColumnSpan и Grid.RowSpan аналогично объединению ячеек в HTML-таблицах.

Рассмотрим панель Grid со следующим содержимым:

```
<Grid>
    <Grid.RowDefinitions>
      <RowDefinition/>
      <RowDefinition/>
      <RowDefinition/>
      <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
      <ColumnDefinition/>
      <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Label Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center"
Content="Регистрация пользователя" />
    <Label Grid.Row="1" Grid.Column="0" Content="ФИО" />
    <TextBox Grid.Row="1" Grid.Column="1" Width="200" />
    <Label Grid.Row="2" Grid.Column="0" Content="Email" />
    <TextBox Grid.Row="2" Grid.Column="1" Width="200" />
    <Button Grid.Row="3" Grid.Column="0" Grid.ColumnSpan="2"
HorizontalAlignment="Center" Content="Зарегистрироваться" />
 </Grid>
```

Пример работы приложения:



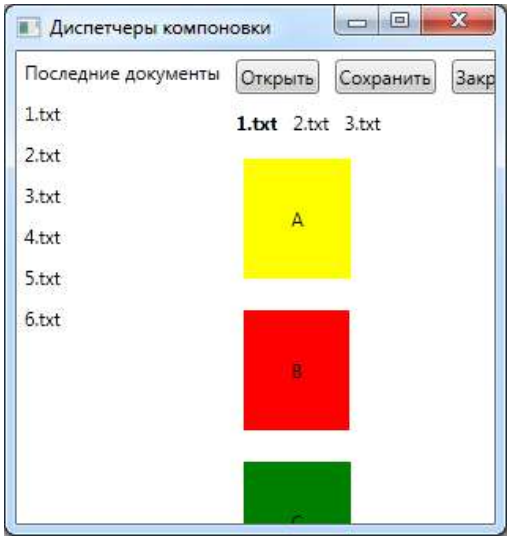
При определении ряда можно задать его высоту с помощью атрибута Height, а при определении столбца можно задать его ширину с помощью атрибута Width. Значение этих атрибутов может быть следующим:

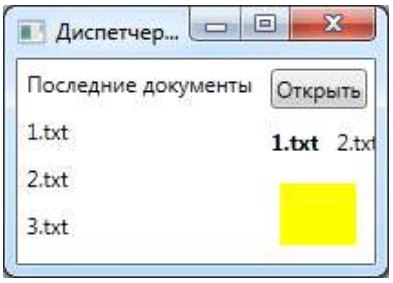
- "Auto" - высота строчки (или ширина колонки) определяется её содержимым;
- "Число" - высота строчки (или ширина колонки) равна указанному числу точек;
- "*" - высота строчки (или ширина колонки) занимает всё свободное пространство. Если строчек

(колонок) с таким значением атрибута несколько, то свободное пространство перераспределяется между ними.

Задание для самостоятельного решения:

Разработать приложение WPF со следующим графическим интерфейсом:





Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 74 Основные элементы управления

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#..

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Разработать WPF-приложение с меню, панелью инструментов и строкой состояния. С помощью пунктов меню пользователь может изменять цвет фона окна, получить информацию о разработчике, а также закрыть окно. Кнопки панели инструментов дублируют команды меню. При наведении на пункты меню или кнопки панели инструментов в строке состояния отображается информация об этих элементах управления.

Задание для самостоятельного решения:

Разработать WPF-приложение «Графический редактор» с выпадающим списком для выбора цвета кисти, ползунком для выбора размеров кисти и зависимыми переключателями для выбора режима работы: «рисование», «редактирование», «удаление».

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 75 Привязка данных

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#..

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Задание 1

Проверьте реакцию среды разработки на неверные значения параметров ElementName и Path. Проанализируйте сообщения, которые выводятся в окне вывода (Вид → Вывод) при построении и при запуске приложения.

Задание 2

Запустите приложение со следующим XAML-кодом:

```
<TextBox x:Name="t1" />
<TextBox x:Name="t2" Text="{Binding ElementName=t1, Path=Text}" />
<Slider x:Name="slider1" />
<Slider x:Name="slider2" Value="{Binding ElementName=slider1, Path=Value}" />
```

Определите различие в поведении полей t1 и t2 и модифицируйте код, чтобы устранить это различие

Задание 3

Дополните пример №2 текстовым полем ввода TextBox, в котором пользователь может ввести размер шрифта, и задайте выражения привязки таким образом, чтобы значение ползунка, текст текстового поля и размер шрифта текстового блока соответствовали друг другу.

Задание для самостоятельного решения:

Модифицируйте приложения, разработанные в предыдущем практическом занятии: удалите как можно больше обработчиков событий и реализуйте ту же функциональность приложения с помощью привязки данных.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 76 Использование стилей в WPF-приложении

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Задание 1

Проверьте, какое значение имеет больший приоритет: значение свойства, указанное в стиле, или значение атрибута элемента.

Задание 2

Модифицируйте WPF-приложение, разработанное в практическом занятии 74: используйте стили для однотипных элементов управления.

