ОГЛАВЛЕНИЕ

3

[Лабораторная работа 12. Основы проектирования приложений Windows Forms 5](#_bookmark1)

[Лабораторная работа 13. Простейшие элементы управления и события 15](#_bookmark2)

[Лабораторная работа 14. Меню и строка состояния в приложениях Windows](#_bookmark3) [Forms 28](#_bookmark3)

[Лабораторная работа 15. Диалоговые окна 56](#_bookmark4)

[Лабораторная работа 16. Обработка табличных данных 70](#_bookmark5)

[Лабораторная работа 17. Технология GDI+ в приложениях Windows Forms 88](#_bookmark6)

[Лабораторная работа 18. Приложения Windows Presentation Foundation 105](#_bookmark7)

[Лабораторная работа 19. Ресурсы, стили, триггеры 127](#_bookmark8)

[Лабораторная работа 20. Механизм привязки WPF 142](#_bookmark9)

[Лабораторная работа 21. Источники привязки произвольного типа 152](#_bookmark10)

# ЛАБОРАТОРНАЯ РАБОТА 12. ОСНОВЫ ПРОЕКТИРОВАНИЯ ПРИЛОЖЕНИЙ WINDOWS FORMS

* 1. **Цель и содержание**

Цель лабораторной работы: научиться создавать приложение Windows в среде MS Visual Studio и программировать алгоритмы с использованием простых элементов управления.

Задачами лабораторной работы являются:

* разработка приложения с использованием Windows Forms,
* изучение основных членов класса Form;
  1. **Теоретическая часть**

Файловая структура проекта MS Visual Studio в случае использования типа проекта Windows Form Application выглядит следующим образом (рис. 12.1):

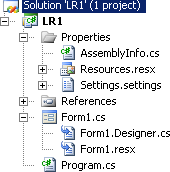


Рисунок 12.1 – Вид окна Solution Explorer для Windows Form Application

Файл Program.cs содержит класс Program и статический метод Main(), с которого начинается выполнение приложения (рис. 12.2).

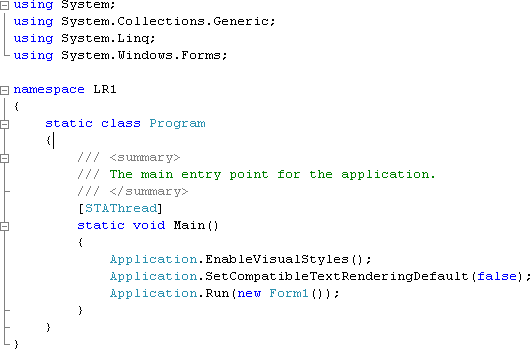


Рисунок 12.2 – Листинг Program.cs

В самом начале файла Program.cs выполняется объявление используемых пространств имен с использованием using.

Класс Program содержит метод Main(), который используя статический метод Run класса Application создает и выводит на экран главную форму приложения: Application.Run (new Form1()).

Таким образом, реализуется один из принципов объектно-ориентированного программирования: разграничение обязанностей (т.е. каждый класс выполняет минимально возможное количество операций).

Класс главной формы Form1 (по умолчанию) представлен двумя связанными C#-файлами. Для отображения содержимого Form1.cs необходимо щелкнутьправой кнопкой мыши в окне проектирования главной формы на самой форме или на пиктограмме Form1.cs в окне Solution Explorer (рис. 12.1) и в появившемся контекстном меню выбрать пункт «View Code».

Листинг Form1.cs представлен на рис 12.3.

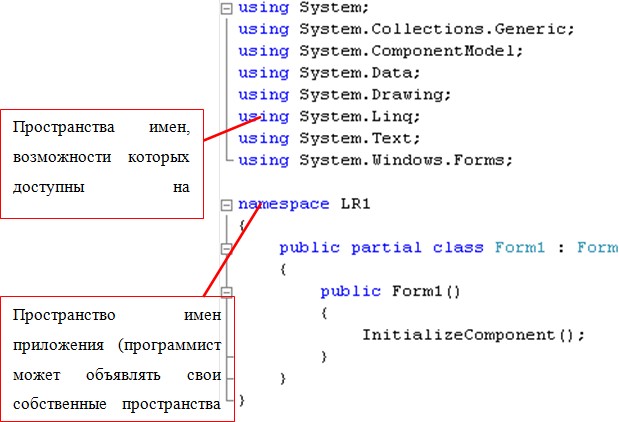


Рисунок 12.3 –Листинг Form1.cs

Конструктор, созданный по умолчанию, вызывает метод InitializeComponent(), который определен в соответствующем файле Form1.Designer.cs (рис. 12.1).

Этот метод создается автоматически, в нем отображаются все изменения, производимые программистом в окне визуального проектирования формы.

Пространства имен, доступные для использования, объявляются в начале файла.

System является базовым пространством имен – в него входят все остальные типы и пространство имен. Конструкция using System; в начале файла программы указывает на то, что весь программный код будет выполняться в данном пространстве имен, поэтому при использовании типов (например Int32), определенных в пространстве System нет необходимости указывать само имя пространства (то есть нет необходимости писать System.Int32).

Программист может самостоятельно вводить пространства имен при написании приложений.

**Обработка событий в режиме проектирования**. Окно проектирования главной формы приложения (рис. 12.4) можно открыть дважды щелкнув на

пиктограмме Form1.cs в окне Solution Explorer (рис. 12.1). При этом откроется вкладка Form1.cs [Design]. Окно свойств примет вид, показанный на рисунке 12.4.

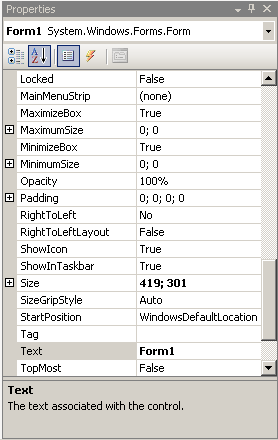


Рисунок 12.4 – Вид окна Свойств (Properties) в режиме проектирования формы

Кнопка позволяет отобразить список событий, характерных для элемента управления, активного в данный момент в окне проектирования. Например, если выделить главную форму приложения, то отобразятся события, характерные для класса Form.



Для отработки механизма регистрации событий, рассмотрим обработку наиболее простых событий – событий от мыши.

Для добавления обработчика события мыши необходимо выбрать соответствующее событие (Click, DoubleClick, MouseEnter, MouseLeave, MouseDown, MouseUp, MouseMove,. MouseHover, MeaseWheel) и дважды щелкнуть на поле, расположенном в соседнем от него столбце в окне свойств (таким образом обработчику присвоится имя по умолчанию). В данном поле можно ввести свое название для нового обработчика, или выбрать из выпадающего списка уже существующие обработчики.

После регистрации события в окне «Properties», автоматически добавляются строки кода в файлы Form1.Designer.cs (регистрируется обработчик события) и Form1.cs (добавляется метод, ассоциированный с данным событием).

Форма приложения должна быть функциональной и максимально эргономичной. Такие простые вещи, как цветовые комбинации, размеры шрифтов и окон могут сделать приложение намного более привлекательным для пользователя.

В случае если свойство Icon формы установлено, а для свойства ControlBox не указано значение false, то значок появится в левом верхнем углу формы. Обычно принято устанавливать здесь значок приложения.

Свойство FormBorderStyle задает тип рамки (а точнее вид окна: диалоговое, окно панели инструментов, окно верхнего уровня), которая появляется вокруг формы. Для этого используется перечисление FormBorderStyle. Допустимые значения этого перечисления: Fixed3D, FixedDialog, FixedSingle, FixedToolWindow, None, Sizable, SizableToolWindow.

Для выполнения лабораторной работы потребуется основная информация о событиях мыши:

Все обработчики событий мыши принимают параметр типа MouseEventArgs. Поступающий на вход обработчика событий мыши объект типа MouseEventArgs позволяет получить дополнительную информацию о действии мыши, путем введения ряда специальных членов класса (табл. 12.1).

Таблица 12.1 – Свойства типа MouseEventArgs

|  |  |
| --- | --- |
| Свойство | Описание |
| Button | Содержит информацию о том, какая клавиша была  нажата, в соответствии с определением перечня MouseButtons |
| Clicks | Содержит информацию о том, сколько раз была нажата  и отпущена клавиша мыши |
| Delta | Содержит информацию со знаком, соответствующее числу щелчков, произошедших при вращении колесика  мыши |
| X | Содержит информацию о координате X |
| Y | Содержит информацию о координате Y |

* 1. **Методика и порядок выполнения работы**
     1. Учебная задача

1. Создайте в среде разработки MS VS проект. В качестве типа проекта выбрать «Windows Application» (или «Windows Forms Application» в зависимости от версии .NET Framework)
2. После создания и сохранения проекта измените программу таким образом, чтобы координаты курсора мыши выводились в заголовке главного окна приложения.

Для этого в обработчик события движения мыши над оконной формой (MouseMove) добавьте строку кода:

Text = string.Format("Координаты: {0}, {1}", e.X, e.Y);

1. Добавьте текстовое поле (TextBox) в режиме разработки (для этого необходимо использовать панель элементов управления «ToolBox»). Дополните обработчик движения мыши таким образом, чтобы в текстовом поле отображалась сумма координат указателя мыши (рисунок 12.5).

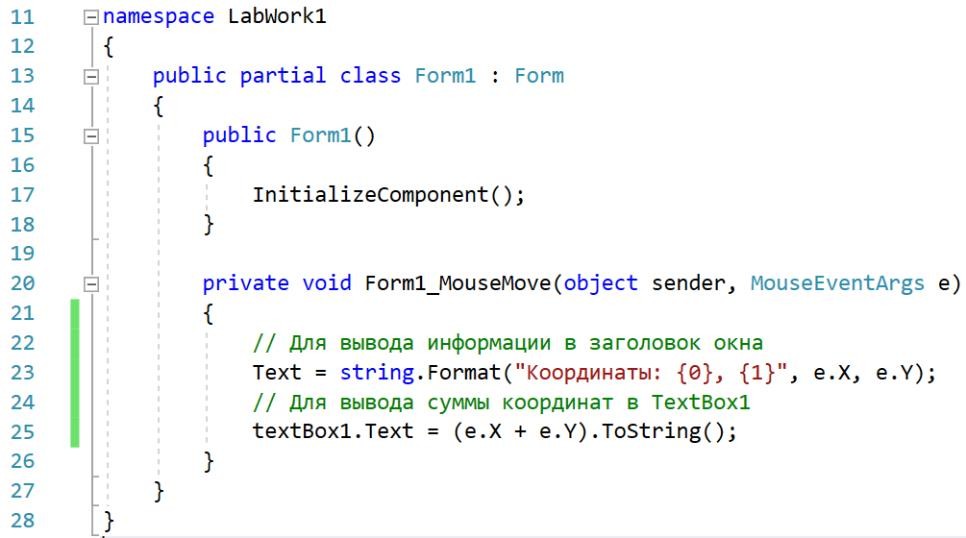


Рисунок 12.5 – Код обработчика перемещения указателя мыши

1. Работающую программу необходимо представить преподавателю.
2. После этого выполните индивидуальное задание в соответствие с вариантом.
   * 1. Индивидуальное задание

В каждом варианте необходимо реализовать следующий функционал:

1. Значения переменных задаются следующим образом: любые две переменные – это координаты курсора мыши; оставшиеся переменные инициализируются пользователем в соответствующих текстовых полях.
2. Переменные в левой части, которые привязаны к координатам указателя мыши также выводятся в «свои» текстовые поля, которые помечены как только для чтения.
3. Результат вычисления выражения выводится в заголовок окна (если значение выражения вычислить не представляется возможным, то выводится **ERROR**).

