Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Северо-Кавказский федеральный университет»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

Методические указания к лабораторным работам

Часть 2 (лабораторные работы 13-24)

для направления подготовки:

09.03.02 «Информационные системы и технологии»,

Квалификация выпускника: бакалавр

Ставрополь, 2026

Н 63 **Объектно-ориентированное программирование:** Методические указания к лабораторным работам в двух частях (часть 2) для студентов направления 09.03.02 «Информационные системы и технологии» / Николаев Е.И. – Ставрополь: Изд-во СКФУ, 2026. – 190 с.

Методические указания к лабораторным работам по дисциплине «Объектно-ориентированное программирование» представляет собой цикл лабораторных работ, посвященный практическим аспектам проектирования на языке C#, разработки и отладки программного обеспечения с использованием инструментария Microsoft Visual Studio. Первая часть лабораторного практикума содержит 12 лабораторных работ. Цель учебного пособия: сформировать у студентов целостный взгляд на современные тенденции в области программной инженерии; обеспечить студентов обширным теоретическим материалом, достаточным для освоения методик разработки приложений; сформировать систему профессиональных компетенций.

УДК 004.41 (075.8)
ББК 22.18я73

Автор:

канд. техн. наук, доцент **Е.И. Николаев**

ОГЛАВЛЕНИЕ

Введение 4
Лабораторная работа 13. Простейшие элементы управления и события 5
Лабораторная работа 14. Меню и строка состояния в приложениях Windows Forms 18
Лабораторная работа 15. Диалоговые окна 46
Лабораторная работа 16. Обработка табличных данных 60
Лабораторная работа 17. Технология GDI+ в приложениях Windows Forms 78
Лабораторная работа 18. Приложения Windows Presentation Foundation 95
Лабораторная работа 19. Ресурсы, стили, триггеры 117
Лабораторная работа 20. Механизм привязки WPF 132
Лабораторная работа 21. Источники привязки произвольного типа 142
Лабораторная работа 22. Асинхронные делегаты 166
Лабораторная работа 23. Передача данных потокам 172
Лабораторная работа 24. Многомодульные приложения 176
Список литературы 188

ВВЕДЕНИЕ

Лабораторный практикум по дисциплине «Объектно-ориентированное программирование» представляет собой цикл лабораторных работ, посвященный практическим аспектам проектирования, разработки и отладки программного обеспечения с использованием инструментария Microsoft Visual Studio. Вторая часть учебного пособия включает 12 лабораторных работ.

Сложность освоения программирования для Windows, в совокупности с постоянным ростом популярности данной операционной системы, указывает на необходимость подготовки специалистов в данной области. Увеличение количества технологий в сфере создания Windows приложений указывает на необходимость выбора отдельного инструментария, поддерживаемого непосредственно производителем операционной системы Windows. В качестве такого инструментария в данных методических указаниях используется IDE MS Visual Studio.

Изучаемые технологии позволяют создавать кроссплатформенные приложения для Windows и Linux, соответственно изучаемые особенности языка C# могут применяться при разработке приложений на основе .NET Framework или .NET Core.

Основное внимание в первой части пособия уделено вопросам, связанным с синтаксисом языка C#; стандартной библиотеке C#; массивам; основами объектно-ориентированного программирования; файловому вводу-выводу.

ЛАБОРАТОРНАЯ РАБОТА 13. ПРОСТЕЙШИЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ И СОБЫТИЯ

13.1. Цель и содержание

Цель лабораторной работы: научиться перенаправлять стандартные потоки ввода-вывода.

Задачи лабораторной работы:

– научиться сохранять состояние потоков ввода-вывода;

– научиться объявлять новые потоки, ассоциированные с файлами;

– научиться решать задачи с вводом-выводом в файлы;

– научиться восстанавливать состояние потоков.

13.2. Теоретическая часть

Для выполнения лабораторной работы необходимо изучить теоретические основы проектирования с использованием стандартных элементов управления библиотеки .NET Framework.

Класс. Button. Класс Button представляет простую командную кнопку и наследуется от класса ButtonBase. Наиболее часто приходится писать код для обработки события Click кнопки. В следующем фрагменте кода реализован обработчик события Click (обработка данного события не представляет сложностей). С помощью метода PerformClick можно эмулировать событие Click на кнопке без реального щелчка на ней пользователем, что может пригодиться для тестирования построенного пользовательского интерфейса. В окне также имеется понятие кнопки по умолчанию, на которой автоматически совершается щелчок, если пользователь нажимает в этом окне клавишу «Enter». Чтобы идентифицировать кнопку как кнопку по умолчанию, в форме, содержащей эту кнопку, необходимо установить свойство AcceptButton в объект кнопки.

Класс CheckBox (рис. 13.1). Элемент управления CheckBox также унаследован от ButtonBase и используется для приема двух или трех состояний от пользователя.

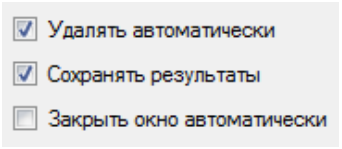


Рисунок 13.1 – Внешний вид CheckBox

Если установить свойство ThreeState в true, то свойство CheckState элемента CheckBox сможет принимать одно их трех значений перечисления CheckState, описанных в табл. 13.1.

Таблица 13.1 – Состоятия CheckBox (возможные значения перечисления CheckState)

Значение	Описание
Checked	Флажок имеет отметку.
Unchecked	Флажок не имеет отметки.
Indeterminate	В этом состоянии флажок становится серым (неопределенное состояние).

Состояние Indeterminate удобно, когда пользователь должен быть уведомлен, что опция не установлена. Если нужно булевское значение, можно проверить свойство Checked.

События CheckedChanged и CheckStateChanged возникают, когда изменяются значения свойств CheckState и Checked. Для флажка с тремя состояниями понадобится присоединиться к событию CheckStateChanged. Перехват этих событий может быть полезен для установки других значений, основанных на новом состоянии CheckBox.

Класс RadioButton (рис. 13.2). Переключатель (кнопки опций) – элемент управления, унаследованный от ButtonBase.

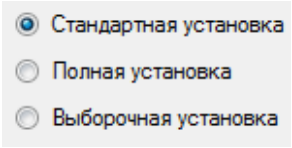


Рисунок 13.2 – Внешний вид RadioButton

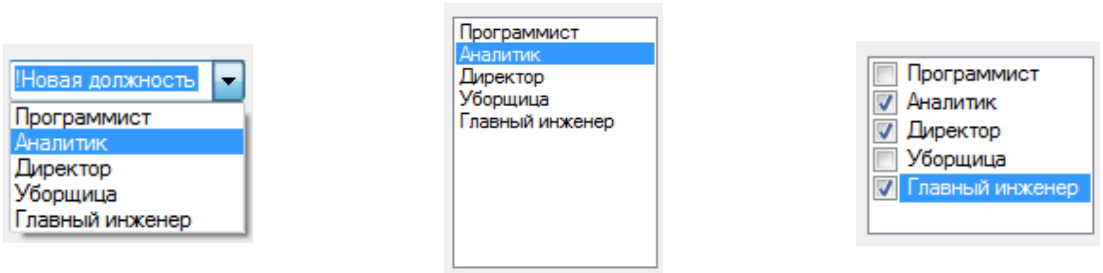
Переключатели обычно используются в группе. Переключатели позволяют пользователю выбирать одну из нескольких опций. При наличии нескольких элементов управления RadioButton в одном контейнере, только один из них может быть выбран в один и тот же момент времени.

Свойство Appearance принимает значение перечисления Appearance, которым может быть Button либо Normal. В случае установки значения Normal переключатель выглядит как маленький кружочек с меткой рядом с ним. Выбор переключателя заполняет этот кружок, а выбор другого переключателя снимает отметку с текущего выбранного переключателя и кружок делается пустым. В случае установки значения Button элемент управления выглядит подобно стандартной кнопке, но работает как переключатель – его выбор означает включение (вдавливание кнопки), а отказ от выбора – выключение (стандартное положение кнопки).

Свойство CheckedAlign определяет местоположение кружка по отношению к тексту метки. Он может быть над меткой, с любой стороны от нее либо под меткой.

Событие CheckedChanged инициируется при любом изменении свойства Checked. Это позволяет предпринимать другие действия на основе нового значения элемента управления.

Классы ComboBox, ListBox и CheckedListBox (рис. 13.3). Элементы управления ComboBox, ListBox и CheckedListBox унаследованы от класса ListControl. Этот класс предоставляет некоторую базовую функциональность управления списками. Наиболее важные аспекты использования списочных элементов управления состоят в добавлении данных и выборе данных из списка.



а) б) в)

Рисунок 13.3 – Внешний вид списков: а – ComboBox; б – ListBox; в – CheckedListBox

На выбор списка влияет то, как он должен использоваться, и тип данных, которые будут помещены в список. Если есть необходимость во множественном выборе, или если пользователю нужно видеть несколько элементов в списке в одно и то же время, то для этого лучше всего подойдут ListBox и CheckedListBox. Если в любой момент времени в списке может быть выбран только один элемент, то предпочесть следует ComboBox.

Для добавления элементов в списки необходимо обратиться к коллекции ListBox.ObjectCollection, которая доступна через свойство Items списка.

Поскольку коллекция хранит объекты, в список может быть добавлен любой допустимый тип .NET. Чтобы идентифицировать элементы, необходимо установить два важных свойства. Первым свойством является DisplayMember. Его настройка сообщает ListControl, какое свойство объекта должно быть отображено в списке. Другое свойство – ValueMember. Оно представляет собой свойство объекта, которое должно возвращаться в качестве значения. Если в список были добавлены строки, для обоих свойств по умолчанию используются строковые значения.

После загрузки данных в список для их получения могут быть использованы свойства Selectedltem и Selectedlndex. Свойство Selectedltem возвращает объект, выбранный в данный момент. Если список предполагает множественный выбор, то должна применяться коллекция Selectedltems для получения выбранных элементов.

Если для наполнения списка применяется DataBinding, то свойство SelectedValue вернет значение свойства выбранного объекта, которое было установлено в свойстве ValueMember.

Элемент ComboBox – это комбинация поля редактирования и окна списка. Стиль ComboBox задается установкой в свойстве DropDownStyle значения перечисления DropDownStyle (таблица 13.2).

Таблица 13.2 – Стили ComboBox (возможные значения перечисления DropDownStyle)

Значение	Описание
DropDown	Текстовая часть комбинированного списка допускает редактирование, и пользователи могут вводить значение. Также они могут щелкать на кнопке со стрелочкой, чтобы отобразить раскрывающийся список элементов.
DropDownList	Текстовая часть не допускает редактирование. Пользователи должны выбирать значения из списка.
Simple	Подобно DropDown, но список является постоянно видимым.

Класс DateTimePicker (рис. 13.4). Элемент управления DateTimePicker позволяет выбирать значение даты или времени (или то и другое) во множестве разных форматов. Значение DateTime можно отобразить в любом стандартном формате времени и даты. Свойство Format принимает значение перечисления DateTimePickerFormat, которое устанавливает формат Long, Short, Time или Custom. Если свойство Format установлено в DateTimePickerFormat .Custom, с помощью свойства CustomFormat можно задать строку, представляющую формат.

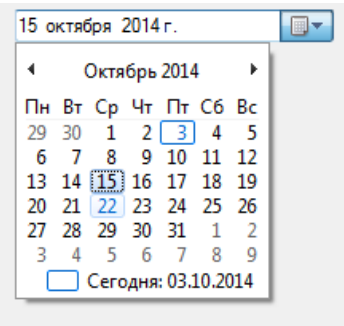


Рисунок 13.4 – Внешний вид DateTimePicker

Свойство Text возвращает строковое представление значения DateTime, а свойство Value – объект DateTime. С помощью свойств MinDate и MaxDate можно устанавливать максимальное и минимальное значения дат. Когда пользователь щелкает на стрелке вниз, отображается календарь, позволяющий выбрать дату. Доступны свойства, которые позволяют изменить внешний вид календаря, указать фоновые цвета заголовка и месяца, а также цвета переднего плана.

13.3. Методика и порядок выполнения работы

13.3.1. Учебная задача

Задание:

Необходимо разработать приложение для вычисления выражения:

$$Zoo = \begin{cases} 1 - X + Y^2 - X^3 + Y^4 - \ldots \\ W \cdot X - T \cdot Y + 7 \end{cases}$$

Решение:

1. Спроектируем экранную форму (рисунок 13.5). Студент может спроектировать форму с отличающимся дизайном и расположением элементов, но форма должна быть эстетически корректна.

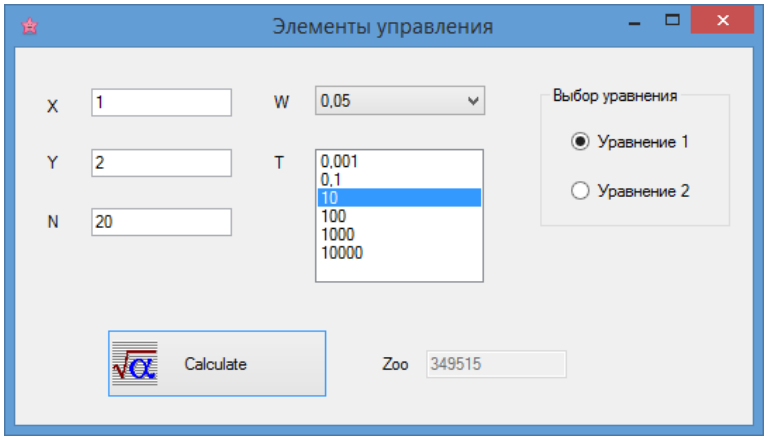


Рисунок 13.5 – Внешний вид экранной формы

2. Добавим обработчик события Click для кнопки расчета:

```
using System;
using System.Windows.Forms;

namespace LabWork1
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
            comboBoxW.SelectedIndex = 0;
            listBoxT.SelectedIndex = 0;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            double Zoo = 0;
            double X = Convert.ToDouble(textBoxX.Text);
            double Y = Convert.ToDouble(textBoxY.Text);
            int N = Convert.ToInt32(textBoxN.Text);
            double W = Convert.ToDouble(comboBoxW.SelectedItem.ToString());
            double T = Convert.ToDouble(listBoxT.SelectedItem.ToString());

           if (eq1.Checked)
           {
               Zoo = 0; double item = 0;
               for (int i = 0; i < N; i++)
               {
                   item = (i % 2 == 0) ? Y : X;
                   Zoo += Math.Pow(-1, i % 2) * Math.Pow(item, i);
               }
           }
           else
           {
               Zoo = W * X - T * Y + 7;
           }
           textBoxZoo.Text = Zoo.ToString();
        }
    }
}
```

Рисунок 13.6 – Код обработчика

3. Запустите приложение. Исправьте ошибки.

13.3.2. Индивидуальное задание

Для выполнения индивидуального задания необходимо разработать приложение для расчета значения выражения с выбором формулы расчета. Дополнительно, необходимо реализовать механизм перехвата исключений.

Вариант	Выражение для вычисления
1	$Z=\begin{cases}1-\frac{X}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{3+a\cdot j}{b\cdot i}.\end{cases}$
2	$Z=\begin{cases}-\frac{1}{2}+\frac{Y}{6}-\frac{X^2}{24}+\frac{Y^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j}{i^j}.\end{cases}$
3	$Z=\begin{cases}-\frac{1}{1}+\frac{Y}{2}-\frac{X^2}{3}+\frac{Y^3}{4}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{j^2+h\cdot i}{i^j+c\cdot j}.\end{cases}$
4	$Z=\begin{cases}-\frac{Y}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i+b\cdot j}{c\cdot i^j}.\end{cases}$
5	$Z=\begin{cases}-\frac{p^2}{2}+\frac{p^3}{6}-\frac{p^4}{24}+\frac{p^5}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b\cdot j}{c\cdot i^3}.\end{cases}$
6	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+a\cdot j^2}{i^3+j^3}.\end{cases}$

7	$Z=\begin{cases}-\frac{X}{2}+\frac{X^2}{6}-\frac{X^3}{24}+\frac{X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i^2}{i^3+b\cdot j^3}.\end{cases}$
8	$Z=\begin{cases}-\frac{1}{2}+\frac{X}{6}-\frac{X^2}{24}+\frac{X^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i}{i^3+j^3}.\end{cases}$
9	$Z=\begin{cases}-\frac{f}{2}+\frac{X\cdot f^2}{6}-\frac{X^2\cdot f^3}{24}+\frac{X^3\cdot f^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot(i+j)}{i\cdot(i+2)}.\end{cases}$
10	$Z=\begin{cases}1-\frac{X}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{3+j}{c\cdot i}.\end{cases}$
11	$Z=\begin{cases}-\frac{1}{2}+\frac{Y}{6}-\frac{X^2}{24}+\frac{Y^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j}{a^j}.\end{cases}$
12	$Z=\begin{cases}-\frac{1}{1}+\frac{Y}{2}-\frac{X^2}{3}+\frac{Y^3}{4}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{j^2+c\cdot i}{i^j+d\cdot j}.\end{cases}$
13	$Z=\begin{cases}-\frac{Y}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i+b\cdot j}{c\cdot i^j}.\end{cases}$
14	$Z=\begin{cases}-\frac{p^2}{2}+\frac{p^3}{6}-\frac{p^4}{24}+\frac{p^5}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b\cdot j}{c^i\cdot i^3}.\end{cases}$

15	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j^2}{i^3+j^3}.\end{cases}$
16	$Z=\begin{cases}-\frac{X}{2}+\frac{2\cdot X^2}{6}-\frac{3\cdot X^3}{24}+\frac{4\cdot X^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{(i+1)^2}{a\cdot i^3+b\cdot j^3}.\end{cases}$
17	$Z=\begin{cases}-\frac{1}{2}+\frac{X}{6}-\frac{X^2}{24}+\frac{X^3}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i}{i^3+b\cdot j^3}.\end{cases}$
18	$Z=\begin{cases}-\frac{f}{2}+\frac{X\cdot f^2}{6}-\frac{X^2\cdot f^3}{24}+\frac{X^3\cdot f^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot(i+j)}{b\cdot i\cdot(i+2)}.\end{cases}$
19	$Z=\begin{cases}-\frac{1}{1}+\frac{Y}{3}-\frac{X^2}{5}+\frac{Y^3}{7}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot j^2+2}{b\cdot i^j+3}.\end{cases}$
20	$Z=\begin{cases}-\frac{Y}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i+b\cdot j}{c\cdot i^j}.\end{cases}$
21	$Z=\begin{cases}-\frac{p^2}{1\cdot 2}+\frac{p^3}{2\cdot 6}-\frac{p^4}{3\cdot 24}+\frac{p^5}{4\cdot 120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+d\cdot j}{c^i\cdot i^3}.\end{cases}$
22	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j^2}{i^3}.\end{cases}$

23	$Z=\begin{cases}-\frac{3\cdot Y}{2}+\frac{5\cdot Y^2}{6}-\frac{7\cdot X^3}{24}+\frac{9\cdot Y^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i+k\cdot j}{c\cdot i^j}.\end{cases}$
24	$Z=\begin{cases}-\frac{3\cdot p^2}{2}+\frac{5\cdot p^3}{6}-\frac{7\cdot p^4}{24}+\frac{9\cdot p^5}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b}{c^i\cdot(i+1)^3}.\end{cases}$
25	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{(a\cdot i+j)^2}{i^3+b\cdot j^3}.\end{cases}$
26	$Z=\begin{cases}-\frac{1}{2}+\frac{Y}{6}-\frac{X^2}{24}+\frac{Y^3}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j}{i^j}.\end{cases}$
27	$Z=\begin{cases}-\frac{1}{1}+\frac{Y}{2}-\frac{X^2}{3}+\frac{Y^3}{4}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{j^2+h\cdot i}{i^j+c\cdot j}.\end{cases}$
28	$Z=\begin{cases}-\frac{Y}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i+b\cdot j}{c\cdot i^j}.\end{cases}$
29	$Z=\begin{cases}-\frac{p^2}{2}+\frac{p^3}{6}-\frac{p^4}{24}+\frac{p^5}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+b\cdot j}{c\cdot i^3}.\end{cases}$
30	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;.\\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+a\cdot j^2}{i^3+j^3}.\end{cases}$

31	$Z=\begin{cases}-\frac{X}{2}+\frac{X^2}{6}-\frac{X^3}{24}+\frac{X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i^2}{i^3+b\cdot j^3}.\end{cases}$
32	$Z=\begin{cases}-\frac{1}{2}+\frac{X}{6}-\frac{X^2}{24}+\frac{X^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot i}{i^3+j^3}.\end{cases}$
33	$Z=\begin{cases}-\frac{f}{2}+\frac{X\cdot f^2}{6}-\frac{X^2\cdot f^3}{24}+\frac{X^3\cdot f^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{a\cdot(i+j)}{i\cdot(i+2)}.\end{cases}$
34	$Z=\begin{cases}1-\frac{X}{2}+\frac{Y^2}{6}-\frac{X^3}{24}+\frac{Y^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{3+j}{c\cdot i}.\end{cases}$
35	$Z=\begin{cases}-\frac{1}{2}+\frac{Y}{6}-\frac{X^2}{24}+\frac{Y^3}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j}{a^j}.\end{cases}$
36	$Z=\begin{cases}-\frac{1}{1}+\frac{Y}{2}-\frac{X^2}{3}+\frac{Y^3}{4}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{j^2+c\cdot i}{i^j+d\cdot j}.\end{cases}$
37	$Z=\begin{cases}-\frac{X}{2}+\frac{p\cdot X^2}{6}-\frac{p^2\cdot X^3}{24}+\frac{p^3\cdot X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{i^2+j^2}{i^3+j^3}.\end{cases}$
38	$Z=\begin{cases}-\frac{X}{2}+\frac{2\cdot X^2}{6}-\frac{3\cdot X^3}{24}+\frac{4\cdot X^4}{120}-..;. \\ \sum_{i=1}^{N}\sum_{j=1}^{R}\frac{(i+1)^2}{a\cdot i^3+b\cdot j^3}.\end{cases}$

39	$Z = \begin{cases} -\frac{1}{2} + \frac{X}{6} - \frac{X^2}{24} + \frac{X^3}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot i}{i^3 + b \cdot j^3}. \end{cases}$
40	$Z = \begin{cases} -\frac{f}{2} + \frac{X \cdot f^2}{6} - \frac{X^2 \cdot f^3}{24} + \frac{X^3 \cdot f^4}{120} - ..;. \\ \sum_{i=1}^{N} \sum_{j=1}^{R} \frac{a \cdot (i+j)}{b \cdot i \cdot (i+2)}. \end{cases}$

13.4. Контрольные вопросы

1. Какое событие элемента управления Button обрабатывается в программах чаще всего?

2. Для чего предназначен компонент CheckBox? Назовите основные свойства класса CheckBox.

3. Опишите назначение элемента RadioButton. Какой внешний вид может принимать данный компонент? Назовите основные свойства класса RadioButton.

4. Назовите классы компонентов для представления списочной информации.

5. Опишите основные свойства класса ListBox. Чем комопнет ListBox отличается от CheckedListBox?

ЛАБОРАТОРНАЯ РАБОТА 14. МЕНЮ И СТРОКА СОСТОЯНИЯ В ПРИЛОЖЕНИЯХ WINDOWS FORMS

14.1. Цель и содержание

Цель лабораторной работы: научиться использовать в программах контекстные меню, главное меню приложения и строки состояния.

Задачами лабораторной работы являются:

– научится использовать контекстные меню для различных визуальных элементов управления,

– научиться использовать главное меню приложения,

– научиться координировать главное меню приложения с контекстными меню и другими элементами управления;

– научится создавать строки состояния с различными функциональными возможностями;

– получить практические навыки программирования обработчиков событий от элементов управления «панель инструментов» и «строка состояния».

14.2. Теоретическое обоснование

14.2.1. Меню в приложениях Windows Forms

В рамках платформы .NET элементом управления для создания системы меню является MenuStrip. Этот элемент управления позволяет создавать как пункты меню, представляющие собой любые подходящие элементы управления. Некоторые общие элементы интерфейса, которые могут содержаться в MenuStrip:

ToolStripMenuItem - традиционный пункт меню;

ToolStripComboBox - встроенный элемент ComboBox;

ToolStripSeparator - простая линия, разделяющая содержимое.

ToolStripTextBox - встроенный элемент TextBox.

MenuStrip содержит строго типизированную коллекцию ToolStripItemCollection. Подобно другим типам коллекции, этот объект поддерживает методы Add(), AddRange(), Remove() и свойство Count. Эта коллекция обычно заполняется не напрямую, а с помощью различных инструментов режима проектирования.

14.2.2. Строка состояния в приложениях Windows Forms

Поддержка элемента управления «строка состояния» реализована в .NET 2.0 использованием класса StatusStrip (также доступен класс StatusBar для проектирования строки состояния). Строка состояния StatusStrip может состоять из произвольного количества панелей, которые могут содержать текстовые и графические данные. Данные панели представлены типом ToolStripStatus. В отличие от StatusBar, элемент управления класса StatuStrip может содержать дополнительные элементы управления, такие как:

1) ToolStripProgressBar – встроенный индикатор выполнения (хода задания);

2) ToolStripDropDownButton – встоенная кнопка, отображающая при щелчке на ней список вариантов;

3) ToolStripSplitButton – отображает варианты списка только при щелчке непосредственно в области раскрывающегосясписка.

14.3. Методика и порядок выполнения работы

14.3.1. Учебная задача

1. Создайте приложение Windows Forms в среде MS Visual Studio.

2. Поместите элемент MenuStrip в главную форму в окне проектирования.

3. Создайте меню со следующей структурой:

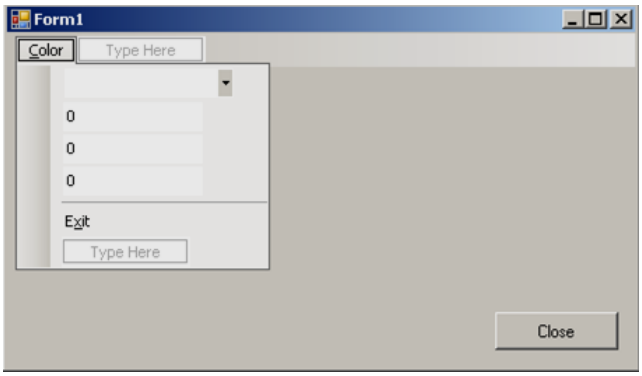


Рисунок 14.1 – Главное меню приложения.

3.1. Для этого свойстве Text пункта меню верхнего уровня установите равным «&Color».

3.2. Первым элементом меню выберите элемент типа ToolStripComboBox, для которого, в режиме проектирования, установите следующие свойства:

DropDownStyle = DropDownList

Items = список значений: белый, красный, черный, синий, желтый

ToolTipText = «Готовые цвета»

3.3. Далее следуют три элемента типа ToolStripTextBox, для каждого задается свойство Text = 0. Дополнительно, для toolStripTextBox1 свойство ToolTipText устанавливается равным «Красный», для toolStripTextBox2 – «Зеленый», toolStripTextBox3 – «Синий».

3.4. Следующий пункт меню типа ToolStripSeparator.

3.5. Последний пункт меню типа ToolStripMenuItem, для которого устанавливаются свойство Text = “E&xit”.

4. После создания меню, необходимо инициализировать начальные значения пункта ToolStripComboBox. Для этого в обработчике события Load главной формы добавьте код:

```
private void Form1_Load(object sender, EventArgs e)
{
    toolStripComboBox1.SelectedIndex = 0;
}
```

5. Для обработки выбора пункта Exit, добавьте в обработчик события Click данного пункта следующий код:

```
private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

6. Запустите приложение, в случае необходимости исправьте ошибки. Обратите внимание на работу пункта меню Exit.

Следующим этапом является обработка выбора пунктов меню:

Для обработки выбора цвета в первом пункте меню выполните следующие действия:

1. Создайте обработчик события SelectedIndexChanged:

```
private void toolStripComboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    switch (toolStripComboBox1.Text)
    {
        case "белый": ; BackColor = Color.White; break;
        case "красный": ; BackColor = Color.Red; break;
        case "черный": ; BackColor = Color.Black; break;
        case "синий": ; BackColor = Color.Blue; break;
        case "желтый": ; BackColor = Color.Yellow; break;

        default: BackColor = SystemColors.Control;  break;
    }
}
```

Рисунок 14.2 – Обработка выбора цвета в списке.

2. Запустите приложение, протестируйте обработчик.

Для задания RGB-компонент цвета с использованием пунктов меню типа ToolStripTextBox выполните следующие действия:

1. Для первого текстового поля в меню создайте обработчик события TextChanged:

```
private void toolStripTextBox1_TextChanged(object sender, EventArgs e)
{
    try
    {
        BackColor = Color.FromArgb(
            Convert.ToInt32(toolStripTextBox1.Text),
            Convert.ToInt32(toolStripTextBox2.Text),
            Convert.ToInt32(toolStripTextBox3.Text));
    }
    catch
    {
        MessageBox.Show("Необходимо ввести целое чило от 0 до 255", "Ошибка в задании цвета");
    }
```

Рисунок 14.3 – Обработка события изменения значения компоненты цвета.

2. Для второго и третьего полей установите обработчики событий TextChanged. В качестве функции-обработчика события для обоих оставшихся диалоговых окон выбрать из списка обработчик toolStripTextBox1_TextChanged первого окна (которое отвечает за ввод компоненты красного цвета).

Создайте кнопку Close, установите для нее обработчик события Click такой же как у пункта меню Exit.

По аналогии с созданием главного меню приложения создайте контекстное приложения, открывающееся при щелчке правой кнопкой мыши на окне приложения. Для этого:

1. Спроектируйте контекстное меню на основе элемента управления ContextMenuStrip.

2. Укажите значение свойства ContextMenuStrip главной формы равным ContextMenuStrip1 (необходимо указать имя созданного вами контекстного меню).

3. Подключите обработчики событий, созданные для главного меню к пунктам контекстного меню.

14.3.2. Индивидуальное задание

1. Создайте приложение Windows Forms в среде MS Visual Studio 2005.

2. Поместите элемент StatusStrip в главную форму в окне проектирования.

3. Поменяйте имя данного элемента управления на MainStatusStrip.

4. Для добавления панелей в строке состояния можно использовать следующие подходы:

– создавать необходимый программный код без использования мастеров;

– добавить нужные элементы в диалоговом окне, появляющемся при выборе ссылки Edit Items (Редактирование элементов) из меню контекстного редактора StatusStrip (рис. 14.4).

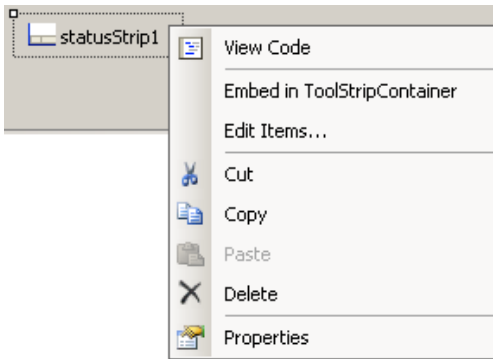


Рисунок 14.4 – Контекстное меню для выбора команды редактирования элементов строки состояния.

– добавить нужные элементы по одному с помощью раскрывающегося меню новых элементов StatusStrip (рис. 14.5).

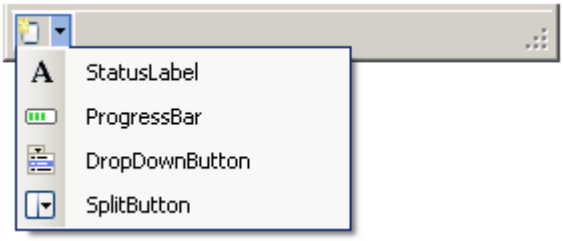


Рисунок 14.5 – Добавление элементов с помощью меню новых элементов.

5. Создайте с использованием вышеописанных методов строку состояния со следующей структурой:

– первый элемент типа ToolStripStatusLabel, для которого необходимо установить следующие свойства: Name = toolStripStatusLabelState, Spring = true, Text = (пусто), TextAlign = TopLeft.

– второй элемент типа ToolStripStatusLabel, для которого необходимо установить следующие свойства: Name = toolStripStatusLabelClock, BorderSides = All, Text = (пусто).

– третий элемент типа ToolStripDropDownButtonDateTime, для которого необходимо создать два элемента меню типа ToolStripMenuItem: первый элемент с Name = toolStripMenuItemDate, Text = «Текущая дата»; второй элемент с Name = toolStripMenuItemTime, Text = «Текущее время».

6. Создайте элемент управления типа Timer с именем timerDateTimeUpdate. Используя окно свойств установите значение свойства Interval в значение 1000 (значение в миллисекундах), а значение Enabled равным True.

7. Определите в проекте новый тип перечня DateTimeFormat, для выяснения того, что должен отображать второй элемент строки состояния – текущее время или текущую дату:

```
public enum DateTimeFormat
{
    ShowTime,
    ShowDate
}
```

8. В классе главного окна определите следующие поля и методы:

8.1. Поле dtFormat типа DateTimeFormat.

8.2. Поле currentCheckedItem типа ToolStripMenuItem.

8.3. Создайте обработчик события Tick для таймера timerDateTimeUpdate с именем timeDateTimeUpdate_Tick.

8.4. Выполните необходимые действия по присвоению начальных значений переменным dtFormat и currentCheckedItem.

8.5. Создайте обработчики события Click (выбор пункта меню) для элементов toolStripMenuItemTime и toolStripMenuItemDate.

После выполнения действий 8.1 – 8.5 класс главного окна будет содержать поля и методы, показанные на рисунке 14.6.