Задание для самостоятельного решения:

Разработайте приложение MultiEdit для одновременной работы с несколькими текстами. Окно должно быть разделено на две части с одинаковыми градиентами. В каждой части окна должно быть несколько многострочных текстовых полей: одно из них большого размера с крупным шрифтом, а остальные маленького размера с мелким шрифтом. То текстовое окно, в котором пользователь набирает текст, должно быть большим, остальные текстовые поля должны быть маленькими. Внешний вид однотипных элементов управления должен определяться с помощью стилей.

Изменить стиль элемента управления в коде можно следующим образом:

(sender as FrameworkElement).Style = (Style)Resources["ИМЯ_СТИЛЯ"];

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 77 Триггеры в WPF-приложениях

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Задание 1

Рассмотрите случай, когда для одного и того же элемента управления срабатывают сразу несколько триггеров, устанавливающих для одного и того же свойства различные значения, и определите правило, по которому определяется приоритет применения элементов Setter этих триггеров.

Задание для самостоятельного решения:

Разработайте WPF-приложение с двумя многострочными текстовыми полями, кнопками «Открыть», «Очистить», «Закрыть» и выпадающим списком для задания внешнего вида текстовых полей. Задайте для текстовых полей одинаковый градиентный фон. Кнопка «Закрыть» должна быть доступна только в том случае, если в обоих текстовых полях нет текста. Задайте для кнопок

различный внешний вид при наведении курсора и при нажатии на них. Внешний вид текстовых полей (тип шрифта, размер шрифта, цвет шрифта) должен меняться в зависимости от значения, выбранного в выпадающем списке.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 78 Трансформация в WPF-приложениях

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

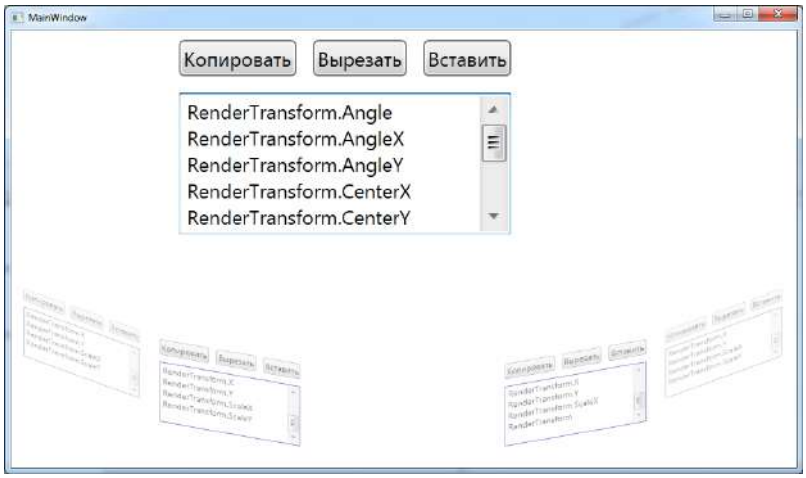
Для любого WPF-приложения с большим количеством элементов управления, разработанного на одном из предыдущих практических занятий, реализуйте масштабирование всего пользовательского интерфейса с помощью ползунка Slider.

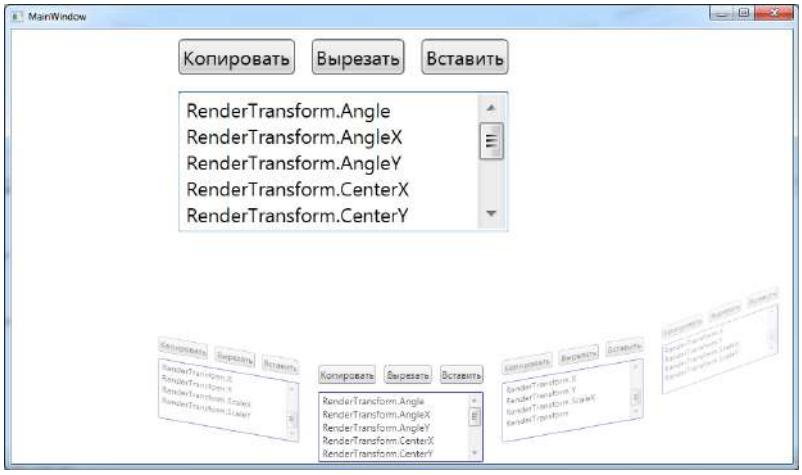
Задание для самостоятельного решения:

Разработайте WPF-приложение «Текстовый редактор» в соответствии с изображениями пользовательского интерфейса, приведенными ниже. Блоки с многострочными текстовыми полями выводятся в нижней части интерфейса «полукругом». При выделении какого-либо блока (событие GotFocus) он плавно перемещается в центральную часть. При потере фокуса (событие LostFocus) блок возвращается в свое исходное состояние. Приложение не должно содержать код на языке C#.

Подсказки:

- используйте диспетчер компоновки Canvas;
- если для анимации не заданы атрибуты To и By, то целевым значением является исходное значение свойства, действующее до начала серии анимаций.





Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 79 Использование кистей в WPF

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;

- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

На основе примера из практического занятия 78 разработайте WPF-приложение с анимацией источника света, светового пятна от него на шаре и отражения шара. Для анимации начала радиального градиента используйте элемент PointAnimation, свойства From и To которого задаются в формате ”X,Y” (пример: To=”0,1”)

Контрольные вопросы:

1. Для чего нужны кисти в WPF?