|  |  |
| --- | --- |
| Вариант | Выражение для вычисления |
| 1 | *Z*  *a*  *x*  *b*  *x*  cos( *y*) *wa* |
| 2 | *F*  *d* 2  *x*  *b*3  *x*2  cos( *y*) |
| 3 | *A*  *d*  *z*  sin(*e*)  cos( *y*)  *e* |
| 4 | *U*  *f*  *e*  sin 2( *e* )  cos( *y*) *w w* |
| 5 | *t*  lg( *f* )  *e*  sin( *w*)  *e*  *t* |
| 6 | *V*  *h*  *e*  sin( *f* )  sin(*b*)  *q*  *a* |
| 7 | *Q*  sin(*h*  *d* )  *o*1  sin( *f* )  sin(*b*)  *eo*1 |
| 8 | *E*  sin( *h*11  *d*12 )  *v*6  sin(*h*3)  sin( *f* ) ln(*h*11) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | *Z*  *z*  *ax*1  | | *e*7  *x* | | | * *x*  cos(*as* \_ 9  *y*) | | | | | | | | | |
|  | | *y* |
|  | | | | | | | | | | | | | | |
| 10 | arg01  arg02   *RES*  | | | | | | |  | | | | | | |  |
| sin(arg 03)  | | | | | | arg04 |  |
| arg01arg5  lg(arg 02) | | | | | | | | | | | | | | |
| 11 | *v*  *e* \_ 2   *U*  | |  | | | | | | |  | | | | | |
| cos(arg3)  | | | | | | *t* |  | | | | | |
| *a* \_ *del* \**tg*(*t*) | | | | | | | | | | | | | | |
| 12 | *t*  *W*  cos(*g*  *q*)  *e*  sin(*e*)  *o*  *W* | | | | | | | | | | | | | | |
| 13 | *Z*  *z* cos( *y*)  *e*7  *x*  cos(*as* \_ 9  *y*) | | | | | | | | | | | | | | |
| 14 | *R*  *q*  | sin 2 (*e*)  cos( *y*) | | | | | |  cos(*s*  *g*) | | | | | | | |
| 15 | *F*  sin( *v*  *x* )  *x*  *b*3  *x*2  cos( *y*) *y* | | | | | | | | | | | | | | |
| 16 | sin( *w*)  *e*  *res*  *F*  *e*  *t*  cos(*w*) *e* | | | | | | | | | | | | | | |
| 17 | *t*  *W*  *tg*(*g*  *q*)  *e*  sin(*e*)  *W*  *e*  ln(20  *x*) *W* | | | | | | | | | | | | | | |
| 18 | *t*  *W*  *e*  cos(*e*)  *W*  *e*  *W* | | | | | | | | | | | | | | |
| 19 | *F*  *d* 2  *x*  *x*  *b*3  *x*2  cos( *y*) | | | | | | | | | | | | | | |
| 20 | *t*  *W*  cos(*g*  *q*)  *e*   *W* | | | | | | sin(*e*)  | | | | | *o e*4 |  | | |
| 21 | *v*3  *e* \_ 2   *U*  | | |  | | | | | | | |  | | | |
| cos(arg3)  | | | | | | | *t* |  | | | |
| *a* \_ *del* \* *tg*(*t*) | | | | | | | | | | | | | | |
| 22 | *F*  sin( *v*  *x* )  *x*  *b*3  *x*2  cos( *y*) *y* | | | | | | | | | | | | | | |
| 23 | *v*  4 *e* \_ 2  sin(12 \* arg3)  cos(arg3)  *t*  *e* \_ 2 1 | | | | | | | | | | | | | | |
| 24 | *w*  *F e* sin( *t* )  *res*   *e*   cos(*w*  *e*) 25  *e* | | | | | | | | | | | | | | |
| 25 | *t*  *az*  cos(*g*  *q*)  *g*  sin(*q*)  12  *q g*3 *az* | | | | | | | | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 26 | *t*  *W*  cos(*g*  *q*)  *e*  sin(*e*)  *o*  *W* | | | | | |
| 27 | *Z*  *z* cos( *y*)  *e*7  *x*  cos(*as* \_ 9  *y*) | | | | | |
| 28 | *R*  *q*  | sin 2 (*e*)  cos( *y*) | |  cos(*s*  *g*) | | |
| 29 | *F*  sin( *v*  *x* )  *x*  *b*3  *x*2  cos( *y*) *y* | | | | | |
| 30 | sin( *w*)  *e*  *res*  *F*  *e*  *t*  cos(*w*) *e* | | | | | |
| 31 | *t*  *W*  *tg*(*g*  *q*)  *e*  sin(*e*)  *W*  *e*  ln(20  *x*) *W* | | | | | |
| 32 | *t*  *W*  *e*  cos(*e*)  *W*  *e*  *W* | | | | | |
| 33 | *F*  *d* 2  *x*  *x*  *b*3  *x*2  cos( *y*) | | | | | |
| 34 | *t*  *W*  cos(*g*  *q*)  *e*   *W* | | sin(*e*)  | | *o e*4 |  |
| 35 | *Z*  *a*  *x*  *b*  *x*  cos( *y*) *wa* | | | | | |
| 36 | *F*  *d* 2  *x*  *b*3  *x*2  cos( *y*) | | | | | |
| 37 | *A*  *d*  *z*  sin(*e*)  cos( *y*)  *e* | | | | | |
| 38 | *U*  *f*  *e*  sin 2( *e* )  cos( *y*) *w w* | | | | | |
| 39 | *t*  lg( *f* )  *e*  sin( *w*)  *e*  *t* | | | | | |
| 40 | *V*  *h*  *e*  sin( *f* )  sin(*b*)  *q*  *a* | | | | | |

* 1. **Контрольные вопросы**

1. Какие файлы описывают класс формы?
2. Какие действия необходимо выполнить для создания обработчика события?
3. Где описывается код обработчика события? В каком файле регистрируется обработчик события (метод привязывается к событию)?
4. Как получить доступ к координатам курсора мыши?
5. Какой класс содержит методы, реализующие математические функции?

# ЛАБОРАТОРНАЯ РАБОТА 13. ПРОСТЕЙШИЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ И СОБЫТИЯ

* 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.
  1. **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.

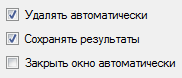


Рисунок 13.1 – Внешний вид CheckBox

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.

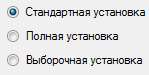


Рисунок 13.2 – Внешний вид RadioButton

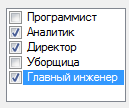
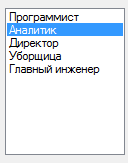
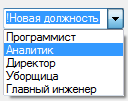
Переключатели обычно используются в группе. Переключатели позволяют пользователю выбирать одну из нескольких опций. При наличии нескольких элементов управления RadioButton в одном контейнере, только один из них может быть выбран в один и тот же момент времени.

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.

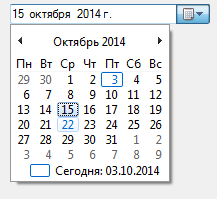


Рисунок 13.4 – Внешний вид DateTimePicker

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.

* 1. **Методика и порядок выполнения работы**
     1. Учебная задача

**Задание:**

Необходимо разработать приложение для вычисления выражения:

*Zoo*  1  *X*  *Y* 2  *X* 3  *Y* 4  ...



*W*  *X*  *T*  *Y*  7

**Решение:**

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

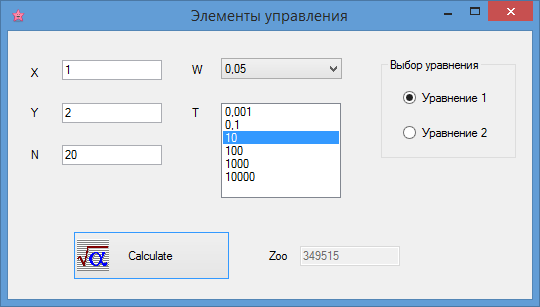


Рисунок 13.5 – Внешний вид экранной формы

1. Добавим обработчик события Click для кнопки расчета:

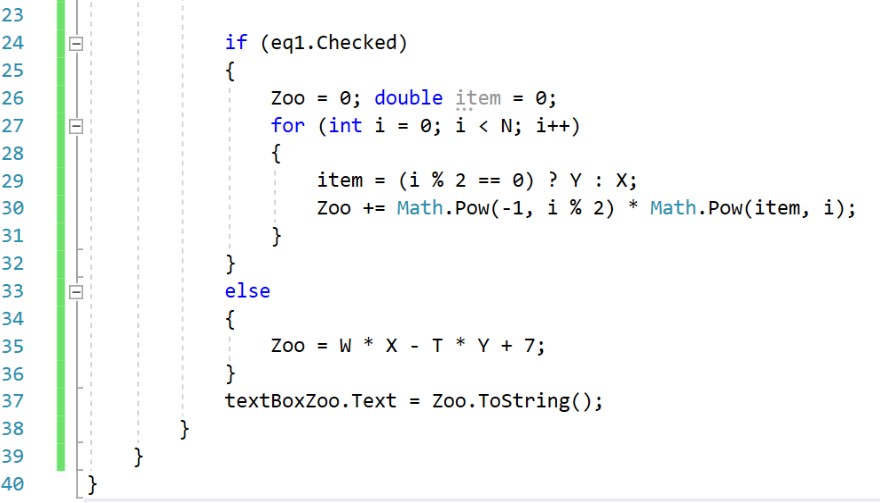
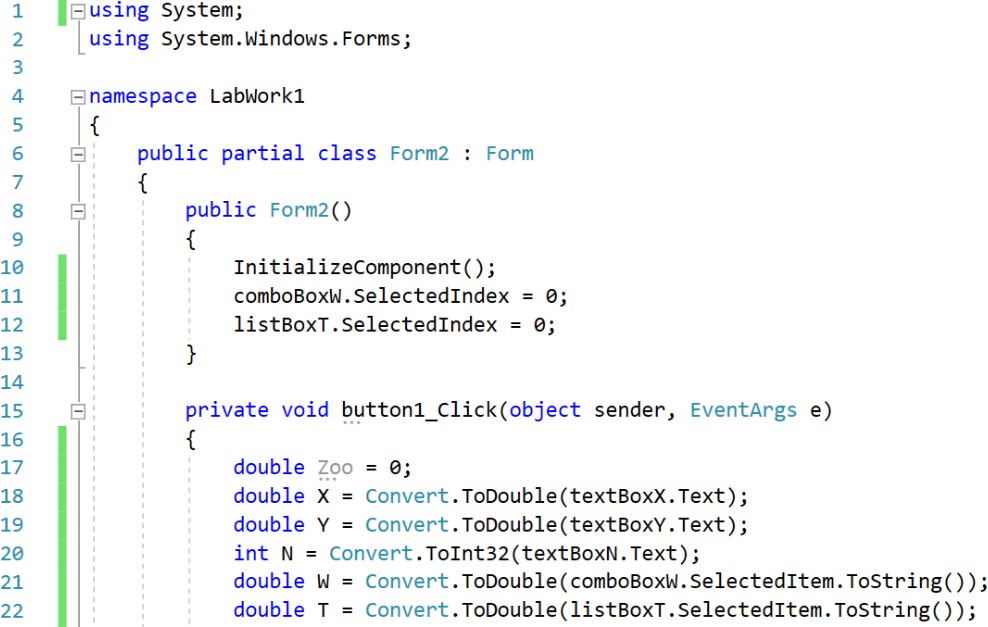