```
public partial class MainForm : Form
    {
        DateTimeFormat dtFormat = DateTimeFormat.ShowTime;
        ToolStripMenuItem currentCheckedItem;

        public MainForm()
        {
            InitializeComponent();
            Text = "Пример строки состояния";
            CenterToScreen();
            BackColor = Color.CadetBlue;
            currentCheckedItem = toolStripMenuItemTime;
            currentCheckedItem.Checked = true;
        }

        private void timerDateTimeUpdate_Tick(object sender, EventArgs e)
        {
            string info = "";
            if (dtFormat == DateTimeFormat.ShowTime)
                info = DateTime.Now.ToLongTimeString();
            else
                info = DateTime.Now.ToLongDateString();
            toolStripStatusLabelClock.Text = info;
        }

        private void toolStripMenuItemDate_Click(object sender, EventArgs e)
        {
            // установка даты
            currentCheckedItem.Checked = false;
            dtFormat = DateTimeFormat.ShowDate;
            currentCheckedItem = toolStripMenuItemDate;
            currentCheckedItem.Checked = true;
        }

        private void toolStripMenuItemTime_Click(object sender, EventArgs e)
        {
            // установка времени
            currentCheckedItem.Checked = false;
            dtFormat = DateTimeFormat.ShowTime;
            currentCheckedItem = toolStripMenuItemTime;
            currentCheckedItem.Checked = true;
        }
    }
```

Рисунок 14.6 – Поля и методы класса главного окна, определяемые программистом.

9. Исправьте ошибки и запустите приложение.

10. Самостоятельно выполните задание: добавьте код, показывающий в строке состояния (на панели toolStripStatusLabelState) координаты курсора мыши.

Выполните задание в соответствие с индивидуальным вариантом, дополнив главное и контекстное меню своими пунктами.

Варианты индивидуальных заданий

Вариант	Задание
1	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + b \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x + y$ или $z = x^2 + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
2	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a^2 \cdot x + (b - a) \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^3 + y}{y^2 + 1}$, $f = \sqrt{y^3}$ или $z = \sqrt{\frac{x}{y+1}} + y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на

	второй панели должно изменяться при смене положения курсора мыши.
3	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x^2 + b \cdot (y - x) + \lg(a \cdot z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\lvert x - y \rvert}$, $f = \sqrt{\lvert \sqrt{x} - \sqrt{y} \rvert}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
4	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{b} + b \cdot \frac{y}{x} + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^4$,

	$f = y^3$ или $z = \sqrt{\frac{x}{y}} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
5	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot z + b \cdot y + \sin(z)}{a + b}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \sqrt{\lvert x^3 - y^2 \rvert}$, $f = \frac{x^2}{y^3}$ или $z = \sin x$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
6	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot \frac{x}{y} + b \cdot \frac{y}{z} + \sin(\frac{z}{x})$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает

	пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^2 + y^3}{\lvert y^3 - x \rvert}$ или «квадраты координат курсора равны (x^2, y^2)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
7	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{b - y} + \frac{b \cdot y + \sin(z)}{a \cdot x}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x + y}{\lvert y - x \rvert}$ или «координаты курсора равны (x, y)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
8	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot \sqrt{x} + b \cdot y + \cos(x) \cdot \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна.

	Задание 2. Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y}{x^2}$, $f = \sqrt{\left\|\sqrt{x} - \sin y\right\|}$ или $z = \cos x^2 + \sin^2 \sqrt{y}$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
9	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\lg(a \cdot x) + \sin(z) - \cos(y)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\left\|x - y\right\|}$, $f = \sqrt{\left\|\sqrt{x} - \sqrt{y}\right\|}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
10	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + \frac{b \cdot y}{\lg(z)} + \frac{\sin(z)}{\cos(y)}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа

	ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z=\frac{x}{\lvert y-x^2\rvert}$, $f=\sqrt{\lvert x-\sqrt{y}\rvert}$ или $z=\cos x+\sin y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
11	**Задание 2.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a\cdot x}{\cos(z)}+\frac{b\cdot y+\sin(z)}{\sqrt{\lvert x-y\rvert}}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z=\frac{x^4}{y+1}$, $f=\sqrt{\lvert y^3+x\rvert}$ или $z=\sqrt{\lvert x-y\rvert}$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.

12	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{\lvert z \rvert} + b \cdot y + \frac{\sin(z)}{\sqrt{\lvert x^2 - y \rvert}}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^4$, $f = y^3$ или $z = \sqrt{\frac{x}{y}} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
13	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{y} + \frac{b \cdot y + \sin(z)}{\lvert x - y^2 \rvert}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^3 + y}{y^2 + 1}$, $f = \sqrt{y^3}$ или $z = \sqrt{\frac{x}{y+1}} + y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на

	второй панели должно изменяться при смене положения курсора мыши.
14	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\sqrt{\lvert a \cdot x + b \cdot y + \sin(z) \rvert}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^3$, $f = \sqrt{y}$ или $z = \sqrt{x} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
15	**Задание 1** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\sqrt{\left\lvert a \cdot x \cdot \sqrt{\left\lvert b \cdot y \cdot \sqrt{\lvert z \rvert} \right\rvert} \right\rvert}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \sqrt{x} + \sqrt{y}$ или $z = \lvert x^2 - y^2 \rvert$, где x и y – текущие координаты

	указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
16	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + b \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x + y$ или $z = x^2 + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
17	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a^2 \cdot x + (b - a) \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^3 + y}{y^2 + 1}$, $f = \sqrt{y^3}$ или $z = \sqrt{\frac{x}{y+1}} + y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
28	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x^2 + b \cdot (y - x) + \lg(a \cdot z)$, где x, y, z - неизвестные, вводимые в поля

	ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\lvert x - y \rvert}$, $f = \sqrt{\lvert \sqrt{x} - \sqrt{y} \rvert}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
19	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{b} + b \cdot \frac{y}{x} + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^4$, $f = y^3$ или $z = \sqrt{\frac{x}{y}} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
20	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot z + b \cdot y + \sin(z)}{a + b}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна.

	Задание 2. Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z=\sqrt{\lvert x^3-y^2\rvert}$, $f=\frac{x^2}{y^3}$ или $z=\sin x$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
21	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a\cdot\frac{x}{y}+b\cdot\frac{y}{z}+\sin(\frac{z}{x})$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z=\frac{x^2+y^3}{\lvert y^3-x\rvert}$ или «квадраты координат курсора равны (x^2, y^2)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
22	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a\cdot x}{b-y}+\frac{b\cdot y+\sin(z)}{a\cdot x}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения:

	$z = \frac{x+y}{\lvert y-x \rvert}$ или «координаты курсора равны (x , y)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
23	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot \sqrt{x} + b \cdot y + \cos(x) \cdot \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y}{x^2}$, $f = \sqrt{\lvert \sqrt{x} - \sin y \rvert}$ или $z = \cos x^2 + \sin^2 \sqrt{y}$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
24	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\lg(a \cdot x) + \sin(z) - \cos(y)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\lvert x - y \rvert}$, $f = \sqrt{\lvert \sqrt{x} - \sqrt{y} \rvert}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.

25	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + \frac{b \cdot y}{\lg(z)} + \frac{\sin(z)}{\cos(y)}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x}{\lvert y - x^2 \rvert}$, $f = \sqrt{\lvert x - \sqrt{y} \rvert}$ или $z = \cos x + \sin y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
26	**Задание 2.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{\cos(z)} + \frac{b \cdot y + \sin(z)}{\sqrt{\lvert x - y \rvert}}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^4}{y+1}$, $f = \sqrt{\lvert y^3 + x \rvert}$ или $z = \sqrt{\lvert x - y \rvert}$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.

27	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{\lvert z \rvert} + b \cdot y + \frac{\sin(z)}{\sqrt{\lvert x^2 - y \rvert}}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^4$, $f = y^3$ или $z = \sqrt{\frac{x}{y}} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
28	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{y} + \frac{b \cdot y + \sin(z)}{\lvert x - y^2 \rvert}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^3 + y}{y^2 + 1}$, $f = \sqrt{y^3}$ или $z = \sqrt{\frac{x}{y+1}} + y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
29	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\sqrt{\lvert a \cdot x + b \cdot y + \sin(z) \rvert}$,

	где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^3$, $f = \sqrt{y}$ или $z = \sqrt{x} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
30	**Задание 1** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\sqrt{\left\lvert a \cdot x \cdot \sqrt{\left\lvert b \cdot y \cdot \sqrt{\lvert z \rvert} \right\rvert} \right\rvert}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \sqrt{x} + \sqrt{y}$ или $z = \left\lvert x^2 - y^2 \right\rvert$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
31	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + b \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает

	пользователю выбрать один из вариантов расчета выражения: $z = x + y$ или $z = x^2 + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
32	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a^2 \cdot x + (b - a) \cdot y + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^3 + y}{y^2 + 1}$, $f = \sqrt{y^3}$ или $z = \sqrt{\frac{x}{y+1}} + y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
33	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x^2 + b \cdot (y - x) + \lg(a \cdot z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\lvert x - y \rvert}$, $f = \sqrt{\lvert \sqrt{x} - \sqrt{y} \rvert}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.

34	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{b} + b \cdot \frac{y}{x} + \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = x^4$, $f = y^3$ или $z = \sqrt{\frac{x}{y}} + y^2$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
35	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot z + b \cdot y + \sin(z)}{a + b}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \sqrt{\lvert x^3 - y^2 \rvert}$, $f = \frac{x^2}{y^3}$ или $z = \sin x$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
36	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot \frac{x}{y} + b \cdot \frac{y}{z} + \sin(\frac{z}{x})$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a,

	b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x^2 + y^3}{\lvert y^3 - x \rvert}$ или «квадраты координат курсора равны (x^2, y^2)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
37	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\frac{a \cdot x}{b - y} + \frac{b \cdot y + \sin(z)}{a \cdot x}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{x + y}{\lvert y - x \rvert}$ или «координаты курсора равны (x, y)», где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
38	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot \sqrt{x} + b \cdot y + \cos(x) \cdot \sin(z)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна.

	Задание 2. Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y}{x^2}$, $f = \sqrt{\left\|\sqrt{x} - \sin y\right\|}$ или $z = \cos x^2 + \sin^2 \sqrt{y}$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
39	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $\lg(a \cdot x) + \sin(z) - \cos(y)$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения: $z = \frac{y - \sqrt{x}}{\left\|x - y\right\|}$, $f = \sqrt{\left\|\sqrt{x} - \sqrt{y}\right\|}$ или $z = \cos x^2 + \sin^2 y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.
40	**Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения $a \cdot x + \frac{b \cdot y}{\lg(z)} + \frac{\sin(z)}{\cos(y)}$, где x, y, z - неизвестные, вводимые в поля ToolStripTextBox; a, b - константы, значения которых выбираются из пунктов типа ToolStripComboBox. Значение выражения выводится в заголовке главного окна. **Задание 2.** Создайте элемент «строка состояния», который содержит панели следующих типов: ToolStripDropDownButton (панель выбора) и ToolStripStatusLabel (отображения текста). Первая панель предлагает пользователю выбрать один из вариантов расчета выражения:

	$z = \frac{x}{\lvert y - x^2 \rvert}$, $f = \sqrt{\lvert x - \sqrt{y} \rvert}$ или $z = \cos x + \sin y$, где x и y – текущие координаты указателя мыши. Значение выбранного выражения на второй панели должно изменяться при смене положения курсора мыши.

14.4. Вопросы для защиты работы

1. Какие элементы управления используются для создания главного меню приложения?

2. Какие элементы управления используются для создания контекстного меню элемента управления?

3. Как в режиме разработки указать элементу управления его контекстное меню?

4. Какие типы данных используются для создания пунктов меню?

5. Поясните механизм синхронизации событий от разных элементов управления.

6. Поясните назначение конструкции try … catch …

7. Какие классы используются для создания строки состояния в .NET?

8. Какие классы используются для создания панелей строки состояния? Каково назначение каждого из этих классов?

9. Какой класс используется для работы с таймером? Какое событие элемента-таймера необходимо обрабатывать для реагирования на смену системного времени?

10. С помощью какого класса можно получить текущие дату и время? Какие методы содержит данный класс?

11. Что позволяет настраивать свойство Spring элементов-панелей строки состояния?

ЛАБОРАТОРНАЯ РАБОТА 15. ДИАЛОГОВЫЕ ОКНА

15.1. Цель и содержание

Цель лабораторной работы: Научиться создавать формы верхнего уровня и модальные диалоговые окна.

Задачами лабораторной работы являются:

– создание окон различного типа (модальные и нет) в ответ на события мыши,

– научиться реализовывать обмен данными между модальными диалоговыми окнами и главной формой;

– изучить принципы работы с диалоговым окном «Выбор цвета»;

– изучить принципы работы с диалоговым окном «Выбор шрифта»;

– изучить принципы работы с диалоговыми окнами, предназначенными для работы с каталогами;

– изучить принципы работы с диалоговыми окнами, предназначенными для работы с файлами.

15.2. Теоретическое обоснование

15.2.1 Пространство имен System.Windows.Forms

Основные компоненты пространства имен System.Windows.Forms:

1. System.Windows.Forms компонуется из различных классов, структур, делегатов, интерфейсов и перечней. Сотни типов пространства имен System.Windows.Forms можно объединить в следующие большие категорий.

2. Базовая инфраструктура. Это типы, представляющие базовые операции программы .NET Forms (Form, Application и т.д.), а также различные типы, обеспечивающие совместимость с разработанными ранее элементами управления ActiveX.

3. Элементы управления. Все типы, используемые для создания пользовательского интерфейса (Button, MenuStrip, ProgressBar, DataGridView и т.д.), являются производными базового класса Control, Элементы управления конфигурируются в режиме проектирования и оказываются видимыми (по умолчанию) во время выполнения.

4. Компоненты. Это типы, не являющиеся производными базового класса Control, но тоже предлагающие визуальные инструменты (ToolTip, ErrorProvider и т.д.) для программ .NET Forms, Многие компоненты (например, Timer) во время выполнения не видимы, но они могут конфигурироваться визуально в режиме проектирования.

5. Диалоговые окна общего вида. Среда Windows Forms предлагает целый ряд стандартных заготовок диалоговых окон для выполнения типичных действий (OpenFileDialog, PrintDialog и т.д.). Кроме того, вы можете создавать свои собственные пользовательские диалоговые окна, если стандартные диалоговые окна по какой-то причине вам не подойдут.

Общее число типов в System.Windows.Forms намного больше 100. В таблице 15.1 описаны наиболее важные из типов System.Windows.Forms, предлагаемых в .NET Framework.

Таблица 15.1 – Базовые типы пространства имен System.Windows.Forms

Классы	Описание
Application	Класс, инкапсулирующий средства поддержки Windows Forms, необходимые любому приложению
Button, CheckBox, ComboBox, DateTimePicker, ListBox, LinkLabel, MaskedTextBox, MonthCalendar, PictureBox, TreeView	Классы, которые (вместе со многими другими классами) определяют различные GUI- элементы
FlowLayoutPanel, TableLayoutPanel	Платформа .NET 2.0 предлагает целый набор администраторов оформления, выполняющих автоматическую корректировку размещения элементов управления в форме при изменении ее размеров
Form	Тип, представляющий главное окно, диалоговое окно или дочернее окно MDI в приложении Windows Forms
ColorDialog, OpenFileDialog, SaveFileDialog, FontDialog,	Представляют различные диалоговые окна, соответствующие стандартным операциям в рамках GUI

PrintPreviewDialog, FolderBrowserDialog	
Menu, MainMenu, Menultem, ContextMenu, MenuStrip, ContextMenuStrip	Типы, используемые для построения оконных и контекстно-зависимых систем меню. Новые (появившиеся в .NET 2.0) элементы управления MenuStrip и ContextMenuStrip позволяют строить меню, содержащие как традиционные пункты меню, так и другие элементы управления (окна текста, комбинированные окна и т.д.)
StatusBar, Splitter, ToolBar, ScrollBar, StatusStrip, ToolStrip	Типы, используемые для добавления в форму стандартных элементов управления

15.2.2. Создание окон

Все окна являются объектами типа Form из пространства имен System.Windows.Forms. Поэтому создание диалогового или модального окна отличается только свойствами, устанавливаемыми для данного окна в окне Properties среды разработки (свойство FormBorderStyle) и методом, которым форма выводится на экран:

Form2.Show() – для немодальной формы.

Form2.ShowDialog() – для модального диалогового окна.

В случае использования диалоговых окон, доступны различные варианты возвращения результатов функцией ShowDialog(). Результат имеет тип DialogResult.

15.2.3 Стандартные диалоговые окна библиотеки

При программировании графических приложений выделяют стандартные задачи, которые часто повторяются на практике: выбор объекта файловой системы, выбор шрифта, выбор цвета. Для решения подобных задач широко используются стандартные диалоговые окна библиотке .NET Framework.

На рис. 15.1 представлен фрагмен панели инструментов, содержащий невизуальные компоненты, которые могут быть размещены в лотке элементов управления формы в режиме проектирвоания.

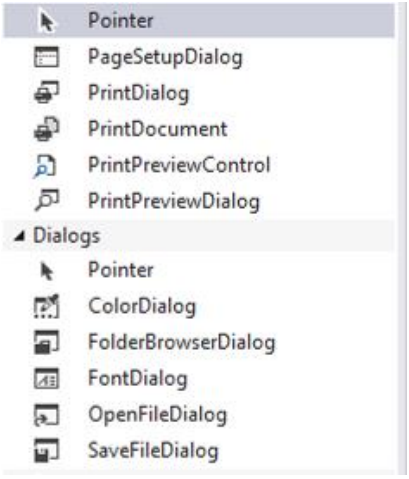


Рисунок 15.1 – Компоненты, обеспечивающие работу с встроенными диалоговыми окнами.

На рисунке показаны иконки диалогов, которые доступны программисту: PageSetupDialog, PrintDialog, PrintPreviewDialog, ColorDialog, FolderBrowserDialog, FontDialog, OpenFileDialog, SaveFileDialog.

После переноса иконки из панели инструментов на экранную форму в режиме проектирвания, в лотке компонентов появляется соответствующий компонент. Следует понимать, что подобные диалоги можно создавать и настраивать как в визуальном режиме (режим проектирования), так и динамически непосредственно в коде (режим выполнения).

15.3. Методика и порядок выполнения работы

15.3.1. Учебная задача

Для изучения принципов работы со стандартными диалоговыми окнами реализуем приложение, которое демонстрирует работу со встроенными диалоговыми окнами.

1. Переместите на форму приложения компоненты в соответствии с рис. 15.2.

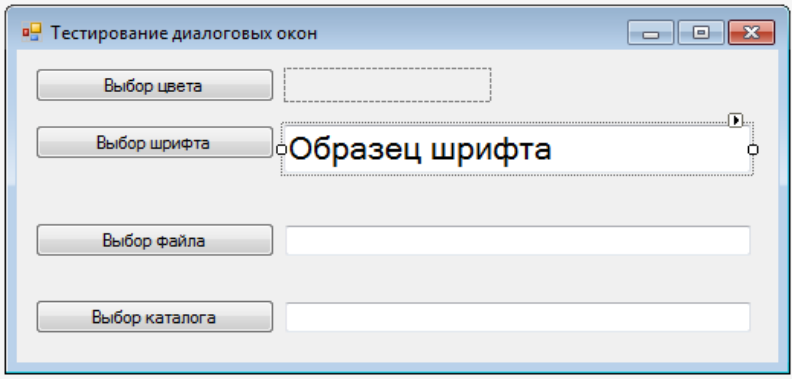


Рисунок 15.2 – Внешний вид экранной формы для работы со стандартными диалоговыми окнами

В таблице 15.3 перечислены компоненты формы и указаны некоторые из их свойств.

Таблица 15.3 – Компоненты формы.

Компонент	Описание
btnChooseColor	Тип Button. Создается для выбора открытия диалога выбора цвета. Свойство Text установлено в «Выбор цвета».
panelChooseColor	Тип Panel. Создается для демонстрации выбранного цвета.
btnChooseFont	Тип Button. Создается для открытия диалога выбора шрифта. Свойство Text установлено в «Выбор шрифта».
txtChooseFont	Тип TextBox. Создается для демонстрации внешнего вида выбранного шрифта.
btnChooseFile	Тип Button. Создается для открытия диалога выбора файла. Свойство Text установлено в «Выбор файла».
txtChoseFile	Тип TextBox. Отображает полный путь к выбранному файлу.
btnChooseFolder	Тип Button. Создается для открытия диалога выбора каталога. Свойство Text установлено в «Выбор каталога».
txtChooseFolder	Тип TextBox. Отображает полный путь к выбранному каталогу.

2. Создайте обработчики нажатия каждой кнопки.

На рис. 15.3 представлен листинг обработчика нажатия btnChooseColor.

```
private void btnChooseColor_Click(object sender, EventArgs e)
{
    // Создание диалогового окна
    ColorDialog dlg = new ColorDialog();
    // Открытие полной версии диалога
    dlg.FullOpen = true;
    // Разрешить пользователю получать справку
    dlg.ShowHelp = true;
    // Установка цвета панели в качестве начального цвета диалога
    dlg.Color = panelChooseColor.BackColor;

    // Получение результата выполнения диалога
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // Установка цвета панели
        panelChooseColor.BackColor = dlg.Color;
    }
}
```

Рисунок 15.3 – Обработчик btnChooseColor

На рис. 15.4 представлен обработчик выбора шрифта.

```
private void btnChooseFont_Click(object sender, EventArgs e)
{
    // Создание диалогового окна
    FontDialog dlg = new FontDialog();
    dlg.ShowColor = true;
    dlg.ShowHelp = true;
    dlg.Font = txtChooseFont.Font;

    // Получение результата выполнения диалога
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // Установка шрифта текстового поля и его цвета
        txtChooseFont.Font = dlg.Font;
        txtChooseFont.ForeColor = dlg.Color;
    }
}
```

Рисунок 15.4 – Обработчик btnChooseFont

На рис. 15.5 представлен код обработчика нажатия кнопки btnChooseFolder.

```
private void btnChooseFolder_Click(object sender, EventArgs e)
{
    // Создание диалогового окна
    FolderBrowserDialog dlg = new FolderBrowserDialog();
    dlg.Description = "выберите папку для демонстрации работы диалога";
    dlg.ShowNewFolderButton = true;
    dlg.SelectedPath = Application.StartupPath;

    // Получение результата выполнения диалога
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // Установка выбранного пути
        txtChooseFolder.Text = dlg.SelectedPath;
    }
}
```

Рисунок 15.5 – Обработчик btnChooseFolder

На рис. 15.6 показан диалог выбора файла.

```
private void btnChooseFile_Click(object sender, EventArgs e)
{
    // Создание диалогового окна
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.InitialDirectory = Application.StartupPath;
    dlg.Filter = "txt files (*.txt)|*.txt |" +
        "Мои файлы (расширения не придумал)|*.xxx|" +
        "Сборки (*.exe)|*.exe";
    dlg.FilterIndex = 3;
    dlg.Title = "Выбор моего файла";


    // Получение результата выполнения диалога
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // Установка выбранного пути к файлу
        txtChoseFile.Text = dlg.FileName;
    }
}
```

Рисунок 15.6 – Обработчик btnChooseFile

3. Реализуйте все обработчики, запустите приложение, убедитесь в его работоспособности. Измените различные настройки диалоговых окон.

15.3.2. Индивидуальное задание

Разработайте приложение для вычисления значения выражения. Кроме расчета значения выражения, реализуйте функционал, который предполагает использование стандартных диалоговых окон (не менее трёх различных типов).

Вариант	Задание
1	Вычислить значение выражения: $U = 1 - \frac{\cos^2(x \cdot t)}{2} + \frac{\sin^3(x \cdot z)}{3} - \frac{\cos^4(x \cdot t)}{4} + \frac{\sin^5(x \cdot z)}{5} - \ldots,$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
2	Вычислить значение выражения: $U = 1 - \frac{\cos^2(x)}{2!} + \frac{\sin^3(x)}{3!} - \frac{\cos^4(x)}{4!} + \frac{\sin^5(x)}{5!} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
3	Вычислить значение выражения: $y = \frac{\sin(x) \cdot x^2}{2!} - \frac{x^2 \cdot \sin(x^3)}{3!} + \frac{\sin(x^3) \cdot x^4}{4!} - \frac{x^4 \cdot \sin(x^5)}{5!} + \ldots,$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
4	Вычислить значение выражения: $h = \frac{x^2}{1 \cdot 3} - \frac{\sin(x^4)}{3 \cdot 5} + \frac{x^6}{5 \cdot 7} - \frac{\sin(x^8)}{7 \cdot 9} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
5	Вычислить значение выражения: $U = 1 - \frac{\sin^2(x) \cdot y}{2} + \frac{\sin^3(x) \cdot y^3}{3} - \frac{\sin^4(x) \cdot y^5}{4} + \frac{\sin^5(x) \cdot y^7}{5} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
6	Вычислить значение выражения: $Z = 1 - \frac{\sin(x^2)}{2} + \frac{\cos(x^3)}{3} - \frac{\sin(x^4)}{4} + \frac{\cos(x^5)}{5} - \ldots$, если количество слагаемых в правой части выражения и целое x пользователь вводит с клавиатуры.

7	Вычислить значение выражения: $U = 1 - \frac{\sin^2(x)}{2} + \frac{\sin^3(x)}{3} - \frac{\sin^4(x)}{4} + \frac{\sin^5(x)}{5} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
8	Вычислить значение выражения: $V = \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
9	Вычислить значение выражения: $s = -\frac{x^2}{2 \cdot 3} + \frac{x^3}{3 \cdot 4} - \frac{x^4}{4 \cdot 5} + \frac{x^5}{5 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
10	Вычислить значение выражения: $h = \frac{y^3 \cdot x^2}{1 \cdot 3} - \frac{y^5 \cdot x^4}{3 \cdot 5} + \frac{y^7 \cdot x^6}{5 \cdot 7} - \frac{y^9 \cdot x^8}{7 \cdot 9} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
11	Вычислить значение выражения: $p = -\frac{x}{1 \cdot 2 \cdot 3} + \frac{x^3}{3 \cdot 4 \cdot 5} - \frac{x^5}{5 \cdot 6 \cdot 7} + \frac{x^7}{7 \cdot 8 \cdot 9} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
12	Вычислить значение выражения: $T = -\frac{x}{1 \cdot 3} + \frac{x^3}{3 \cdot 5} - \frac{x^5}{5 \cdot 7} + \frac{x^7}{7 \cdot 9} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
13	Вычислить значение выражения: $j = -\frac{\sin^3(x)}{1 \cdot 3} + \frac{\sin^5(x^2)}{3 \cdot 5} - \frac{\sin^7(x^3)}{5 \cdot 7} + \frac{\sin^9(x^4)}{7 \cdot 9} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
14	Вычислить значение выражения: $Z = 1 - \frac{\sin(x^2)}{2} + \frac{\cos(x^3)}{3} - \frac{\sin(x^4)}{4} + \frac{\cos(x^5)}{5} - \ldots$, если количество слагаемых в правой части выражения и целое x пользователь вводит с клавиатуры.

15	Вычислить значение выражения: $s = -\frac{y \cdot x^2}{1 \cdot 3} + \frac{z \cdot x^3}{2 \cdot 4} - \frac{y \cdot x^4}{3 \cdot 5} + \frac{z \cdot x^5}{4 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
16	Вычислить значение выражения: $F = -\frac{x^2}{2 \cdot 3} + \frac{y^3}{3 \cdot 4} - \frac{x^4}{4 \cdot 5} + \frac{y^5}{5 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
17	Вычислить значение выражения: $A = -\frac{\sin(t) \cdot \lg(a)}{1!} + \frac{\ln(t^3) \cdot \cos(a^2)}{3!} - \frac{\sin(t^5) \cdot \lg(a^3)}{5!} + \frac{\ln(t^7) \cdot \cos(a^4)}{7!} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
18	Вычислить значение выражения: $E = -\frac{\sin(y) \cdot x^2}{2 \cdot 3} + \frac{\sin(x^3)}{3 \cdot 4} - \frac{\cos(y) \cdot x^4}{4 \cdot 5} + \frac{\sin(x^5)}{5 \cdot 6} - \frac{\sin(y) \cdot x^6}{4 \cdot 5} + \ldots,$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
19	Вычислить значение выражения: $p = -\frac{t \cdot \sin(a)}{1 \cdot 2} + \frac{t^3 \cdot \cos(a^2)}{3 \cdot 4} - \frac{t^5 \cdot \sin(a^3)}{5 \cdot 6} + \frac{t^7 \cdot \cos(a^4)}{7 \cdot 8} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
20	Вычислить значение выражения: $p = -\frac{y \cdot x}{1 \cdot 2 \cdot 3} + \frac{y^2 \cdot x^3}{3 \cdot 4 \cdot 5} - \frac{y^3 \cdot x^5}{5 \cdot 6 \cdot 7} + \frac{y^4 \cdot x^7}{7 \cdot 8 \cdot 9} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
21	Вычислить значение выражения: $U = 1 - \frac{\cos^2(x \cdot t)}{2} + \frac{\sin^3(x \cdot z)}{3} - \frac{\cos^4(x \cdot t)}{4} + \frac{\sin^5(x \cdot z)}{5} - \ldots,$ если

	количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
22	Вычислить значение выражения: $U = 1 - \frac{\cos^2(x)}{2!} + \frac{\sin^3(x)}{3!} - \frac{\cos^4(x)}{4!} + \frac{\sin^5(x)}{5!} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
23	Вычислить значение выражения: $y = \frac{\sin(x) \cdot x^2}{2!} - \frac{x^2 \cdot \sin(x^3)}{3!} + \frac{\sin(x^3) \cdot x^4}{4!} - \frac{x^4 \cdot \sin(x^5)}{5!} + \ldots,$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
24	Вычислить значение выражения: $h = \frac{x^2}{1 \cdot 3} - \frac{\sin(x^4)}{3 \cdot 5} + \frac{x^6}{5 \cdot 7} - \frac{\sin(x^8)}{7 \cdot 9} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
25	Вычислить значение выражения: $U = 1 - \frac{\sin^2(x) \cdot y}{2} + \frac{\sin^3(x) \cdot y^3}{3} - \frac{\sin^4(x) \cdot y^5}{4} + \frac{\sin^5(x) \cdot y^7}{5} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
26	Вычислить значение выражения: $Z = 1 - \frac{\sin(x^2)}{2} + \frac{\cos(x^3)}{3} - \frac{\sin(x^4)}{4} + \frac{\cos(x^5)}{5} - \ldots$, если количество слагаемых в правой части выражения и целое x пользователь вводит с клавиатуры.
27	Вычислить значение выражения: $U = 1 - \frac{\sin^2(x)}{2} + \frac{\sin^3(x)}{3} - \frac{\sin^4(x)}{4} + \frac{\sin^5(x)}{5} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.

28	Вычислить значение выражения: $V = \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
29	Вычислить значение выражения: $s = -\frac{x^2}{2 \cdot 3} + \frac{x^3}{3 \cdot 4} - \frac{x^4}{4 \cdot 5} + \frac{x^5}{5 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
30	Вычислить значение выражения: $h = \frac{y^3 \cdot x^2}{1 \cdot 3} - \frac{y^5 \cdot x^4}{3 \cdot 5} + \frac{y^7 \cdot x^6}{5 \cdot 7} - \frac{y^9 \cdot x^8}{7 \cdot 9} + \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
31	Вычислить значение выражения: $p = -\frac{x}{1 \cdot 2 \cdot 3} + \frac{x^3}{3 \cdot 4 \cdot 5} - \frac{x^5}{5 \cdot 6 \cdot 7} + \frac{x^7}{7 \cdot 8 \cdot 9} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
32	Вычислить значение выражения: $T = -\frac{x}{1 \cdot 3} + \frac{x^3}{3 \cdot 5} - \frac{x^5}{5 \cdot 7} + \frac{x^7}{7 \cdot 9} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
33	Вычислить значение выражения: $j = -\frac{\sin^3(x)}{1 \cdot 3} + \frac{\sin^5(x^2)}{3 \cdot 5} - \frac{\sin^7(x^3)}{5 \cdot 7} + \frac{\sin^9(x^4)}{7 \cdot 9} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
34	Вычислить значение выражения: $Z = 1 - \frac{\sin(x^2)}{2} + \frac{\cos(x^3)}{3} - \frac{\sin(x^4)}{4} + \frac{\cos(x^5)}{5} - \ldots$, если количество слагаемых в правой части выражения и целое x пользователь вводит с клавиатуры.
35	Вычислить значение выражения: $s = -\frac{y \cdot x^2}{1 \cdot 3} + \frac{z \cdot x^3}{2 \cdot 4} - \frac{y \cdot x^4}{3 \cdot 5} + \frac{z \cdot x^5}{4 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
36	Вычислить значение выражения: $F = -\frac{x^2}{2 \cdot 3} + \frac{y^3}{3 \cdot 4} - \frac{x^4}{4 \cdot 5} + \frac{y^5}{5 \cdot 6} - \ldots$, если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.

37	Вычислить значение выражения: $A = -\frac{\sin(t)\cdot\lg(a)}{1!} + \frac{\ln(t^3)\cdot\cos(a^2)}{3!} - \frac{\sin(t^5)\cdot\lg(a^3)}{5!} + \frac{\ln(t^7)\cdot\cos(a^4)}{7!} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
38	Вычислить значение выражения: $E = -\frac{\sin(y)\cdot x^2}{2\cdot 3} + \frac{\sin(x^3)}{3\cdot 4} - \frac{\cos(y)\cdot x^4}{4\cdot 5} + \frac{\sin(x^5)}{5\cdot 6} - \frac{\sin(y)\cdot x^6}{4\cdot 5} + \ldots,$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
39	Вычислить значение выражения: $p = -\frac{t\cdot\sin(a)}{1\cdot 2} + \frac{t^3\cdot\cos(a^2)}{3\cdot 4} - \frac{t^5\cdot\sin(a^3)}{5\cdot 6} + \frac{t^7\cdot\cos(a^4)}{7\cdot 8} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.
40	Вычислить значение выражения: $p = -\frac{y\cdot x}{1\cdot 2\cdot 3} + \frac{y^2\cdot x^3}{3\cdot 4\cdot 5} - \frac{y^3\cdot x^5}{5\cdot 6\cdot 7} + \frac{y^4\cdot x^7}{7\cdot 8\cdot 9} - \ldots$ если количество слагаемых в правой части выражения пользователь вводит с клавиатуры.

15.8. Вопросы для защиты работы

1. Как создаются модальные диалоговые окна?

2. Что такое пространство имен?

3. Какие пространства имен использованы в вашей программе? Какие типы данных из этих пространств использовались?

4. Опишите механизм обработки событий для визуальных элементов управления.

5. С каким модификатором доступности создаются элементы управления формы?

6. Как возвращается информация о нажатой кнопке закрытия модального диалогового окна? Какое свойство диалогового окна используется для этого? Значение какого типа возвращает метод ShowDialog?

7. Для чего предназначен класс OpenFileDialog? Назовите основные методы и свойства класса.

8. Для чего предназначен класс SaveFileDialog? Опишите основные методы и свойства данного класса.

9. Опишите принцип работы с классом ColorDialog. Опишите основные методы и свойства данного класса.

10. Опишите принципы работы с классом FontDialog. Опишите основные методы и свойства данного класса.

ЛАБОРАТОРНАЯ РАБОТА 16. ОБРАБОТКА ТАБЛИЧНЫХ ДАННЫХ

16.1. Цель и содержание

Цель лабораторной работы: научиться использовать механизмы файлового ввода-вывода при работе с данными.

Задачи лабораторной работы:

– научиться применять классы для работы с файлами;

– научиться использовать возможности ввода-вывода для решения практических задач;

– научиться обрабатывать данные в приложениях с файловым вводом-выводом.

16.2. Теоретическая часть

Повторите теоретический материал лабораторной работы и лекционный материал, посвященные файловому вводу-выводу. При выполнении данной лабораторной работы рекомендуется использовать символьный поток.

16.3. Методика и порядок выполнения работы

16.3.1. Учебная задача

Условие задачи. Разработать приложение, отражающее процессы поступления товара в магазине. Пользователь приложения формирует редактируемый список товарных групп («Продукты», «Бытовая химия», «Одежда», «Фрукты», «Полиграфия»). По факту поступления товара пользователь формирует запись со следующими полями: «Наименование товара», «Партия», «Товарная группа», «Цена» «Количество». Описание полей представлены в таблице 16.1. В

программе должен быть реализован автоматический расчет сумм по каждой партии и товарной группе.

Таблица 16.1 – Поля записи регистра поступления товаров

Поле записи	Описание	Тип данных
Наименование товара	Пользователь заполняет самостоятельно.	String
Партия	Документ (приходная накладная) в соответствии с которым принят товар. Пользователь заполняет самостоятельно.	String
Товарная группа	Пользователь может выбрать только из имеющегося справочника товарных групп.	String
Цена	Пользователь вводит самостоятельно.	float
Количество	Пользователь вводит самостоятельно.	float

Для выполнения лабораторной работы необходимо спроектировать многомодульное приложение, использующее файлы для ввода и вывода информации.

На рис. 16.1 представлен фрагмент текстового файла, содержащий исходные данные.

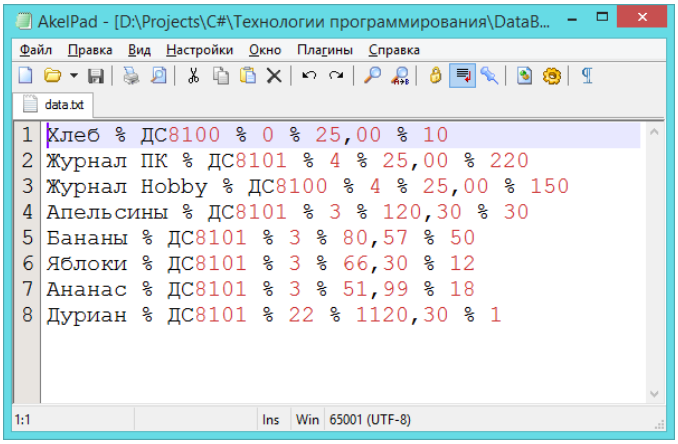


Рисунок 16.1 – Фрагмент исходного файла

Решение. Решение задачи будем реализовывать на базе приложения Windows Forms.

1. Создайте новый проект в среде MS VS типа Windows Forms с именем DataBase. В приложении модифицируйте экранную форму в соответствии с рисунком 16.2.

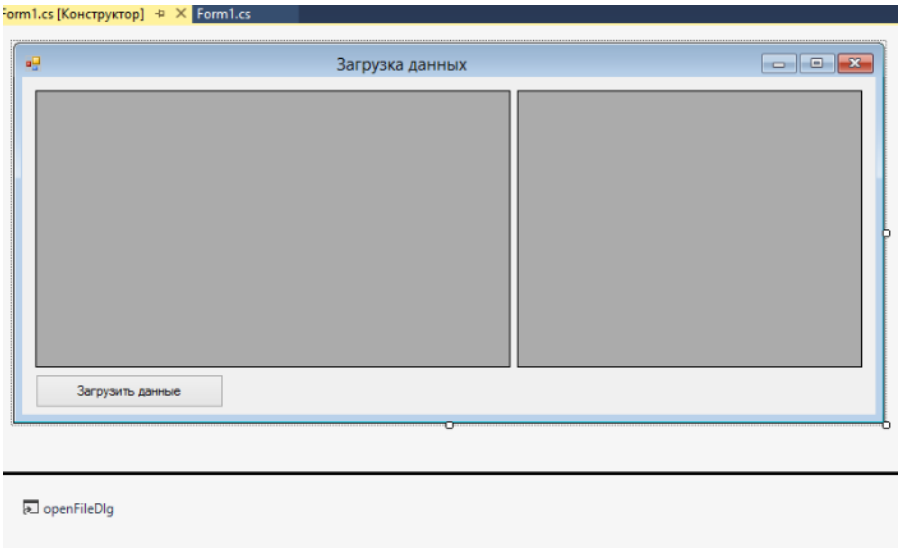


Рисунок 16.2 – Внешний вид экранной формы Form1

В таблице 16.2 представлено описание компонентов экранной формы.

Таблица 16.2 – Компоненты экранной формы Form1

Свойство Name объекта	Тип объекта	Описание
dgvRaw	DataGridView	Таблица для отображения данных из текстового файла. Данные отображаются в том виде, в котором они хранятся в текстовом файле – без дополнительных преобразований
dgvSummary	DataGridView	Таблица для отображения агрегированных данных: суммарная стоимость по каждой товарной категории
btnLoad	Button	Кнопка с текстом «Загрузить данные». Выполняет открытие диалогового окна для выбора файла с данными.
openFileDlg	OpenFileDialog	Невизуальный компонент. Представляет собой стандартное диалоговое окно для выбора файла

2. Выполним декомпозицию решаемой задачи и определим, какие классы понадобятся нам для выполнения поставленной задачи. В таблице 16.3 представлены имена добавляемых классов и их описание.

Таблица 16.3 – Классы проекта

Имя класса	Описание
RawDataItem	Класс для представления одной строки в файле с данными. Этот же класс будет являться источником данных для строки таблицы dgvRaw
SummaryDataItem	Класс для представления одной строки агрегированных данных. Этот же класс будет предоставлять данные для одной строки таблицы dgvSummary
Utils	Вспомогательный класс для отображения номера категории в название товарной категории (в файле хранятся не имена товарных категорий, а их коды), класс Utils будет преобразовывать номера категорий в их наименования
DataInterface	Это не класс, а интерфейс, который будет предоставлять экранной форме необходимые данные
DataStorage	Класс, который будет реализовывать интерфейс DataInterface

3. Самый простой класс – RawDataItem. На рис. 16.3 представлен листинг кода объявления класса RawDataItem для представления данных одной строки исходного файла.