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 80 WPF-калькулятор

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Разработать WPF-калькулятор.

Задание для самостоятельного решения:

Доработать калькулятор – обычный и инженерный.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 81 Выполнение конкурсного задания WorlSkills компетенции Программные решения в бизнесе

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Выполнить конкурсоное задание WorldSkills по компетенции Программные решения для бизнеса.

Задания для самостоятельной работы:

1. Спроектировать и создать базу данных
2. Спроектировать интерфейс программного продукта в соответствии с заданием.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

Практическое занятие 82 Выполнение конкурсного задания WorlSkills компетенции Программные решения в бизнесе. Дифференцированный зачет

ПК1. Выполнять разработку спецификаций отдельных компонентов.

ПК2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.

ПК3. Выполнять отладку программных модулей с использованием специализированных программных средств.

ПК4. Осуществлять тестирование программных модулей.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся должен:

уметь:

- осуществлять разработку кода программного модуля на современных языках программирования;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля.

Цель: Изучить основы верстски графического дизайна на XAML и создания WPF-прилжений на языке C#.

Оборудование: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение: VisualStudio 2019

Порядок выполнения работы:

Выполнить конкурсоное задание WorldSkills по компетенции Программные решения для бизнеса.

Задания для самостоятельной работы:

1. Доработать проект.
2. Выполнить тестирование.

Критерии оценивания:

Практическая работа оценивается следующим образом:

оценка «5» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил пять задач для самотоятельного решения.

оценка «4» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил четыре задач для самотоятельного решения.

оценка «3» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил три задач для самотоятельного решения.

оценка «2» ставится, если:

- обучающийся выполнил задания, ответил на контрольные вопросы и решил менее трех задач для самотоятельного решения.

2.2 Теоретические вопросы для проведения устного опроса

- основные принципы отладки и тестирования программных продуктов;

1. Что такое тестирование?
2. Виды тестирования.
3. Принципы отладки.
4. Методы тестирования.
5. Стадии тестирования.

- работа с базами данных в сети

1. Что такое база данных?

2. Что такое распределенная база данных?

3. Что такое СУБД?

4. По какой технологии можно подключать распределённые базы данных в C++ Builder?

5. Технология InterBase.

6. Технология ADO.NET

- технология WPF

1. Что такое WPF?
2. Что такое XAML?
3. Назовите преимущества WPF.
4. Какие основные компоненты составляют архитектуру WPF?
5. Приведите ключевые иерархии фундаментальных классов WPF.
6. Как осуществляется компоновка элементов управления в WPF?
7. Назовите основные контейнеры компоновки в WPF.
8. Поясните назначение и возможности класса Grid.
9. Поясните назначение и возможности класса StackPanel.
10. Для чего предназначены элементы управления содержимым. Приведите примеры элементов управления содержимым.
11. Поясните назначение класса ContentControl.
12. Какие текстовые элементы управления имеются в WPF? Поясните их назначение.
13. Какие текстовые элементы управления списками имеются в WPF? Поясните их назначение.
14. Какие специализированные элементы управления имеются в WPF? Поясните их назначение.
15. Поясните назначение командной модели WPF. Приведите пример использования команд.
16. Для чего предназначены ресурсы в WPF и как они определяются и используются?

17. Для чего предназначены стили в WPF и как они определяются и используются?

18. Для чего предназначены шаблоны в WPF и как они определяются и используются?

19. Поясните назначение и приведите примеры шаблонов элементов управления.

20. Поясните назначение и приведите примеры шаблонов данных.

КРИТЕРИИ ОЦЕНКИ:

Оценка «отлично» выставляется при 5 правильных ответах на 5 любых вопросов.

Оценка «хорошо» выставляется при 4 правильных ответах на 4 любых вопроса.

Оценка «удовлетворительно» выставляется при 3 правильных ответах на 3 любых вопроса.

При ответе менее чем на 3 любых вопроса выставляется оценка «неудовлетворительно».

2.3. Тесты для проведения тестирования

- методы и средства разработки технической документации

Задание 1. Сопоставьте термины и их определения из государственного стандарта:

__ Документ *(1)*

__ Документация *(2)*

__ Программная продукция *(3)*

__ Программное изделие *(4)*

__ Программный документ *(5)*

__ Эксплуатационный документ *(6)*

Укажите соответствие для всех 6 вариантов ответа:

1) Уникально обозначенный блок информации для использования человеком, такой как отчёт, спецификация, руководство или книга

2) Набор из одного или более связанных документов

3) Результат процесса разработки программного обеспечения, то есть ПО, выпускаемое для использования

4) Программа на носителе данных, являющаяся продуктом промышленного производства

5) Документ, содержащий сведения, необходимые для разработки, изготовления, эксплуатации и сопровождения программного изделия

6) Программный документ, содержащий сведения, необходимые для обеспечения функционирования и эксплуатации программного изделия

Задание 2. Выберите, к какому виду документации - программной или эксплуатационной - относится тот или иной документ:

__ Ведомость держателей подлинников *(1)*

__ Программа и методика испытаний *(1)*

__ Пояснительная записка *(1)*

__ Описание языка *(2)*

__ Формуляр *(2)*

__ Описание применения *(2)*

Укажите соответствие для всех 6 вариантов ответа:

1) Программная документация

2) Эксплуатационная документация

Задание 3. Введите двузначное число - серию ГОСТ ЕСПД.