Рисунок 13.6 – Код обработчика

1. Запустите приложение. Исправьте ошибки.
   * 1. 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 |  *X Y* 2 *X* 3 *Y* 4  1 2  6  24  120  ..;.  *Z*     *N R* 3  *a*  *j*   .   *i* 1 *j* 1 *b*  *i* |
| 2 |  1  *Y*  *X* 2  *Y* 3    2 6 24 120 ..;.  *Z*     *N R i*2  *j*    *j* .  *i* 1 *j* 1 *i* |
| 3 |  1  *Y*  *X* 2  *Y* 3    1 2 3 4 ..;.  *Z*     *N R j*2  *h*  *i*    *j* .  *i* 1 *j* 1*i*  *c*  *j* |
| 4 |  *Y*  *Y* 2  *X* 3  *Y* 4    2 6 24 120 ..;.  *Z*     *N R a*  *i*  *b*  *j*    *j* .  *i* 1 *j* 1 *c*  *i* |
| 5 |  *p*2  *p*3  *p*4  *p*5    2 6 24 120 ..;.  *Z*     *N R i*2  *b*  *j*    3 .  *i* 1 *j* 1 *c*  *i* |
| 6 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R i*2  *a*  *j*2    3 3 .  *i* 1 *j* 1 *i*  *j* |

|  |  |
| --- | --- |
| 7 |  *X*  *X* 2  *X* 3  *X* 4      2 6 24 120 ..;.  *Z*     *N R a*  *i*2    3 3 .  *i* 1 *j* 1*i*  *b*  *j* |
| 8 |  1  *X*  *X* 2  *X* 3    2 6 24 120 ..;.  *Z*     *N R a*  *i*    3 3 .  *i* 1 *j* 1*i*  *j* |
| 9 |  *f*  *X*  *f* 2  *X* 2  *f* 3  *X* 3  *f* 4    2 6 24 120 ..;.  *Z*     *N R a*  (*i*  *j*)    .  *i* 1 *j* 1 *i*  (*i*  2) |
| 10 |  *X Y* 2 *X* 3 *Y* 4  1  2  6  24  120  ..;.  *Z*     *N R* 3  *j*    .  *i* 1 *j* 1 *c*  *i* |
| 11 |  1  *Y*  *X* 2  *Y* 3    2 6 24 120 ..;.  *Z*     *N R i*2  *j*    *j* .  *i* 1 *j* 1 *a* |
| 12 |  1  *Y*  *X* 2  *Y* 3    1 2 3 4 ..;.  *Z*     *N R j*2  *c*  *i*    *j* .  *i* 1 *j* 1*i*  *d*  *j* |
| 13 |  *Y*  *Y* 2  *X* 3  *Y* 4    2 6 24 120 ..;.  *Z*     *N R a*  *i*  *b*  *j*    *j* .  *i* 1 *j* 1 *c*  *i* |
| 14 |  *p*2  *p*3  *p*4  *p*5    2 6 24 120 ..;.  *Z*     *N R i*2  *b*  *j*    *i* 3 .  *i* 1 *j* 1 *c*  *i* |

|  |  |
| --- | --- |
| 15 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R i*2  *j*2    3 3 .  *i* 1 *j* 1 *i*  *j* |
| 16 |  *X*  2  *X* 2  3  *X* 3  4  *X* 4    2 6 24 120 ..;.  *Z*     *N R* (*i*  1)2    3 3 .  *i* 1 *j* 1*a*  *i*  *b*  *j* |
| 17 |  1  *X*  *X* 2  *X* 3    2 6 24 120 ..;.  *Z*     *N R a*  *i*    3 3 .  *i* 1 *j* 1*i*  *b*  *j* |
| 18 |  *f*  *X*  *f* 2  *X* 2  *f* 3  *X* 3  *f* 4    2 6 24 120 ..;.  *Z*     *N R a*  (*i*  *j*)    .  *i* 1 *j* 1*b*  *i*  (*i*  2) |
| 19 |  1  *Y*  *X* 2  *Y* 3    1 3 5 7 ..;.  *Z*     *N R a*  *j*2  2    *j* .  *i* 1 *j* 1 *b*  *i*  3 |
| 20 |  *Y*  *Y* 2  *X* 3  *Y* 4    2 6 24 120 ..;.  *Z*     *N R a*  *i*  *b*  *j*    *j* .  *i* 1 *j* 1 *c*  *i* |
| 21 |  *p*2  *p*3  *p*4  *p*5    1 2 2  6 3  24 4 120 ..;.  *Z*     *N R i*2  *d*  *j*    *i* 3 .  *i* 1 *j* 1 *c*  *i* |
| 22 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R i*2  *j*2    3 .  *i* 1 *j* 1 *i* |

|  |  |
| --- | --- |
| 23 |  3  *Y*  5  *Y* 2  7  *X* 3  9  *Y* 4    2 6 24 120 ..;.  *Z*     *N R i*  *k*  *j*    *j* .  *i* 1 *j* 1 *c*  *i* |
| 24 |  3  *p*2  5  *p*3  7  *p*4  9  *p*5      2 6 24 120 ..;.  *Z*     *N R i*2  *b*    *i* 3 .  *i* 1 *j* 1*c*  (*i*  1) |
| 25 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R* (*a*  *i*  *j*)2    3 3 .  *i* 1 *j* 1 *i*  *b*  *j* |
| 26 |  1  *Y*  *X* 2  *Y* 3    2 6 24 120 ..;.  *Z*     *N R i*2  *j*    *j* .  *i* 1 *j* 1 *i* |
| 27 |  1  *Y*  *X* 2  *Y* 3    1 2 3 4 ..;.  *Z*     *N R j*2  *h*  *i*    *j* .  *i* 1 *j* 1*i*  *c*  *j* |
| 28 |  *Y*  *Y* 2  *X* 3  *Y* 4    2 6 24 120 ..;.  *Z*     *N R a*  *i*  *b*  *j*    *j* .  *i* 1 *j* 1 *c*  *i* |
| 29 |  *p*2  *p*3  *p*4  *p*5    2 6 24 120 ..;.  *Z*     *N R i*2  *b*  *j*    3 .  *i* 1 *j* 1 *c*  *i* |
| 30 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R i*2  *a*  *j*2    3 3 .  *i* 1 *j* 1 *i*  *j* |

|  |  |
| --- | --- |
| 31 |  *X*  *X* 2  *X* 3  *X* 4      2 6 24 120 ..;.  *Z*     *N R a*  *i*2    3 3 .  *i* 1 *j* 1*i*  *b*  *j* |
| 32 |  1  *X*  *X* 2  *X* 3    2 6 24 120 ..;.  *Z*     *N R a*  *i*    3 3 .  *i* 1 *j* 1*i*  *j* |
| 33 |  *f*  *X*  *f* 2  *X* 2  *f* 3  *X* 3  *f* 4    2 6 24 120 ..;.  *Z*     *N R a*  (*i*  *j*)    .  *i* 1 *j* 1 *i*  (*i*  2) |
| 34 |  *X Y* 2 *X* 3 *Y* 4  1  2  6  24  120  ..;.  *Z*     *N R* 3  *j*    .  *i* 1 *j* 1 *c*  *i* |
| 35 |  1  *Y*  *X* 2  *Y* 3    2 6 24 120 ..;.  *Z*     *N R i*2  *j*    *j* .  *i* 1 *j* 1 *a* |
| 36 |  1  *Y*  *X* 2  *Y* 3    1 2 3 4 ..;.  *Z*     *N R j*2  *c*  *i*    *j* .  *i* 1 *j* 1*i*  *d*  *j* |
| 37 |  *X p*  *X* 2 *p*2  *X* 3 *p*3  *X* 4   2  6  24  120  ..;.  *Z*     *N R i*2  *j*2    3 3 .  *i* 1 *j* 1 *i*  *j* |
| 38 |  *X*  2  *X* 2  3  *X* 3  4  *X* 4    2 6 24 120 ..;.  *Z*     *N R* (*i*  1)2    3 3 .  *i* 1 *j* 1*a*  *i*  *b*  *j* |

|  |  |
| --- | --- |
| 39 |  1  *X*  *X* 2  *X* 3    2 6 24 120 ..;.  *Z*     *N R a*  *i*    3 3 .  *i* 1 *j* 1*i*  *b*  *j* |
| 40 |  *f*  *X*  *f* 2  *X* 2  *f* 3  *X* 3  *f* 4    2 6 24 120 ..;.  *Z*     *N R a*  (*i*  *j*)    .  *i* 1 *j* 1*b*  *i*  (*i*  2) |

* 1. **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

* 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».
  1. **Теоретическое обоснование**
     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.

* + 1. 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.
   1. **Методика и порядок выполнения работы**
      1. Учебная задача
4. Создайте приложение Windows Forms в среде MS Visual Studio.
5. Поместите элемент MenuStrip в главную форму в окне проектирования.
6. Создайте меню со следующей структурой:

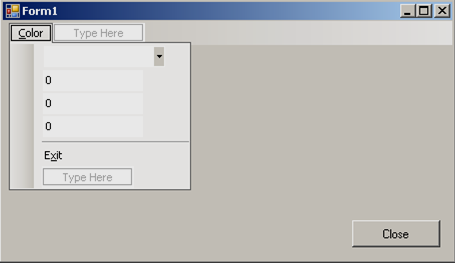


Рисунок 14.1 – Главное меню приложения.

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

* 1. 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».
  2. Следующий пункт меню типа ToolStripSeparator.
  3. Последний пункт меню типа ToolStripMenuItem, для которого устанавливаются свойство Text = “E&xit”.

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



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



1. 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:

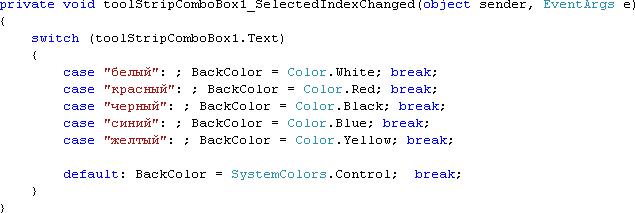


Рисунок 14.2 – Обработка выбора цвета в списке.

1. 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:

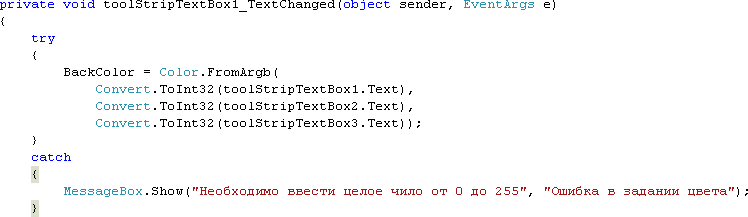


Рисунок 14.3 – Обработка события изменения значения компоненты цвета.

1. 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.
   * 1. Индивидуальное задание
4. Создайте приложение Windows Forms в среде MS Visual Studio 2005.
5. Поместите элемент StatusStrip в главную форму в окне проектирования.
6. Поменяйте имя данного элемента управления на MainStatusStrip.
7. 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).

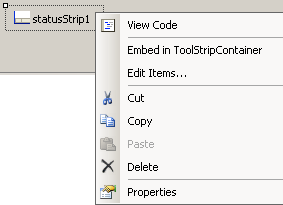


Рисунок 14.4 – Контекстное меню для выбора команды редактирования элементов строки состояния.

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

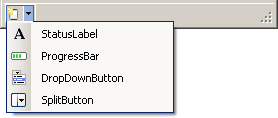
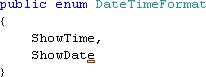


Рисунок 14.5 – Добавление элементов с помощью меню новых элементов.

1. 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».

1. 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.
2. Определите в проекте новый тип перечня DateTimeFormat, для выяснения того, что должен отображать второй элемент строки состояния – текущее время или текущую дату:



1. В классе главного окна определите следующие поля и методы:
   1. Поле dtFormat типа DateTimeFormat.
   2. Поле currentCheckedItem типа ToolStripMenuItem.
   3. Создайте обработчик события Tick для таймера timerDateTimeUpdate с именем timeDateTimeUpdate\_Tick.
   4. Выполните необходимые действия по присвоению начальных значений переменным dtFormat и currentCheckedItem.
   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.