```
7   namespace DataBase
8   {
9       class RawDataItem
10      {
11          public String Name { get; set; }
12          public int Group { get; set; }
13          public String Part { get; set; }
14          public float Price { get; set; }
15          public float Count { get; set; }
16          public float Sum
17          {
18              get { return Count * Price;  }
19          }
20      }
21  }
```

Рисунок 16.3 – Исходный код класса RawDataItem

4. Следующий простой класс – SummaryDataItem, листинг которого представлен на рис. 16.4.

```
7    namespace DataBase
8    {
9        class SummaryDataItem
10       {
11           public String GroupName { get; set; }
12           public float GroupSum { get; set; }
13       }
14   }
```

Рисунок 16.4 – Исходный код класса SummaryDataItem

5. Реализуйте класс Utils (рисунок 16.5).

```
9    class Utils
10   {
11       private static Dictionary<int, String> dict;
12       static Utils()
13       {
14           if(dict == null)
15           {
16               dict = new Dictionary<int, string>(5);
17               dict.Add(0, "Продукты");
18               dict.Add(1, "Бытовая химия");
19               dict.Add(2, "Одежда");
20               dict.Add(3, "Фрукты");
21               dict.Add(4, "Полиграфия");
22           }
23       }
24
25       public static String GetGroupByNumber(int number)
26       {
27           if (dict.ContainsKey(number))
28               return dict[number];
29           else
30               return "???";
31       }
32   }
```

Рисунок 16.5 – Исходный код класса Utils

Класс Utils содержит единственный статический метод GetGroupByNumber(int number),который возвращает наименование товарной категории по ее номеру. Пары «номер-наименование» хранятся в закрытом словаре, построение которого происходит в статическом конструкторе класса Utils.

6. Реализуем интерфейс DataInterface, который будет использоваться для получения данных в экранной форме (рис. 16.6).

```
7   namespace DataBase
8   {
9       interface DataInterface
10      {
11          List<RawDataItem> GetRawData();
12          List<SummaryDataItem> GetSummaryData();
13      }
14  }
```

Рисунок 16.6 – Исходный код интерфейсного типа DataInterface

Интерфейс содержит два метода:

1) GetRawData() – возвращает данные для таблицы dgvRaw;

2) GetSummaryData() – возвращает данные для таблицы dgvSummary.

7. Рассмотрим определение класса DataStorage (рис. 16.7).

```
10  class DataStorage : DataInterface
11  {
12      public bool IsReady...
20      private List<RawDataItem> rawdata;
21      private List<SummaryDataItem> sumdata;
22      private char devider = '%';
23      private DataStorage(){ }
24
25      private bool InitData(String datapath)...
59
60      private void BuildSummary()...
81
82      public static DataStorage DataCreator(String path)...
90
91      public List<RawDataItem> GetRawData()...
98
99      public List<SummaryDataItem> GetSummaryData()...
106 }
```

Рисунок 16.7 – Исходный код класса DataStorage

Описание полей и методов класса DataStorage представлено в таблице 16.4.

Таблица 16.4 – Поля и методы класса DataStorage

Имя поля/метода	Тип / тип возврата	Описание
rawData	List<RawDataItem>	Закрытое свойство для хранения списка элементов типа RawDataItem
sumData	List<SummaryDataItem>	Закрытое свойство для хранения списка элементов типа SummaryDataItem
devider	char	Поле для хранения символа-разделителя, которым отделяются поля в исходном файле
IsReady	bool	Возвращает true, если поле rawdata заполнено, в противном случае – false
DataStorage( )		Закрытый конструктор (пустой)
InitData(String datapath)	bool	Метод считывает данные из файла datapath и заполняет коллекцию rawData
BuildSummary()	void	Метод для расчета агрегирующих показателей по отдельным категориям: создает коллекцию sumData
DataCreator(String path)	DataStorage	Статический метод для возврата нового объекта типа DataStorage
GetRawData( )	List<RawDataItem>	Метод является реализацией интерфейса DataInterface. Возвращает данные для таблицы dgvRaw
GetSummaryData( )	List<SummaryDataItem>	Метод является реализацией интерфейса DataInterface. Возвращает данные для таблицы dgvSummary

Листинг класса DataStorage представлен на рис. 16.8.

```
10      class DataStorage : DataInterface
11      {
12          public bool IsReady
13          {
14              get
15              {
16                  if (rawdata == null) return false;
17                  else return true;
18              }
19          }

21          private List<RawDataItem> rawdata;
22          private List<SummaryDataItem> sumdata;
23          private char devider = '%';
24          private DataStorage(){ }
```

```
60          private void BuildSummary()
61          {
62              Dictionary<int, float> tmp = new Dictionary<int, float>();
63              foreach (var item in rawdata)
64              {
65                  if (tmp.ContainsKey(item.Group))
66                      tmp[item.Group] += item.Sum;
67                  else
68                      tmp[item.Group] = item.Sum;
69              }
70
71              sumdata = new List<SummaryDataItem>();
72              foreach (var item in tmp)
73              {
74                  sumdata.Add(new SummaryDataItem()
75                  {
76                      GroupName = Utils.GetGroupByNumber(item.Key),
77                      GroupSum = item.Value
78                  });
79              }
80          }
```

```
26          private bool InitData(String datapath)
27          {
28              rawdata = new List<RawDataItem>();
29
30              try
31              {
32                  StreamReader sr = new StreamReader(datapath, Encoding.UTF8);
33                  String line;
34                  while ((line = sr.ReadLine()) != null)
35                  {
36                      string[] items = line.Split(devider);
37                      var item = new RawDataItem()
38                      {
39                          Name = items[0].Trim(),
40                          Part = items[1].Trim(),
41                          Group = Convert.ToInt32(items[2].Trim()),
42                          Price = Convert.ToSingle(items[3].Trim()),
43                          Count = Convert.ToSingle(items[4].Trim())
44                      };
45                      rawdata.Add(item);
46
47                  }
48
49                  sr.Close();
50                  BuildSummary();
51
52              } catch(IOException ex)
53              {
54                  return false;
55              }
56
57              return true;
58          }
```

```
82          public static DataStorage DataCreator(String path)
83          {
84              DataStorage d = new DataStorage();
85              if (d.InitData(path))
86                  return d;
87              else
88                  return null;
89          }
90
91          public List<RawDataItem> GetRawData()
92          {
93              if (this.IsReady)
94                  return rawdata;
95              else
96                  return null;
97          }
98
99          public List<SummaryDataItem> GetSummaryData()
100         {
101             if (this.IsReady)
102                 return sumdata;
103             else
104                 return null;
105         }
```

Рисунок 16.8 – Листинг класса DataStorage

8. В классе Form1 добавьте обработчик события btnLoad_Click( ) и метод ShowData( ). Листинги методов представлены на рисунках 16.9 и 16.10.

```
22          private void btnLoad_Click(object sender, EventArgs e)
23          {
24              openFileDlg.InitialDirectory = Application.StartupPath;
25              if(openFileDlg.ShowDialog() == DialogResult.OK)
26              {
27                  ShowData(openFileDlg.FileName);
28              }
29          }
```

Рисунок 16.9 – Обработчик нажатия кнопки btnLoad

```
31          private void ShowData(String datapath)
32          {
33              try
34              {
35                  data = DataStorage.DataCreator(datapath);
36              }
37              catch (Exception ex)
38              {
39                  MessageBox.Show("При загрузке данных что-то сломалось");
40              }
41
42              dgvRaw.DataSource = data.GetRawData();
43              dgvRaw.ReadOnly = true;
44              dgvSummary.DataSource = data.GetSummaryData();
45              dgvSummary.ReadOnly = true;
46
47          }
```

Рисунок 16.10 – Метод ShowData класса Form1

9. После реализации всех классов, запустите приложение. Приложение выглядит как указано на рис. 16.11.

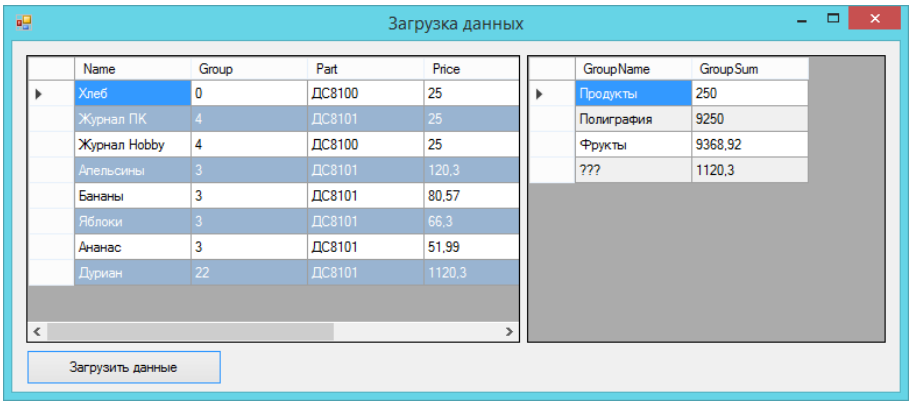


Рисунок 16.11 – Внешний вид окна приложения в режиме выполнения

16.3.2. Индивидуальное задание

Разработайте приложение для обработки табличных данных в соответствии с представленным фрагментом файла данных.

Вариант	Задание

1	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

2	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.

3	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.

4	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.

5	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.

6	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк): 1 Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1 2 Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7 3 ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19 4 ПО Office 365 % ПО % 11.00 % Склад №19 5 Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7 Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.
7	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк): 1 Дом*6*150*3500000 2 Квартира*1*49*790000 3 Квартира*2*59*1500000 4 Квартира*3*70*3000000 5 Комната*1*12*500000 Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.
8	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк): 1 Привет с того света | Детектив | 150.00 2 Кто подставил кролика | Детектив | 200.10 3 Почемучка | Детская литература | 60.22 4 Капитошка | Детская литература | 230.14 5 Звезда 112 | Фантастика | 12.90 Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.
9	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк): 1 Молоко \ Продукты \ 40.00 \ 1 2 Тряпка \ Хозтовары \ 10.00 \ 5 3 Порошок \ Хозтовары \ 750.00 \ 10 4 Мясо \ Продукты \ 350 \ 5 5 Швабра \ Хозтовары \ 900.00 \ 5 Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.
10	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк): 1 Рис 1 кг. / 120.11 / 54 / Магнит 2 Ручка шариковая / 23.00 / 15 / ТЦ Карандаш 3 Тетрадь 12 л. / 15.00 / 200 / ГеоКнига 4 Тетрадь 60 л. / 90.70 / 300 / ГеоКнига 5 Торт 1 кг. / 450 / 4 / Магнит Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.
11	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк): 1 Иванов С.Е.| доцент | ИСТ | 56.00 2 Петров А.А. | профессор | ИБАС | 100.23 3 Новиков П.С. | ст. преподаватель | ИСТ | 35.00 4 Ганотченко Е.П. | профессор | ИБАС | 200.45 5 Петров А.А. | профессор | ИБАС | 105.67 Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

12	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк): 1 ВАЗ-2107; Автоваз; 140000.57; 16000 2 Гога-1; Хуань Вей Трейд; 50.00; 1000 3 ВАЗ-5000; Автоваз; 1400000.00; 1000 4 X35; Hyundai; 750000; 1 Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.
13	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк): 1 Самолет*Москва*Пекин*30000.00 2 Автобус*Ставрополь*Москва*3000.00 3 Автобус*Невинномысск*Ставрополь*10.00 4 Автобус*Москва*Владивосток*50000 5 Самолет*СПб*Москва*1500 Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.
14	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк): 1 Иванов Е.А.*ИСТ-б-о-121*3*2 2 Петров А.А.*ИСТ-б-о-111*4*1 3 Кононов А.Е.*БАС-б-о-111*4*0 4 Катченко Е.А.*БАС-б-о-121*3*1 Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.
15	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк): 1 Молоко, 0.5 л % Продукты % 10.00 % 23.00 2 Хлеб белый % Продукты % 15.00 % 22.00 3 Молоко, 1.0 л % Продукты % 18.00 % 30.00 4 Туалетный утенок % Химия % 11.00 % 17.58 5 Фейри, 0.5 л % Химия % 17.00 % 88.00 Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.
16	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк): 1 Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1 2 Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7 3 ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19 4 ПО Office 365 % ПО % 11.00 % Склад №19 5 Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7 Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.
17	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк): 1 Дом*6*150*3500000 2 Квартира*1*49*790000 3 Квартира*2*59*1500000 4 Квартира*3*70*3000000 5 Комната*1*12*500000 Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».

18	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».

19	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.

20	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

21	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

22	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.

23	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.

24	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.

25	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.

26	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.

27	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.

28	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.

29	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.

30	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.

31	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

32	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.

33	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.

34	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.

35	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.

36	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.

37	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».

38	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».

39	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.

40	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

16.4. Контрольные вопросы

1. Какие классы для работы с файловой системой вы знаете?

2. Что такое класс потока? Перечислите классы потоков для работы с файлами?

3. Для чего используются интерфейсные типы? Приведите примеры.

4. Какие классы отвечают за представление файлов в программе?

5. Опишите последовательность действий при необходимости записать одну строку в файл. Приведите примеры использования различных классов.

6. Опишите принципы работы с байтовым потоком. Приведите пример кода для записи и считывания файла с использованием байтового потока.

7. Чем байтовый поток отличается от символьного?

ЛАБОРАТОРНАЯ РАБОТА 17. ТЕХНОЛОГИЯ GDI+ В ПРИЛОЖЕНИЯХ WINDOWS FORMS

17.1. Цель и содержание

Цель лабораторной работы: научиться использовать инструменты визуализации двумерной графики, предоставляемые пространством имен GDI+.

Задачами лабораторной работы являются:

– научится использовать основные типы, предоставляемые библиотекой GDI+;

– научиться использовать различные режимы вывода графики;

– освоить принципы построения графиков, диаграмм;

– научиться создавать интерактивные приложения;

– научиться обрабатывать события в графических приложениях;

– научиться использовать класс Timer;

– научиься реализовывать механизм анимации в приложениях Windows.

17.2. Теоретическое обоснование

17.2.1 Основы GDI+

В таблице 17.1 приводится описание базовых пространств имен GDI+.

Таблица 17.1 – Базовые пространства имен GDI+

Пространство имен	Описание
System.Drawing	Базовое пространство имен GDI+, определяющее множество типов для основных операций визуализации (шрифты, перья, кисти и т.д.), а также тип Graphics.
System.Drawing.Drawing2D	Содержит описание типов, используемых для более сложной двумерной/векторной графики (градиентные кисти, стили окончания линий, геометрические трансформации и т.д.)

Пространство имен	Описание
System.Drawing.Imaging	Содержит описание типов, обеспечивающих обработку графических изображений (изменение палитры, извлечение метаданных изображения, работа с метафайлами и т.д.)
System.Drawing.Printing	Содержит описание типов, обеспечивающих отображение графики на печатной странице, непосредственное взаимодействие с принтером.
System.Drawing.Text	Дает возможность управлять коллекциями шрифтов

При создании приложения Windows с использованием мастера, ссылка на System.Drawing.dll устанавливается автоматически.

Обзор пространства имен System.Drawing.

В таблице 17.2 приводится описание основных типов, необходимых при работе с графикой.

Таблица 17.2 – Базовые типы пространства имен System.Drawing

Тип	Описание
Bitmap	Тип, инкапсулирующий данные изображения
Brush, Brushes, SolidBrush, SystemBrushes, TextureBrush	Объекты Brush используются для заполнения внутренних областей графических форм (прямоугольников, эллипсов и т.д.)
BufferedGraphics	Новый тип .NET 2.0, обеспечивающий графический буфер для двойной буферизации, которая используется для уменьшения или полного исключения эффекта мелькания, возникающего при перерисовке
Color SystemColors	Определяют ряд статических свойств, доступных только для чтения и используемых для получения нужного цвета при использовании перьев и кистей
Font, FontFamily	Тип Font инкапсулирует характеристики данного шрифта (название, плотность, размер и т.д.). FontFamily предлагает абстракцию для группы шрифтов, имеющих аналогичный дизайн, но определенные вариации стиля

Graphics	Представляет реальную поверхность нанесения изображения, а также предлагает ряд методов для визуализации текста, изображений и геометрических шаблонов
Pen Pens SystemPens	Pens – это объекты, используемые для построения линий, кривых. Тип Pen определяет ряд статических свойств, возвращающих новый объект Pen заданного цвета.
Point, PointF	Структуры, представляющие точку
Rectangle RectangleF	Структуры, представляющие прямоугольник
Size SizeF	Структуры, представляющие размер (в виде целого или вещественного числа)

Утилитарные типы System.Drawing.

Многие из методов рисования, определенные объектом System.Drawing.Graphics, требуют указать позицию или область, в которой требуется отобразить данный элемент. Другие методы требуют указания размеров, координат, границ геометрического образа.

Тип Point(F) поддерживает ряд полезных членов:

- перегруженные операции: +, -, ==, != ;

- доступ к значениям (x,y): X, Y;

- поле IsEmpty возвращает значение true, если x и y установлены в 0.

Пример:

Типы **Rectangle(F)**, как и Point, необходимы в большинстве GUI-приложений. Одним из наиболее полезных методов типа Rectangle является Contains(), который позволяет выяснить, находится ли данный тип Point или Rectangle в рамках границ некоторого другого объекта.

Пример:

Класс Region представляет внутреннюю часть геометрической фигуры, поэтому конструкторы класса Region требуют, чтобы им на вход был предоставлен некоторый геометрический объект.

Например, имея внутреннюю часть фигуры, вы можете манипулировать ею с использованием различных членов.

Класс Graphics.

Класс System.Windows.Graphics – это интерфейс для функциональных возможностей GDI+. Этот класс не только предоставляет поверхность, на которой размещаются изображения (поверхность формы, поверхность элемента управления или область в памяти), но определяет также члены, которые позволяют отображать текст, изображения (пиктограммы, точечные рисунки и т.д.) и самые разные геометрические формы.

Класс Graphics не допускает непосредственного создания своего экземпляра с помощью ключевого слова new, поскольку класс не имеет открытых конструкторов.

Для получения доступа к объекту Graphics необходимо создатьобработчик события WM_PAINT, для этого для формы необходимо зарегистрировать обработчик события Paint. В качестве параметра метода-обработчика будет передан объект типа PaintEventArgs. В обработчике события Paint можно будет получить объект Graphics следующим образом:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics new_graf = e.Graphics;

    new_graf.DrawRectangle(10, 10, 100, 100);

}
```

Рисунок 17.1 – Получение объекта Graphics через параметр PaintEventArgs обработчика события Paint

17.3.2 Обработка события таймера

Для организации анимации в приложениях Windows часто используется класс Timer. Класс достаточно простой и после создания объекта класса программисту требуется только одно событие Tick, которое наступает когда указанный интервал таймера заканчивается. Интрвал устанавливается свойством Interval.

Существую еще два метода, необходимых для работы таймера: Start() – запускает таймер; Stop() – останавливает выполнение таймера.

17.3. Методика и порядок выполнения работы

17.3.1. Рисование графика функции

Для овладения практическими навыками визуализации результатов расчетов реализуйте пример: создайте приложение для отображения графика по заданной функции: $F = a \cdot x^{-p} \cdot \sin(\mathrm{k} \cdot \mathrm{x} + \mathrm{b})$.

Для решения поставленной задачи реализуйте следующие действия:

1. Создайте приложение Windows Forms.

2. Спроектируйте форму в соответствии с представленным шаблоном (рис. 10.2).

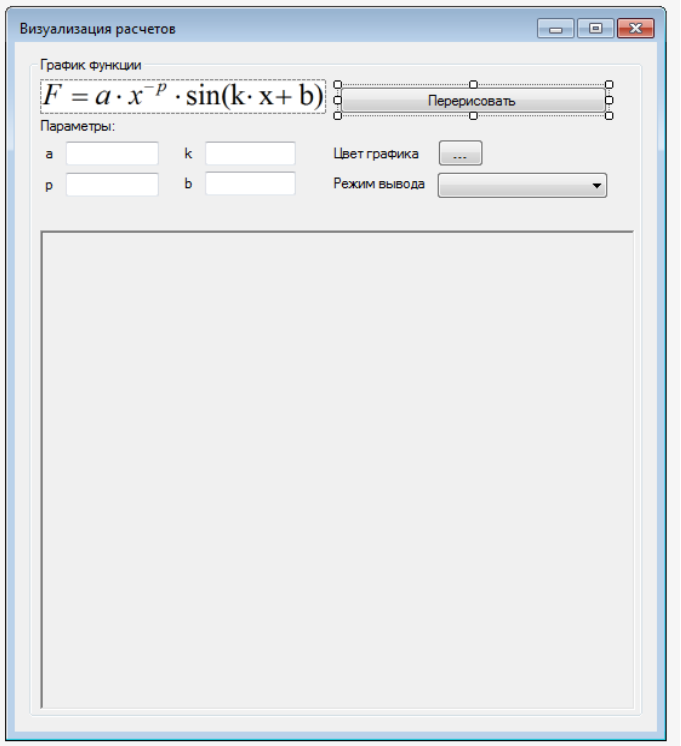


Рисунок 10.2 – Проект формы приложения

В таблице 10.1 приведена дополнительная информация о компонентах, размещенных на форме приложения.

Таблица 10.1 – Элементы управления формы главного окна

Имя элемента управления	Описание

txtA	Элемент управления типа TextBox. Предназначен для ввода коэффициента функции a.
txtP	Элемент управления типа TextBox. Предназначен для ввода коэффициента функции p.
txtK	Элемент управления типа TextBox. Предназначен для ввода коэффициента функции k.
txtB	Элемент управления типа TextBox. Предназначен для ввода коэффициента функции b.
btnChooseColor	Элемент управления типа Button. Предназначен для открытия диалога выбора цвета графика.
comboMode	Элемент управления типа ComboBox. Предназначен для выбора режима отображения графика: точками или в виде линии. Заполните свойство Items и добавьте два элемента: «Точки», «Линия».
panelGraf	Элемент управления типа Panel. Предназначен для отображения графика функции. Свойство Height установлено в 400.
btnRedraw	Элемент управления типа Button. Предназначен для перерисовки графика.

3. В теле класса главной формы приложения определите следующие свойства (рис. 10.3).

```
public partial class Form1 : Form
{
    // Цвет графика
    Color color = Color.Red;
    // Режим отображения
    int mode = 1;
    // Шаг расчета функции
    double dx = 5;
    // Интервал изменения x
    double x0 = 0.0, xn = 400.00;
    // Коэффициенты функции
    double a = 5, b = 100, p = -0.5, k = 2;
```

Рисунок 10.3 – Добавление полей класса

4. В конструктре экранной формы установите значения элементов управления в соответствии со значениями служебных переменных (рис. 10.4).

```
public Form1()
{
    InitializeComponent();

    // Инициализация компонентов формы значениями переменных
    // Выбор режима
    comboMode.SelectedIndex = mode;
    // Установка коэффициентов
    txtA.Text = a.ToString();
    txtB.Text = b.ToString();
    txtK.Text = k.ToString();
    txtP.Text = p.ToString();
}
```

Рисунок 10.4 – Конструктор класса

5. Реализуйте метод вычисления значений функции в соответствии с условием задачи (рис. 10.5).

```
private double MyFunc(double x)
{
    if (x == 0)
        return 0;
    else
        return a * Math.Pow(x, -p) * Math.Sin(k * x + b);
}
```

Рисунок 10.5 – Метод для вычисления значений функции

6. Реализуйте обработчики: нажатия кнопки «Перерисовать» (рис. 10.6), книпки выбора цвета графика (рис. 10.7) и выбора режима рисования графика (рис. 10.8).

```
private void btnRedraw_Click(object sender, EventArgs e)
{
    a = Convert.ToDouble(txtA.Text);
    b = Convert.ToDouble(txtB.Text);
    k = Convert.ToDouble(txtK.Text);
    p = Convert.ToDouble(txtP.Text);
    panelGraf.Invalidate();
}
```

Рисунок 10.6 – Обработчик нажатия кнопки btnRedraw

```
private void btnChooseColor_Click(object sender, EventArgs e)
{
    ColorDialog dlg = new ColorDialog();
    dlg.Color = color;
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // еустанавливаем новый цвет
        color = dlg.Color;
        // перерисовываем панель с графиком
        panelGraf.Invalidate();
    }
}
```

Рисунок 10.7 – Обработчик нажатия кнопки btnChooseColor

```
private void comboMode_SelectedIndexChanged(object sender, EventArgs e)
{
    // Устанавливаем режим рисования графика
    mode = ((ComboBox)sender).SelectedIndex;
    // Перерисовываем панель
    panelGraf.Invalidate();
}
```

Рисунок 10.8 – Обработчик выбора пункта в comboMode

Внешний вид экранной формы разработанного приложения представлен на рис. 10.9

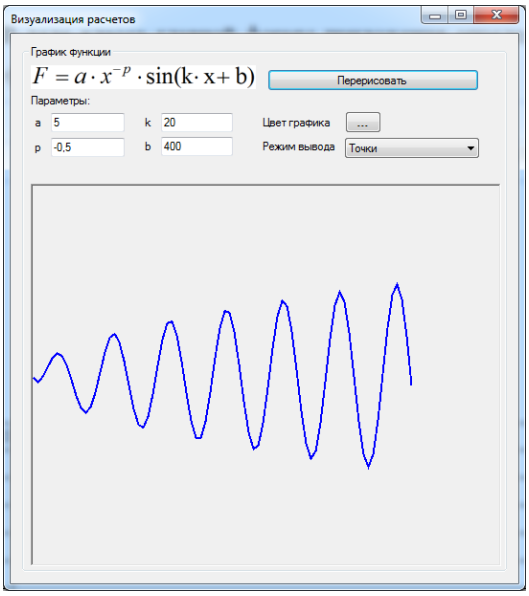


Рисунок 10.9 – Приложение в режиме выполнения

7. Запустите приложение. Протестируйте работу программы при различных значениях параметров.

8. Доработайте приложение: реализуйте механизм вывода графика отдельными точками, так как в представленном варианте график выводится в виде отрезков, соединяющих отдельные точки графика.

9. Разработайте приложение, в котором визуализируются данные сложной функции (учащийся может определить график функции самостоятельно).

17.3.2 Анимация средствами GDI+

1. Создайте приложение Windows Forms.

2. Дополните класс экранной формы вспомогательными переменными (рис. 17.10).

```
public partial class Form2 : Form
{
    private Point pos = new Point(100, 100);
    private int R = 50;
    double T = 0;
```

Рисунок 17.10 – Переменные класса формы

3. Перетащите из панели инструментов компонент Timer на экранную форму. Присвойте ему имя MainTimer. Установите свойство Interval в значение 500.

4. Модифицируйте обработчик события Paint экранной формы (рис. 17.11).

```
private void Form2_Paint(object sender, PaintEventArgs e)
{
    Graphics gr = e.Graphics;
    gr.DrawEllipse(new Pen(Color.Blue, 10), pos.X - R, pos.Y - R, R, R);
}
```

Рисунок 17.11 – Обработчик события Paint

5. Реализуйте обработчик события Tick таймера MainTimer (рис. 17.12).

```
private void MainTimer_Tick(object sender, EventArgs e)
{
    T += 0.1;
    pos.X = (int)(300 + 150 * Math.Sin(T));
    pos.Y = (int)(300 + 100 * Math.Cos(T));
    Invalidate();

}
```

Рисунок 17.12 – Обработчик события Tick

6. Для запуска и остановки таймера добавьте вызов метода Start() в конструктор класса, и вызов метода Stop() в обработчик события FormClosing главной формы (рис. 17.13).

```
public Form2()
{
    InitializeComponent();
    MainTimer.Start();
}
private void Form2_FormClosing(object sender, FormClosingEventArgs e)
{
    MainTimer.Stop();
}
```

Рисунок 17.13 – Запуск и остановка таймера

17.3.3 Интерактивная двумерная графика

Выполните и разберите технологию решения учебного задания: разработайте приложение, которое позволяет пользователю взаимодействовать с GDI-объектами. Необходимо вывести на форму простую фигуру и сделать доступным ее перетаскивание.

Для выполнения задания необходимо выполнить следующие действия:

1. Создайте приложение Windows Forms.
2. Добавить в класс формы служебные переменные (рис. 17.14).
3. Модифицируйте метод обработки события перерисовки окна (рис. 17.15).

```
public partial class Form1 : Form
{
    private Point location = new Point(50,50);
    private int R = 50;
    Boolean DropStarted = false;
```

Рисунок 17.14 – Служебные переменные класса

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    // Рисуем круг в точке location
    Graphics gr = e.Graphics;
    gr.DrawEllipse(new Pen(Color.Red, 5), location.X - R, location.Y - R, R, R);
}
```

Рисунок 17.15 – Обработчик Paint

4. Реализуйте обработчик события MouseDown для формы приложения (рис. 17.16).