Запишите число:

_____ *(19.)*

Задание 4. Из предложенных литер составьте код ГОСТа, в котором приведены основные термины и определения, касающиеся программной документации.

Составьте слово из букв:

19-400.80 -> _____ (*19.004-80)*

Задание 5. В приведённом списке выберите документы, которые относятся к программной документации

Выберите несколько из 5 вариантов ответа:

1) Спецификация

2) Программа и методика испытаний

3) Техническое задание

4) Пояснительная записка

5) Формуляр

Задание 6. В приведённом списке выберите документы, которые относятся к эксплуатационной документации.

Выберите несколько из 6 вариантов ответа:

1) Спецификация

2) Программа и методика испытаний

3) Техническое задание

4) Формуляр

5) Описание применения

6) Пояснительная записка

Задание 7. Сопоставьте эксплуатационный документ и его содержание:

__ Сведения для проверки, обеспечения функционирования и настройки программы на условия конкретного применения *(1)*

__ Сведения для эксплуатации программы *(2)*

__ Сведения для обеспечения процедуры общения оператора с вычислительной системой в процессе выполнения программы *(3)*

__ Сведения для применения тестовых и диагностических программ при обслуживании технических средств *(4)*

Укажите соответствие для всех 4 вариантов ответа:

1) Руководство системного программиста

2) Руководство программиста

3) Руководство оператора

4) Руководство по техническому обслуживанию

Задание 8. Введите недостающее слово.

Руководство по техническому обслуживанию содержит сведения для применения тестовых и ... программ при обслуживании технических средств

Запишите ответ:

_____ (*диагностических)*

Задание 9. Введите недостающее слово.

Рабочая документация на автоматизированную систему - это часть документации на АС, необходимой для изготовления, строительства, монтажа и наладки автоматизированной системы в целом, а также входящих в систему программно-технических, программно-... комплексов и компонентов технического, программного и информационного обеспечения

Запишите ответ:

_____ *(методических)*

Задание 10. Укажите истинность или ложность вариантов ответа.

Укажите истинность или ложность вариантов ответа:

__ Сведения о логической структуре и функционировании программы указаны в описании программы *(Да)*.

__ Схема алгоритма приведена в описании программы *(Нет)*.

__ Формуляр содержит основные характеристики программы, комплектность и сведения об эксплуатации программы *(Да)*.

__ Состав программы и документации на нее указывается в спецификации *(Да)*.

__ Необходимые стадии и сроки разработки приведены в описании программы *(Нет)*.

__ Обоснование принятых технических и технико-экономических решений содержится в пояснительной записке *(Да)*.

- операционные системы Android, IOS

1. Назовите подходящий источник безопасной загрузки приложений Android.

+Google Play

iTunes

сайты с бесплатным ПО

магазин приложений Apple

2. Какие две функции можно выполнить с помощью кнопки «Домой» на мобильном устройстве iOS? (Выберите два варианта.)

+те же функции, что и с помощью кнопок навигации Android

запуск интерфейса TouchFLO

сброс устройства до заводских настроек по умолчанию

блокирование устройства

+запуск голосового управления

3. Назовите два потенциальных преимущества для пользователей от обеспечения доступа с правами «root» и взлома защиты мобильных устройств. (Выберите два варианта.)

Блокирование корневого каталога устройства.

Возможное воздействие на работу сети сотовой связи оператора.

+Возможность тонкой настройки операционной системы для увеличения скорости работы устройства.

+Интерфейс пользователя может быть модифицирован в значительной степени.

4. Какая функция обеспечения безопасности мобильного устройства требует от пользователя определенным образом провести пальцем по экрану для разблокирования устройства?

простой секретный код

скольжение

+шаблон

дистанционное стирание

5. Какой компонент может быть заменен пользователем в большинстве устройств?

ЦП

+карта памяти

сенсорный экран

ОЗУ

6. После обновления мобильного устройства некоторые приложения больше не работают, а другие работают хорошо. Пользователь может делать звонки и получать электронную почту. Какие два типа проблем могут быть причиной этой ситуации? (Выберите два варианта.)

проблема с монтажной платой

+проблема с памятью

проблема с сенсорным экраном

проблема со службой сотовой связи

+проблема с программным обеспечением

7. Какие случай или ситуация не позволят пользователю обновить ОС на устройстве Android?

Не были установлены необходимые исправления.

Не было получено разрешение от производителя.

Авторские права на программное обеспечение не были получены перед установкой.

+Оборудование не соответствует спецификациям.

8. Какие два элемента являются общими для устройств Android и iOS и поддерживают дистанционное резервное копирование? (Выберите два варианта.)

+календарь

сообщения

настройки

+контакты

фотографии

9. Какое утверждение о беспроводных подключениях на мобильном устройстве Android является верным?