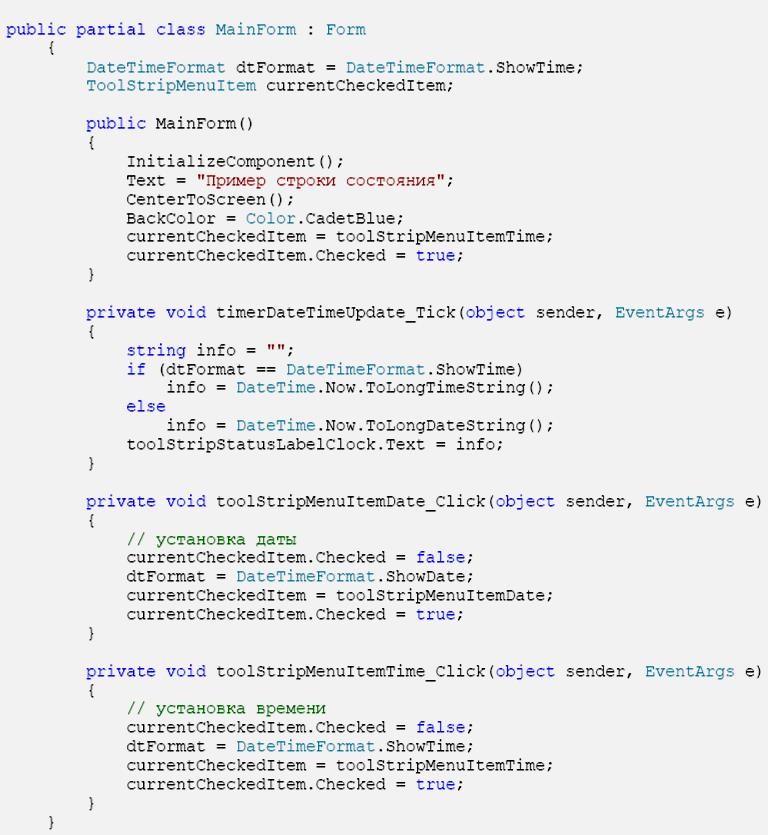


Рисунок 14.6 – Поля и методы класса главного окна, определяемые программистом.

1. Исправьте ошибки и запустите приложение.
2. 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*  *x*  *b*  *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  *x*  (*b*  *a*)  *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* 3 *x*  3  , *f*  *y* или *z*   *y* , где *x* и *y* – текущие  *y* 2  1 *y*  1  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*  *x*2  *b*  ( *y*  *x*)  lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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 *a*  *x*  *b*  *y*  sin( *z*) , где  *b x*  *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*  *x*  *y* 2 , где *x* и *y* – текущие координаты указателя  *y*  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 *a*  *z*  *b*  *y*  sin( *z*) , где  *a*  *b*  *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:  2  *z*  *x*3  *y* 2 , *f*  *x* или *z*  sin *x* , где *x* и *y* – текущие координаты  *y*3  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*  *x*  *b*  *y*  sin( *z* ) ,  *y 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: |

|  |  |
| --- | --- |
|  | *x*2  *y*3  *z*  или «квадраты координат курсора равны ( *x*2 , *y* 2 )», где *x* и  *y*3  *x*  *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 *a*  *x*  *b*  *y*  sin( *z*) ,  *b*  *y a*  *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*  *x*  *y* или «координаты курсора равны ( *x* , *y* )», где *x* и *y* –  *y*  *x*  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*  *x*  *b*  *y*  cos(*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*  *y* ,  *x*2  *f*  *x*  sin *y* 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. |
| 9 | **Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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*  *x*  *b*  *y*  sin( *z*) ,  lg( *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*  *x* , *f*  *x*  *y* или *z*  cos *x*  sin *y* , где *x* и *y* – текущие  *y*  *x*2  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 *a*  *x*  *b*  *y*  sin( *z*) ,  cos(*z*) *x*  *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  *x*4  пользователю выбрать один из вариантов расчета выражения: *z*   *y*  1  , *f*  *y* 3  *x* или *z*  *x*  *y* , 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 *a*  *x*  *b*  *y*  sin( *z*) ,  *z x*2  *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*  *x*4 ,  *f*  *y* 3 или *z*  *x*  *y* 2 , где *x* и *y* – текущие координаты указателя  *y*  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 *a*  *x*  *b*  *y*  sin( *z*) , где  *y x*  *y*2  *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* 3 *x*  3  , *f*  *y* или *z*   *y* , где *x* и *y* – текущие  *y* 2  1 *y*  1  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 *a*  *x*  *b*  *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*3 , *f*  *y* или *z*  *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 |  | | |  |
| *a*  *x*  | *b*  *y*  | *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. | | | | |
| 16 | **Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения *a*  *x*  *b*  *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  *x*  (*b*  *a*)  *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* 3 *x*  3  , *f*  *y* или *z*   *y* , где *x* и *y* – текущие  *y* 2  1 *y*  1  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*  *x*2  *b*  ( *y*  *x*)  lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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 *a*  *x*  *b*  *y*  sin( *z*) , где  *b x*  *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*  *x*  *y* 2 , где *x* и *y* – текущие координаты указателя  *y*  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 *a*  *z*  *b*  *y*  sin( *z*) , где  *a*  *b*  *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:  2  *z*  *x*3  *y* 2 , *f*  *x* или *z*  sin *x* , где *x* и *y* – текущие координаты  *y*3  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*  *x*  *b*  *y*  sin( *z* ) ,  *y 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:  *x*2  *y*3  *z*  или «квадраты координат курсора равны ( *x*2 , *y* 2 )», где *x* и  *y*3  *x*  *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 *a*  *x*  *b*  *y*  sin( *z*) ,  *b*  *y a*  *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*  *x*  *y* или «координаты курсора равны ( *x* , *y* )», где *x* и *y* –  *y*  *x*  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*  *x*  *b*  *y*  cos(*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*  *y* ,  *x*2  *f*  *x*  sin *y* 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. |
| 24 | **Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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*  *x*  *b*  *y*  sin( *z*) ,  lg( *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*  *x* , *f*  *x*  *y* или *z*  cos *x*  sin *y* , где *x* и *y* – текущие  *y*  *x*2  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 *a*  *x*  *b*  *y*  sin( *z*) ,  cos(*z*) *x*  *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  *x*4  пользователю выбрать один из вариантов расчета выражения: *z*   *y*  1  , *f*  *y* 3  *x* или *z*  *x*  *y* , 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 *a*  *x*  *b*  *y*  sin( *z*) ,  *z x*2  *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*  *x*4 ,  *f*  *y* 3 или *z*  *x*  *y* 2 , где *x* и *y* – текущие координаты указателя  *y*  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*  *x*  *b*  *y*  sin( *z*) , где  *y x*  *y*2  *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* 3 *x*  3  , *f*  *y* или *z*   *y* , где *x* и *y* – текущие  *y* 2  1 *y*  1  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 *a*  *x*  *b*  *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*3 ,  *f*  *y* или *z*  *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 |  | | |  |
| *a*  *x*  | *b*  *y*  | *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. | | | | |
| 31 | **Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения *a*  *x*  *b*  *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  *x*  (*b*  *a*)  *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* , *f*  *y*3 или *z*  *x*  *y* , где *x* и *y* – текущие  3  *y* 2  1 *y*  1  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*  *x*2  *b*  ( *y*  *x*)  lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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 *a*  *x*  *b*  *y*  sin( *z*) , где  *b x*  *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*  *x*  *y* 2 , где *x* и *y* – текущие координаты указателя  *y*  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 *a*  *z*  *b*  *y*  sin( *z*) , где  *a*  *b*  *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:  2  *z*  *x*3  *y* 2 , *f*  *x* или *z*  sin *x* , где *x* и *y* – текущие координаты  *y*3  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*  *x*  *b*  *y*  sin( *z* ) ,  *y 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:  *x*2  *y*3  *z*  или «квадраты координат курсора равны ( *x*2 , *y* 2 )», где *x* и  *y*3  *x*  *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 *a*  *x*  *b*  *y*  sin( *z*) ,  *b*  *y a*  *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*  *x*  *y* или «координаты курсора равны ( *x* , *y* )», где *x* и *y* –  *y*  *x*  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*  *x*  *b*  *y*  cos(*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*  *y* ,  *x*2  *f*  *x*  sin *y* 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. |
| 39 | **Задание 1.** Создайте и синхронизируйте пункты главного и контекстного меню для вычисления выражения lg( *a*  *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*  *y*  *x* , *f*  *x*  *y* или *z*  cos *x*2  sin 2 *y* , где *x* и *y* – текущие  *x*  *y*  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*  *x*  *b*  *y*  sin( *z*) ,  lg( *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*  *x* , *f*  *x*  *y* или *z*  cos *x*  sin *y* , где *x* и *y* – текущие  *y*  *x*2  координаты указателя мыши. Значение выбранного выражения на  второй панели должно изменяться при смене положения курсора мыши. |

* 1. **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

* 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.
  1. Теоретическое обоснование
     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, ContextMenu, ContextMenuStrip | Menultem, MenuStrip, | 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  стандартных элементов управления | в | форму |

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.

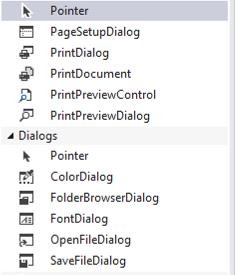


Рисунок 15.1 – Компоненты, обеспечивающие работу с встроенными диалоговыми окнами.

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).

* 1. **Методика и порядок выполнения работы**
     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.

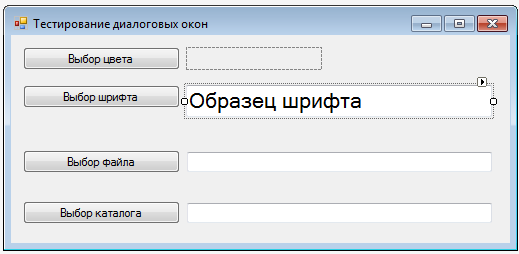


Рисунок 15.2 – Внешний вид экранной формы для работы со стандартными диалоговыми окнами

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. |

1. Создайте обработчики нажатия каждой кнопки.

На рис. 15.3 представлен листинг обработчика нажатия btnChooseColor.

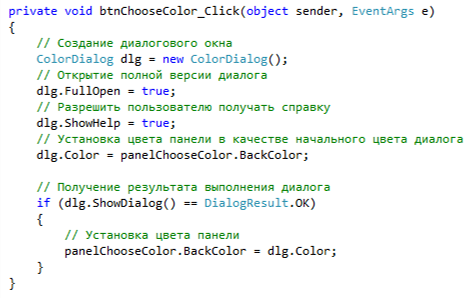


Рисунок 15.3 – Обработчик btnChooseColor

На рис. 15.4 представлен обработчик выбора шрифта.

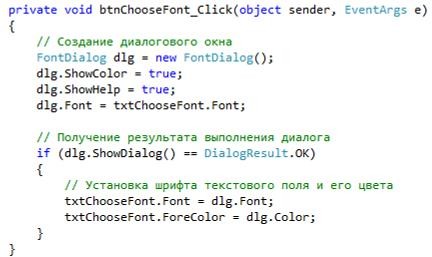


Рисунок 15.4 – Обработчик btnChooseFont

На рис. 15.5 представлен код обработчика нажатия кнопки btnChooseFolder.

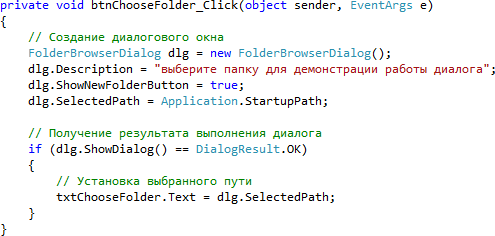


Рисунок 15.5 – Обработчик btnChooseFolder На рис. 15.6 показан диалог выбора файла.