```
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    int curX = e.X, curY = e.Y;
    Boolean InX = (curX >= location.X - R && curX <= location.X + R);
    Boolean InY = (curY >= location.Y - R && curY <= location.Y + R);

    if (InX && InY)
        DropStarted = true;
}
```

Рисунок 17.16 – Обработчик MouseDown

5. Реализуйте обработчик события MouseUp для формы приложения (рис. 17.17).

```
private void Form1_MouseUp(object sender, MouseEventArgs e)
{
    if (DropStarted)
    {
        location.X = e.X + R/2;
        location.Y = e.Y + R/2;
        Invalidate();
    }
}
```

Рисунок 17.17 – Обработчик MouseUp

6. Запустите приложение, протестируйте механизм перетаскивания объектов.

17.3.4. Индивидуальное задание

1. Создайте приложение Windows Forms в среде Visual Studio.

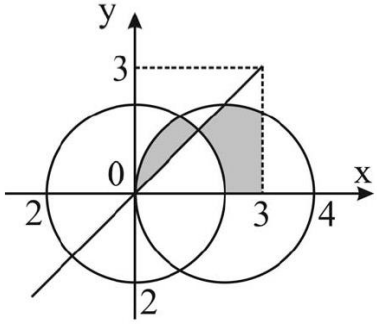
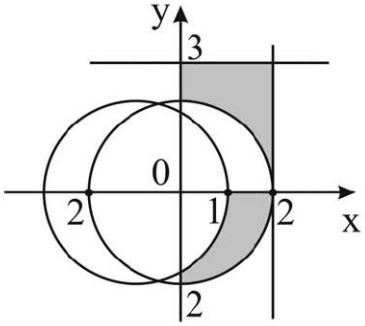
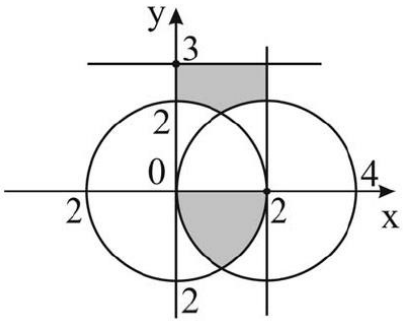
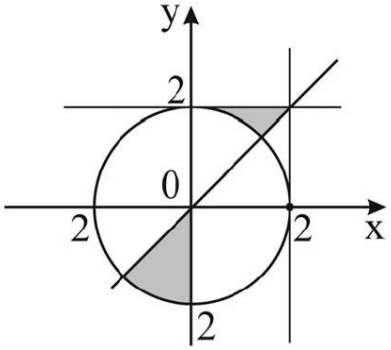
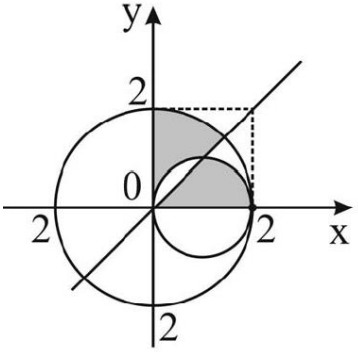
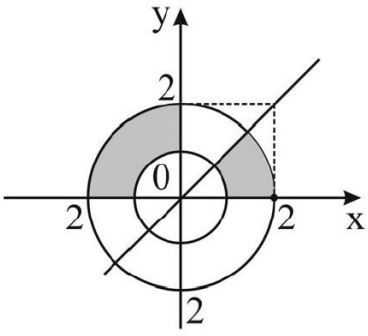
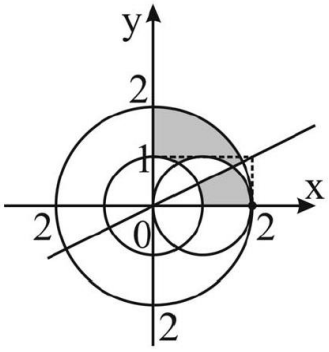
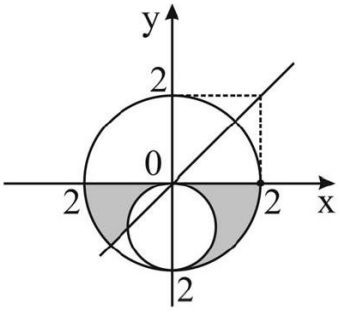
2. Выведите на поверхность формы рисунок, в соответствие с индивидуальным заданием.

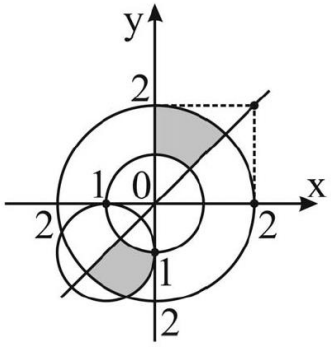
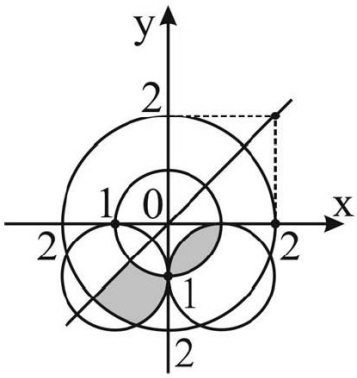
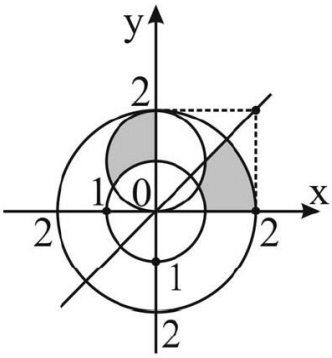
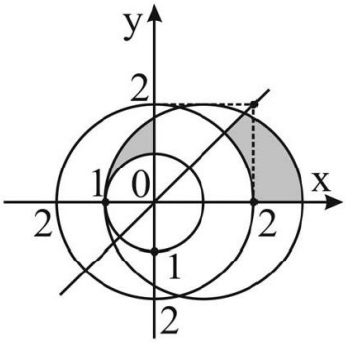
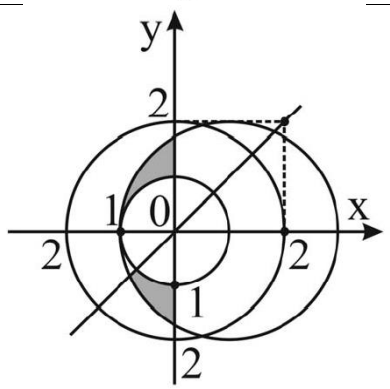
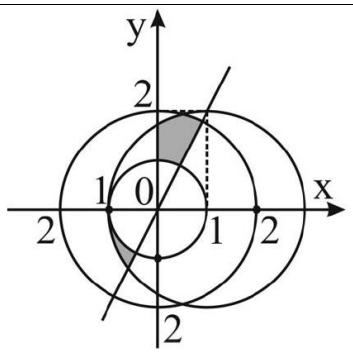
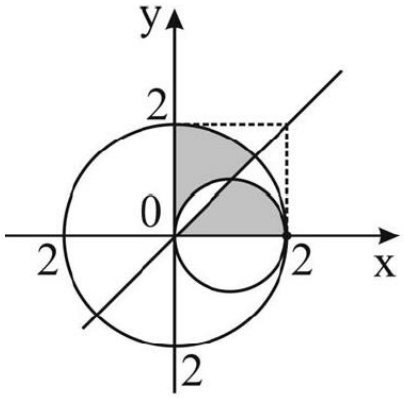
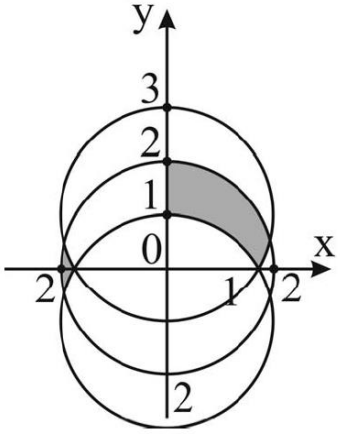
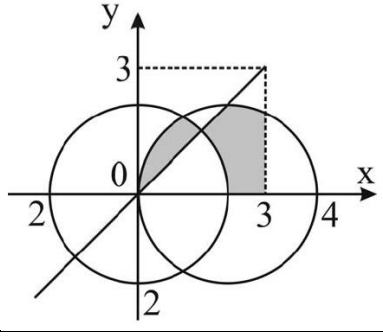
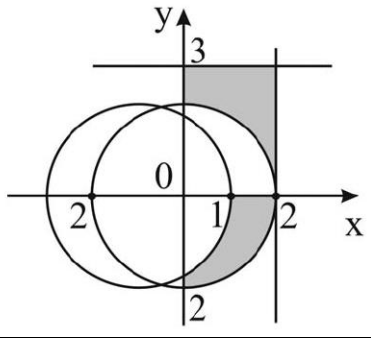
3. Реализуйте обработчик события изменения координат курсора мыши таким образом, чтобы при попадании курсора мыши в закрашенную область, соответствующий закрашенный регион менял свой цвет.

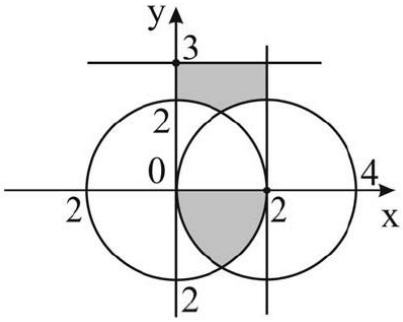
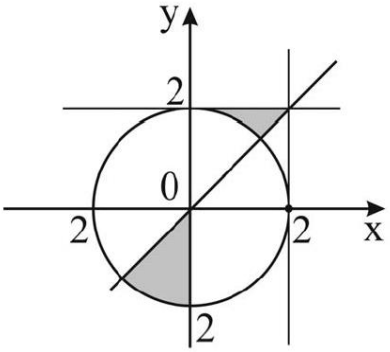
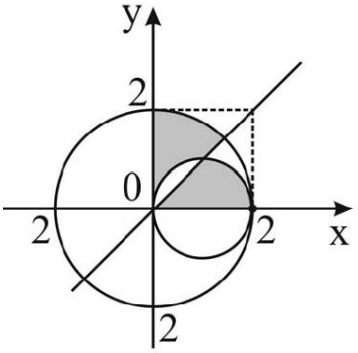
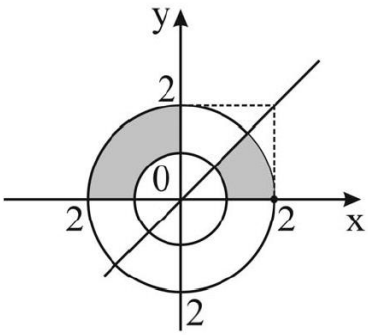
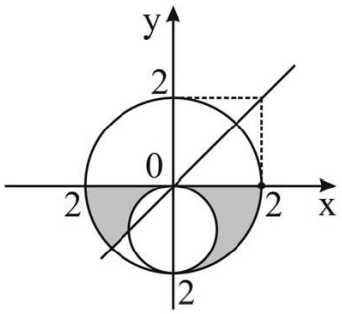
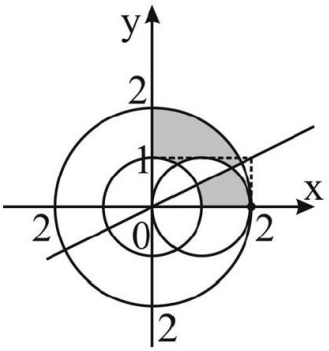
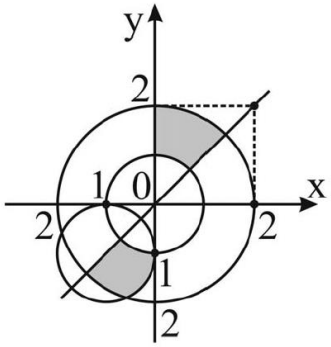
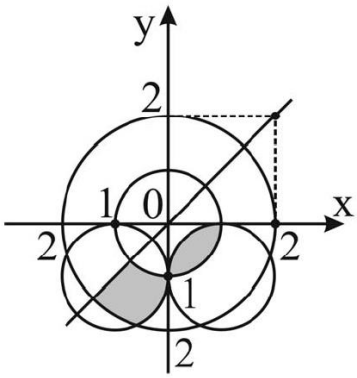
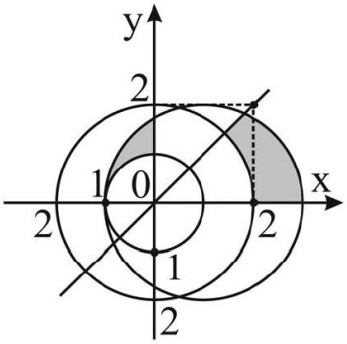
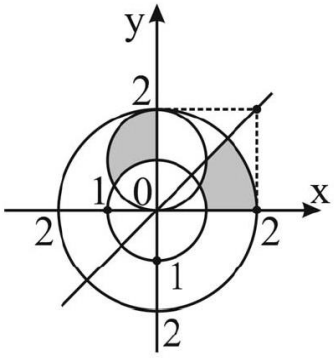
4. Реализуйте возможность выбора цвета закрашенных областей с использованием стандартного диалога операционной системы для выбора цвета.

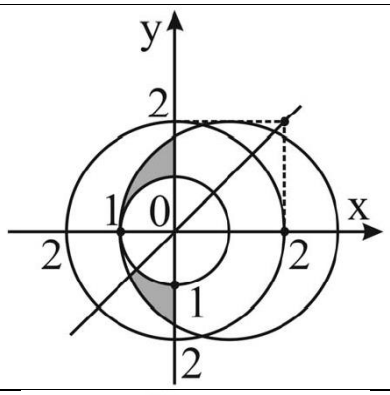
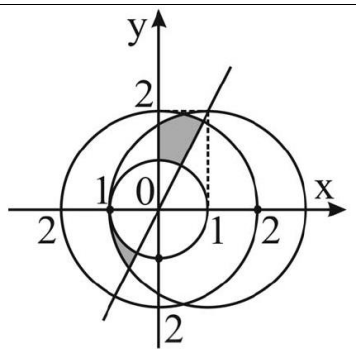
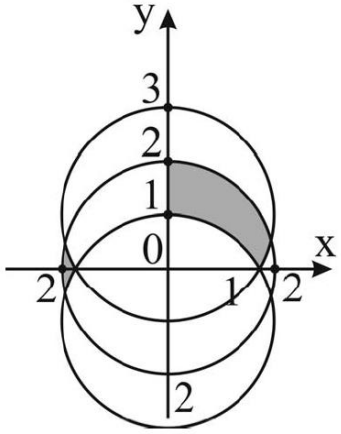
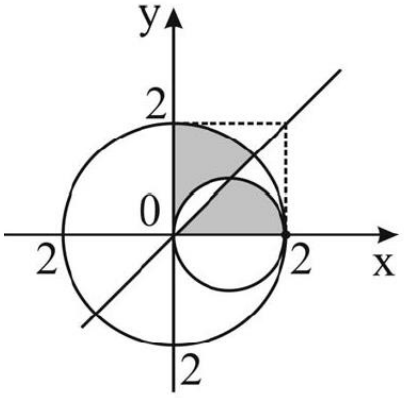
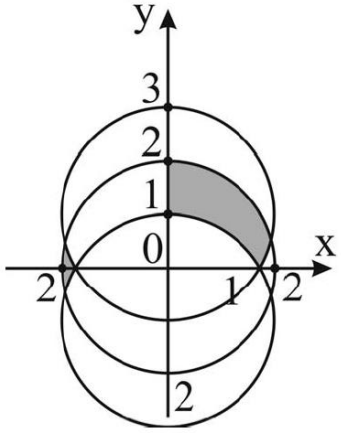
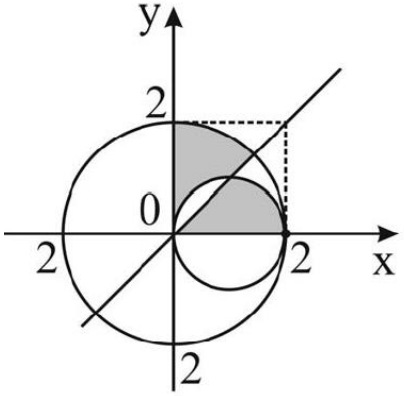
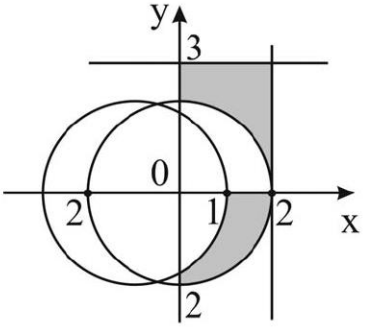
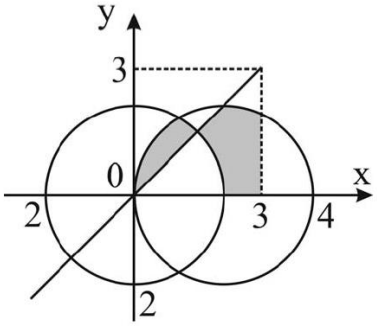
5. Реализуйте возможность перетаскивания пользователем подписей к рисункам (названия осей, цифры).

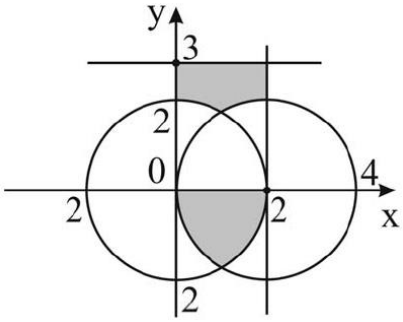
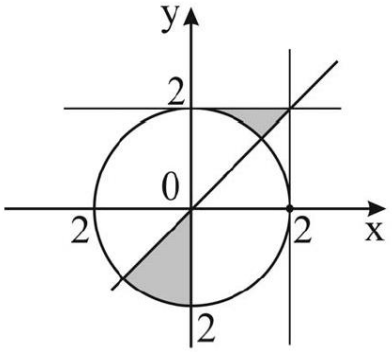
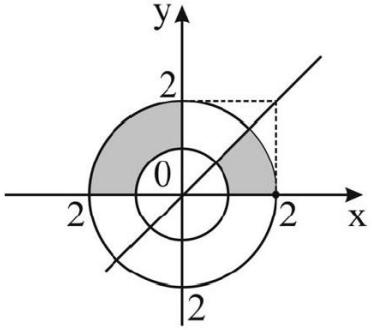
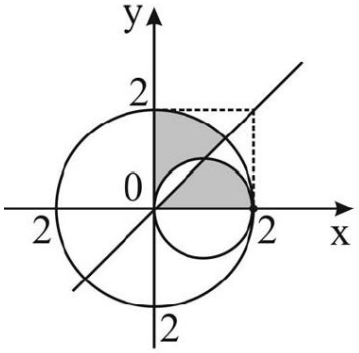
6. Реализуйте анимацию: циклическое изменение размеров шрифта подписей, анимацию цвета элементов и т.п. (что анимировать – выбирает разработчик). Необходимо предусмотреть возможность остановки и запуска анимации.

Вариант	Задание	Вариант	Задание
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	
27		28	

29		30	
31		32	
33		34	
35		36	

37		38	
39		40	

17.4. Вопросы для защиты работы

1. Назовите основные типы, необходимые для работы с GDI+.

2. Какие пространства имен доступны для работы с графикой?

3. Опишите возможные конструкторы класса Point.

4. В чем состоит особенность конструктора класса Region?

5. Как получить доступ к объекту класса Graphics?

6. Опишите способ, с помощью которого можно реализовать перехват события мыши над фигурами GDI+.

7. Для чего предназначен класс GraphicPath?

8. Чем отличаются типы Point и PointF?

9. Какие элементы могут обрабатывать событие Paint?

10. Для чего используется тип PathPointType?

11. Через какой тип происходит рисование в обработчике события Paint?

12. Опишите механизм выбора объекта GDI.

13. Какие события участвуют в процессе интерактивного взаимодействия с GDI объектами графики?

14. Какое событие необходимо задействовать, если добавить в программу отображение контура объекта при перетаскивании?

15. Опишите механизм реализации анимации с использованием GDI+.

16. Опишите назначение основных свойств и методов класса Timer.

17. Для чего предназначено событие Tick класса Timer?

18. Сколько таймеров может присутствовать в программе?

ЛАБОРАТОРНАЯ РАБОТА 18. ПРИЛОЖЕНИЯ WINDOWS PRESENTATION FOUNDATION

18.1. Цель и содержание

Цель лабораторной работы: научиться проектировать интерфейс приложений на основе технологии WPF.

Задачами лабораторной работы являются:

– изучить возможности работы с классом Window;

– изучить принципы использования контейнеров компоновки;

– научиться обрабатывать события простых элементов управления.

18.2. Теоретическое обоснование

18.2.1 Класс Window.

Объект Window – основной элемент традиционных приложений, в нем отображается их основное содержимое. В Windows Presentation Foundation (WPF) за объектом Window скрывается обычное окно Win32. Операционная система не различает окна с WPF- и Win32-содержимым: обрамление рисуется одинаково, на панели задач они неотличимы друг от друга и т. д. Обрамление (Chrome) – это просто другое название области окна, которая обрамляет клиентскую область и в числе прочего содержит кнопки Minimize (Свернуть), Maximize (Развернуть) и Close (Закрыть).

Объект Window – это абстракция окна Win32 (так же, как класс Form в Windows Forms), содержащая ряд простых методов и свойств. Создав объект Window, программист может показать его на экране с помощью метода Show, скрыть – методом Hide() (это то же самое, что присвоить свойству Visibility значение Hidden или Collapsed) и закрыть навсегда – методом Close().

Внешним видом Window можно управлять с помощью таких свойств, как Icon (иконка приложения), Title (интерпретируется как заголовок окна) и WindowStyle. Для управления положением на экране служат свойства Left и Top. Более осмысленного поведения можно добиться, присваивая свойству WindowStartupLocation значение CenterScreen или CenterOwner. Короче говоря, с помощью установки свойств можно делать почти все, что принято ожидать от окна. Скажем, если установить для Topmost значение true, то окно будет всегда отображаться поверх других, а если присвоить ShowInTaskbar значение false, то значок окна не будет показываться на панели задач.

Всякий раз, как окно становится активным или неактивным (например, из-за того, что пользователь переключается между разными окнами), возникает событие Activated или Deactivated объекта Window.

На рис. 18.1 представлено определение окна с использованием языка разметки XAML. Данное определение генерируется Visual Studio по умолчанию.

```
1  <Window x:Class="WpfExample.Window1"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Window1" Height="300" Width="300">
5      <Grid>
6
7      </Grid>
8  </Window>
```

Рисунок 18.1 – Шаблон окна приложения, генерируемого по умолчанию.

Окно может содержать только один дочерний элемент. Для размещения в окне множества дочерних элементов управления в WPF используется подход: сначала в фрейме окна размещается специализированная панель (контейнер компоновки), а затем, в данном контейнере располагаются элементы управления.

18.2.2 Контейнеры компоновки

Для размещения элементов управления в WPF используются специализированные компоненты: контейнеры компоновки. В технологии Windows Forms применяется механизм абсолютного позиционирования элементов

управления: задаются координаты верхнего левого угла элемента (свойства Top, Left) и его размер (свойства Width и Height).

В WPF компоновка определяется используемым контейнером. Окно в WPF должно реализовывать следующие принципы:

1. Элементы (такие как элементы управления) не должны иметь явно установленных размеров. Вместо этого они растут, чтобы уместить свое содержимое. Например, кнопка увеличивается при добавлении в нее текста. Можно ограничить элементы управления приемлемыми размерами, устанавливая максимальное и минимальное их значение.

2. Элементы не указывают свою позицию в экранных координатах. Вместо этого они упорядочиваются своим контейнером на основе размера, порядка и (необязательно) другой информации, специфичной для контейнера компоновки. Для добавления пробелов между элементами служит свойство Margin.

3. Контейнеры компоновки «разделяют» доступное пространство между своими дочерними элементами. Они пытаются обеспечить для каждого элемента его предпочтительный размер (на основе его содержимого), если только позволяет свободное пространство. Они могут также выделять дополнительное пространство одному или более дочерним элементам.

4. Контейнеры компоновки могут быть вложенными. Типичный пользовательский интерфейс начинается с Grid – наиболее развитого контейнера, и содержит другие контейнеры компоновки, которые организуют меньшие группы элементов, такие как текстовые поля с метками, элементы списка, значки в панели инструментов, колонка кнопок и т.д.

Все контейнеры компоновки WPF являются панелями, которые унаследованы от абстрактного класса System.Windows.Controls.Panel (рис. 18.2). Класс Panel добавляет небольшой набор возможностей всем производным классам.

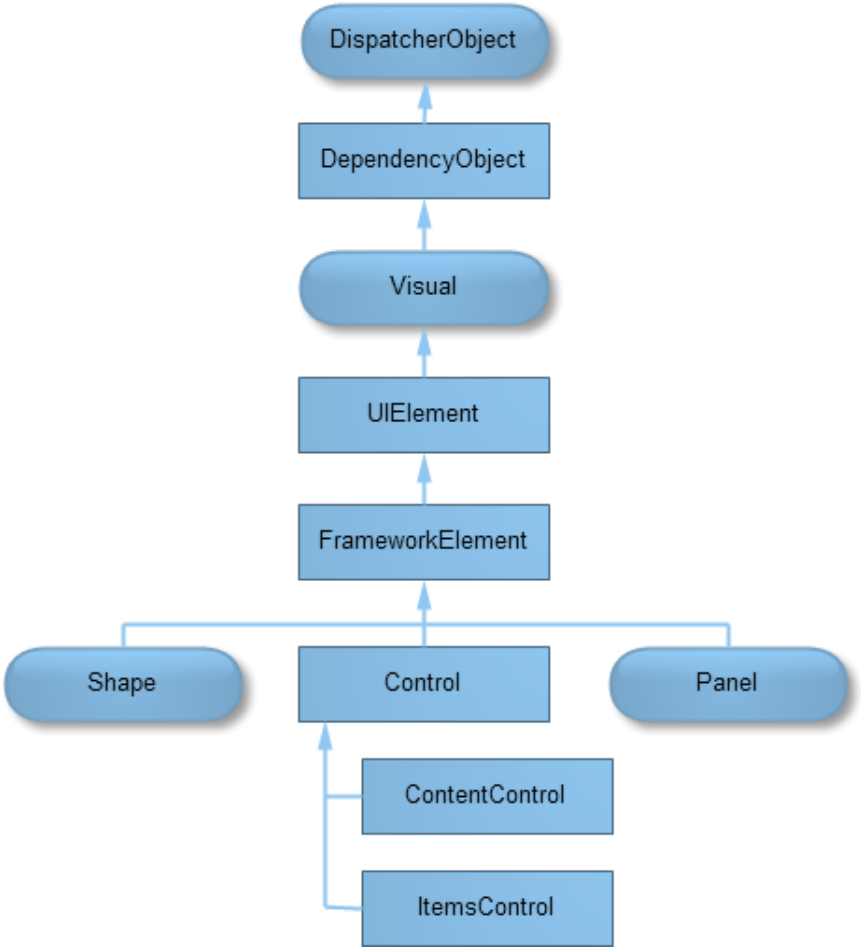


Рисунок 18.2 – Иерархия классов WPF

Сам по себе базовый класс Panel – это абстрактный класс, не допускающий создания объектов. WPF предлагает набор производных от Panel классов, которые можно использовать для организации компоновки. Основные контейнеры компоновки перечислены в табл. 18.1. Как и все элементы управления WPF, а также большинство визуальных элементов, эти классы находятся в пространстве имен System.Windows.Controls.

Таблица 18.1 – Контейнеры компоновки WPF

Контейнер	Описание
StackPanel	Размещает элементы в горизонтальном или вертикальном стеке. Этот контейнер компоновки обычно используется в небольших разделах крупного и более сложного окна.

WrapPanel	Размещает элементы в последовательностях строк с переносом. В горизонтальной ориентации WrapPanel располагает элементы в строке слева направо, затем переходит к следующей строке. В вертикальной ориентации WrapPanel располагает элементы сверху вниз, используя дополнительные колонки для дополнения оставшихся элементов.
DockPanel	Выравнивает элементы по краю контейнера.
Grid	Выстраивает элементы в строки и колонки невидимой таблицы. Это один из наиболее гибких и широко используемых контейнеров компоновки.
UniformGrid	Помещает элементы в невидимую таблицу, устанавливая одинаковый размер для всех ячеек. Данный контейнер компоновки используется нечасто.
Canvas	Позволяет элементам позиционироваться абсолютно — по фиксированным координатам. Этот контейнер компоновки более всего похож на традиционный компоновщик Windows Forms, но не предусматривает средств привязки и стыковки. В результате это неподходящий выбор для окон переменного размера.

18.2.2 Контейнер StackPanel

На рис. 18.3 представлен простой пример компоновки элементов с использованием контейнера StackPanel.

```
1  <Window x:Class="WpfExample.Window31"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Window31" Height="127" Width="300">
5      <StackPanel>
6          <Button>Кнопка 1</Button>
7          <Button>Кнопка 2</Button>
8          <Button>Кнопка 3</Button>
9          <Button>Кнопка 4</Button>
10     </StackPanel>
11 </Window>
```

Рисунок 18.3 – Использование StackPanel

Результат подобной компоновки представлен на рис. 18.4. Без дополнительной информации о порядке, координатах и размерах кнопок с помощью контейнера StackPanel удалось расположить элементы вертикально.

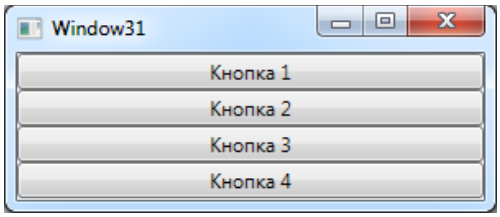


Рисунок 18.4 – Окно с контейнером StackPanel

По умолчанию панель StackPanel располагает элементы сверху вниз, устанавливая высоту каждого из них такой, которая необходима для отображения его содержимого.

В данном примере это значит, что размер кнопок устанавливается достаточно большим для спокойного размещения текста внутри них. Все элементы растягиваются на полную ширину StackPanel, которая равна ширине окна. Если вы расширите окно, StackPanel также расширится, и кнопки растянутся, чтобы заполнить ее.

StackPanel может также использоваться для упорядочивания элементов в горизонтальном направлении за счет установки свойства Orientation (рис. 18.5):

<StackPanel Orientation=llHorizontalll>

```
1  <Window x:Class="WpfExample.Window31"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Window31" Height="127" Width="300">
5      <StackPanel Orientation="Horizontal">
6          <Button>Кнопка 1</Button>
7          <Button>Кнопка 2</Button>
8          <Button>Кнопка 3</Button>
9          <Button>Кнопка 4</Button>
10     </StackPanel>
11 </Window>
```

Рисунок 18.5 – Использование StackPanel

Результат представлен на рис. 18.6

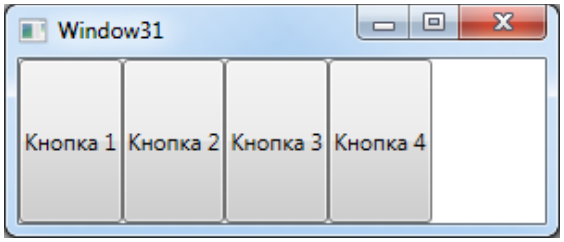


Рисунок 18.6 – Использование горизонтального размещения StackPanel.

Размещение элементов управления определяется контейнером, но элементы управления могут повлиять на этот процесс, явно задавая свойства. Свойства элементов, которые влияют на размещение в контейнере представлены в таблице 18.2.

Таблица 18.2 – Свойства компоновки

Наименование свойства	Описание
HonzontalAlignment	Определяет позиционирование дочернего элемента внутри контейнера компоновки, когда имеется дополнительное пространство по горизонтали. Доступные значения: Center, Left, Right или Stretch.
VerticalAlignment	Определяет позиционирование дочернего элемента внутри контейнера компоновки, когда имеется дополнительное пространство по вертикали. Доступные значения: Center, Top, Bottom или Stretch.
Margin	Добавляет некоторое пространство вокруг элемента. Свойство Margin – это экземпляр структуры System.Windows.Thickness, с отдельными компонентами для верхней, нижней, левой и правой граней.
MinWidth и MinHeight	Устанавливает минимальные размеры элемента. Если элемент слишком велик, чтобы поместиться в его контейнер компоновки, он будет усечен.
MaxWidth и MaxHeight	Устанавливает максимальные размеры элемента. Если контейнер имеет свободное пространство, элемент не будет увеличен сверх указанных пределов, даже если свойства HonzontalAlignment и VerticalAlignment установлены в Stretch.
Width и Height	Явно устанавливают размеры элемента. Эта установка переопределяет значение Stretch для свойств HorizontalAlignment и VerticalAlignment. Однако данный размер не будет установлен, если выходит за пределы, заданные в MinWidth, MinHeight, MaxWidth и MaxHeight.

18.2.3 Элемент Border

Border не является одной из панелей компоновки, но это удобный элемент, который часто используется.

Класс Border принимает единственную порцию вложенного содержимого (которым часто является панель компоновки) и добавляет фон или рамку вокруг него. Для работы с Border понадобятся свойства, перечисленные в табл. 18.3.

Таблица 18.3 – Свойства класса Border

Наименование свойства	Описание
Background	С помощью объекта Brush устанавливает фон, который появляется под содержимым. Можно использовать сплошной цвет либо что-нибудь более сложное.
BorderBrush и BorderThickness	Устанавливают цвет рамки, который появляется на границе объекта Border, и толщину рамки. Для отображения рамки потребуется установить оба свойства.
CornerRadius	Позволяет скруглить углы рамки. Чем больше значение CornerRadius, тем заметнее эффект.
Padding	Добавляет пространство между рамкой и содержимым внутри нее.

На рис. 18.7 представлен фрагмент кода XAML, использующий данные свойства.