+Если устройство выходит за пределы диапазона какой-либо сети Wi-Fi, оно может подключиться к сотовой сети передачи данных, если такая функция включена.

ОС Android не поддерживает беспроводную связь с использованием шифрования WPA2.

Если сеть Wi-Fi защищена паролем, эта сеть настраивается на устройстве вручную.

Без широковещательной рассылки имени сети (SSID) устройство не сможет подключиться к сети.

10. Что обозначает термин «режим точки доступа» в контексте мобильных устройств?

подключение мобильного устройства к порту USB компьютера для подзарядки мобильного устройства

подключение мобильного устройства к гарнитуре

подключение мобильного устройства к сети сотовой связи 4G

+подключение мобильного устройства к другому мобильному устройству или компьютеру для совместного использования сетевого подключения

11. Какое утверждение о синхронизации мобильного устройства с компьютером является верным?

+Для синхронизации данных на устройстве iOS на компьютере необходимо установить iTunes.

Синхронизация iOS может быть выполнена только через подключение по USB.

Данные на устройствах Android невозможно синхронизировать с компьютером.

Motocast USB можно использовать для синхронизации данных на устройстве iOS.

КРИТЕРИИ ОЦЕНКИ:

При выполнении заданий ставится отметка:

«отлично» - за правильное выполнение более 85% заданий.

«хорошо» - за 70-85% правильно выполненных заданий,

«удовлетворительно» - за 50-70% правильно выполненных заданий,

При выполнении менее 50% ставится оценка «неудовлетворительно»

2.4 Задания для проведения дифференцированного зачета

Теоретические вопросы:

1 Объектно-ориентированное программирование. Основы визуального программирования интерфейса

2 Работа в интегрированной среде разработки (ИСР) C++ Builder (проекты C++ Builder, включение в проект новой формы, размещение компонентов на форме)

3 Инструментальные средства поддержки разработки кода

4 Отладка приложений

5 Архитектура системы компонентов

6 Организация управления приложением

7 Разработка графического интерфейса (проектирование окон с изменяемыми размерами, обработка событий мыши, обработка событий клавиатуры, формы)

8 Перетаскивание объектов (технология Drag&Drop)

9 Встраивание и буксировка объектов (технология Drag&Doc)

10MDI-приложения

11Порождение дочерних процессов

12Управление окнами внешних программ

13Организация приложения с несколькими потоками

14Технология ADO

15Основные запросы SQL

16Компоненты для работы с технологией ADO

17FastReport

18Технология InterBase

19Компоненты для работы с технологией InterBase

20Фреймворк FireMonkey. Разработка мобильных приложений

21Использование Microsoft Visual Studio (C++) для создания прикладных приложений: что такое .Net. Общеязыковая среда выполнения

22Использование Microsoft Visual Studio (C++) для создания прикладных приложений: Интернет-программирование

23Использование Microsoft Visual Studio (C++) для создания прикладных приложений: Программирование взаимодействия приложения с базой данных

24Использование Microsoft Visual Studio (C++) для создания прикладных приложений: Программирование взаимодействия приложения с базой данных MySQL

25Использование Microsoft Visual Studio (C++) для создания прикладных приложений: Доступ к базам данных с помощью ADO.NET

26Использование Microsoft Visual Studio (C#) для создания прикладных приложений: что такое .Net. Общеязыковая среда выполнения

27Использование Microsoft Visual Studio (C#) для создания прикладных приложений: Программирование взаимодействия приложения с базой данных

28Использование Microsoft Visual Studio (C#) для создания прикладных приложений: Программирование взаимодействия приложения с базой данных MSSQL

29Использование Microsoft Visual Studio (C#) для создания прикладных приложений: Программирование взаимодействия приложения с базой данных MySQL

30Использование Microsoft Visual Studio (C#) для создания прикладных приложений: Доступ к базам данных с помощью ADO.NET

Типовые задания для оценки освоения МДК 01.02:

Задание 1: Разработать приложение, которое «приветствует» пользователя

Задание 2: Разработать приложение «Сопротивление» для вычисления сопротивления электрической цепи, состоящей их двух сопротивлений. Сопротивления могут быть соединены последовательно или параллельно. Если величина сопротивления цепи превышает 1 000 Ом, то результат должен быть выведен в килоомах.

Задание 3: Разработать приложение «Электроэнергия»

Задание 4: Разработать приложение «Двигающаяся кнопка»

Задание 5: Разработать приложение «Прыгающая кнопка»

Задание 6: Разработать приложение «Вычисление процентов»

Задание 7: Разработать приложение, которое позволяет выполнить пересчет скорости ветра из «метров в секунду» в «километры в час».

Задание 8: Разработать приложение для пересчета массы из фунтов в килограммы (1 фунт =409,5 грамм). Кнопка **Пересчет** должна быть доступна только в том случае, если пользователь ввел исходные данные. Разрешается вводить целые и вещественные значения (разделитель точка и только одна). Добавить на форму кнопку, при щелчке по которой удаляются значения из полей ввода и вывода.