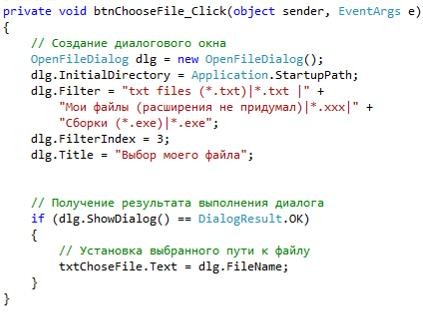


Рисунок 15.6 – Обработчик btnChooseFile

1. Реализуйте все обработчики, запустите приложение, убедитесь в его работоспособности. Измените различные настройки диалоговых окон.
   * 1. 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:  cos2 (*x*  *t*) sin 3 (*x*  *z*) cos4 (*x*  *t*) sin 5 (*x*  *z*)  *U*  1      ..., если  2 3 4 5  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 2 | Вычислить значение выражения:  2 3 4 5  *U*  1  cos (*x*)  sin (*x*)  cos (*x*)  sin (*x*)  ... , если количество слагаемых в  2! 3! 4! 5!  правой части выражения пользователь вводит с клавиатуры. |
| 3 | Вычислить значение выражения:  sin( *x*)  *x*2 *x*2  sin( *x*3 ) sin( *x*3 )  *x*4 *x*4  sin( *x*5 )  *y*      ..., если  2! 3! 4! 5!  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 4 | 2 4 6 8  Вычислить значение выражения: *h*  *x*  sin( *x* )  *x*  sin( *x* )  ... , если  1 3 3  5 5  7 7  9  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 5 | Вычислить значение выражения:  *U*  1 sin (*x*)  *y*  sin (*x*)  *y*  sin (*x*)  *y*  sin (*x*)  *y* ... , если количество  2 3 3 4 5 5 7    2 3 4 5  слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 6 | Вычислить значение выражения:  2 3 4 5  *Z*  1  sin( *x* )  cos( *x* )  sin( *x* )  cos( *x* )  ... , если количество слагаемых в  2 3 4 5  правой части выражения и целое *x* пользователь вводит с клавиатуры. |

|  |  |
| --- | --- |
| 7 | Вычислить значение выражения:  2 3 4 5  *U*  1 sin (*x*)  sin (*x*)  sin (*x*)  sin (*x*) ..., если количество слагаемых в  2 3 4 5  правой части выражения пользователь вводит с клавиатуры. |
| 8 | 2 3 4 5  Вычислить значение выражения: *V*  *x*  *x*  *x*  *x*  ... , если  2! 3! 4! 5!  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 9 | 2 3 4 5  Вычислить значение выражения: *s*   *x*  *x*  *x*  *x*  ..., если  2  3 3  4 4  5 5  6  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 10 | Вычислить значение выражения:  *y*  *x*  *y*  *x*  *y*  *x*  *y*  *x*  ,  3 2 5 4 7 6 9 8  *h* ...  1 3 3 5 5  7 7  9  если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 11 | 3 5 7  Вычислить значение выражения: *p*   *x*  *x*  *x*  *x*  ...  1 2  3 3  4  5 5  6  7 7  8  9  если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 12 | 3 5 7  Вычислить значение выражения: *T*   *x*  *x*  *x*  *x*  ..., если  1 3 3  5 5  7 7  9  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 13 | Вычислить значение выражения:  3 5 2 7 3 9 4  *j*   sin (*x*)  sin (*x* )  sin (*x* )  sin (*x* )  ... , если количество слагаемых  1 3 3  5 5  7 7  9  в правой части выражения пользователь вводит с клавиатуры. |
| 14 | Вычислить значение выражения:  2 3 4 5  *Z*  1  sin( *x* )  cos( *x* )  sin( *x* )  cos( *x* )  ... , если количество слагаемых в  2 3 4 5  правой части выражения и целое *x* пользователь вводит с клавиатуры. |

|  |  |
| --- | --- |
| 15 | *y*  *x*2 *z*  *x*3 *y*  *x*4 *z*  *x*5  Вычислить значение выражения: *s*       ...  1 3 2  4 3  5 4  6  , если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 16 | *x*2 *y*3 *x*4 *y*5  Вычислить значение выражения: *F*       ..., если  2  3 3  4 4  5 5  6  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 17 | Вычислить значение выражения:  sin(*t*)  lg( *a*) ln(*t* 3 )  cos(*a*2 ) sin(*t* 5 )  lg( *a*3 ) ln(*t* 7 )  cos(*a*4 )  *A*       ...  1! 3! 5! 7!  если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 18 | Вычислить значение выражения:  sin( *y*)  *x*2 sin( *x*3 ) cos( *y*)  *x*4 sin( *x*5 ) sin( *y*)  *x*6  *E*        ..., если  2  3 3  4 4  5 5  6 4  5  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 19 | Вычислить значение выражения:  *t*  sin(*a*) *t* 3  cos(*a*2 ) *t* 5  sin(*a*3) *t* 7  cos(*a*4 )  *p*       ... если  1 2 3  4 5  6 7  8  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 20 | Вычислить значение выражения:  *y*  *x y* 2  *x*3 *y*3  *x*5 *y* 4  *x*7  *p*       ... если количество слагаемых  1 2  3 3  4  5 5  6  7 7  8  9  в правой части выражения пользователь вводит с клавиатуры. |
| 21 | Вычислить значение выражения:  cos2 (*x*  *t*) sin 3 (*x*  *z*) cos4 (*x*  *t*) sin 5 (*x*  *z*)  *U*  1      ..., если  2 3 4 5 |

|  |  |
| --- | --- |
|  | количество слагаемых в правой части выражения пользователь вводит  с клавиатуры. |
| 22 | Вычислить значение выражения:  2 3 4 5  *U*  1  cos (*x*)  sin (*x*)  cos (*x*)  sin (*x*)  ... , если количество слагаемых в  2! 3! 4! 5!  правой части выражения пользователь вводит с клавиатуры. |
| 23 | Вычислить значение выражения:  sin( *x*)  *x*2 *x*2  sin( *x*3) sin( *x*3)  *x*4 *x*4  sin( *x*5 )  *y*      ..., если  2! 3! 4! 5!  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 24 | 2 4 6 8  Вычислить значение выражения: *h*  *x*  sin( *x* )  *x*  sin( *x* )  ... , если  1 3 3  5 5  7 7  9  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 25 | Вычислить значение выражения:  *U*  1 sin (*x*)  *y*  sin (*x*)  *y*  sin (*x*)  *y*  sin (*x*)  *y* ... , если количество  2 3 3 4 5 5 7    2 3 4 5  слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 26 | Вычислить значение выражения:  2 3 4 5  *Z*  1  sin( *x* )  cos( *x* )  sin( *x* )  cos( *x* )  ... , если количество слагаемых в  2 3 4 5  правой части выражения и целое *x* пользователь вводит с клавиатуры. |
| 27 | Вычислить значение выражения:  2 3 4 5  *U*  1 sin (*x*)  sin (*x*)  sin (*x*)  sin (*x*) ..., если количество слагаемых в  2 3 4 5  правой части выражения пользователь вводит с клавиатуры. |

|  |  |
| --- | --- |
| 28 | 2 3 4 5  Вычислить значение выражения: *V*  *x*  *x*  *x*  *x*  ... , если  2! 3! 4! 5!  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 29 | 2 3 4 5  Вычислить значение выражения: *s*   *x*  *x*  *x*  *x*  ..., если  2  3 3  4 4  5 5  6  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 30 | Вычислить значение выражения:  *y*  *x*  *y*  *x*  *y*  *x*  *y*  *x*  ,  3 2 5 4 7 6 9 8  *h* ...  1 3 3 5 5  7 7  9  если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 31 | 3 5 7  Вычислить значение выражения: *p*   *x*  *x*  *x*  *x*  ...  1 2  3 3  4  5 5  6  7 7  8  9  если количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 32 | 3 5 7  Вычислить значение выражения: *T*   *x*  *x*  *x*  *x*  ..., если  1 3 3  5 5  7 7  9  количество слагаемых в правой части выражения пользователь вводит с клавиатуры. |
| 33 | Вычислить значение выражения:  3 5 2 7 3 9 4  *j*   sin (*x*)  sin (*x* )  sin (*x* )  sin (*x* )  ... , если количество слагаемых  1 3 3  5 5  7 7  9  в правой части выражения пользователь вводит с клавиатуры. |
| 34 | Вычислить значение выражения:  2 3 4 5  *Z*  1  sin( *x* )  cos( *x* )  sin( *x* )  cos( *x* )  ... , если количество слагаемых в  2 3 4 5  правой части выражения и целое *x* пользователь вводит с клавиатуры. |
| 35 | *y*  *x*2 *z*  *x*3 *y*  *x*4 *z*  *x*5  Вычислить значение выражения: *s*       ...  1 3 2  4 3  5 4  6  , если количество слагаемых в правой части выражения пользователь  вводит с клавиатуры. |
| 36 | *x*2 *y*3 *x*4 *y*5  Вычислить значение выражения: *F*       ..., если  2  3 3  4 4  5 5  6  количество слагаемых в правой части выражения пользователь вводит  с клавиатуры. |

|  |  |
| --- | --- |
| 37 | Вычислить значение выражения:  sin(*t*)  lg( *a*) ln(*t* 3 )  cos(*a*2 ) sin(*t* 5 )  lg( *a*3 ) ln(*t* 7 )  cos(*a*4 )  *A*       ...  1! 3! 5! 7!  если количество слагаемых в правой части выражения пользователь  вводит с клавиатуры. |
| 38 | Вычислить значение выражения:  sin( *y*)  *x*2 sin( *x*3 ) cos( *y*)  *x*4 sin( *x*5 ) sin( *y*)  *x*6  *E*        ..., если  2  3 3  4 4  5 5  6 4  5  количество слагаемых в правой части выражения пользователь вводит  с клавиатуры. |
| 39 | Вычислить значение выражения:  *t*  sin(*a*) *t* 3  cos(*a*2 ) *t* 5  sin(*a*3) *t* 7  cos(*a*4 )  *p*       ... если  1 2 3  4 5  6 7  8  количество слагаемых в правой части выражения пользователь вводит  с клавиатуры. |
| 40 | Вычислить значение выражения:  *y*  *x y* 2  *x*3 *y*3  *x*5 *y* 4  *x*7  *p*       ... если количество слагаемых  1 2  3 3  4  5 5  6  7 7  8  9  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**

* 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.
  1. **Теоретическая часть**

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

* 1. **Методика и порядок выполнения работы**
     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.

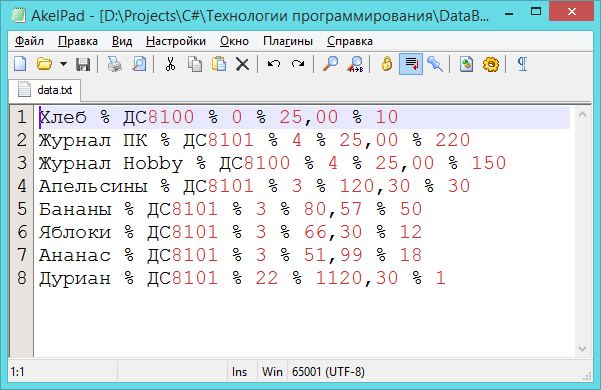


Рисунок 16.1 – Фрагмент исходного файла

**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.

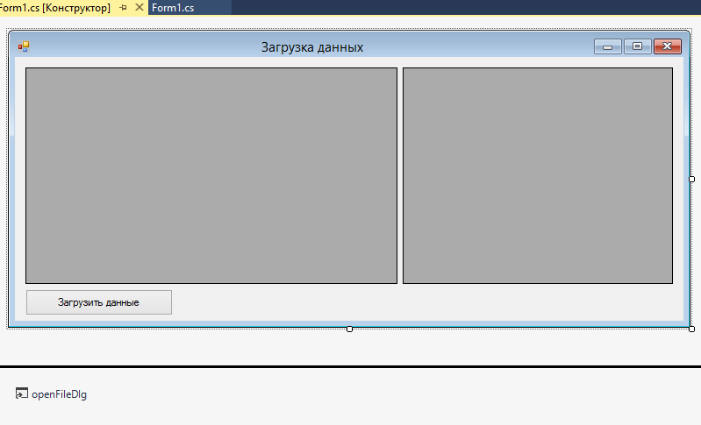


Рисунок 16.2 – Внешний вид экранной формы Form1

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 |

1. 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 |

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

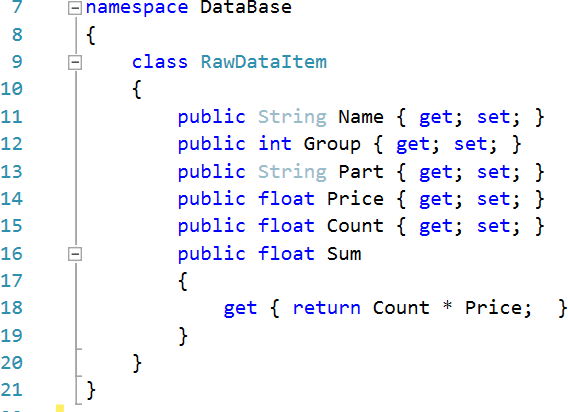


Рисунок 16.3 – Исходный код класса RawDataItem

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

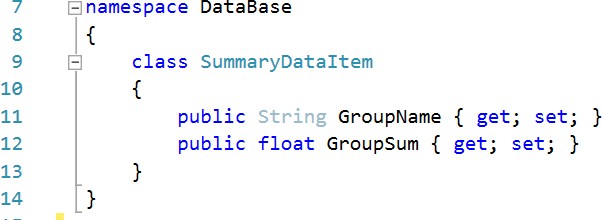


Рисунок 16.4 – Исходный код класса SummaryDataItem

1. Реализуйте класс Utils (рисунок 16.5).