```
1  <Window x:Class="WpfExample.Window31"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Window31" Height="127" Width="300">
5      <Border Background="GreenYellow" BorderBrush="Red"
6              BorderThickness="5" CornerRadius="10" Padding="10">
7          <StackPanel Orientation="Vertical">
8              <Button>Кнопка 1</Button>
9              <Button>Кнопка 2</Button>
10             <Button>Кнопка 3</Button>
11             <Button>Кнопка 4</Button>
12         </StackPanel>
13     </Border>
14
15 </Window>
```

Рисунок 18.7 – Использование свойств Border

На рис. 18.8 показано результирующее окно.

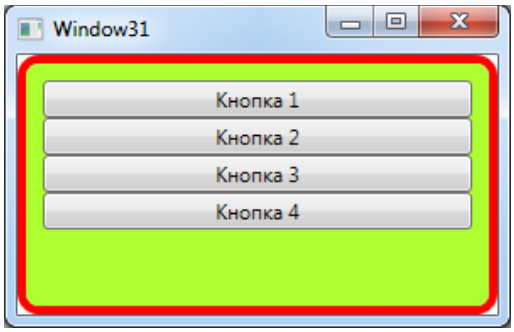


Рисунок 18.8 – Результат применения Border

18.2.4 Элементы WrapPanel и DockPanel

Панель WrapPanel располагает элементы управления в доступном пространстве – по одной строке или колонке за раз. По умолчанию свойство WrapPanel.Orientation установлено в Horizontal; элементы управления располагаются слева направо, затем – в следующих строках. Установка значения Vertical для свойства WrapPanel. Orientation приводит к размещению элементов в нескольких колонках.

Подобно StackPanel, панель WrapPanel действительно предназначена для управления мелкими деталями пользовательского интерфейса, а не компоновкой всего окна. Например, WrapPanel можно использовать для удержания вместе кнопок в элементе управления, подобном панели инструментов.

Панель DockPanel обеспечивает более интересный вариант компоновки. Эта панель растягивает элементы управления вдоль одной из внешних границ. Простейший способ представить это – вообразить панель инструментов, которая присутствует в верхней части многих Windows-приложений. Такие панели инструментов пристыковываются к верхней части окна. Как и в случае StackPanel, пристыкованные элементы должны выбрать один аспект компоновки. Например, если вы пристыковать кнопку к верхней части DockPanel, она растянется на всю ширину DockPanel, но получит высоту, которая ей понадобится (на основе своего содержимого и свойства MinHeight). С другой стороны, если пристыковать кнопку к левой стороне контейнера, ее высота будет растянута для заполнения контейнера, но ширина будет установлена по необходимости.

Дочерние элементы DockPanel прикрепляются к одной из сторон панели посредством задания присоединенного свойства Dock. Элементы на самом деле не имеют этого свойства – они приобретают его после помещения в контейнер.

На рис. 18.9 показан пример использования DockPanel.

```
<Window x:Class="WpfExample.DockPanelExample"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="DockPanelExample" Height="300" Width="300">
    <DockPanel>
        <Button Content="кнопка 1" DockPanel.Dock="Top"/>
        <Button Content="кнопка 2" DockPanel.Dock="Bottom"/>
        <Button Content="кнопка 3" DockPanel.Dock="Left"/>
        <Button Content="кнопка 4" DockPanel.Dock="Right"/>
        <Button Content="кнопка 5"/>
    </DockPanel>
</Window>
```

Рисунок 18.9 – Использование DockPanel

На рис. 18.10 представлен результат.

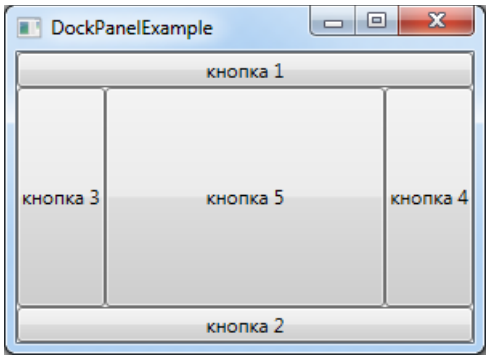


Рисунок 18.10 – Контейнер DockPanel в окне приложения

18.2.5 Контейнер Grid

Элемент управления Grid – это наиболее мощный контейнер компоновки в WPF. Большая часть того, что можно достичь с помощью других элементов управления компоновкой, также возможно и в Grid. Контейнер Grid является идеальным инструментом для разбиения окна на меньшие области, которыми можно управлять с помощью других панелей.

При добавлении в Visual Studio нового документа XAML для окна автоматически добавляются дескрипторы Grid в качестве контейнера первого уровня, вложенного внутрь корневого элемента Window.

Хотя Grid задуман как невидимый элемент, можно установить свойство Grid.ShowGridLines в true и получить наглядное представление о нем. Это средство на самом деле не предназначено для украшения окна. В действительности это средство для облегчения отладки, которое предназначено для того, чтобы наглядно показать, как Grid разделяет пространство на отдельные области. Благодаря ему, появляется возможность точно контролировать то, как Grid выбирает ширину колонок и высоту строк.

Создание компоновки на основе Grid – поэтапный процесс. Сначала выбирается необходимое количество колонок и строк. Затем каждому содержащемуся элементу назначается соответствующая строка и колонка, тем самым помещая его в правильное место.

Колонки и строки создаются путем заполнения объектами коллекции Grid.ColumnDefinitions и Grid.RowDefinitions. Например, если необходимо создать две строки и три колонки, то используются следующие дескрипторы:

```
<Window x:Class="WpfExample.GridExample"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="GridExample" Height="300" Width="300">
    <Grid>
        <Grid Background="GreenYellow">
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition/>
                <ColumnDefinition/>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <Button Grid.Column="0" Grid.Row="0" Content="Кнопка (0, 0)"/>
            <Button Grid.Column="1" Grid.Row="1" Content="Кнопка (1, 1)"/>
            <Button Grid.Column="2" Grid.Row="2" Content="Кнопка (2, 2)"/>
            <Button Grid.Column="2" Grid.Row="0" Content="Кнопка (2, 0)"/>
            <Button Grid.Column="0" Grid.Row="2" Content="Кнопка (0, 2)"/>
        </Grid>
    </Grid>
</Window>
```

Рисунок 18.11 – Контейнер Grid

Результирующее окно представлено на рис. 18.12.

Рисунок 18.12 – Расположение кнопок в контейнере Grid

18.3. Методика и порядок выполнения работы

18.3.1. Учебная задача

Для закрепления теоретического материала и получения практических навыков работы с технологий WPF решим следующую задачу:

Условие. Реализовать приложение для вычисления выражения:

$$S = \begin{cases} \sin(f \cdot a),\ f \in \{4,5,6,7,8,9\}; \\ \cos(f \cdot a) + \sin(f \cdot b),\ f \in \{10,20,30,40\}; \\ c \cdot a^2 + d \cdot b^2,\ c \in \{0,1\},\ d \in \{-1,0,1\}; \\ \sum_{i=0}^{d} (c+a)^i,\ d = const,\ c \in \{0,1,2,3,4,5\}. \end{cases}$$

Решение. В примере вычисление выражения может быть организовано по четырем возможным направлениям, причем критерии выбора одной из формул не определены. Это значит, что необходимо дать пользователю возможность производить выбор формулы для расчета. Решение задачи реализуем последовательно, выполняя следующие стадии:

1. Создайте приложение WPF Application. В качестве имени проекта необходимо указать «WpfExample». При разработке приложения необходимо придерживаться какого-либо наброска интерфейса, например, изображенного на рис. 16.13.

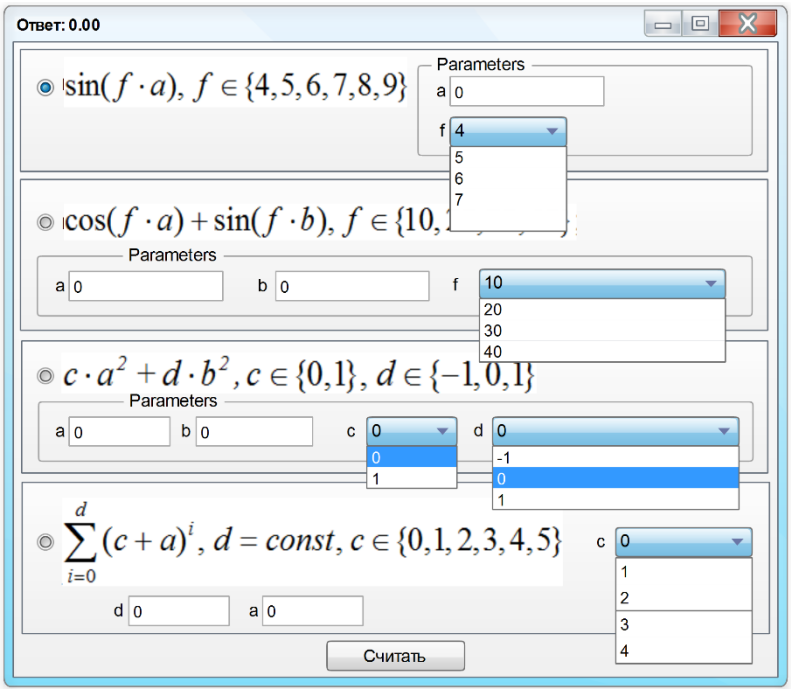


Рисунок 16.13 – Проект формы: приблизительное расположение элементов управления формы

На рис. 16.13 представлен приблизительный интерфейс, но даже из этого макета можно выделить следующие особенности:

– форма разбита на 4 области, каждая из которых представляет собой панель с параметрами для вычисления выражения;

– каждая из четырех панелей формы может быть отмечена пользователем посредством выбора с помощью RadioButton;

– при нажатии кнопки «Считать» результат вычисления, выбранного пользователем выражения, выводится в заголовок окна.

2. Перейдем к созданию интерфейса с использованием языка XAML. Для начала определим контейнер компоновки верхнего уровня. По умолчанию в классе Window установлен контейнер Grid. Удалите его и добавьте контейнер StackPanel. Измените свойства Window. В результате этих действий получим код:

```
1  <Window x:Class="WpfExample.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="Ответ: "
5          Height="420" Width="480"
6          WindowStartupLocation="CenterScreen"
7          SizeToContent="Height">
8      <StackPanel>
9          <RadioButton IsChecked="True" Margin="5">
10             <WrapPanel...>
37         </RadioButton>
38         <RadioButton Margin="5">
39             <StackPanel...>
57         </RadioButton>
58         <RadioButton Margin="5">
59             <StackPanel...>
81         </RadioButton>
82         <RadioButton Margin="5">
83             <WrapPanel...>
96         </RadioButton>
97         <Button Content="Считать" Width="100"/>
98     </StackPanel>
99 </Window>
```

Рисунок 16.14 – Создание контейнера и добавление элементов в контейнер

Рассмотрим данный пример подробнее:

– в строке 1 расположен открывающий дескриптор <Window> главного окна приложения; в строках 4-7 устанавливаются атрибуты главного окна (поясните, какие именно свойства окна задаются данными атрибутами);

– в строке 8 определяется контейнер <StackPanel> (контейнер <Grid> был удален);

– внутри контейнера StackPanel определены четыре элемента RadioButton и кнопка «Считать»; обратите внимание, что внутри каждого элемента RadioButton добавляется свой контейнер компановки (в первом и четвертом – WrapPanel, во втором и третьем – StackPanel).

3. Для удобства отображения вложенности элементов в Visual Studio предусмотрен просмотр элементов окна в виде дерева. Откройте окно «Document Outline» (рис. 18.15) и убедитесь, что иерархия элементов соответствует задуманной.

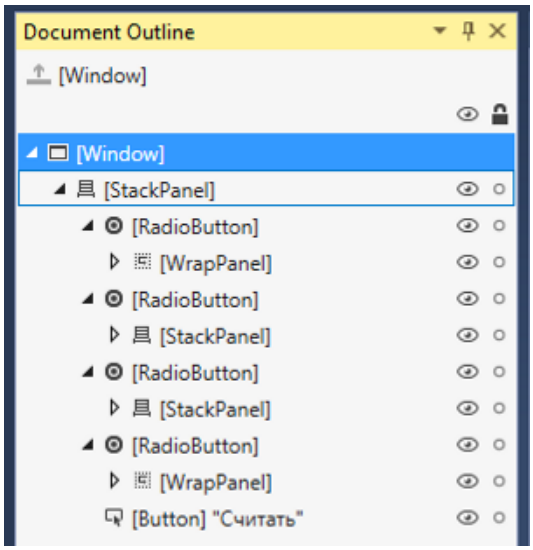


Рисунок 18.15 – Просмотр иерархии элементов главного окна

4. Далее произведем наполнение каждого элемента RadioButton. При дальнейшей разработке присвоим элементам имена, чтобы обращаться к ним из кода на языке C#.

4.1 На рис. 18.16 представлено содержимое первого RadioButton. После создания данного фрагмента кода необходимо проконтролировать вложенность элементов с использованием окна «Document Outline» (рис. 18.17).

```
9         <RadioButton IsChecked="True" Margin="5" Name="Radio1">
10            <WrapPanel>
11                <Image Source="p1.png" Height="30"/>
12                <GroupBox Header="Parameters">
13                    <Grid>
14                        <Grid.RowDefinitions>
15                            <RowDefinition/>
16                            <RowDefinition/>
17                        </Grid.RowDefinitions>
18                        <Grid.ColumnDefinitions>
19                            <ColumnDefinition/>
20                            <ColumnDefinition Width="100"/>
21                        </Grid.ColumnDefinitions>
22                        <Label Content="a" Grid.Column="0" Grid.Row="0"/>
23                        <TextBox Text="0,00" Grid.Column="1" Grid.Row="0"
24                                 MinWidth="70" Name="R1TextA"/>
25                        <Label Content="f" Grid.Column="0" Grid.Row="1"/>
26                        <ComboBox Grid.Column="1" Grid.Row="1" SelectedIndex="0"
27                                  Name="R1ComboF">
28                            <ComboBoxItem>4</ComboBoxItem>
29                            <ComboBoxItem>5</ComboBoxItem>
30                            <ComboBoxItem>6</ComboBoxItem>
31                            <ComboBoxItem>7</ComboBoxItem>
32                            <ComboBoxItem>8</ComboBoxItem>
33                            <ComboBoxItem>9</ComboBoxItem>
34                        </ComboBox>
35                    </Grid>
36                </GroupBox>
37            </WrapPanel>
38        </RadioButton>
```

Рисунок 18.16 – Дескриптор RadioButton для представления первой формулы

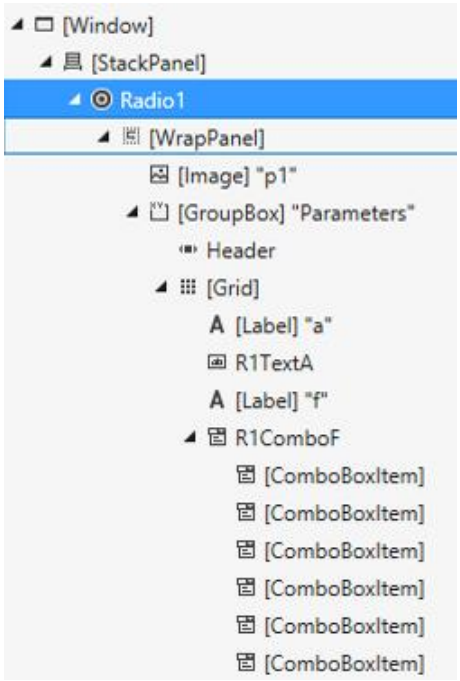


Рисунок 18.17 – Дескриптор RadioButton (иерархическое представление)

4.2 На рис. 18.18 представлено наполнение второго RadioButton для представления параметров вычисления второй формулы выражения. И снова необходимо просмотреть иерархию элементов управления с использованием окна «Document Outline».

```
 9  <RadioButton IsChecked="True" Margin="5" Name="Radio1"...>
39  <RadioButton Margin="5" Name="Radio2">
40      <StackPanel>
41          <Image Height="23" Source="p2.png"/>
42          <GroupBox Header="Parameters">
43              <StackPanel Orientation="Horizontal">
44                  <Label Content="a"/>
45                  <TextBox Text="0,00" MinWidth="70" Name="R2TextA"/>
46                  <Label Content="b"/>
47                  <TextBox Text="0,00" MinWidth="70" Name="R2TextB"/>
48                  <Label Content="f"/>
49                  <ComboBox MinWidth="100" Name="R2ComboF">
50                      <ComboBoxItem>10</ComboBoxItem>
51                      <ComboBoxItem>20</ComboBoxItem>
52                      <ComboBoxItem>30</ComboBoxItem>
53                      <ComboBoxItem>40</ComboBoxItem>
54                  </ComboBox>
55              </StackPanel>
56          </GroupBox>
57      </StackPanel>
58  </RadioButton>
```

Рисунок 18.18 – Дескриптор RadioButton для представления второй формулы

4.3 На рис. 16.19 представлен код XAML для третьего дескриптора RadioButton.

```
9   <RadioButton IsChecked="True" Margin="5" Name="Radio1"...>
39  <RadioButton Margin="5" Name="Radio2"...>
59  <RadioButton Margin="5" Name="Radio3">
60      <StackPanel>
61          <Image Source="p3.png" Height="30"/>
62          <GroupBox Header="Parameters">
63              <StackPanel Orientation="Horizontal">
64                  <Label Content="a"/>
65                  <TextBox Text="0,00" MinWidth="50" Name="R3TextA"/>
66                  <Label Content="b"/>
67                  <TextBox Text="0,00" MinWidth="50" Name="R3TextB"/>
68                  <Label Content="c"/>
69                  <ComboBox MinWidth="50" SelectedIndex="0" Name="R3ComboC">
70                      <ComboBoxItem Content="0"/>
71                      <ComboBoxItem Content="1"/>
72                  </ComboBox>
73                  <Label Content="d"/>
74                  <ComboBox MinWidth="70" SelectedIndex="0" Name="R3ComboD">
75                      <ComboBoxItem Content="-1"/>
76                      <ComboBoxItem Content="0"/>
77                      <ComboBoxItem Content="1"/>
78                  </ComboBox>
79              </StackPanel>
80          </GroupBox>
81      </StackPanel>
82  </RadioButton>
```

Рисунок 18.19 – Дескриптор RadioButton для представления третьей формулы

4.4 Затем наполним содержимым четвертый дескриптор RadioButton (рис. 18.20).

```
84  <RadioButton Margin="5" Name="Radio4">
85      <WrapPanel>
86          <Image Source="p4.png" Height="60"/>
87          <Label Content="c" VerticalAlignment="Center"/>
88          <ComboBox MinWidth="70" VerticalAlignment="Center" Name="R4ComboC"
89                    SelectedIndex="0">
90              <ComboBoxItem Content="0"/>
91              <ComboBoxItem Content="1"/>
92              <ComboBoxItem Content="2"/>
93          </ComboBox>
94          <Label Content="d" VerticalAlignment="Center"/>
95          <TextBox Text="0,00" Width="70"
96                   VerticalAlignment="Center" Name="R4TextD"/>
97          <Label Content="a" VerticalAlignment="Center"/>
98          <TextBox Text="0,00" Width="70"
99                   VerticalAlignment="Center" Name="R4TextA"/>
100     </WrapPanel>
101 </RadioButton>
```

Рисунок 18.20 – Дескриптор RadioButton для представления четвертой формулы

5. В результате выполнения этапов 1 – 4 получим главное окно приложения, внешний вид которого показан на рис. 18.21 (показано окно в режиме

проектирования). На рис. 18.22 показано это же окно в режиме выполнения приложения.

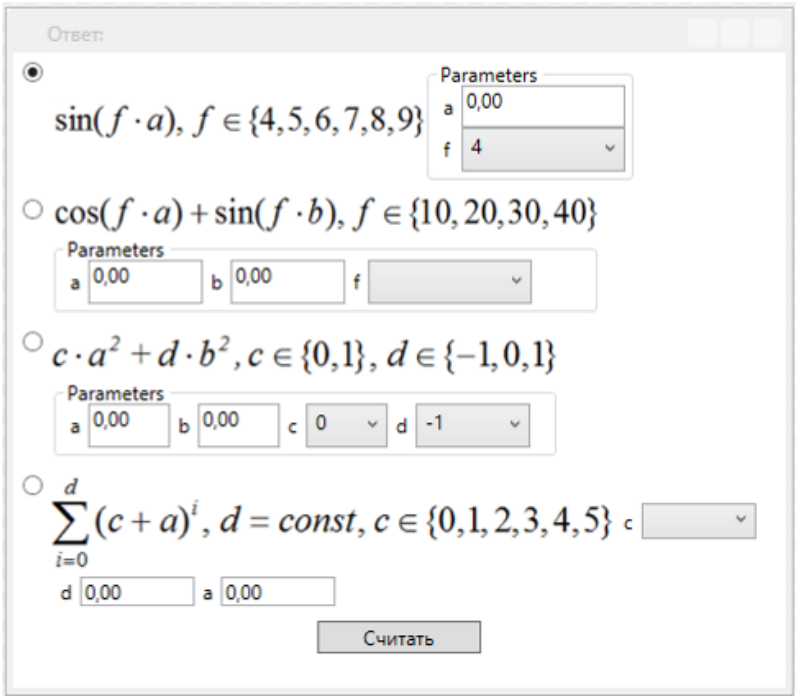


Рисунок 18.21 – Главное окно приложения в режиме проектирования

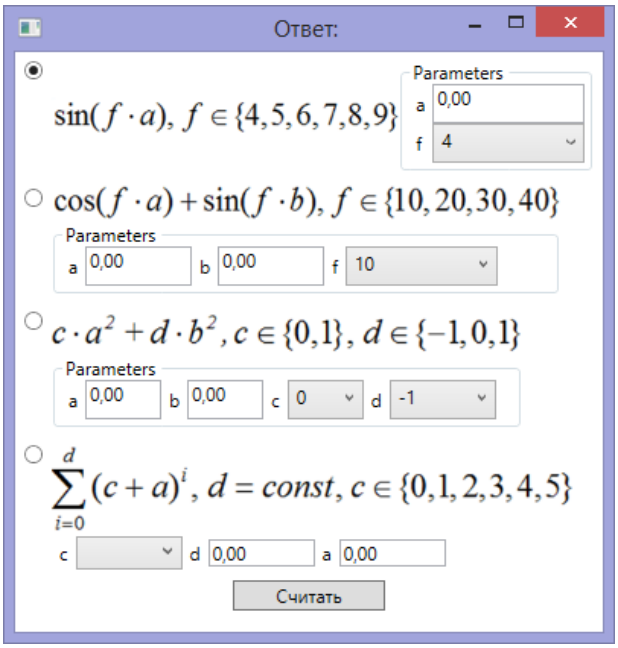


Рисунок 18.22 – Главное окно приложения в режиме выполнения

6. Окно приложения создано, теперь необходимо написать код обработчика события нажатия кнопки «Считать». Для этого просто добавьте соответствующий атрибут в определение кнопки (рис. 18.23).

```
99          <Button Content="Считать" Width="100"
100                 Click="Calc_Click"/>
```

Рисунок 18.23 – Добавление обработчика нажатия кнопки.

В результате выполнения данного действия Visual Studio откроет редактор кода C# и загрузит в него файл MainWindow.xaml.cs. В отличие от файла MainWindow.xaml (в котором определен код XAML окна), в данном файле содержится листинг кода на языке C#.

Добавьте в обработчик события нажатия кнопки следующий код:

```
29          private void Calc_Click(object sender, RoutedEventArgs e)
30          {
31              // Определяем, какой RadioButton выбран
32              // и в зависимости от этого расчитываем нужную формулу
33              if (Radio1.IsChecked.GetValueOrDefault())
34              {
35                  double a = Convert.ToDouble(R1TextA.Text);
36                  double f = Convert.ToDouble(R1ComboF.Text);
37                  this.Title = "Ответ: " + Math.Sin(f*a).ToString("F");
38              }
39
40              if (Radio2.IsChecked.GetValueOrDefault())
41              {
42                  double a = Convert.ToDouble(R2TextA.Text);
43                  double b = Convert.ToDouble(R2TextB.Text);
44                  double f = Convert.ToDouble(R2ComboF.Text);
45                  this.Title = "Ответ: "
46                      + (Math.Cos(f*a) + Math.Sin(f*b)).ToString("F");
47              }
48
49              if (Radio3.IsChecked.GetValueOrDefault())
50              {
51                  double a = Convert.ToDouble(R3TextA.Text);
52                  double b = Convert.ToDouble(R3TextB.Text);
53                  double c = Convert.ToDouble(R3ComboC.Text);
54                  double d = Convert.ToDouble(R3ComboD.Text);
55                  this.Title = "Ответ: "
56                      + (c*a*a + d*b*b).ToString("F");
57              }
58
```

```
59              if (Radio4.IsChecked.GetValueOrDefault())
60              {
61                  double a = Convert.ToDouble(R4TextA.Text);
62                  double d = Convert.ToDouble(R4TextD.Text);
63                  double c = Convert.ToDouble(R4ComboC.Text);
64                  double res = 1;
65                  for (int i = 0; i < d; i++)
66                  {
67                      res = res*(c + a) + 1;
68                  }
69                  this.Title = "Ответ: "
70                      + res.ToString("F");
71              }
72          }
```

Рисунок 18.24 – Код обработчика нажатия кнопки «Считать»

Представленный код простой и в пояснениях не нуждается.

7. Запустите приложение. Протестируйте работу программы при различных значениях входных параметров.

8. Выполните задание в соответствии с индивидуальным вариантом. В каждом варианте необходимо реализовать вычисление выражения. Окно приложения необходимо разрабатывать с использованием технологии WPF. Необходимо самостоятельно спроектировать внешний вид окна, реализовать адаптивное расположение элементов управления.

18.3.2. Индивидуальное задание

1.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{p^i \cdot x^3 + f^j \cdot y^2}{i \cdot j}$.	2.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{t^i \cdot x^i + f^j \cdot y^j}{t \cdot i \cdot j}$.
3.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{p^i \cdot y^j}{i \cdot j}$	4.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\sin(x) \cdot x^i + f^j \cdot y^j}{i \cdot j}$
5.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\sin(x) \cdot x^i + \cos(y) \cdot y^j}{i \cdot j}$	6.	$Z = \sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\lvert x^i \rvert + \cos(y) \cdot y^j}{i \cdot j}$

7.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{i\cdot x^i+j\cdot y^j}{i\cdot j}$	8.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{i\cdot\cos(x^i)+j\cdot\sin(y^j)}{i\cdot j}$
9.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\cos(x^i)+j}{i\cdot j}$	10.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\cos(y^i)+j\cdot x}{i\cdot j}$
11.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\sin(y^i)+i\cdot x}{(i+1)\cdot j}$	12.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\sin(y\cdot x)+i\cdot y}{(i+1)\cdot(j+2)}$
13.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{2\cdot x^i+3\cdot y^j}{i^2\cdot j^2}$	14.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{i\cdot x^{2\cdot i}+j\cdot y^{2\cdot j}}{i^2\cdot j^2}$
15.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{p^{i-1}\cdot x+f^{j-1}\cdot y^2}{i\cdot j}$	16.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{a^{i-1}\cdot x^i+f^j\cdot y^j}{(i+1)\cdot j}$
17.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{5\cdot x+3\cdot y}{i^2\cdot j^2}$	18.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\cos(y^i)+j\cdot\sin(x)}{i\cdot(j-1)}$
19.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{\lvert x^i\rvert+\cos(y)\cdot y^j}{(i+1)\cdot j}$	20.	$Z=\sum_{i=1}^{N}\sum_{j=1}^{K}\frac{(p-1)^i\cdot y^j}{i\cdot j}$

18.4. Вопросы для защиты работы

1. Опишите основные свойства класса Window.
2. Что такое контейнер компоновки? Для чего они используются? Опишите свойства класса Panel.
3. Назовите основные контейнеры компоновки. Дайте характеристику и опишите особенности контейнеров компоновки.
4. Что такое присоединенное свойство? Приведите примеры.
5. Какой контейнер применяется в окне приложения как контейнер компоновки по умолчанию?
6. Какие свойства элементов управления влияют на положение элементов внутри контейнера компоновки?

ЛАБОРАТОРНАЯ РАБОТА 19. РЕСУРСЫ, СТИЛИ, ТРИГГЕРЫ

19.1. Цель и содержание

Цель лабораторной работы: научиться использовать инструменты XAML, позволяющие гибко настраивать и управлять внешним видом приложения.

Задачами лабораторной работы являются:

– научиться использовать ресурсы в приложениях WPF,

– научиться использовать стили;

– научится задействовать механизм триггеров.

19.2. Теоретическое обоснование

19.2.1 Ресурсы

Система ресурсов WPF представляет собой просто способ поддержания вместе набора полезных объектов, таких как наиболее часто используемые кисти, стили или шаблоны, что существенно упрощает работу с ними.

В WPF ресурсы можно определять в коде или в различных местах внутри разметки (вместе с отдельными элементами управления, внутри отдельных окон или во всем приложении).

Ресурсы обладают рядом важных преимуществ:

1. Эффективность. Ресурсы позволяют определять объект один раз и затем использовать его в нескольких местах внутри разметки. Это упрощает код и делает его намного эффективнее.

2. Сопровождаемость. Ресурсы позволяют переносить низкоуровневые детали форматирования (вроде размеров шрифтов) в центральное место, где их легко изменять. Это своего рода XAML-эквивалент создания констант в коде.

3. Адаптируемость. После отделения определенной информации от остальной части приложения и ее помещения в раздел ресурсов появляется возможность ее динамической модификации.

Каждый элемент включает свойство Resources, в котором хранится словарная коллекция ресурсов (представляющая собой экземпляр класса ResourceDictionary). Эта коллекция ресурсов может хранить объект любого типа с индексацией по строке. Хотя каждый элемент имеет свойство Resources (которое определено в классе FrameworkElement), чаще всего ресурсы определяются на уровне окна.

Рассмотрим пример (рис. 19.1) использования ресурсов для хранения данных различного типа.

```
<Window x:Class="ComplexDataExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="MainWindow" Height="350" Width="525">
    <Window.Resources>
        <SolidColorBrush x:Key="SolidBrush" Color="Black"/>
        <sys:Double x:Key="FontSize">32</sys:Double>
        <sys:Double x:Key="StrokeSize">5</sys:Double>
        <ImageBrush x:Key="BrushWithoutStretch" ImageSource="Images/example.png"
                    TileMode="Tile" Stretch="None"/>
        <ImageBrush x:Key="BrushWithTitle" ImageSource="Images/example1.png"
                    TileMode="Tile" Stretch="Uniform"
                    Viewport="0,0 0.2, 0.3"
                    ViewportUnits="RelativeToBoundingBox" />
    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions...>
        <Grid.RowDefinitions...>

        <Button Foreground="{StaticResource SolidBrush}"
                Background="{StaticResource BrushWithoutStretch}"
                FontSize="{StaticResource FontSize}"
                Margin="5">
            Click Me!
        </Button>
        <Ellipse Grid.Column="1"
```

```
34                  Fill="{StaticResource BrushWithTitle}"
35                  Stroke="{StaticResource SolidBrush}"
36                  StrokeThickness="{StaticResource StrokeSize}"/>
37          <Rectangle Grid.Column="0" Grid.Row="1" Margin="5"
38                     Stroke="{DynamicResource OwnBrush}"
39                     StrokeThickness="50">
40              <Rectangle.Resources>
41                  <LinearGradientBrush x:Key="OwnBrush">
42                      <GradientStop Color="Red" Offset="0"/>
43                      <GradientStop Color="DeepPink" Offset="0.2"/>
44                      <GradientStop Color="GreenYellow" Offset="0.4"/>
45                      <GradientStop Color="DarkCyan" Offset="0.6"/>
46                      <GradientStop Color="Aqua" Offset="0.8"/>
47                      <GradientStop Color="Black" Offset="1"/>
48                  </LinearGradientBrush>
49              </Rectangle.Resources>
50          </Rectangle>
51          <Button Grid.Row="1" Grid.Column="1" Margin="5"
52                  Background="{StaticResource BrushWithTitle}"
53                  FontSize="{StaticResource FontSize}"
54                  BorderBrush="{StaticResource SolidBrush}">
55              Another!
56          </Button>
57      </Grid>
58  </Window>
```

Рисунок 19.1 – Код XAML, использующий ресурсы

Обратите внимание на места определения ресурсов и синтаксические конструкции для их подключения.

Результат подобной разметки представлен на рис. 19.2.

Рисунок 19.2 – Снимок окна приложения для демонстрации использования ресурсов

19.2.2 Стили

Стили (style) представляют собой важный инструмент для организации и повторного использования вариантов форматирования. Вместо того чтобы заполнять XAML-файл повторяющимся кодом разметки для установки деталей вроде полей, отступов, цветов и шрифтов, можно просто создавать набор охватывающих все эти детали стилей, а затем применять эти стили по мере необходимости, устанавливая единственное свойство.

Стилем называется коллекция значений свойств, которые могут применяться к элементу. Система стилей WPF играет ту же роль, которую играет стандарт каскадных таблиц стилей (Cascading Style Sheet — CSS) в HTML-разметке.

Стили WPF поддерживают триггеры, которые позволяют изменять стиль элемента управления при изменении какого-то свойства.

Рассмотрим пример определения внешнего вида элемента с использованием ресурсов (рис. 19.3).

```
<Window x:Class="ComplexDataExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="MainWindow" Height="150" Width="400">
    <Window.Resources>
        <sys:Double x:Key="DefaultWidth">150</sys:Double>
        <sys:Double x:Key="DefaultHeght">50</sys:Double>
        <Thickness x:Key="StrokeSize">10</Thickness>
        <FontFamily x:Key="FF">Times New Roman</FontFamily>
        <LinearGradientBrush x:Key="DefaultBrush">
            <GradientStop Color="AliceBlue" Offset="0.2"/>
            <GradientStop Color="Aquamarine" Offset="0.7"/>
        </LinearGradientBrush>
    </Window.Resources>
```

```
16      <Grid>
17          <Grid.ColumnDefinitions>
18              <ColumnDefinition/>
19              <ColumnDefinition/>
20          </Grid.ColumnDefinitions>
21          <Button Grid.Column="0" FontFamily="{StaticResource FF}"
22                  Width="{StaticResource DefaultWidth}"
23                  Height="{StaticResource DefaultHeght}"
24                  BorderThickness="{StaticResource StrokeSize}"
25                  Background="{StaticResource DefaultBrush}">
26              Button One
27          </Button>
28          <Button Grid.Column="1">
29              Button Two
30          </Button>
31      </Grid>
32  </Window>
```

Рисунок 19.3 – Определение внешнего вида элементов управления с использованием ресурсов

Результат подобной разметки представлен на рис. 19.4.

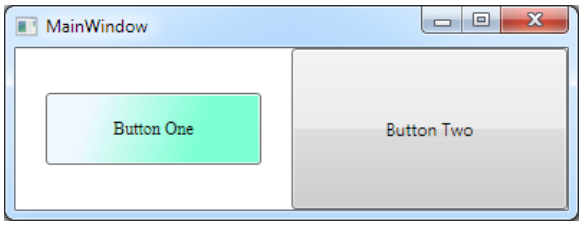


Рисунок 19.4 – Внешний вид элементов управления при использовании ресурсов

Для того, чтобы изменить внешний вид второй кнопки в соответствии с выбранным внешним видом, необходимо установить все свойства, примененные к первой кнопке (то есть повторить строки 21–25 листинга на рис. 19.3 применительно ко второй кнопке). Такой код XAML будет больше, обслуживать его будет сложнее. Еще сильнее усложнится код если в приложении будет множество элементов управления, оформленных в одном визуальном стиле. Для решения этой задачи необходимо применить стили WPF.

На рис. 19.5 представлен рефакторинг кода (из рис. 19.3) с использованием механизма стилей.