Задание 9: Разработать приложение, которое вычисляет скорость (км/час), с которой спортсмен пробежал дистанцию. Количество минут задается целым числом, секунд – вещественным.

Задание 10: Разработать приложение для вычисления стоимости покупки с учетом скидки. Скидка 1% предоставляется, если сумма покупки больше 300 руб., 2% – если больше 500 руб., 3% – если больше 1000 руб. Информация о предоставленной скидке должна быть выведена в диалоговое окно.

Задание 11: Разработать приложение для вычисления стоимости проезда на автомобиле на дачу.

Задание 12: Разработать приложение «Вычисление функций по заданным формулам»

Задание 13: Разработать приложение «Собери 50!» – компьютерную версию одной из головоломок Самуэля Ллойда: из заданного набора чисел надо выбрать те, сумма которых составит 50. Числа, которые избрал Лллойд для своей головоломки: 25, 27, 3, 12, 6, 15, 9, 30, 21, 19.

Задание 14: Разработать приложение компьютерную версию головоломки: из изображенных пяти сброшенных флажков установить все. Но при выборе одного флажка меняются состояния двух следующих.

Задание 15: Разработать приложение, которое тестирует учащегося по информатике и математике.

Задание 16: Разработать приложение, выполняющее следующие действия: после запуска программы в окне изображается две полосы

прокрутки. Вертикальная полоса будет управлять движением по вертикали, горизонтальная – по горизонтали. Наводя указатель на одну из двух фигур, можно выбирать, какая из этих фигур связана с полосами прокрутки.

Задание 17: Текстовый файл содержит несколько вопросов и 4 варианта ответа, из которых только один ответ верный (помечен символом +). Файл имеет следующую структуру:

Тема теста (дисциплина)

/вопрос 1

- ответ 1

+ ответ 2

- ответ 3

- ответ 4

/вопрос 2

+ вопрос 1

- вопрос 2

- вопрос 3

- вопрос 4

Создать текстовый файл с тестом из 5 вопросов в соответствии с заданной структурой.

Разработать приложение, которое позволяет выбирать файл с тестовыми заданиями и проводить последовательное тестирование. Предусмотреть вывод результат тестирования на экран.

Задание 18: Создать программу, выполняющую следующие действия: после запуска программы в окне изображается строка меню (Файл, Выход). При выборе пункта меню Файл, появляются пункты меню (Рисунки, Выход). При выбору пункта меню Рисунки появляется вложенное меню, состоящее из двух пунктов (Облака, Лес). По щелчку правой кнопкой мыши появляется контекстное меню. Выбрать по пункту другой рисунок. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка. Если выбрать любой из пунктов Выход, работа программы завершается.

Задание 19: Текст задания: Создать программу «Работа с StringGrid». Дана функция 1/(5-cos(x)), в компонент Edit1 вводится минимальное значение интервала, в Edit2 вводится максимальное значение интервала, в Edit3 – шаг табуляции. Компонент StringGrid на две строки x и y. Провести табулирование функции, значения которой будут отображаться в StringGrid.

Задание 20: Разработать приложение «Состав компьютера». При выборе соответствующего пункта меню или пиктограммы на панели инструментов на экране отображается графическое изображение устройства и информация о назначении устройства. При наведении курсора отображается всплывающая подсказка о названии устройства.

Задание 21: Разработать приложение, реализующее основные функции текстового редактора:

- форматирование шрифта для выделенного контекста;
- копирование и перемещение выделенного контекста;
- выравнивание абзацев;
- поиск и замена в тексте;
- открытие и сохранение текстового файла;

Все функции приложения должны быть доступны через главное меню и панель инструментов.

Задание 22: Разработать приложение, выполняющее следующие действия: после запуска программы в окне изображается три поля. По щелчку мышью на кнопке «Случайный выбор» из трех слов составляется предложение случайным образом. Для выхода из программы необходимо щелкнуть мышью на закрывающейся кнопке в строке заголовка.

Задание 23: Разработать приложение «Игра в пятнашки». Алгоритм игры следующий: в прямоугольной коробке находится 15 фишек, на которых написаны числа от 1 до 15. Размер коробки - 4x4, таким образом, в коробке одна пустая ячейка. В начале игры фишки перемешаны. Задача игрока состоит в том, чтобы, не вынимая фишки из коробки, выстроить фишки в правильном порядке (по возрастанию).

Задание 24: Разработать приложение «Тригонометрические функции». Приложение должно работать следующим образом:

- после загрузки на экран выводится окно-заставка, которое отображается 5 сек;
- через 5 секунд заставка пропадает, на экран выводится основное окно;
- в основном окне отображается таблица со значениями тригонометрических функций с заданным шагом.

Задание 25: Разработать приложение, выполняющее следующие действия: после запуска программы, в окне отображается два движка. Необходимо выбрать два числовых значения и найти их произведение. Если выбирается одно число, то находится его квадрат.

Задание 26: Разработайте приложение «Клавиатура». Приложение должно работать следующим образом:

при щелчке по одной из «клавиш» в текстовое окно добавляется выбранная буква или пробел;

при щелчке по кнопке «Очистить», текст полностью исчезает.

Кнопки с буквами – это массив объектов (командные кнопки или метки).