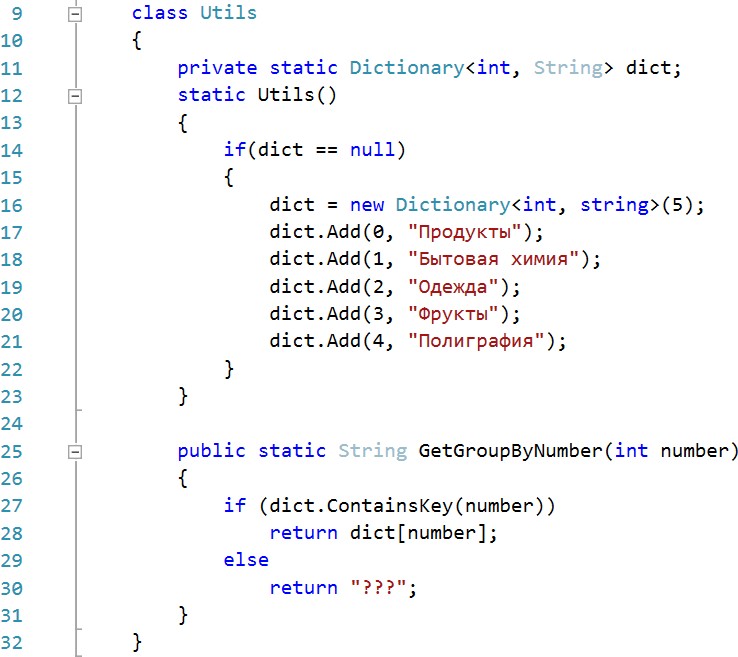


Рисунок 16.5 – Исходный код класса Utils

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.

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

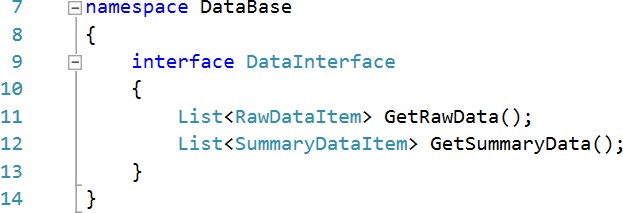


Рисунок 16.6 – Исходный код интерфейсного типа DataInterface Интерфейс содержит два метода:

1. GetRawData() – возвращает данные для таблицы dgvRaw;
2. GetSummaryData() – возвращает данные для таблицы dgvSummary.
3. Рассмотрим определение класса DataStorage (рис. 16.7).

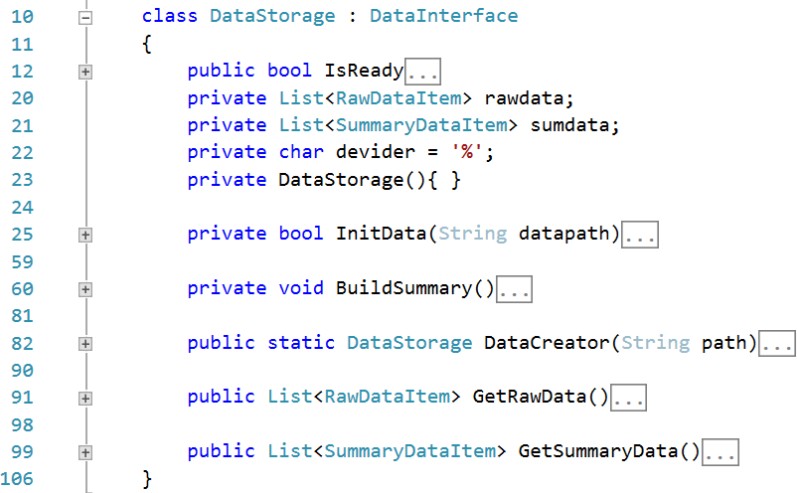


Рисунок 16.7 – Исходный код класса DataStorage

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.

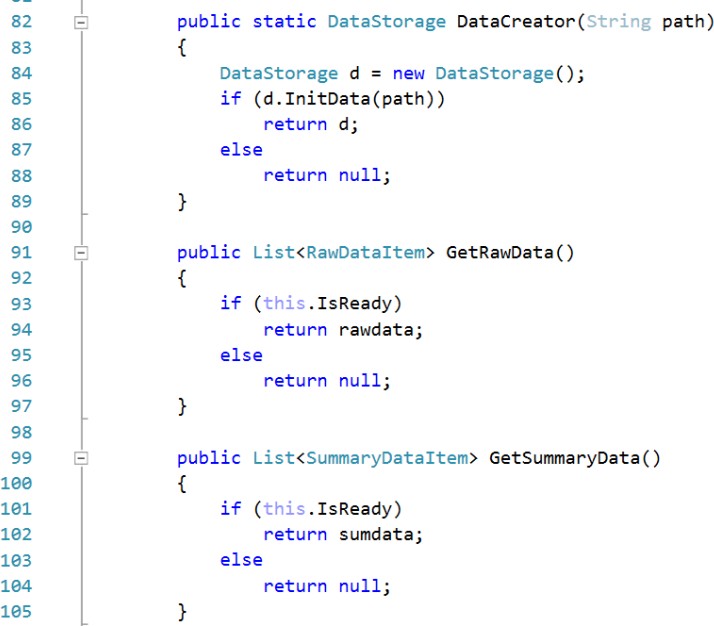
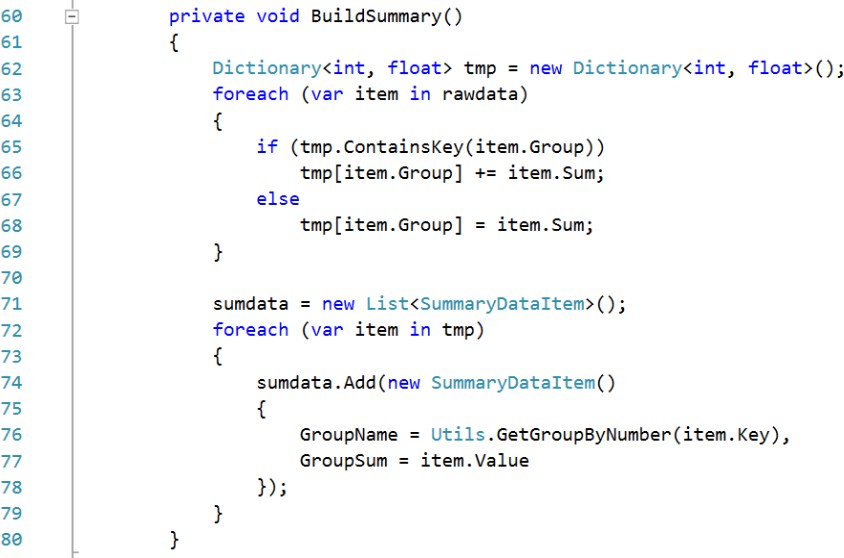
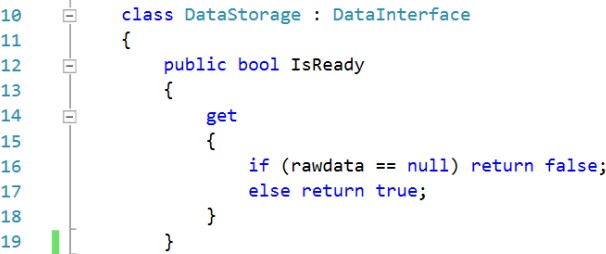


Рисунок 16.8 – Листинг класса DataStorage

1. В классе Form1 добавьте обработчик события btnLoad\_Click( ) и метод ShowData( ). Листинги методов представлены на рисунках 16.9 и 16.10.

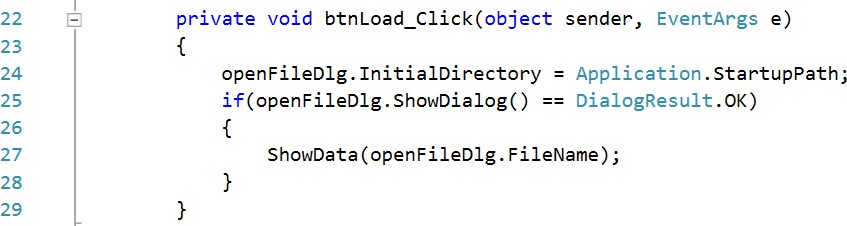


Рисунок 16.9 – Обработчик нажатия кнопки btnLoad

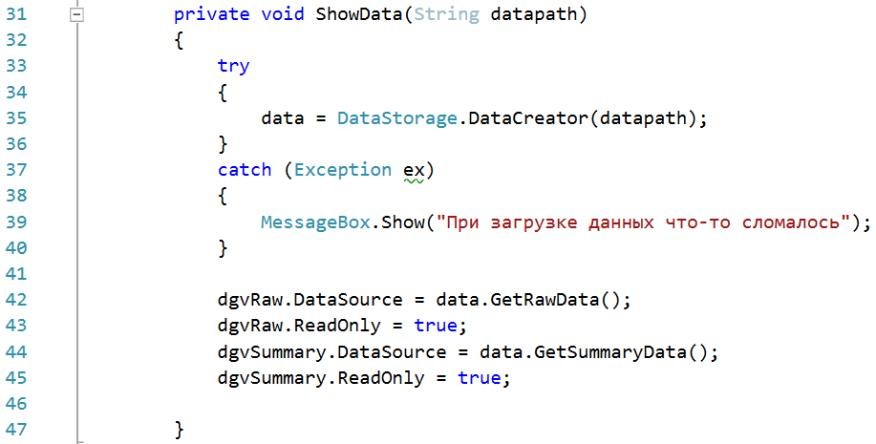


Рисунок 16.10 – Метод ShowData класса Form1

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

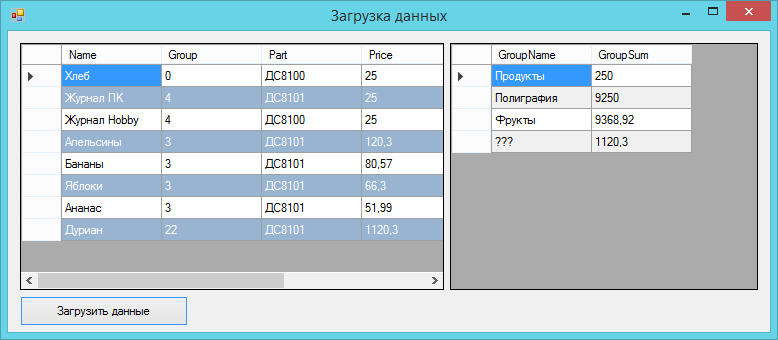


Рисунок 16.11 – Внешний вид окна приложения в режиме выполнения

* + 1. 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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом. |

* 1. **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

* 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.
  1. **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:

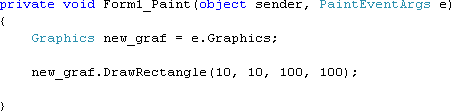


Рисунок 17.1 – Получение объекта Graphics через параметр PaintEventArgs обработчика события Paint

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.