```
<Window x:Class="ComplexDataExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="MainWindow" Height="150" Width="400">
    <Window.Resources>
        <LinearGradientBrush x:Key="DefaultBrush">
            <GradientStop Color="AliceBlue" Offset="0.2"/>
            <GradientStop Color="Aquamarine" Offset="0.7"/>
        </LinearGradientBrush>
        <Style x:Key="DefaultStyle">
            <Setter Property="Control.Width" Value="150"/>
            <Setter Property="Control.Height" Value="50"/>
            <Setter Property="Control.BorderThickness" Value="10"/>
            <Setter Property="Control.FontFamily" Value="Times New Roman"/>
            <Setter Property="Control.Background" Value="{StaticResource DefaultBrush}"/>
        </Style>

    </Window.Resources>
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Button Grid.Column="0"
                Style="{StaticResource DefaultStyle}">
            Button One
        </Button>
        <Button Grid.Column="1"
                Style="{StaticResource DefaultStyle}">
            Button Two
        </Button>
    </Grid>
</Window>
```

Рисунок 19.5 – Код XAML с использованием стилей

Код существенно сократился. Стили также могут использовать ссылки на ресурсы. Внешний вид двух кнопок установлен всего одной строкой – установкой свойства Style.

Стиль (фактически, это класс System.Windows.Style) определяется в словаре ресурсов. Как и ресурс, стиль может быть определен у различных элементов управления (хотя чаще определяется у элементов управления верхнего уровня).

В таблице 19.1 представлены основные свойства класса Style.

Таблица 19.1 – Свойства класса Style

Свойство	Описание
Setters	Коллекция объектов Setter или EventSetter, которые устанавливают значения для свойств и присоединяют обработчики событий автоматически

Triggers	Коллекция объектов, унаследованных от класса TriggerBase, которые позволяют автоматически изменять настройки стиля. Настройки стиля могут модифицироваться, например, при изменении значения какого-то другого свойства или при поступлении какого-нибудь события
Resources	Коллекция ресурсов, которые должны использоваться со стилями.
BasedOn	Свойство, которое позволяет создавать более специализированный стиль, наследующий (и дополнительно переопределяющий) параметры другого стиля
TargetType	Свойство, которое идентифицирует тип элемента, к которому применяется данный стиль. Это свойство позволяет создавать объекты Setter, влияющие только на определенные элементы, а также объекты Setter, автоматически вступающие в силу для всех элементов подходящего типа

В объекте типа Style размещается коллекция Setters с объектами Setter, по одному для каждого свойства, которое подлежит установке. В каждом объекте Setter указывается имя свойства Property, на которое он влияет, и значение Value, которое он должен применять к этому свойству. Как и все ресурсы, объект стиля имеет ключевое имя, по которому его можно при необходимости извлекать из коллекции.

Если бы в примере на рис. 19.5 было много кнопок, то можно обойтись без установки Style для каждой кнопки. Для этого необходимо описать стиль следующим образом:

<Style x:Key="{x:Type Button}">

Стиль с таким ключом будет применен ко всем кнопкам данного окна. Можно также установить свойство TargetType:

<Style TargetType="Button">

19.2.3 Триггеры

С помощью триггеров можно автоматизировать процесс внесения простых изменений в стили, для которых требуется написание рутинной логики обработки событий. Например, можно обеспечить реакцию на изменение значения свойства и соответствующим образом автоматически подстроить стиль.

Триггеры связываются со стилями через коллекцию Style.Triggers. Каждый стиль может иметь любое количество триггеров, а каждый триггер является экземпляром класса, унаследованного от System.Windows.TriggerBase (таблица 19.2).

Таблица 19.2 – Классы, унаследованные от TriggerBase

Класс	Описание
Trigger	Простейшая форма триггера. Он следит за изменением в свойстве зависимости и затем использует средство установки для изменения стиля
MultiTrigger	Похож на Trigger, но поддерживает проверку множества условий. Этот триггер вступает в действие, только если удовлетворены все заданные условия
DataTrigger	Работает с привязкой данных. Он похож на Trigger, но следит за изменением в любых связанных данных
MultiDataTrigger	Объединяет множество триггеров данных
EventTrigger	Наиболее сложный триггер. Он применяет анимацию, когда возникает соответствующее событие

Продемонстрировать использование простого триггера можно с использованием примера (рис. 19.6), который дополняет стиль, представленный на рис. 19.5.

```
<Style x:Key="{x:Type Button}">
    <Style.Triggers>
        <Trigger Property="Control.IsMouseOver" Value="True">
            <Setter Property="Control.Foreground" Value="Red" />
            <Setter Property="Control.RenderTransform">
                <Setter.Value>
                    <RotateTransform Angle="10" CenterX="75" CenterY="25"></RotateTransform>
                </Setter.Value>
            </Setter>
        </Trigger>
    </Style.Triggers>
    <Setter Property="Control.Width" Value="150"/>
    <Setter Property="Control.Height" Value="50"/>
    <Setter Property="Control.BorderThickness" Value="10"/>
    <Setter Property="Control.FontFamily" Value="Times New Roman"/>
    <Setter Property="Control.Background" Value="{StaticResource DefaultBrush}"/>
</Style>
```

Рисунок 19.6 – Простой триггер

Из рис. 19.6 очевидно на какое событие среагирует триггер, а также как изменится внешний вид любой кнопки окна при наступлении условного события (или при срабатывании триггера). Свойства изменяются путем добавления объектов типа Setter в коллекцию триггера Trigger.Setters.

К одному элементу может быть применено любое количество триггеров (главное, чтобы отдельные триггеры не конфликтовали).

При использовании мульти триггера можно задать несколько условий срабатывания (рис. 19.7).

```
<Style x:Key="{x:Type Button}">
    <Style.Triggers>
        <MultiTrigger>
            <MultiTrigger.Conditions>
                <Condition Property="Button.IsKeyboardFocused" Value="true"/>
                <Condition Property="Button.IsMouseOver" Value="True"/>
            </MultiTrigger.Conditions>
            <MultiTrigger.Setters>
                <Setter Property="Button.Width" Value="160"/>
                <Setter Property="Button.Height" Value="55"/>
                <Setter Property="Button.FontSize" Value="26"/>
            </MultiTrigger.Setters>
        </MultiTrigger>
        <Trigger Property="Control.IsMouseOver" Value="True">
            <Setter Property="Control.Foreground" Value="Red" />
            <Setter Property="Control.RenderTransform">
                <Setter.Value>
                    <RotateTransform Angle="10" CenterX="75" CenterY="25"/>
                </Setter.Value>
            </Setter>
        </Trigger>
    </Style.Triggers>
    <Setter Property="Control.Width" Value="150"/>
    <Setter Property="Control.Height" Value="50"/>
    <Setter Property="Control.BorderThickness" Value="10"/>
    <Setter Property="Control.FontFamily" Value="Times New Roman"/>
    <Setter Property="Control.Background" Value="{StaticResource DefaultBrush}"/>
</Style>
```

Рисунок 19.7 –Триггер, срабатывающий при наступлении нескольких условий

В приведенном листинге MultiTrigger срабатывает только в случае, если наступает одновременно два условия. При этом определенный ранее триггер не отменен – он продолжает функционировать. Нутри класса MultiTrigger присутствуют две коллекции: Setters и Conditions.

19.3. Методика и порядок выполнения работы

19.3.1. Учебная задача

Рассмотрим пример выполнения задания с использованием механизмов ресурсов, стилей и триггеров.

Условие задания: разработайте приложение, демонстрирующее N графических примитивов (эллипсов, прямоугольников) со случайными значениями

положения, размера и стиля. Число N задается пользователем. Фигуры должны быть интерактивны, т.е. реагировать на какие-либо действия пользователя (любое событие).

Для решения данной задачи необходимо выполнить следующие действия:

1. Создайте проект WPF-приложения. Реализуйте XAML-разметку (рис. 19.8), основным назначением которой является задание стилей и поведения (триггеров) фигур.

```xaml
<Window x:Class="ComplexDataExample.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:sys="clr-namespace:System;assembly=mscorlib"
        Title="MainWindow" Height="400" Width="800">
    <Window.Resources>
        <Style x:Key="style1">
            <Setter Property="Shape.Fill">
                <Setter.Value>
                    <LinearGradientBrush>
                        <GradientStop Offset="0" Color="Aqua"/>
                        <GradientStop Offset="0.5" Color="WhiteSmoke"/>
                        <GradientStop Offset="1.5" Color="DarkGreen"/>
                    </LinearGradientBrush>
                </Setter.Value>
            </Setter>
            <Setter Property="Shape.Stroke" Value="Blue"/>
            <Setter Property="Shape.StrokeThickness" Value="5"/>
        </Style>
        <Style x:Key="style2">
            <Setter Property="Shape.Fill" Value="Blue"/>
            <Setter Property="Shape.Stroke" Value="BlueViolet"/>
            <Setter Property="Shape.StrokeThickness" Value="7"/>
        </Style>
        <Style x:Key="style3">
            <Setter Property="Shape.StrokeThickness" Value="3"/>
            <Setter Property="Shape.Fill" Value="LawnGreen"/>
            <Setter Property="Shape.Stroke" Value="ForestGreen"></Setter>
        </Style>
    </Window.Resources>
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition/>
                <ColumnDefinition Width="Auto"/>
            </Grid.ColumnDefinitions>
            <TextBox Name="FigureCount"
                     ToolTip="Inter figere Count"
                     Margin="5" Text="10"/>
            <Button Grid.Column="1" Click="Button_Click">Generate Shapes</Button>
        </Grid>
        <Canvas Name="MainCanvas" Grid.Row="1" Margin="5">

        </Canvas>
    </Grid>
</Window>
```

Рисунок 19.8 – Разметка XAML для каркаса приложения

2. Добавьте обработчик события для кнопки «Generate Shapes» (рис. 19.9)

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    int N = 10;
    try
    {
        N = Convert.ToInt32(FigureCount.Text);
    }
    catch (Exception ee)
    {
        this.Title = "Только целое число!";
    }

    GenerteShapes(N);
}
```

Рисунок 19.9 – Обработчик события нажатия кнопки «Generate Shapes»

3. Реализуйте функцию GenerateShapes(int N), которая случайным образом выберет фигуру, ее стиль, размер и положение, а также добавит фигуру в Canvas (рис. 19.10).

```
private void GenerteShapes(int N)
{
    Random rndShapeType = new Random(DateTime.Now.Millisecond);
    Random rndStyle = new Random(DateTime.Now.Second);
    Random rndPosition = new Random(DateTime.Now.Minute);
    Random rndSize = new Random(DateTime.Now.Minute);

    for (int i = 0; i < N; i++)
    {
        Shape currentShape;
        int shapeType = rndShapeType.Next(0, 2);
        if(shapeType == 0)
            currentShape = new Ellipse();
        else
            currentShape = new Rectangle();

        int shapeStyle = rndStyle.Next(0, 3) + 1;
        String styleName = "style" + shapeStyle.ToString();
        Style currentStyle = (Style) this.FindResource(styleName);
        currentShape.Style = currentStyle;

        currentShape.Width = rndSize.Next(10, 200);
        currentShape.Height = rndSize.Next(10, 100);

        MainCanvas.Children.Add(currentShape);
        Canvas.SetLeft(currentShape, rndPosition.Next(5, 750));
        Canvas.SetTop(currentShape, rndPosition.Next(5, 370));
    }
}
```

Рисунок 19.10 – Реализация функции GenerateShapes(int N)

Результат (скриншот) работающего приложения показан на рис 19.11.

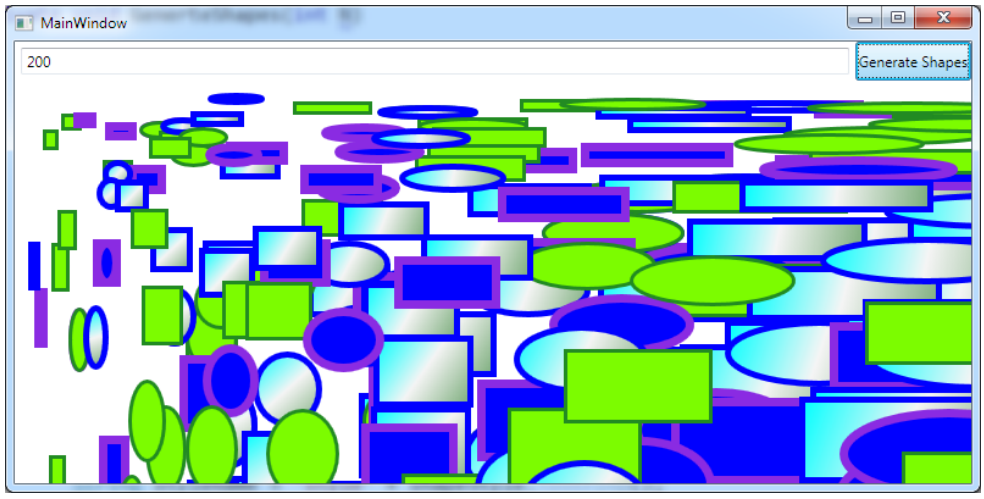


Рисунок 19.11 – Скриншот работающего приложения

На рис. 19.11 видно, что в процессе работы приложения используются оба типа фигур, а также задействованы все три стиля.

19.3.2. Индивидуальное задание

Разработайте собственные стили. По аналогии с задачей, рассмотренной в рамках лабораторной работы реализуйте генерацию любых стилизованных объектов (фигур, элементов управления). Необходимо реализовать не менее четырёх собственных стиля, а также продемострировать применение триггеров.

19.4. Вопросы для защиты работы

1. Что такое стиль?
2. Что такое ресурс?
3. Что такое триггер?
4. Опишите механизмы применения стилей.
5. Опишите механизмы применения триггеров.
6. Какие типы ресурсов вы знаете?
7. У каких элементов WPF может быть коллекция ресурсов?

ЛАБОРАТОРНАЯ РАБОТА 20. МЕХАНИЗМ ПРИВЯЗКИ WPF

20.1. Цель и содержание

Цель лабораторной научиться применять механизм привязки элементов управления.

Задачами лабораторной работы являются:

– изучить теоретические аспекты функционирования механизма привязки;

– научиться осуществлять привязку элементов управления в различных режимах.

20.2. Теоретическое обоснование

20.2.1. Основы привязки элементов управления

Простейший сценарий привязки данных подразумевает ситуацию, когда исходным объектом является элемент WPF, а исходным свойством – свойство зависимости. Свойства зависимости имеют встроенную поддержку уведомлений об изменениях. В результате, когда значение свойства зависимости изменяется в исходном объекте, привязанное свойство целевого объекта немедленно обновляется. Это именно то, что требуется, и происходит оно без необходимости построения любой дополнительной инфраструктуры.

Рассмотрим пример выполнения привязки. Окно содержит два элемента: Slider (ползунок) и TextBlock (текстовый блок) с единственной строкой текста. Перемещение ползунка вправо приводит к увеличению размера шрифта текста, а перемещение влево – к уменьшению размера шрифта. Такое поведение реализуется через Slider.ValueChanged.

Привязка WPF позволяет реализовать данную задачу проще. На рис. 20.1 представлен листинг XAML для окна WPF.

```
1  <Window x:Class="Task5.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="MainWindow" Height="200" Width="500">
5      <Grid>
6          <Grid.RowDefinitions>
7              <RowDefinition Height="Auto"/>
8              <RowDefinition/>
9          </Grid.RowDefinitions>
10         <Slider Name="sldSource" Value="14" Maximum="80" Minimum="1"
11                 TickFrequency="6" TickPlacement="Both"
12                 Grid.Row="0" Margin="5" AutoToolTipPlacement="TopLeft"/>
13         <TextBlock Name="txtTarget" Text="Образец текста" Grid.Row="1"
14                    TextAlignment="Center" VerticalAlignment="Center"
15                    FontSize="{Binding ElementName=sldSource, Path=Value}"
16                    Foreground="Red"/>
17     </Grid>
18 </Window>
```

Рисунок 20.1 – Листинг XAML главного окна приложения

В данном листинге на холсте главного окна размещен двухстрочный контейнер компоновки Grid. В него помещены два элемента управления:

– элемент Slider (с именем sldSource);

– элемент TextBlock (с именем txtTarget).

Кроме визуальных свойств (отступов, размеров и т.п.) в приведенном коде присутствует синтаксическая конструкция языка XAML, позволяющая связать размер шрифта элемента txtTarget к текущему значению ползунка sldSource.

В строке 15 листинга показано выражение привязки. Синтаксис данного выражения следующий:

ИмяЦелевогоСвойства =

"{

Binding ElementName = *ИмяЭлементаИсточника*,

Path = *ИмяСвойстваИсточника*

}"

Целевое свойство должно быть свойством зависимости.

Таким образом, свойство FontSize элемента txtTarget устанавливается не явно через указание численного значения, а с использованием привязки к свойству-источнику Value элемента-источника sldSource.

Рассмотренную привязку можно представить графически следующим образом:

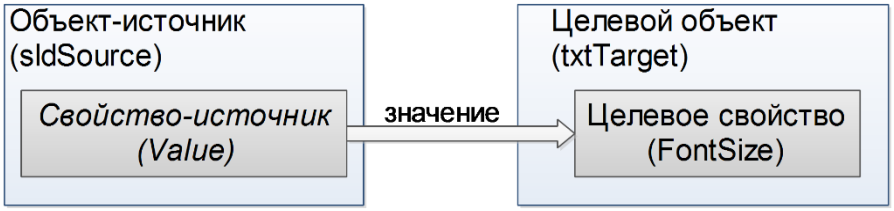


Рисунок 20.2 – Привязка целевого свойства к свойству-источнику значения, содержащемуся в элементе-источнике

Если во время привязки возникла ошибка (например, вместо свойства Value указано несуществующее свойство) – обрабатываться она не будет. WPF не генерирует исключения для уведомления о проблемах привязки данных. Во время отладки приложения эта информация появляется в выходном окне Visual Studio.

На рис. 20.3 представлен внешний вид окна работающего приложения. В результате включения привязки в код XAML изменения размера шрифта элемента достигнуты без использования кода на языке C#.

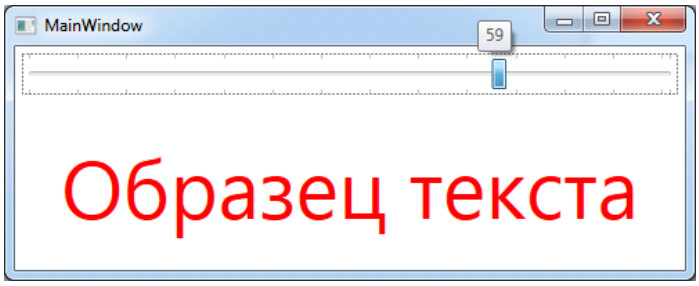


Рисунок 20.3 – Результат применения привязки

20.2.2. Режимы привязки

Особенность привязки данных заключается в том, что целевое свойство обновляется автоматически, независимо от того, как модифицируется источник.

Усложним ранее рассмотренный пример: в главное окно добавлены кнопки «Нормальный размер», «Крупный размер». Данные кнопки устанавливают по нажатию размер шрифта элемента txtTarget в значения 20 и 60 соответственно.

На рис. 20.4 приведен листинг XAML полученного окна.

```
<Window x:Class="Task5.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="500">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
        <Slider Name="sldSource" Value="14" Maximum="80" Minimum="1"
                TickFrequency="6" TickPlacement="Both"
                Grid.Row="0" Margin="5" AutoToolTipPlacement="TopLeft"/>
        <TextBlock Name="txtTarget" Text="Образец текста" Grid.Row="1"
                   TextAlignment="Center" VerticalAlignment="Center"
                   FontSize="{Binding ElementName=sldSource, Path=Value}"
                   Foreground="Red"/>
        <StackPanel Orientation="Horizontal" Grid.Row="2" HorizontalAlignment="Center">
            <Button Name="btnNormal" Content="Нормальный размер" Margin="5" Width="150"
                    Click="btnNormal_OnClick"/>
            <Button Name="btnLarge" Content="Крупный размер" Margin="5" Width="150"
                    Click="btnLarge_Click"/>
        </StackPanel>
    </Grid>
</Window>
```

Рисунок 20.4 – Окно приложения с добавленными элементами управления

Для добавленных кнопок созданы обработчики нажатия. На рис. 20.5 представлен листинг обработчика нажатия кнопки «Нормальный размер». На рис. 20.6 – кнопки «Крупный размер».

```
private void btnNormal_OnClick(object sender, RoutedEventArgs e)
{
    sldSource.Value = 30;
}
```

Рисунок 20.5 – Обработчик нажатия кнопки «Нормальный размер», устанавливающий значение Value ползунка sldSource

```
private void btnLarge_Click(object sender, RoutedEventArgs e)
{
    txtTarget.FontSize = 60;
}
```

Рисунок 20.6 – Обработчик нажатия кнопки «Крупный размер», устанавливающий значение FontSize элемента txtTarget.

Обработчик на рис. 20.5 установит положение ползунка sldSource в положение 30. Так как значение Value ползунка изменится, то механизм привязки обновит значение размера шрифта элемента txtTarget автоматически.

Обработчик на рис. 20.6 вместо выражения привязки (рис. 20.4 строка 16) запишет числовое значение 60. Размер шрифта будет задан, но привязка будет аннулирована и дальнейшее перемещение ползунка пользователем или нажатием кнопки «Нормальный размер» не будет приводить к изменению размера шрифта TextBlock.

Такое поведение обусловлено режимом привязки. Привязка в рассмотренном примере – односторонняя (рис. 20.2). При изменении значения свойства-источника значение целевого свойства меняется, но обратное утверждение не действительно. В WPF существуют различные варианты привязок (таблица 20.1).

Устранить недостаток, проявленный в примере можно модифицировав выражение привязки следующим образом (рис. 20.7):

```
14        <TextBlock Name="txtTarget" Text="Образец текста" Grid.Row="1"
15                   TextAlignment="Center" VerticalAlignment="Center"
16                   FontSize="{Binding ElementName=sldSource, Path=Value, Mode=TwoWay}"
17                   Foreground="Red"/>
```

Рисунок 20.7 – Явное указание режима привязки.

В качестве значения режима привязки Mode указывается значение перечисления BindingMode. После указанной на рис. 20.7 модификации все кнопки будут работать в окне, не разрушая привязку.

Таблица 20.1 – Значения перечисления BindingMode.

Значение	Описание
OneWay	Целевое свойство обновляется при изменениях исходного свойства.
TwoWay	Целевое свойство обновляется при изменениях исходного свойства, а исходное свойство обновляется при изменении целевого свойства.
OneTime	Целевое свойство устанавливается изначально на основе значения исходного свойства. Однако с этого момента изменения игнорируются (если только привязка не устанавливается на совершенно другой объект или не вызывается BondingExpression.UpdateTarget()). Обычно этот режим используется для сокращения накладных расходов, если известно, что целевое свойство не изменится.

OneWayToSource	Подобно OneWay, но действует в обратном направлении. Исходное свойство обновляется, когда изменяется целевое свойство (что может показаться несколько странным), но целевое свойство никогда не обновляется.
Default	Этот тип привязки зависит от целевого свойства. Это либо TwoWay (для устанавливаемых пользователем свойств, таких как TextBox.Text), либо OneWay (для всего остального). Все привязки используют данный подход, если только не указано иное.

Таким образом, можно применять синтаксис привязки элементов управления следующего вида:

ИмяЦелевогоСвойства =

"{

Binding ElementName = *ИмяЭлементаИсточника*,

Path = *ИмяСвойстваИсточника*,

Mode = *РежимПривязки*

}"

На рис. 20.8 представлена схема взаимодействия компонентов привязки.

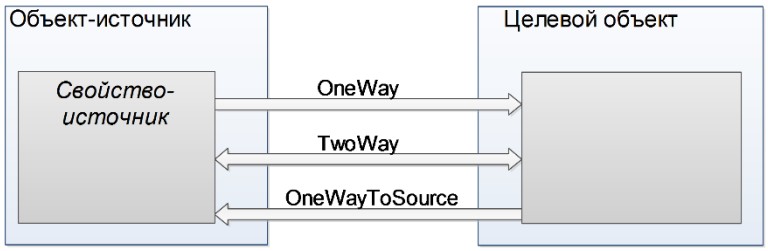


Рисунок 20.8 – Варианты привязок

Привязку можно реализовать в коде C#. На рис. 20.9 показан вариант реализации привязки на языке C#, синтаксис которой представлен на рис. 20.7. Оба варианта (с использованием XAML или C#) реализуют один и тот же механизм.

```
public MainWindow()
{
    InitializeComponent();
    Binding b = new Binding();
    b.Source = sldSource;
    b.Path = new PropertyPath("Value");
    b.Mode = BindingMode.TwoWay;
    txtTarget.SetBinding(TextBlock.FontSizeProperty, b);
}
```

Рисунок 20.9 – Реализация привязки на языке C#

Существуют методы для удаления привязок. Метод ClearBinding() принимает ссылку на свойство зависимости, которое имеет привязку, подлежащую удалению, а метод ClearAllBindings() удаляет все привязки данных элемента (рис. 20.10).

```
private void MainWindow_OnUnloaded(object sender, RoutedEventArgs e)
{
    BindingOperations.ClearAllBindings(txtTarget);
    // или
    BindingOperations.ClearBinding(txtTarget, TextBlock.FontSizeProperty);
}
```

Рисунок 20.10 – Удаление привязки в коде C#

Привязка чаще реализуется в коде XAML, но использование реализации на языке C# эффективно в случаях:

1. Создание динамических привязок. Если необходимо тонко настраивать привязку на основе другой информации времени выполнения или создавать разные привязки в зависимости от обстоятельств, имеет смысл делать это в коде. (В качестве альтернативы можно было бы определить все необходимые привязки в коллекции Resources окна и просто добавить код, который вызывает SetBinding() с соответствующим объектом привязки.)

2. Удаление привязки. Чтобы удалить привязку и получить возможность установки свойства обычным образом, понадобится помощь метода ClearBinding() или ClearAllBindings(). Недостаточно просто присвоить новое значение свойству: в случае использования двунаправленной привязки установленное значение распространится на привязанный объект, и оба свойства останутся синхронизированными.

20.2.3. Множественные привязки

Программист может осуществлять не только одиночные привязки (один целевой объект – один источник), разработчикам WPF доступны множественные привязки (один целевой элемент – множество источников). На рис. 20.11 показан вариант кода XAML, демонстрирующий привязку различных свойств элемента txtTarget к различным свойствам-источникам.

```
1  <Window x:Class="Task5.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          Title="MainWindow" Height="400" Width="500" Unloaded="MainWindow_OnUnloaded">
5      <Grid>
6          <Grid.RowDefinitions>
7              <RowDefinition Height="Auto"/>
8              <RowDefinition/>
9              <RowDefinition Height="Auto"/>
10             <RowDefinition Height="Auto"/>
11         </Grid.RowDefinitions>
12         <Slider Name="sldSource" Value="14" Maximum="80" Minimum="1"
13                 TickFrequency="6" TickPlacement="Both"
14                 Grid.Row="0" Margin="5" AutoToolTipPlacement="TopLeft"/>
15         <DockPanel Grid.Row="1">
16             <ListBox Name="listColor" DockPanel.Dock="Left" Width="150" SelectedIndex="0">
17                 <ListBoxItem Content="Красный" Tag="Red"/>
18                 <ListBoxItem Content="Зеленый" Tag="Green"/>
19                 <ListBoxItem Content="Сисний" Tag="Blue"/>
20                 <ListBoxItem Content="Другой" Tag="#58aa45"/>
21                 <ListBoxItem Content="Розовый" Tag="Pink"/>
22             </ListBox>
23             <!--Целевой элемент привязки: -->
24             <TextBlock Name="txtTarget" DockPanel.Dock="Right"
25                    TextAlignment="Center" VerticalAlignment="Center"
26                    FontSize="{Binding Value, ElementName=sldSource, Mode=TwoWay}"
27                    Foreground="{Binding ElementName=listColor, Path=SelectedItem.Tag}"
28                    Text="{Binding Text, ElementName=txtTextSource}" />
29         </DockPanel>
30         <TextBox Name="txtTextSource" Text="Введите текст" Grid.Row="2"/>
31         <StackPanel Orientation="Horizontal" Grid.Row="3" HorizontalAlignment="Center">
32             <Button Name="btnNormal" Content="Нормальный размер" Margin="5" Width="150"
33                     Click="btnNormal_OnClick"/>
34             <Button Name="btnLarge" Content="Крупный размер" Margin="5" Width="150"
35                     Click="btnLarge_Click"/>
36         </StackPanel>
37     </Grid>
38 </Window>
```

Рисунок 20.11 – Применение множественной привязки

Обратите внимание на строки 24-28, в которых реализуются привязки, листинга на рис. 20.11.

На рис. 20.12 представлен внешний вид полученного окна приложения.

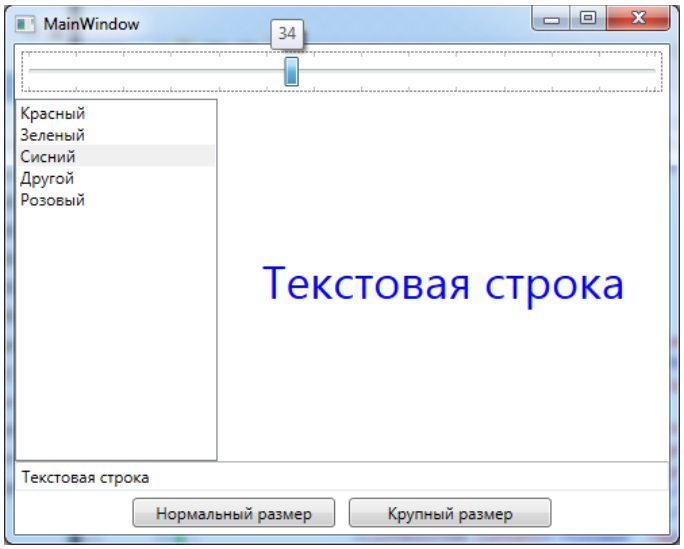


Рисунок 20.12 – Применение множественной привязки

20.3. Методика и порядок выполнения работы

Для закрепления материала о привязках элементов управления и получения практических навыков программирования приложений с использованием привязки к элементам управления, реализуйте приложение, удовлетворяющее следующим требованиям:

– в программе должно быть реализовано не менее 5 привязок к элементам управления;

– каждая привязка должна получать данные из свойства-источника, причем все свойства-источники должны быть различного типа: строковые, логические, целочисленные, вещественные, цвет, имя шрифта и т.д.

– приложение должно демонстрировать различные режимы привязок;

– одна из привязок должна создаваться динамически.

20.4. Вопросы для защиты работы

1. Что такое привязка WPF?
2. Поясните синтаксис привязки. Какие ключевые слова и для каких целей используются при оформлении привязки с помощью XAML?

3. Какие режимы привязки существуют? Поясните назначение всех режимов привязки.

4. Поясните схему привязки элементов управления. Назовите основные элементы данной схемы.

5. Что означают понятия «целевой элемент», «элемент-источник привязки», «целевое свойство», «свойство-источник»?

6. Какое ограничение на источник привязки существует в WPF?

ЛАБОРАТОРНАЯ РАБОТА 21. ИСТОЧНИКИ ПРИВЯЗКИ ПРОИЗВОЛЬНОГО ТИПА

21.1. Цель и содержание

Цель лабораторной работы: научиться использовать механизм привязки с объектами-источниками пользовательского типа.

Задачами лабораторной работы являются:

– научиться осуществлять привязку элементов к источнику, который является объектом пользовательского класса;

– научиться разрабатывать многослойные (multi-tier) приложения WPF.

21.2. Теоретическое обоснование

21.2.1. Многомодульные приложения

При разработке реальных информационных программных систем очень редко используются одномодульные приложения (в виде единственного исполняемого файла .exe). Оптимальным вариантом является разбиение приложения на отдельные уровни, каждый из которых представляется отдельной сборкой (модулем). На рис. 21.1 представлена стандартная схема многоуровнего приложения (multi tier application), которая включает: уровень доступа к данным (Data tier), уровень бизнес-логики (Logic tier) и уровень презентаций (Presentation Tier).

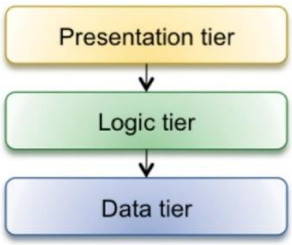


Рисунок 21.1 – Multy tier Application

Такое разделение на отдельные слои позволяет осуществлять многопользовательскую параллельную разработку сложного приложения.

21.2.2. Привязка к произвольным объектам

При реализации механизма привязки необходимо помнить, что целевое свойство должно быть свойством зависимости. Но на объект-источник ограничения отсутствуют. Это означает, что программист может привязать к свойству зависимости любое другое свойство любого другого объекта, независимо от природы его создания.

В предыдущих лабораторных работах рассматривались простые механизмы привязки, основанные на отношении между двумя элементами управления WPF, в котором один элемент – целевой объект привязки, другой – источник привязки.

Но гораздо чаще программисту приходится извлекать данные не из визуального элемента управления, а из объекта определенного бизнес-класса (то есть класса, описанного программистом и предоставляющего необходимые данные, а также реализующего определенный набор действий). В данном механизме необходимо учитывать ограничение – данные для привязки должны находиться в общедоступных свойствах.

При привязке к объекту, который не является элементом управления WPF, пользоваться свойством привязки ElementName не представляется возможным. Вместо него программист должен использовать одно из следующих свойств:

1. Source. Ссылка, указывающая на исходный объект; другими словами, это объект, поставляющий данные. Существует несколько подходов: извлечение объекта источника из ресурса, программная генерация объекта источника или получение источника от поставщика данных.

2. RelativeSource. Указывает на исходный объект, основываясь на отношениях между исходным и целевым объектом. То есть, можно привязать целевой объект к самому себе (сам целевой объект используется в качестве объекта-источника), к родительскому элементу или к определенному элементу, отстоящему от целевого на N уровней в дереве элементов.

3. DataContext. Если источник не был указан с помощью свойства Source или RelativeSource, то среда WPF производит поиск в дереве элементов, начиная с текущего элемента. Она проверяет свойство DataContext каждого элемента и использует первый из них, который не равен null. Свойство DataContext исключительно полезно, когда нужно привязать несколько свойств одного объекта к разным элементам, потому что можно установить свойство DataContext высокоуровневого объекта контейнера, вместо его установки непосредственно на целевой элемент.

В рамках данной лабораторной работы будет изучен механизм использования свойства DataContext, как наиболее простого и широко применяемого.

21.3. Методика и порядок выполнения работы

21.3.1. Учебное задание

Необходимо написать приложение для обработки информации о товарах. О каждом товаре необходимо обрабатывать информацию:

– код;

– наименование;

– цена;

– количество;

– описание.