Задание 27: Разработать игровое приложение, которое работает следующим образом:

после загрузки приложения картинки (массив объектов) меняют свои координаты случайным образом (скорость перемещения отрегулируйте сами)

при щелчке по любой картинке, она становится невидимой;

работа программы заканчивается тогда, когда все изображения становятся невидимыми. Примерный вид окна представлен на рисунке.

Задание 28: В верхней части окна приложения непрерывно движется объект "Противник". В нижней части окна – объект "Охотник", который может перемещаться влево и вправо, а также стрелять в "противника" вверх пулей. При попадании пули в "Противника" объект "Охотник" увеличивает

свои баллы. Цель может быть опасна, т.к. сбрасывает объект "Бомбочки", которые при попадании в охотника уменьшают его баллы.

В начале игры у "Охотника" 10 баллов. Игра заканчивается победой, если "Охотник" удваивает баллы, и поражением, если баллы равны нулю.

Задание 29: Разработать игру "Клавиатурный тренажер", который работает следующим образом: сверху вниз "падает" буква (метка с одной буквой, код буквы – случайное число от 192 до 233). если буква "упала", то сверху начинает падать другая буква. Игрок должен нажать соответствующую клавишу на клавиатуре. Программа должна регистрировать количество ошибок (клавиша нажата неверно или буква "упала" и кнопка не была нажата). Если количество ошибок превысит 10, то игрок проиграл.

Задание 30: Расположить на форме таймер, фигуру "круг и фигуру "прямоугольник". Разработать приложение, которое работает следующим образом: прямоугольник выполняет роль лапты, он перемещается влево и вправо. Круг – это мячик, который постоянно прямолинейно двигается при столкновении с границами формы или лаптой меняет свое направление на противоположное.

Задание 31: Разработать приложение "Колобок", которое должно работать следующим образом: "колобок" постоянно жует (открывает/закрывает рот) и прямолинейно движется. При нажатии клавиш управления курсором "колобок" меняет направление движения и "смотрит" туда, куда движется.

Задание 32:

Удалить из массива A размером N, состоящего из целых чисел (положительных и отрицательных), все отрицательные числа. Новый массив не создавать. Для заполнения массива использовать фунцию random(kod) – генератор случайных равномерно распределенных целых чисел от 0 до (int)kod. Значение N вводить из Edit, значения массива A – из компоненты StringGrid. Результат вывести в компоненту StringGrid.

Задание 33: Разработать приложение, реализующее пузырьковую сортировку элементов одномерного массива. Предусмотреть визуализацию алгоритма сортировки через использование массива объектов. Объекты имеют различную высоту, каждая из которых соответствует значению элемента массива.

Задание 34: Разработать приложение, которое выполняет следующие действия: после запуска программы появляется надпись "Брось кубик". По щелчку мышью на кнопке "Бросок кубика" появляется сообщение, выдающее числа-очки в диапазоне 0 - 6. Для выхода из программы необходимо щелкнуть мышью на закрывающей кнопке в строке заголовка.

Задание 35: Разработать приложение, выполняющие действия строкового калькулятора.

Задание 36: Разработать приложение, использующее технологию Drag&Drop для работы пользователя с интерфейсом программы.

Задание 37: Разработать приложение "Файловый менеджер", позволяющее пользователю работать с файловой системой используемого компьютера: просмотр содержимого внешних носителей информации, просмотр дерева каталогов диска, просмотр содержимого выбранного каталога, удаление и переименование файла, определение объема свободной памяти диска.

Задание 38: Разработать приложение, выполняющее следующие действия: после запуска программы в оке изображается светофор с тремя лампочками, способными реагировать на наведение указателя мыши. Когда указатель мыши наведен на лампочку, она меняет свой цвет.

Задание 39: В СУБД Access создана база данных "Телефонный справочник". Используя технологию ADO, разработать приложение, которое будет работать с этой базой.

Задание 40: Создать Web-браузер.

Задание 41: В окне программы График отображается график изменения курса доллара.

Задание 42: В окне программы Диаграмма отображается график изменения курса доллара.

Задание 43: Разработать приложение, позволяющее воспроизводить звуковые файлы различных форматов, приложение должно выводить информацию о длине звукового файла, времени воспроизведения и времени, которое осталось до конца звучания файла.

Задание 44: Разработать приложение, реализующее движение битового образа (картинки) на фоне другой картинки.

Задание 45: Программа ОСАГО позволяет рассчитать размер страховой премии, подлежащей уплате по договору обязательного страхования гражданской ответственности. Демонстрирует использование компонента ComboBox, обработку одной функцией событий от нескольких компонентов. Программа спроектирована таким образом, что кнопка OK доступна только в том случае, если введены все данные, необходимые для расчета.

Задание 46: Программа Просмотр иллюстраций позволяет просмотреть файлы формата JPEG. Выбор рабочей папке выполняется в стандартном окне Выбор папки. Иллюстрации можно просматривать по кадрам или в режиме слайд-шоу. Частоту смену кадров в режиме слайд-шоу определяет значение свойства Interval таймера.