* 1. **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*  *x* *p*  sin(k x b).

Для решения поставленной задачи реализуйте следующие действия:

1. Создайте приложение Windows Forms.
2. Спроектируйте форму в соответствии с представленным шаблоном (рис. 10.2).

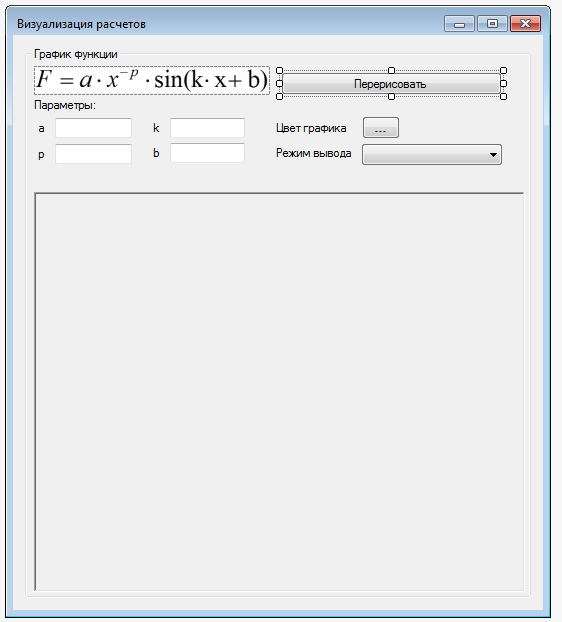


Рисунок 10.2 – Проект формы приложения

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. |

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

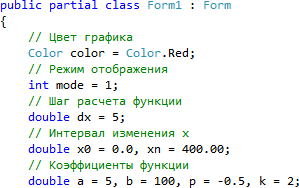


Рисунок 10.3 – Добавление полей класса

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

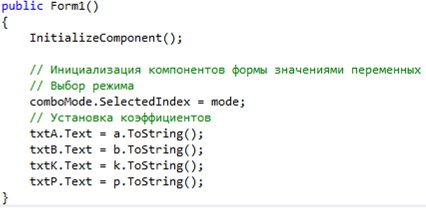


Рисунок 10.4 – Конструктор класса

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

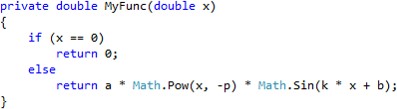


Рисунок 10.5 – Метод для вычисления значений функции

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

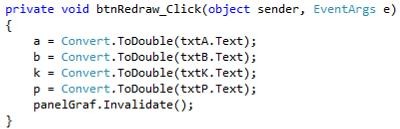


Рисунок 10.6 – Обработчик нажатия кнопки btnRedraw

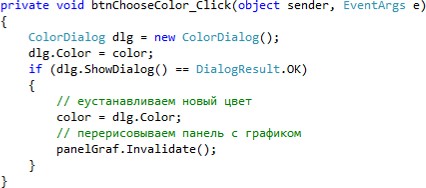


Рисунок 10.7 – Обработчик нажатия кнопки btnChooseColor

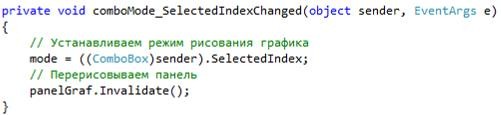


Рисунок 10.8 – Обработчик выбора пункта в comboMode

Внешний вид экранной формы разработанного приложения представлен на рис. 10.9

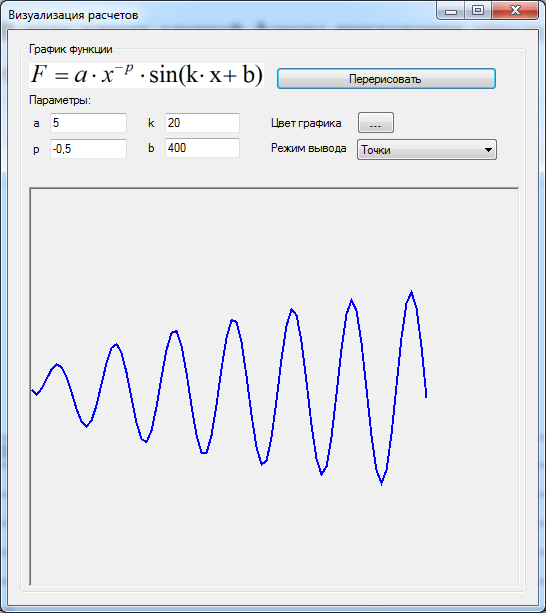


Рисунок 10.9 – Приложение в режиме выполнения

1. Запустите приложение. Протестируйте работу программы при различных значениях параметров.
2. Доработайте приложение: реализуйте механизм вывода графика отдельными точками, так как в представленном варианте график выводится в виде отрезков, соединяющих отдельные точки графика.
3. Разработайте приложение, в котором визуализируются данные сложной функции (учащийся может определить график функции самостоятельно).
   * 1. Анимация средствами GDI+
4. Создайте приложение Windows Forms.
5. Дополните класс экранной формы вспомогательными переменными (рис. 17.10).

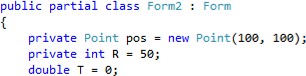


Рисунок 17.10 – Переменные класса формы

1. Перетащите из панели инструментов компонент Timer на экранную форму. Присвойте ему имя MainTimer. Установите свойство Interval в значение 500.
2. Модифицируйте обработчик события Paint экранной формы (рис. 17.11).

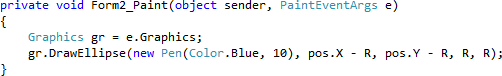


Рисунок 17.11 – Обработчик события Paint

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

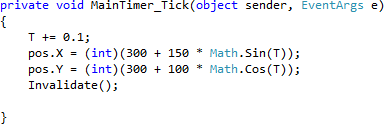


Рисунок 17.12 – Обработчик события Tick

1. 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).

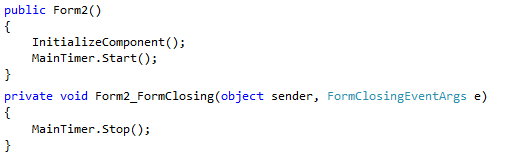


Рисунок 17.13 – Запуск и остановка таймера

* + 1. 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).

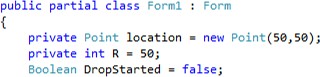


Рисунок 17.14 – Служебные переменные класса

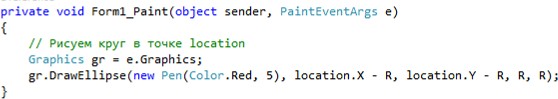


Рисунок 17.15 – Обработчик Paint

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

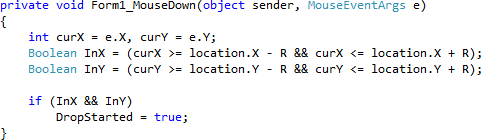


Рисунок 17.16 – Обработчик MouseDown

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

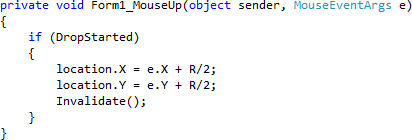


Рисунок 17.17 – Обработчик MouseUp

1. Запустите приложение, протестируйте механизм перетаскивания объектов.

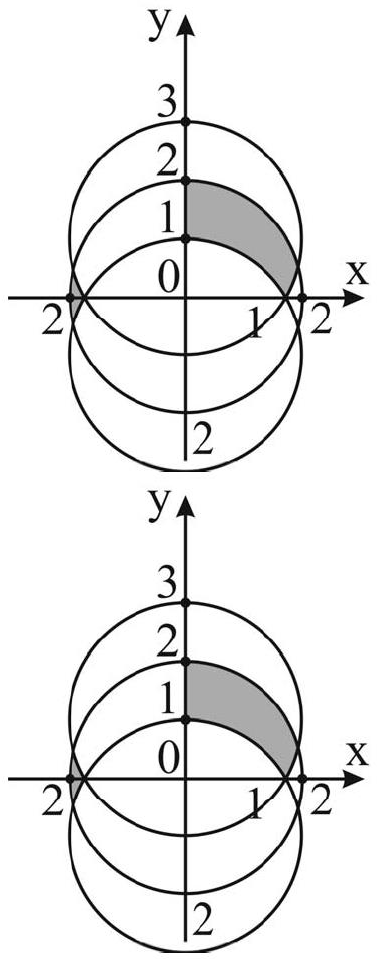
17.3.4. Индивидуальное задание

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 |  |

* 1. 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 соытия мыши над фигурами GDI+.
7. Для чего предназначе 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 интрактивного 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 рпограмме?

# ЛАБОРАТОРНАЯ РАБОТА 18. ПРИЛОЖЕНИЯ WINDOWS PRESENTATION FOUNDATION

* 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.
  1. **Теоретическое обоснование**
     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 Тор. 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.

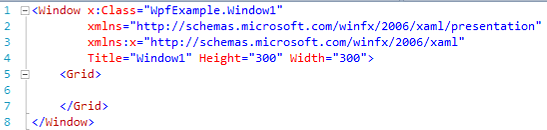


Рисунок 18.1 – Шаблон окна приложения, генерируемого по умолчанию.

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.

* + 1. 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.

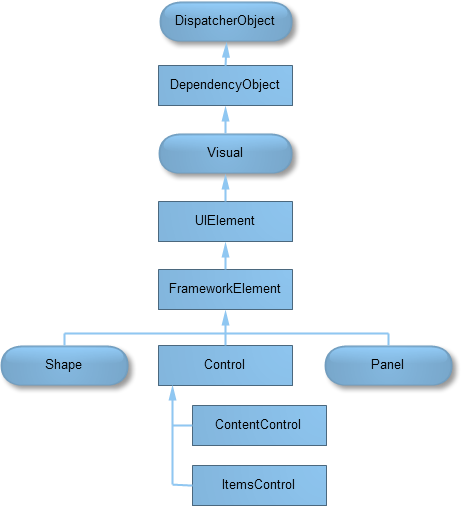


Рисунок 18.2 – Иерархия классов WPF

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. |

* + 1. Контейнер StackPanel

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



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.

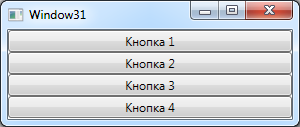


Рисунок 18.4 – Окно с контейнером StackPanel

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>

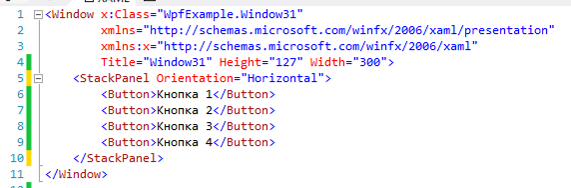


Рисунок 18.5 – Использование StackPanel Результат представлен на рис. 18.6

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. |

* + 1. 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.

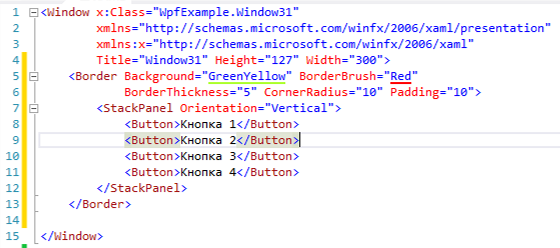


Рисунок 18.7 – Использование свойств Border На рис. 18.8 показано результирующее окно.

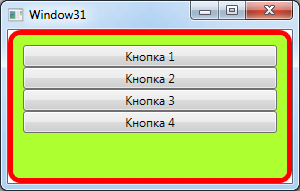


Рисунок 18.8 – Результат применения Border

* + 1. 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.