Для освоения методики привязки к объектам произвольного класса, а также получения навыков разработки многомодульных приложений, необходимо реализовать три этапа:

1. Разработать слой доступа к данным.

2. Разработать слой бизнес-логики.

3. Разработать слой презентаций.

Каждый слой должен быть реализовываться в отдельном проекте.

Для решения задачи выполните следующую последовательность действий:

1. В окне Visual Studio создайте пустое решение с именем «Shop» (не проект!). Для этого выберите тип проекта «Visual Studio Solution» (Blank Solution) (рис. 21.1).

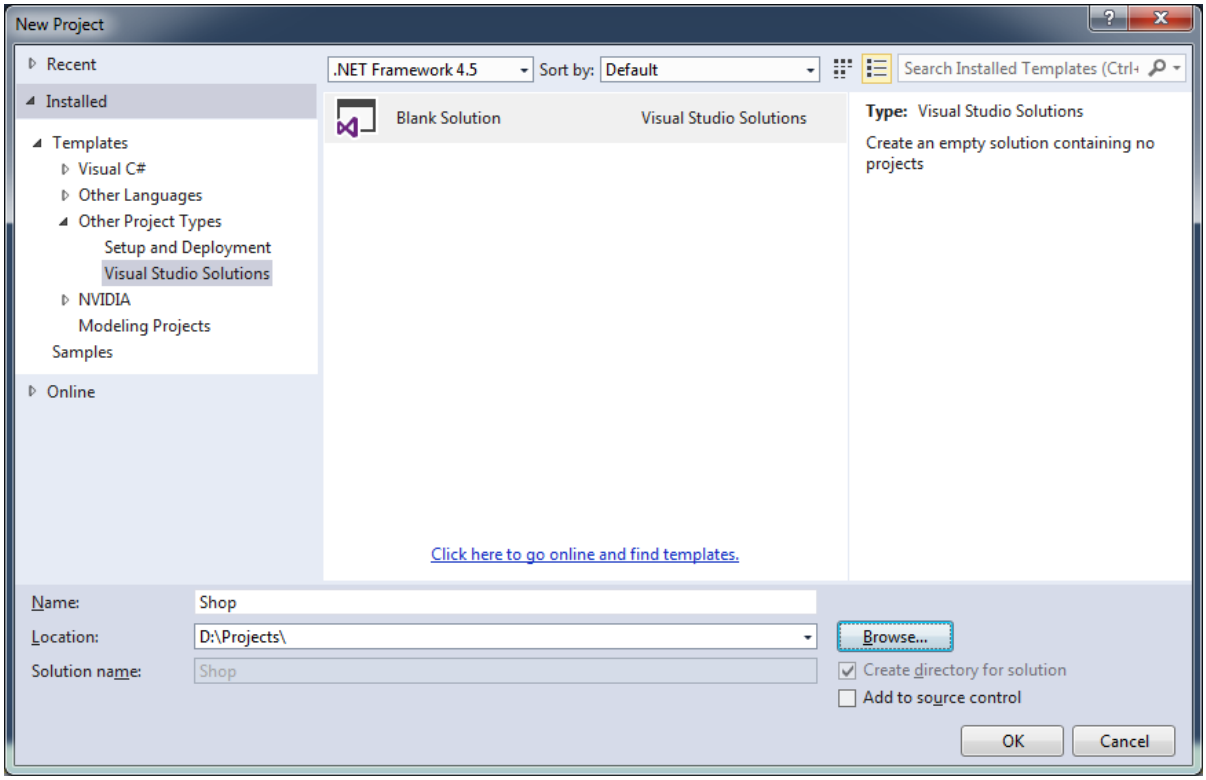


Рисунок 21.1 – Создание пустого решения Visual Studio

2. В обозревателе проектов появится новое пустое решение с именем «Shop» (рисунок 21.2).

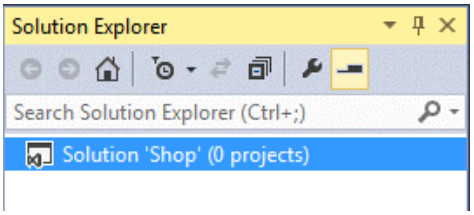


Рисунок 21.2 – Вид обозревателя проекто после создания пустого решения «Shop»

3. Добавьте в решение «Shop» три проекта (таблица 21.1).

Таблица 21.1 – Проекты, добавляемые в решение «Shop»

Имя проекта	Тип проекта	Назначение
DataTier	C# → Windows → Class Library	Библиотека классов для работы с данными. Данный проект представляет уровень доступа к данным в трехуровневой архитектуре.

LogicTier	C# → Windows → Class Library	Библиотека классов, представляющая все необходимые данные и методы для пользовательского интерфейса. Проект представляет уровень бизнес-логики в трехуровневой архитектуре.
PresentationTier	C# → Windows → WPF Application	Проект оконного приложения Windows Presentation Foundation. Данный проект представляет собой уровенб презентаций и содержит интерфейс программы.

На рисунке 21.3 показан внешний вид обозревателя проектов после добавления новых проектов.

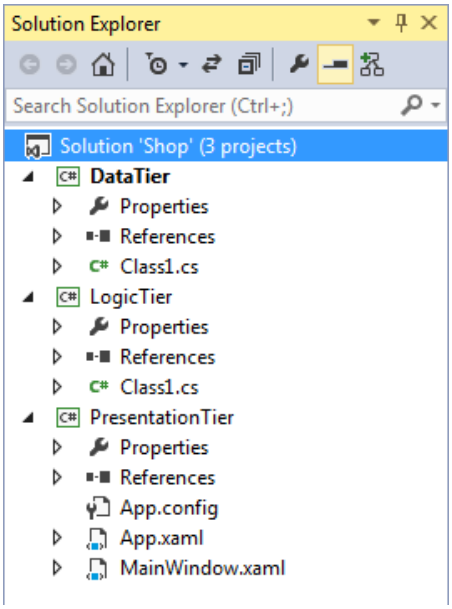


Рисунок 21.3 – Вид обозревателя проектов после добавления отдельных проектов

Наиболее простой способ решения данной задачи – реализовать уровни (tiers) в следующей последовательности с целью избежания архитектурных ошибок разработки:

1) Программирование уровня DataTier. Легче всего реализовать данный уровень первым, т.к. он в большей степени зависит от данных (в данном случае от предметной области из условия задания), чем от Windows-приложения.

2) Программирование уровня PresentationTier. Реализация данного уровня не требует знаний в области программирования, т.к. презентация WPF – это внешний

вид оконного приложения Windows. Данный уровень предполагает, в основном, работу дизайнера.

3) Программирование уровня LogicTier. Цель данного слоя приложения – связать уровни DataTier и PresentationTier.

4. Проект DataTier будет содержать классы, необходимые для представления данных (таблица 21.2).

Таблица 21.2 – Классы проекта «DataTier»

Имя проекта	Назначение
Товар	Представляет отдельный товар; содержит свойства, определяемые заданием
ВсеТовары	Представляет список товаров; содержит методы сохранения и загрузки товаров на носителе

На рис. 21.4 представлен листинг класса «Товар».

```
7  namespace DataTier
8  {
       11 references
9      public class Товар
10     {
           7 references
11         public String Код { get; set; }
           7 references
12         public String Наименование { get; set; }
           8 references
13         public float Цена { get; set; }
           7 references
14         public int Количество { get; set; }
           6 references
15         public String Описание { get; set; }
16     }
17 }
```

Рисунок 21.4 – Определение класса «Товар»

На рис. 21.5 представлен листинг класса «ВсеТовары». Класс «ВсеТовары» содержит два метода:

– метод ПолучитьВсеТовары(), возвращающий список товаров;

– метод СохранитьВсеТовары() для сохранения списка товаров на внешнем носителе.

```
7  namespace DataTier
8  {
       1 reference
9      public class ВсеТовары
10     {
           1 reference
11         public static List<Товар> ПолучитьВсеТовары()
12         {
13             List<Товар> list = new List<Товар>();
14
15             list.Add(
16                 new Товар()
17                 {
18                     Код = "001", Наименование = "ОС Windows 8", Количество = 10, Цена = 40.99f,
19                     Описание = "Современная операционная система. Версия 8"
20                 });
21
22             list.Add(
23                 new Товар()
24                 {
25                     Код = "002", Наименование = "3D Max", Количество = 2, Цена = 500.99f,
26                     Описание = "Система визуализации и рендеринга от Autodesk Corp."
27                 });
28
29             list.Add(
30                 new Товар()
31                 {
32                     Код = "003", Наименование = "Total Commander 1.00",
33                     Количество = 100, Цена = 0.5f, Описание = " - "
34                 });
35
36             list.Add(
37                 new Товар()
38                 {
39                     Код = "004-001", Наименование = "MS SQL Server",
40                     Количество = 5, Цена = 150.00f, Описание = "СУБД от Microsoft Corp."
41                 });
42
43             return list;
44         }
45
           0 references
46         public static void СохранитьВсеТовары(List<Товар> товары)
47         {
48
49         }
50     }
51 }
```

Рисунок 21.5 – Определение класса «ВсеТовары»

Классы, представленные на рис. 21.4 и 21.5, являются минимальной абстракцией объектов предметной области. Метод ПолучитьВсеТовары() должен считывать список товаров из файла или получать его из базы данных, но в данном примере осуществляется построение списка в коде. Метод СохранитьвсеТовары() оставим пустым. Все операции сохранения и загрузки данных, взаимодействие с внешними носителями, осуществляются через слой DataTier.

5. Уровни DataTier и LogicTier реализованы в виде библиотек, в которых отсутствует метод Main(). То есть данные проекты не предполагают запуска.

Библиотеки LogicTier и DataTier после выполнения построения будут оформлены в сборки *.dll. На данном шаге решения задачи уровень DataTier реализован, откомпилируйте данный проект и исправьте ошибки, если они есть.

6. Для взаимодействия с пользователем необходимо реализовать уровень презентаций. Проект «PresentationTier» будет содержать главную форму оконного приложения.

На рис. 21.6 представлен внешний вид формы MainWindow.xaml в режиме проектирования.

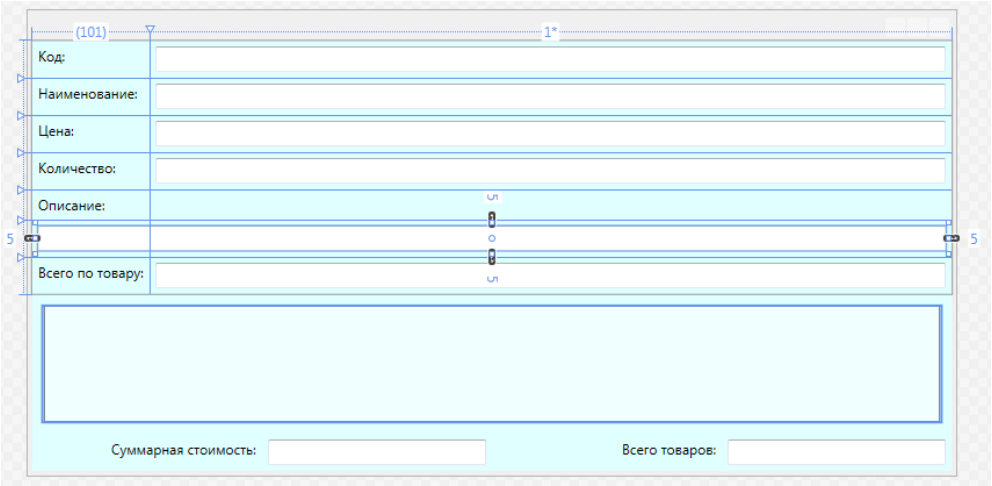


Рисунок 21.6 – Дизайн MainWindow.xaml

Рисунок 21.7 иерархию элементов управления в главном окне.

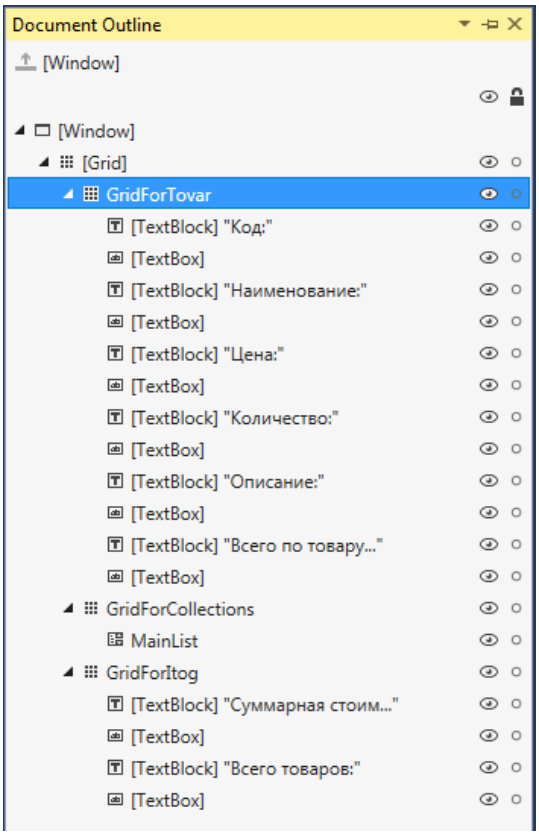


Рисунок 21.7 – Иерархическая структура элементов в окне MainWindow.xaml

Из рисунка 21.7 видно, что окно Window является корневым элементом в визуальном дереве. На втором уровне иерархии расположен элемент Grid без установленного свойства Name, который вмещает в себя все остальные элементы. На третьем уровне расположены три элемента типа Grid: GridForTovar (содержит поля для отображения информации о конкретном товаре), GridForCollections (содержит поле MainList типа ListBox для отображения списка товаров), GridForItog (область для вывода аггрегирующих показателей).

На рис. 21.8 представлен код XAML для представленного на рисунках 21.6, 21.7 окна.

```
<Window x:Class="PresentationTier.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="{Binding НаименованиеМагазина}" Height="400" Width="800"
        WindowStartupLocation="CenterScreen" Background="LightCyan">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid Grid.Row="0" Name="GridForTovar"
              DataContext="{Binding ElementName=MainList, Path=SelectedItem}">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"></ColumnDefinition>
                <ColumnDefinition></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <TextBlock Margin="5" Grid.Row="0" Text="Код:"/>
            <TextBox Margin="5" Grid.Row="0" Grid.Column="1"
                     Text="{Binding КодТовара}"/>
            <TextBlock Margin="5" Grid.Row="1" Text="Наименование:"/>
            <TextBox Margin="5" Grid.Row="1" Grid.Column="1"
                     Text="{Binding НаименованиеТовара}"/>
            <TextBlock Margin="5" Grid.Row="2" Text="Цена:"/>
            <TextBox Margin="5" Grid.Row="2" Grid.Column="1"
                     Text="{Binding ЦенаТовара}"/>
            <TextBlock Margin="5" Grid.Row="3" Text="Количество:"/>
            <TextBox Margin="5" Grid.Row="3" Grid.Column="1"
                     Text="{Binding КоличествоТовара}"/>
            <TextBlock Margin="5" Grid.Row="4" Text="Описание:"/>
            <TextBox Margin="5" Grid.Row="5"
                     Grid.Column="0" Grid.ColumnSpan="2"
                     Text="{Binding ОписаниеТовара}"/>
            <TextBlock Margin="5" Grid.Row="6" Text="Всего по товару:"/>
            <TextBox Margin="5" Grid.Row="6" Grid.Column="1"
                     Text="{Binding СуммарнаяСтоимостьПозиции, Mode=OneWay}"/>
        </Grid>
```

```
47          <Grid Grid.Row="1" Name="GridForCollections">
48              <ListBox Name="MainList" ItemsSource="{Binding СписокТоваров, Mode=OneWay}"
49                       DisplayMemberPath="ПредставлениеТовара" Background="Azure"
50                       Margin="10"/>
51          </Grid>
52          <Grid Grid.Row="2" Name="GridForItog">
53              <Grid.ColumnDefinitions>
54                  <ColumnDefinition/>
55                  <ColumnDefinition/>
56                  <ColumnDefinition/>
57                  <ColumnDefinition/>
58              </Grid.ColumnDefinitions>
59              <TextBlock Margin="5" Text="Суммарная стоимость:" Grid.Column="0"
60                         HorizontalAlignment="Right"/>
61              <TextBox Margin="5" Grid.Column="1"
62                       Text="{Binding Path=СуммарнаяСтоимость, Mode=OneWay}" />
63              <TextBlock Margin="5" Text="Всего товаров:" Grid.Column="2"
64                         HorizontalAlignment="Right"/>
65              <TextBox Margin="5" Grid.Column="3"
66                       Text="{Binding Path=СуммарноеКоличество, Mode=OneWay}"/>
67          </Grid>
68      </Grid>
69  </Window>
```

Рисунок 21.8 – Код XAML главного окна

Как и в случае использования привязки к элементам управления в данном коде также присутствуют привязки. В некоторых элементах управления устанавливаются различные свойства с использованием {Binding …}. В связи с этим необходимо дать некоторые пояснения по коду XAML, представленному на рисунке 21.8:

Таблица 21.3 – Особенности кода XAML, связанные с выполнением привязки данных

Номер строки	Исходный код и пояснения
4	Title="{Binding НаименованиеМагазина}" Производится установка заголовка окна. Значение свойства Title привязывается к полю данных НаименованиеМагазина. Такого поля на данном этапе проектирования приложения не существует, поля данных НаименованиеМагазина будет реализовано позже.
29	Text="{Binding КодТовара}" Значение свойства Text элемента управления TextBox привязывается к полю данных КодТовара. Такого поля на данном этапе проектирования приложения не существует, поля данных КодТовара будет реализовано позже.

32, 35, 38, 42	Значение свойства Text элемента управления TextBox привязывается к полю данных, значение которого будет отображено в данном TextBox. Аналогично с механизмом привязки, продемонстрированным в строке 29.
45	Text="{Binding СуммарнаяСтоимостьПозиции, Mode=OneWay} Значение свойства Text элемента управления TextBox привязывается к полю данных СуммарнаяСтоимостьПозиции в режиме OneWay. Аналогичен предыдущим привязкам, но не предполпгает обновления источника привязки при изменении целевого свойства.
48, 49	ItemsSource="{Binding СписокТоваров, Mode=OneWay}" DisplayMemberPath="ПредставлениеТовара" Элемент ListBox отображает не единственное значение, а целый список элементов. Поэтому для привязки к источнику этих элементов применяется свойство ItemsSource. Очевидно, что если нам необходимо выводить список товаров, то нужно указать, какое именно поле будет выводиться в этом списке пользователю; для этого устанавливает свойство DisplayMemberPath. Поле данных «ПредставлениеТовара» на данном этапе не реализовано.
62, 66	Text="{Binding Path=СуммарнаяСтоимость, Mode=OneWay}" Text="{Binding Path=СуммарноеКоличество, Mode=OneWay}" Механизм привязки аналогичен продимонстрированому в строке 45.
13	DataContext="{Binding ElementName=MainList, Path=SelectedItem}" Фактически, выполняется привязка одного элемента управления к другому. Производится привязка свойства DataContext элемента Grid к элементу MainList типа ListBox. Но не ко всему списку, а только к объекту, заданному свойством Path, то есть к свойству ListBox.SelectedItem (текущий выбранный элемент).

После верстки главного окна с использование инструментария XAML приложение WPF может быть запущено. Внешний вид созданного приложения представлен на рис. 21.9.

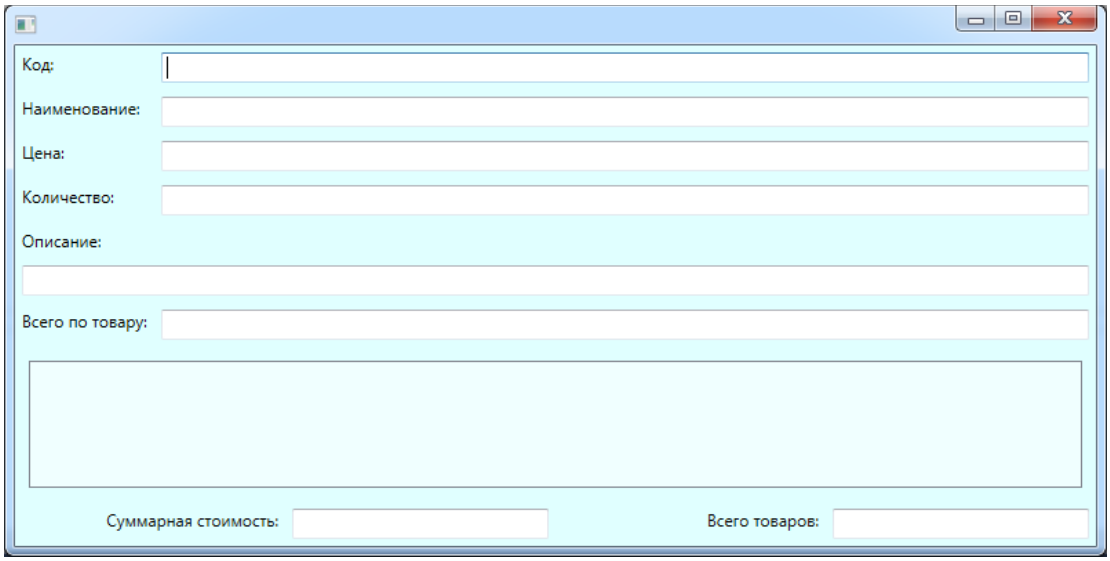


Рисунок 21.9 – Главное окно приложения в режиме выполнения

На данном этапе приложение функционирует, но нет связи с данными. Несмотря на то, что осуществлялась привязка к несуществующим данным, приложение все равно работает. Теперь в нашем приложении две сборки: DataTier.dll и PresentationTier.exe. Реализуем связующее звено – LogicTier.dll.

7. В проекте LogicTier создадим классы, представленные в таблице 21.4.

Таблица 21.4 – Особенности кода XAML, связанные с выполнением привязки данных

Имя класса	Описание
ТоварнаяПозиция	Это класс, который «оборачивает» класс «Товар» из проекта DataTier. Класс «ТоварнаяПозиция» дополняет «сырой» класс данных всем необходимым для вышестоящих уровней приложения.
Магазин	Данный класс «оборачивает» класс «ВсеТовары» из проекта DataTier.

На рисунке 21.10 представлен листинг класса «Товарная позиция».

```
8  namespace LogicTier
9  {
10     public class ТоварнаяПозиция
11     {
12         private Товар _товар;
13
14         public ТоварнаяПозиция(Товар р)
15         {
16             _товар = р;
17         }
18         public String КодТовара
19         {
20             get { return _товар.Код; }
21             set { _товар.Код = value; }
22         }
23
24         public String НаименованиеТовара
25         {
26             get { return _товар.Наименование; }
27             set { _товар.Наименование = value; }
28         }
29
30         public float ЦенаТовара
31         {
32             get { return _товар.Цена; }
33             set { _товар.Цена = value; }
34         }
35
36         public int КоличествоТовара
37         {
38             get { return _товар.Количество; }
39             set { _товар.Количество = value; }
40         }
41
42         public String ОписаниеТовара
43         {
44             get { return _товар.Описание; }
45             set { _товар.Описание = value; }
46         }
47
48         public float СуммарнаяСтоимостьПозиции
49         {
50             get { return _товар.Цена * _товар.Количество; }
51         }
52
53         public String ПредставлениеТовара
54         {
55             get { return _товар.Код + " : " + _товар.Наименование
56                 + " (" + _товар.Цена.ToString("C") + ")"; }
57         }
58     }
59 }
```

Рисунок 21.10 – Код класса «ТоварнаяПозиция»

В представленном коде производится связь между уровнем презентаций и уровнем данных: для каждой привязки к единичному товару в коде XAML

(таблица 21.3) в данном классе создается одноименное поле. Если привязка производилась в режиме OneWay, поле может быть объявлено только для чтения (например, поле ПредставлениеТовара).

В свою очередь, каждое поле из класса «ТоварнаяПозиция» является оберткой над соответствующим полем класса «Товар» (т.е. представляет это поле или вычисляется на основе нескольких полей).

Листинг класса «Магазин» представлен на рисунок 21.11.

```
8  namespace LogicTier
9  {
10     public class Магазин
11     {
12         private List<ТоварнаяПозиция> _товары = new List<ТоварнаяПозиция>();
13
14         public Магазин()
15         {
16             List<Товар> tmp = ВсеТовары.ПолучитьВсеТовары();
17             foreach (var t in tmp)
18             {
19                 _товары.Add(new ТоварнаяПозиция(t));
20             }
21         }
22
23         public List<ТоварнаяПозиция> СписокТоваров
24         {
25             get { return _товары; }
26         }
27
28         public String НаименованиеМагазина {
29             get { return "Наш магазин"; }
30         }
31
32         public float СуммарнаяСтоимость {
33             get
34             {
35                 return _товары.Sum(p => p.СуммарнаяСтоимостьПозиции);
36             }
37         }
38
39         public float СуммарноеКоличество
40         {
41             get
42             {
43                 return _товары.Sum(p => p.КоличествоТовара);
44             }
45         }
46     }
47 }
```

Рисунок 21.11 – Определение класса «Магазин»

Класс «Магазин» – это «обертка» над классом «ВсеТовары».

После компиляции и исправления ошибок получим три сборки: DataTier.dll, LogicTier.dll, PresentationTier.exe.

8. Завершающий этап разработки приложения: связь класса MainWindow и класса LogicTier. Для выполнения данной связи необходимо задать свойство DataContext главного окна. На рисунок 21.12 представлен измененный код конструктора MainWindow.

```
17  namespace PresentationTier
18  {
19      /// <summary>
20      /// Interaction logic for MainWindow.xaml
21      /// </summary>
22      public partial class MainWindow : Window
23      {
24          public MainWindow()
25          {
26              Магазин logicTier = new Магазин();
27              this.DataContext = logicTier;
28              InitializeComponent();
29          }
30      }
31  }
```

Рисунок 21.12 – Конструктор MainWindow

Из листинга видно, что создается объект класса «Магазин» и устанавливается в качестве источника привязки через свойство DataContext для окна MainWindow. После этого все привязки начинают получать данные. Внешний вид окна полученного приложения представлен на рисунок 21.13.

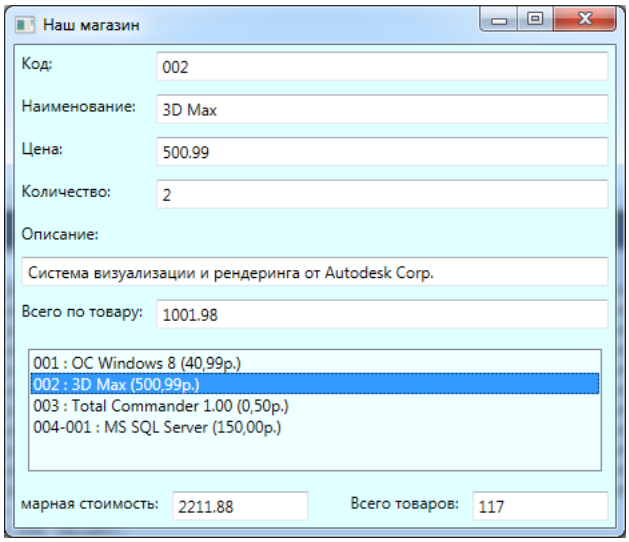


Рисунок 21.13 – Результат решения задачи

9. Выполните задание в соответствии с индивидуальным вариантом.

21.3.2. Индивидуальное задание

В каждом варианте необходимо реализовать многоуровневое приложение для обработки данных из файла. Формат текстового файла представлен в каждом варианте. В отличии от рассмотренного примера, приложение должно считывать информацию из файла. Приложение должно быть многомодульным (как в реализованном примере).

Вариант	Задание

1	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

2	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

3	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.

4	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.

5	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.

6	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.

7	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.

8	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.

9	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.

10	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.

11	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.

12	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

13	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.

14	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.

15	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.

16	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.

17	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.

18	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».

19	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».

20	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.

21	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

22	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

23	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк): 1 ВАЗ-2107; Автоваз; 140000.57; 16000 2 Гога-1; Хуань Вей Трейд; 50.00; 1000 3 ВАЗ-5000; Автоваз; 1400000.00; 1000 4 X35; Hyundai; 750000; 1 Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.
24	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк): 1 Самолет*Москва*Пекин*30000.00 2 Автобус*Ставрополь*Москва*3000.00 3 Автобус*Невинномысск*Ставрополь*10.00 4 Автобус*Москва*Владивосток*50000 5 Самолет*СПб*Москва*1500 Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.
25	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк): 1 Иванов Е.А.*ИСТ-б-о-121*3*2 2 Петров А.А.*ИСТ-б-о-111*4*1 3 Кононов А.Е.*БАС-б-о-111*4*0 4 Катченко Е.А.*БАС-б-о-121*3*1 Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.
26	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк): 1 Молоко, 0.5 л % Продукты % 10.00 % 23.00 2 Хлеб белый % Продукты % 15.00 % 22.00 3 Молоко, 1.0 л % Продукты % 18.00 % 30.00 4 Туалетный утенок % Химия % 11.00 % 17.58 5 Фейри, 0.5 л % Химия % 17.00 % 88.00 Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.
27	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк): 1 Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1 2 Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7 3 ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19 4 ПО Office 365 % ПО % 11.00 % Склад №19 5 Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7 Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.
28	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк): 1 Дом*6*150*3500000 2 Квартира*1*49*790000 3 Квартира*2*59*1500000 4 Квартира*3*70*3000000 5 Комната*1*12*500000 Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.

29	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.

30	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.

31	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.

32	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

33	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.

34	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.

35	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк): 1 Иванов Е.А.*ИСТ-б-о-121*3*2 2 Петров А.А.*ИСТ-б-о-111*4*1 3 Кононов А.Е.*БАС-б-о-111*4*0 4 Катченко Е.А.*БАС-б-о-121*3*1 Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.
36	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк): 1 Молоко, 0.5 л % Продукты % 10.00 % 23.00 2 Хлеб белый % Продукты % 15.00 % 22.00 3 Молоко, 1.0 л % Продукты % 18.00 % 30.00 4 Туалетный утенок % Химия % 11.00 % 17.58 5 Фейри, 0.5 л % Химия % 17.00 % 88.00 Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.
37	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк): 1 Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1 2 Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7 3 ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19 4 ПО Office 365 % ПО % 11.00 % Склад №19 5 Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7 Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.
38	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк): 1 Дом*6*150*3500000 2 Квартира*1*49*790000 3 Квартира*2*59*1500000 4 Квартира*3*70*3000000 5 Комната*1*12*500000 Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».
39	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк): 1 Привет с того света | Детектив | 150.00 2 Кто подставил кролика | Детектив | 200.10 3 Почемучка | Детская литература | 60.22 4 Капитошка | Детская литература | 230.14 5 Звезда 112 | Фантастика | 12.90 Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».
40	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк): 1 Молоко \ Продукты \ 40.00 \ 1 2 Тряпка \ Хозтовары \ 10.00 \ 5 3 Порошок \ Хозтовары \ 750.00 \ 10 4 Мясо \ Продукты \ 350 \ 5 5 Швабра \ Хозтовары \ 900.00 \ 5 Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.

21.4. Вопросы для защиты работы

1. Чем отличается привязка к элементам управления от привязки к произвольным объектам?
2. Поясните синтаксис привязки к данным. Какие ключевые слова и для каких целей используются при оформлении привязки с помощью XAML?
3. Какие режимы привязки существуют? Поясните назначение всех режимов привязки.
4. Поясните схему привязки элементов управления. Назовите основные элементы данной схемы.
5. Что означают понятия «целевой элемент», «элемент-источник привязки», «целевое свойство», «свойство-источник»?
6. Какое ограничение на источник привязки существует в WPF?
7. Какое ограничение существует на целевое свойство привязки?

ЛАБОРАТОРНАЯ РАБОТА 22. АСИНХРОННЫЕ ДЕЛЕГАТЫ

22.1. Цель и содержание

Цель лабораторной работы: научиться использовать делегаты для организации многопоточного приложения.

Задачи лабораторной работы:

– научиться объявлять, инициализировать и запускать потоки с использованием пользовательских делегатов;

– научиться запускать потоки с использованием библиотечных делегатов Action<T> и Func<T>;

– научиться запускать параллельные потоки с использованием лямбда-выражений.

22.2. Теоретическое обоснование

Наиболее простым способом для создания потока является определение делегата и его вызов асинхронным образом. Класс Delegate поддерживает и возможность асинхронного вызова методов. Для решения поставленной задачи класс Delegate создает отдельный поток.

Чтобы посмотреть в действии на асинхронные возможности делегатов, сначала нужно создать метод, выполнение которого занимает определенное время. Например, ниже показан метод TakesAWhile (), который благодаря вызову Thread.Sleep( ) выполняется минимум столько времени, сколько в миллисекундах задано во втором аргументе.

```
static int TakesAWhile(int data, int ms)
{
    Console.WriteLine("TakesAWhile запущен");
    Thread.Sleep(ms);
    Console.WriteLine("TakesAWhile завершен");
    return ++data;
}
```

Чтобы обеспечить вызов этого метода из делегата, понадобится определить тип делегата с тем де самым параметром и возвращаемым типом:

```
public delegate int TakesAWhileDelegate(int data, int ms);
```

Теперь можно применять различные приемы для асинхронного вызова данного делегата и возврата результатов.

Одним из таких приемов является опрос и проверка, завершил ли делегат свою работу. Созданный класс delegate предоставляет метод BeginInvoke ( ), в котором могут передаваться входные параметры, определенные вместе с типом делегата. Метод Beginlnvoke ( ) всегда имеет два дополнительных параметра типа AsyncCallback и object, которые будут рассматриваться позже. Сейчас главный интерес представляет возвращаемый тип метода – IAsyncResult. С помощью IAsyncResult можно извлекать информацию о делегате и проверять, завершил ли он свою работу, что и демонстрируется в примере с применением свойства IsCompleted. Цикл while продолжает выполняться в главном потоке программы до тех пор, пока делегат не завершит работу.

```
static void Main(string[] args)
{
    // синхронный вызов метода
    // TakesAWhileA, 3000);

    // асинхронный вызов метода с применением делегата
    TakesAWhileDelegate dl = TakesAWhile;

    IAsyncResult ar = dl.BeginInvoke(1, 3000, null, null);
    while (!ar.IsCompleted)
    {
        // выполнение еще каких-нибудь Сераций в главном потоке
        Console.Write(".");
        Thread.Sleep(50);
    }
    int result = dl.EndInvoke(ar);

    // вывод результата
    Console.WriteLine("result: {0}", result);
}
```

После запуска этого приложения можно увидеть, что главный поток и поток делегата выполняются параллельно, а после завершения работы потока делегата главный поток прекращает проход по циклу. Если главный поток завершает выполнение, не дожидаясь завершения работы делегата, поток делегата останавливается.