Задание 47: Программа Любимый напиток демонстрирует использование компонента ComboBox. Списки компонентов формируются во время работы программы (делает это конструктор формы). Пользователь может добавить элементы в списки компонентов.

Задание 48: Разработка программы "Графики", отображает график функции y=sin(x).

Задание 49: Разработать приложение для рекурсивного построения изображения

Задание 50: Разработать WPF-калькулятор.

Дифференцированный зачет (5 семестр)

Количество вопросов: 1-19

Количество вариантов: 25 (1-25)

Время выполнения каждого задания: 30 мин.

Дифференцированный зачет (6 семестр)

Количество вопросов: 20-30

Количество вариантов: 25 (26-50)

Время выполнения каждого задания: 30 мин.

Оборудование:

Оборудование рабочих мест обучающихся: ноутбук Dell Inspiron 15 5000 Series

Программное обеспечение:

- Microsoft Visual Studio 2019;
- C++ Builder (Embarcadero RAD Studio XE2).

Литература для обучающегося:

Перечень учебных изданий, Интернет-ресурсов, дополнительной литературы
Основные источники:

Учебные издания, рекомендованные МО РФ и УМЦ СПО ФАС, для образовательных учреждений среднего профессионального образования.

1. Немцова Т.И., Голова С.Ю., Терентьев А.И. Программирование на языке высокого уровня. Программирование на языке С++: учебное пособие / под редакцией Л.Г. Гагариной. – М.: ИД "ФОРУМ": ИНФРА-М, 2016. – 512 с.: ил. – (Профессиональное образование)
2. Подбельский В.В. Программирование. Базовый курс C#: учебник для СПО / В.В. Подбельский . – Москва: Издательство Юрайт, 2019 – 369 с.
3. Серкова Е.Г. Основы алгоритмизации и программирования (ОП.4): практикум/ Е.Г. Серкова. – Ростов н\Д: Феникс, 2019. – 188 с.

Дополнительные источники:

1. Ашарина И.В., Крупская Ж.Ф. Язык С++ и объектно-ориентированное программирование в С++. Лабораторный практикум. Учебное пособие для вузов. – М.: Горячая линия – Телеком , 2015. – 232 с.
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. 3-е издание (переработанное). – СПб.: Наука и Техника, 2015. – 480 с.
3. Гагарина Л.Г., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: учебное пособие / под ред. Л.Г. Гагариной. – М.: ИД "ФОРУМ": ИНФРА-М, 2015. – 400 с.: ил. – (Высшее образование)
4. Катупия Янта, Бентли Ким Управление электронными устройствами на С++. Разработка практических приложений. / Перевод с англ. Бакомчев И.В. – М.: ДМК Пресс, 2016. – 442 с.
5. Колдаев В.Д. Основы алгоритмизации и программирования: учебное пособие / под ред. проф. Л.Г. Гагариной. – М.: ИД «ФОРУМ»: ИНФРА-М, 2015. – 416 с. с.: ил. – (Профессиональное образование)
6. Культин Н.Б. С++ Builder в задачах и примерах. – СПб.: БХВ-Петербург, 2007. – 336 с.

7. Тимофеев В.В. Самоучитель C++ как он есть. – М.: Издательство Бином, 2014. – 336 с., ил.
8. Лафоре Р. Объектно-ориентированное программирование в C++. Классика Computer Science. 4-е изд. – СПб.: Питер, 2016. – 928 с.
9. Прата, Стивен. Язык программирования C++. Лекции и упражнения, 6-е изд.:Пер.с англ. –М.:ООО «И.Д. Вильямс»,2016 – 1248 с.;ил.
10. Сикорд Роберт С Безопасное программирование на C и C++, 2-е изд.: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2015. – 496 с.
11. Шилдт, Герберт C++: базовый курс, 3-е изд.: Пер. с англ.. – М.: ООО «И.Д. Вильямс», 2015. – 642 с.

КРИТЕРИИ ОЦЕНКИ

Оценка «отлично» выставляется при выполнении 3 приложений на выбор студента из предложенных вариантов и правильных ответов на 5 вопросов.

Оценка «хорошо» выставляется при выполнении 2 приложений на выбор студента из предложенных вариантов и правильных ответов на 4 вопроса.

Оценка «удовлетворительно» выставляется при выполнении 1 приложения на выбор студента из предложенных вариантов и правильных ответов на 3 вопроса.

При отсутствии хотя бы одного приложения выставляется оценка «неудовлетворительно».

ПРИЛОЖЕНИЕ А
(обязательное)

Лист ознакомления обучающихся

ЛИСТ ОЗНАКОМЛЕНИЯ ОБУЧАЮЩИХСЯ
с формами, процедурой текущего, рубежного контроля знаний, промежуточной аттестации по дисциплинам, профессиональным модулям, содержанием комплекта оценочных средств

Дисциплина МДК.01.02.Прикладное программирование
код и наименование

Группа 3-П-1, 3-П-2

Специальность 09.02.03 Программирование в компьютерных системах
код и наименование

Преподаватель Барилова С.В.

№	ФИО обучающихся	Подпись	Примечание
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			
25			
26			

Преподаватель _____ С.В. Барилова

Председатель УМО _____ О.А. Афиногенова