В данном разделе методических указаний рассмотрен самый простой метод запуска метода в отдельном потоке – с использованием асинхронного делегата.

Рассмотрен также механизм передачи параметров в поток, выполняемый параллельно (асинхронно); возврат результата из асинхронно выполняемого метода. Необходимо обратить внимание на реализацию механизма ожидания завершения работы асинхронного метода.

22.3. Методика и порядок выполнения работы

1. Создайте приложение на языке C# в MS Visual Studio.

2. В соответствии с индивидуальным вариантом разработайте требуемый тип делегата (пользовательский, библиотечный или лямбда-выражение).

3. Реализуйте асинхронное выполнение метода на основе разработанного делегата с возможностью мониторинга процесса выполнения, передачи параметров в метод и получения результата работы метода.

Индивидуальное задание.

Вариант	Тип делегата	Решаемая задача (результат метода)	Входные параметры
1	пользовательский	Метод возвращает сумму элементов матрицы целых случайных чисел	Два параметра: размер матрицы
2	библиотечный	Метод возвращает разницу максимального и минимального элементов матрицы целых случайных чисел	Два параметра: размер матрицы
3	лямбда-выражение	Метод возвращает логическое значение, указывающее существует ли заданное число в массиве целых случайных чисел	Два параметра: размер массива и искомый элемент
4	пользовательский	Метод возвращает матрицу случайных битовых значений (0 или 1), формируемую случайным образом	Два параметра: размер матрицы
5	библиотечный	Метод возвращает строку максимальной длины на основе переданного массива строк	Один параметр: массив (или список) строк
6	лямбда-выражение	Метод возвращает подмножество элементов массива случайных чисел, которые делятся на 3	Один параметр: размер исходного массива
7	пользовательский	Метод возвращает число – количество элементов присутствующих в обоих массивах случайных чисел.	Два параметра: размеры массивов

8	библиотечный	Метод возвращает среднее арифметическое элементов матрицы случайных чисел.	Два параметра: размер матрицы
9	лямбда-выражение	Метод возвращает результат шифрования строки: каждый исходный символ строки заменяется шифрованным символом, код которого на n больше кода исходного символа.	Два параметра: исходная строка, число сдвига n
10	пользовательский	Метод возвращает статистику для строки: список пар (символ строки, количество вхождений)	Один параметр: анализируемая строка
11	библиотечный	Метод возвращает разницу максимального и минимального элементов матрицы.	Два параметра: размер матрицы
12	лямбда-выражение	Метод возвращает подмножество элементов массива случайных чисел, которые являются четными	Один параметр: размер исходного массива
13	пользовательский	Метод возвращает количество нечетных элементов в матрице случайных чисел	Два параметра: размер матрицы
14	библиотечный	Метод возвращает подмножество элементов массива случайных чисел, которые отличаются от заданного числа не более чем на 4.	Два параметра: размер массива, значеное число
15	лямбда-выражение	Метод возвращает логическое значение, указывающее существует ли заданный символ в строке	Два параметра: строка и искомый символ
16	пользовательский	Метод возвращает два числа – максимальные элементы массивов случайных чисел.	Два параметра: размеры массивов
17	библиотечный	Метод возвращает скадярное произведение двух случайных векторов.	Один параметр: размер векторов
18	лямбда-выражение	Метод возвращает сумму двух матриц случайных чисел	Два параметра: размер матриц
19	пользовательский	Метод возвращает количество вхождений заданного элемента в матрице вещественных чисел	Два параметра: матрица и целевой элемент
20	библиотечный	Метод возвращает подмножество элементов массива случайных чисел, которые делятся на 6	Один параметр: размер исходного массива
21	лямбда-выражение	Метод возвращает результат сложения двух случайных векторов целых чисел	Один параметр: размер векторов
22	пользовательский	Метод возвращает количество вхождений заданного символа в строке	Два параметра: строка и целевой символ
23	библиотечный	Метод возвращает подмножество четных элементов матрицы случайных чисел	Два параметра: размер матрицы

24	лямбда-выражение	Метод возвращает статистику массива случайных целых чисел: список пар (число массива, количество вхождений)	Один параметр: размер массива
25	пользовательский	Метод возвращает логическое значение, указывающее существует ли заданный элемент в матрице чисел double	Два параметра: матрица и искомый элемент

22.4. Вопросы для защиты работы

1. Поясните назначение типа IAsyncResult.
2. Для чего используется метод Thread.Sleep( )?
3. Поясните механизм возврата значения из метода асинхронного делегата.
4. Как произвести возврат более одного значения из метода?
5. Какая разница существует между библиотечными делегатами, пользовательскими типами делегатов и лямбда-выражениями? Являются ли эти делегаты взаимозаменяемыми при реализации асинхронного вызова методов?

ЛАБОРАТОРНАЯ РАБОТА 23. ПЕРЕДАЧА ДАННЫХ ПОТОКАМ

23.1. Цель и содержание

Цель лабораторной работы: научиться создавать многопоточные приложения с возможностью передачи параметров в параллельно выполняемые потоки.

Задачи лабораторной работы:

– изучить механизм передачи параметров в поток;

– научиться решать практические задачи с использованием потоков с параметрами;

– изучить возможные методы передачи параметров.

23.2. Теоретическое обоснование

Передавать потоку какие-то данные можно двумя способами: в конструкторе Thread с помощью делегата ParameterizedThreadStart либо за счет создания специального класса и определения метода потока как метода экземпляра, что позволит инициализировать данные экземпляра перед запуском потока.

Для передачи данных потоку необходим какой-то класс или структура, где бы можно было сохранить данные. В рассматриваемом примере определяется структура Data, которая содержит строку, но в принципе передавать можно любой объект:

```
public struct Data
{
    public string Message;
}
```

В случае применения делегата ParameterizedThreadStart точка входа в поток должна обязательно принимать параметр типа object и возвращать void. Объект можно привести к нужному типу. Например, ниже показано, как вывести на консоль сообщение:

```
static void ThreadMainWithParameters(object o)
{
    Data d = (Data)o;
    Console.WriteLine("Выполняется в потоке, получено {0}", d.Message);
}
```

В конструкторе класса Thread можно назначить новую точку входа ThreadMainWithParameters и вызвать метод Start ( ), передав ему переменную d:

```
static void Main()
{
    var d = new Data { Message = "Info" };
    var t2 = new Thread(ThreadMainWithParameters);
    t2.Start(d);
}
```

Другим способом для передачи данных новому потоку является определение класса (например, MyThread) с необходимыми полями, а также главного метода потока как метода экземпляра этого класса:

```
public class MyThread
{
    private string data;
    public MyThread(string data)
    {
        this.data = data;
    }
    public void ThreadMain()
    {
        Console.WriteLine("Выполняется в потоке, data: {0}", data);
    }
}
```

В этом случае можно создать объект класса MyThread и передать его и метод ThreadMain ( ) конструктору класса Thread. Поток сможет получать доступ к данным:

```
static void Main()
{
    var obj = new MyThread("info");
    var t3 = new Thread(obj.ThreadMain);
}
```

23.3. Методика и порядок выполнения работы

1. Создайте консольное приложение в среде MS Visual Studio.

2. С помощью параметризированных потоков выполните решение задачи в соответствии с индивидуальным вариантом.

Индивидуальное задание.

Вариант	Решаемая задача (результат метода)	Входные параметры
1	Метод находит подмножество элементов массива случайных чисел, которые отличаются от заданного числа не более чем на 4.	Два параметра: размер массива, значеное число

2	Метод находит логическое значение, указывающее существует ли заданный символ в строке	Два параметра: строка и искомый символ
3	Метод находит два числа – максимальные элементы массива случайных чисел.	Два параметра: размеры массивов
4	Метод находит скадярное произведение двух случайных векторов.	Один параметр: размер векторов
5	Метод находит сумму двух матриц случайных чисел	Два параметра: размер матриц
6	Метод находит количество вхождений заданного элемента в матрице вещественных чисел	Два параметра: матрица и целевой элемент
7	Метод находит результат сложения двух случайных векторов целых чисел	Один параметр: размер векторов
8	Метод находит количество вхождений заданного символа в строке	Два параметра: строка и целевой символ
9	Метод находит подмножество четных элементов матрицы случайных чисел	Два параметра: размер матрицы
10	Метод находит статистику массива случайных целых чисел: список пар (число массива, количество вхождений)	Один параметр: размер массива
11	Метод возвращает логическое значение, указывающее существует ли заданный элемент в матрице чисел double	Два параметра: матрица и искомый элемент
12	Метод находит сумму элементов матрицы целых случайных чисел	Два параметра: размер матрицы
13	Метод находит разницу максимального и минимального элементов матрицы целых случайных чисел	Два параметра: размер матрицы
14	Метод находит логическое значение, указывающее существует ли заданное число в массиве целых случайных чисел	Два параметра: размер массива и искомый элемент
15	Метод возвращает матрицу случайных битовых значений (0 или 1), формируемую случайным образом	Два параметра: размер матрицы
16	Метод возвращает строку максимальной длины на основе переданного массива строк	Один параметр: массив (или список) строк
17	Метод возвращает подмножество элементов массива случайных чисел, которые делятся на 3	Один параметр: размер исходного массива
18	Метод находит число – количество элементов присутствующих в обоих массивах случайных чисел.	Два параметра: размеры массивов
19	Метод находит среднее арифметическое элементов матрицы случайных чисел.	Два параметра: размер матрицы
20	Метод находит результат шифрования строки: каждый исходный символ строки заменяется шифрованным символом, код которого на n больше кода исходного символа.	Два параметра: исходная строка, число сдвига n
21	Метод находит подмножество элементов массива случайных чисел, которые делятся на 6	Один параметр: размер исходного массива
22	Метод находит статистику для строки: список пар (символ строки, количество вхождений)	Один параметр: анализируемая строка
23	Метод находит разницу максимального и минимального элементов матрицы.	Два параметра: размер матрицы
24	Метод находит подмножество элементов массива случайных чисел, которые являются четными	Один параметр: размер исходного массива

25	Метод находит количество нечетных элементов в матрице случайных чисел	Два параметра: размер матрицы

23.4. Вопросы для защиты работы

1. Опишите возможные пути передачи параметров в поток.

2. Подумайте, можно ли передать в несколько потоков один и тот же параметр (ссылку на объект)? Ответ обоснуйте.

ЛАБОРАТОРНАЯ РАБОТА 24. МНОГОМОДУЛЬНЫЕ ПРИЛОЖЕНИЯ

24.1. Цель и содержание

Цель лабораторной работы: научиться проектировать консольные приложения на основе нескольких сборок.

Задачи лабораторной работы:

– научиться управлять несколькими разрабатываемыми проектами в одном решении;

– научиться организовывать взаимодействие нескольких модулей приложения.

24.2. Теоретическое обоснование

В терминологии технологической платформы .NET не используется понятие «исполняемый файл». Подобные файлы (*.dll или *.exe) характерны для компиляторов небезопасного кода.

В технологии .NET используется термин «сборка». Сборка – это модуль, в котором хранится компилированный управляемый код. Она похожа на классический исполняемый файл (*.exe) или dll-библиотеку, но имеет важное свойство – она полностью себя описывает. Сборки содержат метаданные, которые включают в себя сведения о сборке и обо всех определенных внутри нее типах, методах и т.д. Сборка может быть частной (доступной только одному приложению) или разделяемой (доступной любому приложению Windows).

В предыдущих лабораторных работах студентам предлагалось спроектировать консольные приложения, которые представляют собой одномодульные приложения – exe-файлы.

В данной лабораторной работе требуется спроектировать приложение, которое состоит из нескольких сборок (один модуль – файл *.exe, остальные – dll).

24.3. Методика и порядок выполнения работы

Для выполнения лабораторной работы спроектируем следующее приложение:

Требуется разработать приложение-опрос. Пользователю последовательно задаются вопросы, за каждый вариант ответа начисляются определенные баллы. После завершения опросы выводится сумма набранных баллов и какое-либо резюме. Такая программа может использоваться для оценки остаточных знаний в форме тестирования, для составления опросов, для самообследования и т.п.

Перед началом создания приложения, то есть перед непосредственным программированием, необходимо определиться: какие объекты и явления предметной области и связи между ними должен выявить программист. Очевидно, что это:

– вопрос;

– ответ (несколько для каждого вопроса);

– балл соответствующий конкретному ответу;

– итоговые резюме или оценки, которые выводятся в зависимости от количества набранных баллов.

Второе, что необходимо сделать осуществить физическую декомпозицию данного приложения, то есть сколько отдельных модулей будет содержать приложение. В данном случае:

– 1-ый модуль в виде файла *.exe для запуска приложения;

– 2-ой модуль в виде файла *.dll, содержащий вопросы и ответы;

– 3-ий модуль в виде файла *.dll, содержащий сообщения о результатах тестирования (опроса).

Декомпозиция, логическая и физическая, завершена. Для реализации прложения необходимо выполнить следующие шаги:

1. Создайте консольное приложение в среде MS VS (проект назовите Quiz). После выполнения всех необходимых действий окно Solution Explorer примет следующий вид:

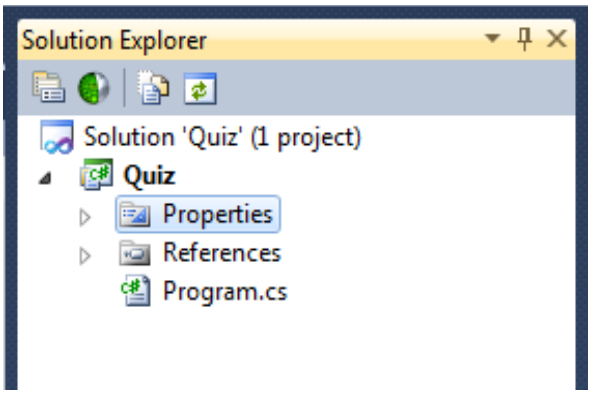


Рисунок 24.1 – Структура одномодульного приложения

В окне отображено одно решение (Solution) с именем «Quiz» и один проект в рамках решения (с именем «Quiz»). Проект содержит файл кода Program.cs, в котором определена функция Main. Этот проект будет откомпилирован в файл Quiz.exe (то есть файл для запуска).

2. Создадим еще один файл в рамках проекта Quiz, который будет содержать класс Testing, позволяющий реализовывать логику тестирования, то есть вывод вопросов в определенной последовательности, ввод выбора пользователя и т.д. Для этого вызовем контекстное меню проекта (рис. 24.2) и создадим новый файл с именем Testing.cs.

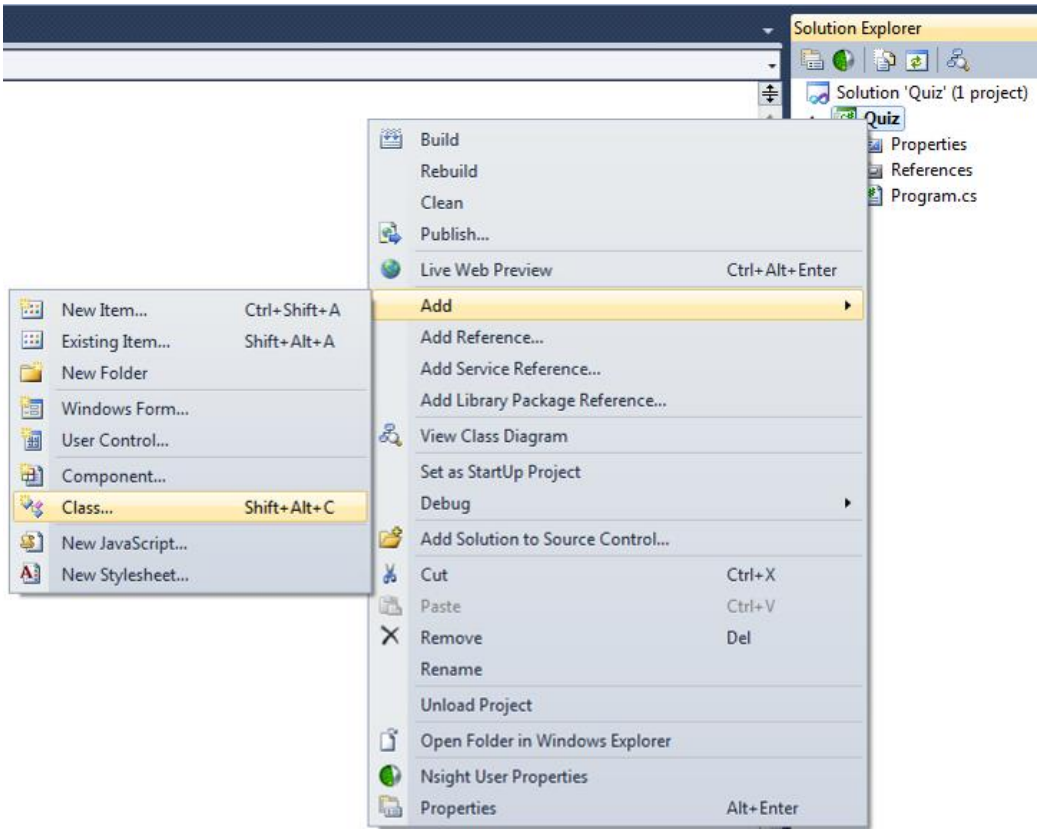


Рисунок 24.2 – Добавления нового класса к проекту

После выполнения данного действия в окне Solution Explorer состав проекта изменится (рис. 24.3).

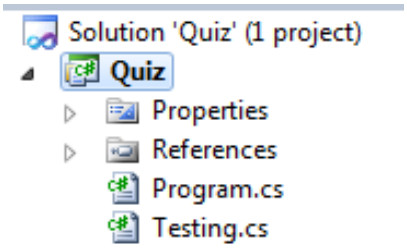


Рисунок 24.3 – Проект с несколькими файлами исходного кода

На данном этапе класс Testing не содержит никакой логики.

3. Создадим второй модуль в виде dll-библиотеки, которая будет содержать вопросы и ответы. Для этого вызовем команду контекстного меню решения Quiz (рис. 24.4).

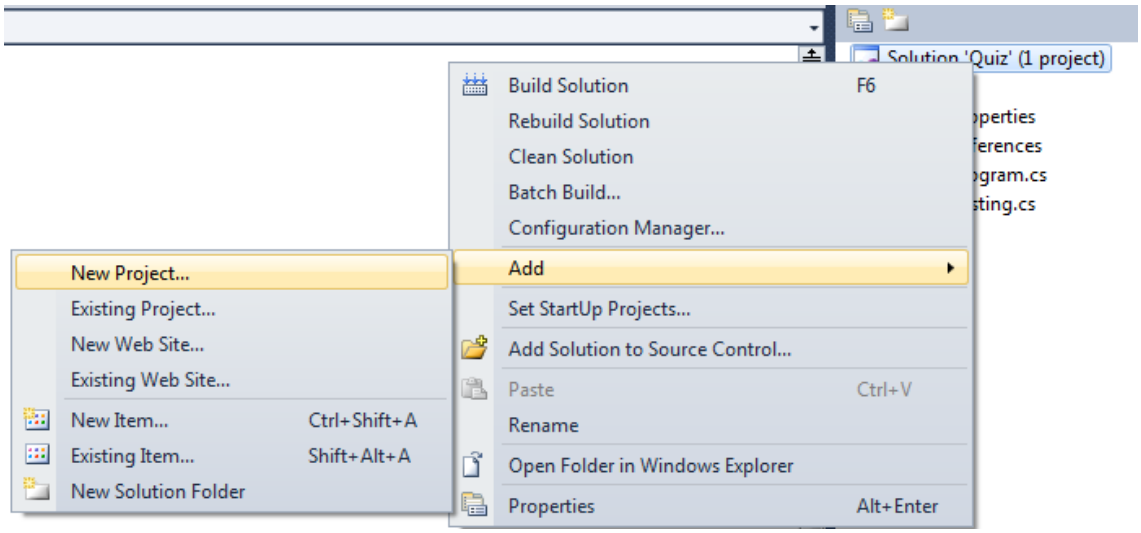


Рисунок 24.4 – Добавление проекта к решению VS

4. В появившемся диалоговом окне выберем тип проекта «Class Library», в качестве имени проекта укажем «Task» (рис. 24.5)

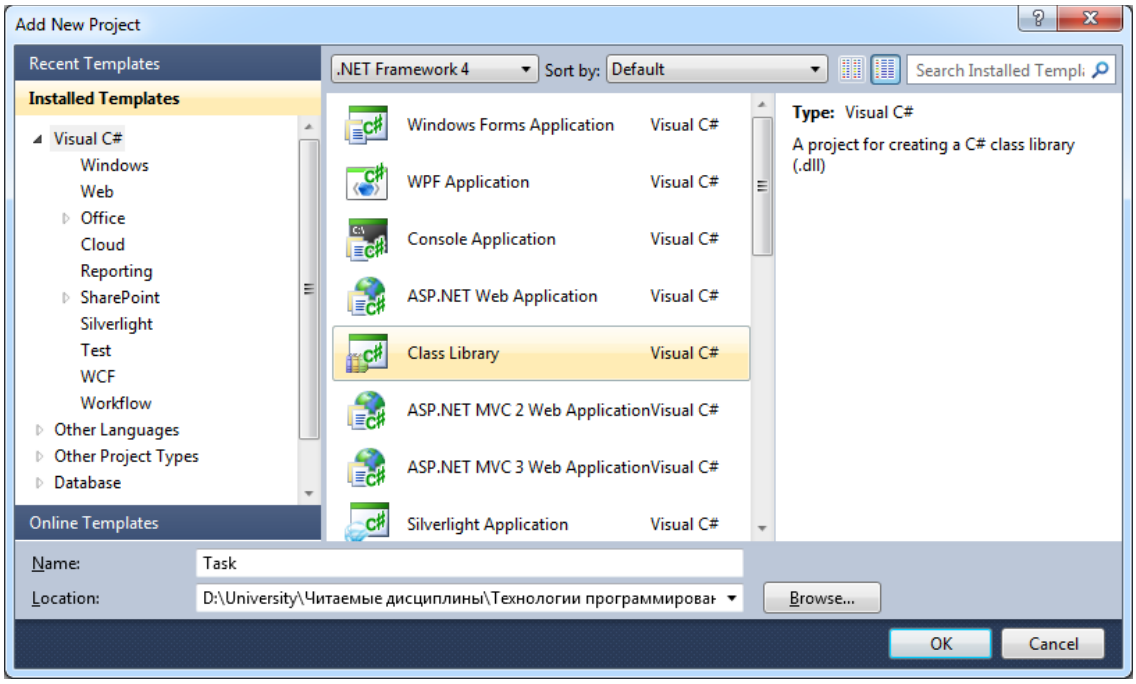


Рисунок 24.5 –Добавление библиотеки классов к решению Quiz.

После добавления нового проекта окно Solution Explorer примет вид, показанный на рис. 24.6

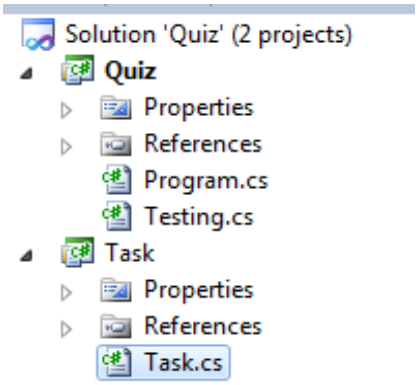


Рисунок 24.6 –Решение с несколькими проектами.

Два проекта в одном решении Quiz, при этом проект Quiz помечен полужирным шрифтом. Это означает, что при вызове команды Run будет запущен именно этот проект. Следует обратить внимание, что файл Task.cs проекта Task содержит только пустой класс без кода. Проект Task в целом не содержит функции Main, то есть не может быть запушен на выполнение.

5. Аналогичным образом добавим к решению еще один проект с именем «Result» и типом «Class Library». Окно Solution Explorer примет вид:

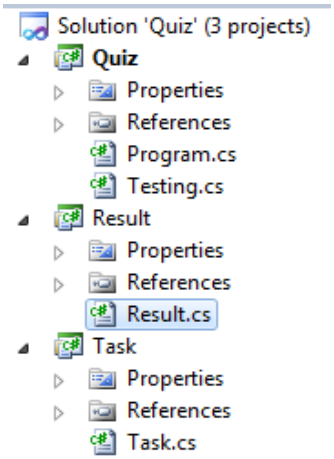


Рисунок 24.7 – Все требуемые проекты, в соответствие с физической декомпозицией, добавлены к решению

Далее переходим к наполнению классов требуемым функционалом.

6. Модифицируем файл Task.cs проекта Task следующим образом:

```
namespace Task
{
    // Класс для представления одного тестового задания
    public class SingleTest
    {
        public string Question {get; set;} // вопрос
        public List<string> Answers; // Ответы
        public List<int> Balls; // Баллы за ответы
    }

    // Статический класс для представления списка тестовых заданий
    public static class Task
    {
        // Список отдельных вопросов с ответами
        public static List<SingleTest> AllQuestions;

        // Статический конструктор для инициализации коллекции
        static Task()
        {
            AllQuestions = new List<SingleTest>()
            {
                new SingleTest()
                {
                    Question = "Main - точка входа программы ...",
                    Answers = new List<string>() { "1", "7", "Не знаю" },
                    Balls = new List<int>() { 20, 5, 0 }
                },
                new SingleTest()
                {
                    Question = "Какой оператор не является оператором цикла?",
                    Answers = new List<string>() { "while", "for", "if" },
                    Balls = new List<int>() { -5, 0, 10 }
                },
                new SingleTest()
                {
                    Question = "Как обозначить составной оператор?",
                    Answers = new List<string>() { "{ ... }", "( ... )", "< ... >" },
                    Balls = new List<int>() { 50, 0, 0 }
                },
                new SingleTest()
                {
                    Question = "Что обозначает символ ; ",
                    Answers = new List<string>() { "Пустой оператор", "Конец программы", "Не знаю" },
                    Balls = new List<int>() { 50, 20, 0 }
                }
            };
        }
    }
}
```

Рисунок 24.8 – Классы проекта Task

Следует обратить внимание на то, что класс Task статический, то есть не требует создания объектов.

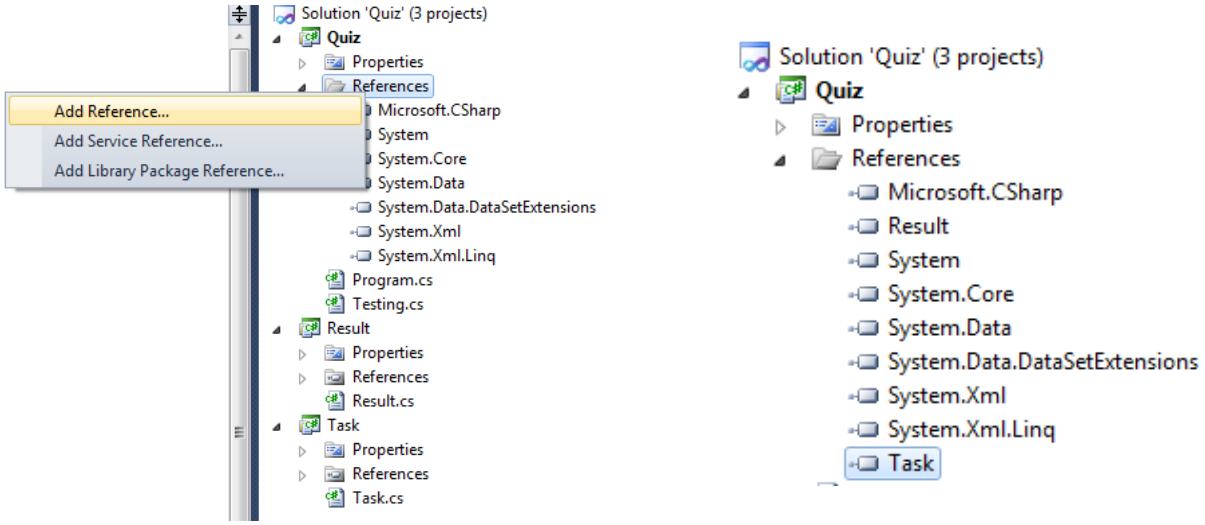
7. Модифицируйте файл Result.cs проекта Result следующим образом:

```
namespace Result
{
    public static class Result
    {
        public static List<string> Messages;

        // Статический конструктор
        static Result()
        {
            Messages = new List<string>()
            {
                "Вы набрали менее 10 баллов! Плохо!",
                "Вы набрали не менее 10 и менее 40 баллов! Нормально!",
                "Вы набрали не менее 40 баллов! Отлично"
            };
        }

        // Метод для возвращения сообщения
        public static string GetMessage(int Ball)
        {
            string mes = "";

            if (Ball < 10)
                mes = Messages[0];
            else if (Ball < 40)
                mes = Messages[1];
            else
                mes = Messages[2];

            return mes;
        }
    }
}
```

Рисунок 24.9 – Класс проекта Result

Создан один статический файл, для представления результирующих сообщений.

8. Теперь модифицируем класс Tasting проекта Quiz. В своей работе класс будет использовать ранее созданные классы Task и Result, которые находятся в других проектах. Поэтому необходимо установит ссылки на данные проекты. Для этого вызовите команду (рис. 24.10а) контекстного меню папки References проекта Quiz. После этого ссылки на сборки (откомпилированные проекты Task и Result) появятся в списке ссылок Reference (рис. 24.10б).



а *б*

Рисунок 24.10 – Добавление ссылки на проекты: а – команда контекстного меню; б – список ссылок на проекты

9. Модифицируем файл Testing.cs для реализации логики тестирования (рис. 24.11).

```
class Testing
{
    public string DoTesting()
    {
        int SumBal = 0;
        int ans = 0;

        foreach (Task.SingleTest p in Task.Task.AllQuestions)
        {
            Console.WriteLine("//////////////////////////");
            Console.WriteLine(p.Question);
            Console.WriteLine("//////////////////////////");
            Console.WriteLine("1. "+p.Answers[0]);
            Console.WriteLine("2. " + p.Answers[1]);
            Console.WriteLine("3. " + p.Answers[2]);
            Console.Write("Выберите номер ответа > ");
            ans = Convert.ToInt32(Console.ReadLine());
            SumBal += p.Balls[ans-1];
        }

        return Result.Result.GetMessage(SumBal);
    }
}
```

Рисунок 24.11 – Класс Testing

10. Модифицируем функцию Main:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Quiz
{
    class Program
    {
        static void Main(string[] args)
        {
            Testing newTest = new Testing();

            Console.WriteLine("\n\n"+newTest.DoTesting());

            Console.ReadKey();
        }
    }
}
```

Рисунок 24.12 – Функция Main

Запустите полученное приложение. Внимательно проанализируйте код приложения. Попробуйте изменить вопросы и их количество.

Затем, выполните индивидуальное задание.

Варианты индивидуальных заданий

Спроектируйте многомодульное приложение (**3 модуля**), реализующее поставленную задачу. Перед выполнением приложения необходимо выполнить декомпозицию задачи и выявить основные объекты предметной области.

Варианты	Структура данных
1, 6, 11, 16, 21	Приложение по двум числам a и b (стороны прямоугольника) рассчитывает периметр, диагональ и площадь прямоугольника.
2, 7, 12, 17, 22	Приложение по трем сторонам (a, b, c) треугольника рассчитывает периметр и площадь треугольника.
3, 8, 13, 18, 23	Приложение по значениям R и h (радиус и высота цилиндра) рассчитывает площадь поверхности и объем цилиндра.

Варианты	Структура данных
4, 9, 14, 19, 24	Приложение из нескольких прямоугольников со сторонами a и b выбирает фигуру наибольшей площади и фигуру, имеющую наибольшую диагональ.
5, 10, 15, 20, 25	Приложение из нескольких треугольников со сторонами a, b и c выбирает фигуру наименьшей и наибольшей площади.

24.4. Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.

2. Цели лабораторной работы.

3. Ответы на контрольные вопросы.

4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в письменном виде сдается преподавателю.

24.5. Вопросы для защиты работы

1. Что такое статический класс?

2. Что такое сборка?

3. Как определить проект по умолчанию в многомодульном решении?

4. Какими способами можно разместить в файлах определение класса?

5. Какой проект начнет выполняться первым если несколько из них в одном решении содержат функцию Main?

6. Что необходимо сделать для того, чтобы появилась возможность использовать классы, определенные в другом проекте?

24.6. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [5], [6-8].

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Павлова, Е. А. Технологии разработки современных информационных систем на платформе Microsoft .NET : учебное пособие / Павлова Е. А. - Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. - 128 с. - Книга находится в базовой версии ЭБС IPRbooks. - ISBN 978-5-9963-0003-7.

2. Пугачев, С. В. Разработка приложений для Windows 8 на языке C# / С. Пугачев, А. Шериев, К. Кичинский. - СПб. : БХВ-Петербург, 2013. - 416 с. : ил. - (Профессиональное программирование). - ISBN 978-56-9775-0846-9.

Дополнительная литература

1. Зиборов, В. В. Visual C# 2012 на примерах / Виктор Зиборов. - Санкт-Петербург : БХВ-Петербург, 2013. - 473 с. : ил. ; 24 см. - ISBN 978-5-9775-0888-9.

2. Прайс, Д. Visual C#2.0.NET. Полное руководство : [учеб. пособие] : пер. с англ. / Дж. Прайс, М. Гандрэрлой. - СПб. : КОРОНА-век, 2011. - 736 с. : ил. - Прил.: с. 673. - ISBN 978-5-7931-0555-2.

3. Чеповский, А. Common Intermediate Language и системное программирование в Microsoft .NET / А. Чеповский ; А. Макаров ; С. Скоробогатов. - 2-е изд., исправ. - Москва : Национальный Открытый Университет «ИНТУИТ», 2016. - 399 с. - (Основы информатики и математики). - ISBN 5-94774-410-4.

4. Khang A. Professional WPF and C# Programming: Practical Software Development Using WPF and C#. / Alex Khang. Independently published (6 May 2019). - 405 p.

5. A. Troelsen, P. Japikse. Pro C# 8 with .NET Core 3: Foundational Principles and Practices in Programming / Andrew Troelsen, Phil Japikse. Apress; 9th ed. edition (16 Aug. 2026). - 1223 p.

Перечень Интернет-ресурсов

1. http://metanit.com - сайт посвящен различным языкам и технологиям программирования, компьютерам, мобильным платформам и ИТ-технологиям.

2. https://msdn.microsoft.com/magazine – Интернет-журнал о технологиях разработки Microsoft.

3. https://professorweb.ru – информационный ресурс, посвященный разработке приложений на основе технологии .NET.

Методические указания к лабораторным работам по дисциплине

«Объектно-ориентированное программирование»

Часть 2 (лабораторные работы 13-24)

для студентов направления

09.03.02 «Информационные системы и технологии»

Составитель: Е.И. Николаев