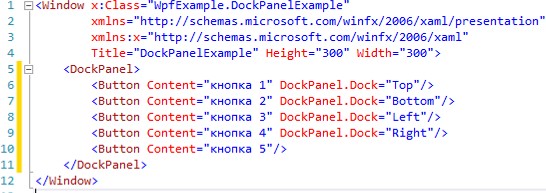
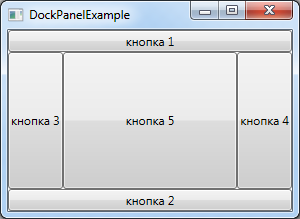


Рисунок 18.9 – Использование DockPanel На рис. 18.10 представлен результат.

Рисунок 18.10 – Контейнер DockPanel в окне приложения



* + 1. 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:



Рисунок 18.11 – Контейнер Grid Результирующее окно представлено на рис. 18.12.

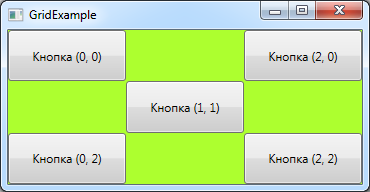


Рисунок 18.12 – Расположение кнопок в контейнере Grid

* 1. **Методика и порядок выполнения работы**
     1. Учебная задача

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

Условие. Реализовать приложение для вычисления выражения:

sin( *f*

 *a*), *f* {4,5, 6, 7,8,9};

cos( *f*  *a*)  sin( *f*

 *b*),

*f* {10, 20,30, 40};

*S*   *2 2*

*c*  *a + d*  *b , c* {0,1}, *d* {1, 0,1};

 *d*



 (*c*  *a*)*i* , *d*  *const*, *c* {0,1, 2,3, 4,5}.

 *i* 0

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.

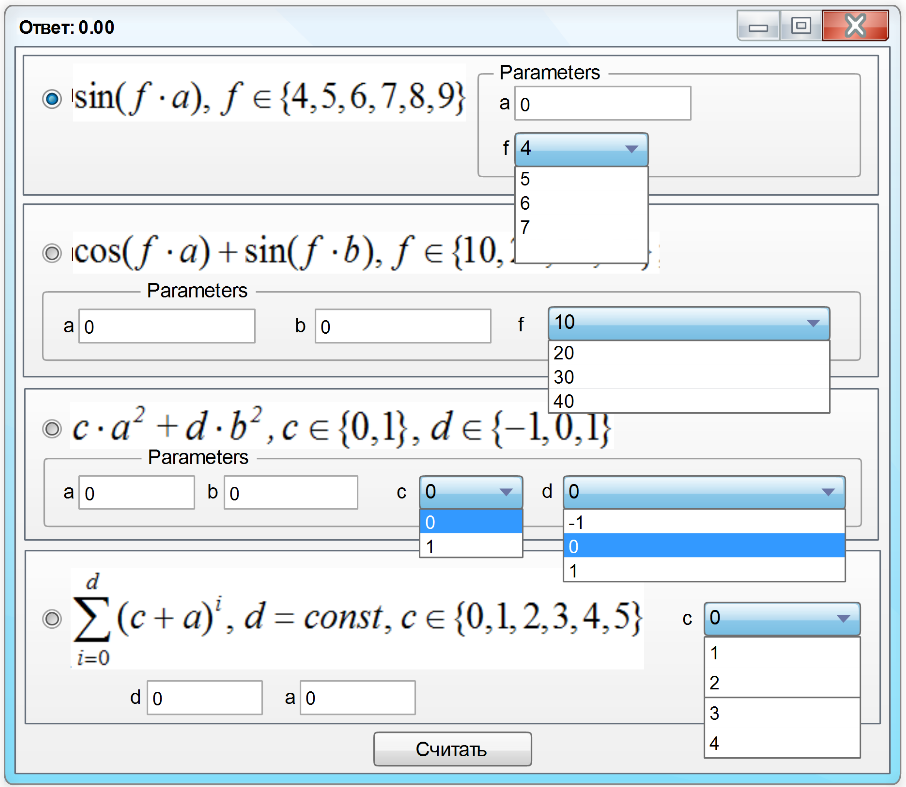


Рисунок 16.13 – Проект формы: приблизительное расположение элементов управления формы

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.

1. 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:



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).

1. 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.

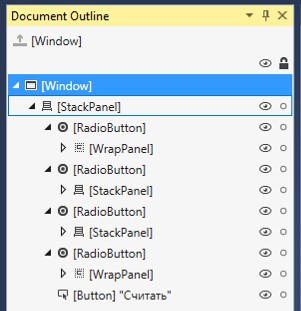


Рисунок 18.15 – Просмотр иерархии элементов главного окна

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



Рисунок 18.16 – Дескриптор RadioButton для представления первой формулы

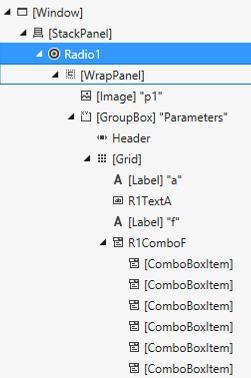


Рисунок 18.17 – Дескриптор RadioButton (иерархическое представление)

* 1. 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».



Рисунок 18.18 – Дескриптор RadioButton для представления второй формулы

* 1. На рис. 16.19 представлен код XAML для третьего дескриптора RadioButton.



Рисунок 18.19 – Дескриптор RadioButton для представления третьей формулы

* 1. Затем наполним содержимым четвертый дескриптор RadioButton (рис. 18.20).



Рисунок 18.20 – Дескриптор RadioButton для представления четвертой формулы

1. 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.

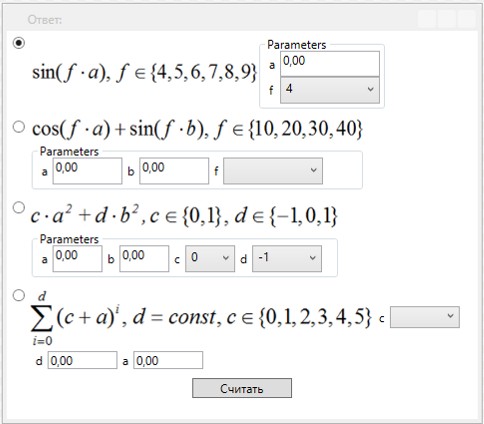


Рисунок 18.21 – Главное окно приложения в режиме проектирования

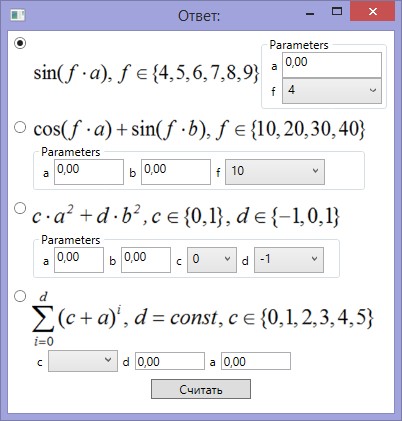


Рисунок 18.22 – Главное окно приложения в режиме выполнения

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



Рисунок 18.23 – Добавление обработчика нажатия кнопки.

В результате выполнения данного действия Visual Studio откроет редактор кода С# 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:

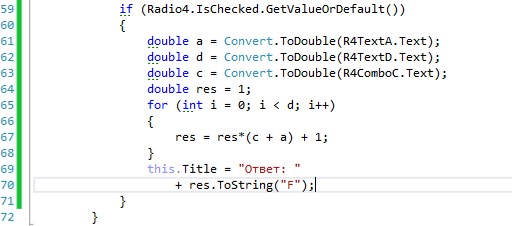
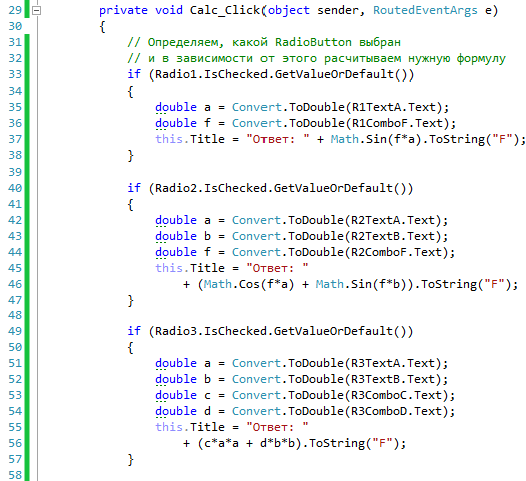


Рисунок 18.24 – Код обработчика нажатия кнопки «Считать» Представленный код простой и в пояснениях не нуждается.

1. Запустите приложение. Протестируйте работу программы при различных значениях входных параметров.
2. 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.
   * 1. Индивидуальное задание

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | *N K pi*  *x*3  *f j*  *y*2  *Z*   *i*  *j* .  *i* 1 *j* 1 | 2. | *N K ti*  *xi*  *f j*  *y j*  *Z*   *t*  *i*  *j* .  *i* 1 *j* 1 |
| 3. | *N K pi*  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 | 4. | *N K* sin(*x*)  *xi*  *f j*  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 |
| 5. | *N K* sin(*x*)  *xi*  cos( *y*)  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 | 6. | *N K xi*  cos( *y*)  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| 7. | *N K i*  *xi*  *j*  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 | 8. | *N K i*  cos(*xi* )  *j*  sin( *y j* )  *Z*   *i*  *j*  *i* 1 *j* 1 |
| 9. | *N K* cos(*xi* )  *j Z*   *i*  *j*  *i* 1 *j* 1 | 10. | *N K* cos( *yi* )  *j*  *x Z*   *i*  *j*  *i* 1 *j* 1 |
| 11. | *N K* sin( *yi* )  *i*  *x Z*   (*i*  1)  *j*  *i* 1 *j* 1 | 12. | *Z*  *N* *K* sin( *y*  *x*)  *i*  *y*  *i*1 *j*1 (*i*  1)  ( *j*  2) |
| 13. | *N K* 2  *xi*  3  *y j*  *Z*   *i*2  *j*2  *i* 1 *j* 1 | 14. | *N K i*  *x*2*i*  *j*  *y*2 *j*  *Z*   *i*2  *j*2  *i* 1 *j* 1 |
| 15. | *N K pi* 1  *x*  *f j* 1  *y*2  *Z*   *i*  *j*  *i* 1 *j* 1 | 16. | *N K ai* 1  *xi*  *f j*  *y j*  *Z*   (*i*  1)  *j*  *i* 1 *j* 1 |
| 17. | *Z*  *N* *K* 5  *x*  3  *y*  *i* 1 *j* 1 *i*2  *j*2 | 18. | *N K* cos( *yi* )  *j*  sin(*x*) *Z*   *i*  ( *j* 1)  *i* 1 *j* 1 |
| 19. | *N K xi*  cos( *y*)  *y j*  *Z*   (*i* 1)  *j*  *i* 1 *j* 1 | 20. | *N K* ( *p* 1)*i*  *y j*  *Z*   *i*  *j*  *i* 1 *j* 1 |

* 1. **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

* 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.
  1. **Теоретическое обоснование**
     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.



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

* + 1. 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).

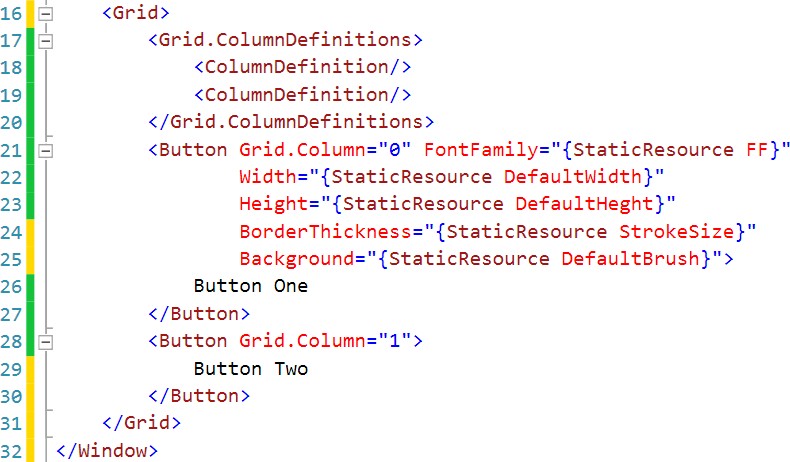
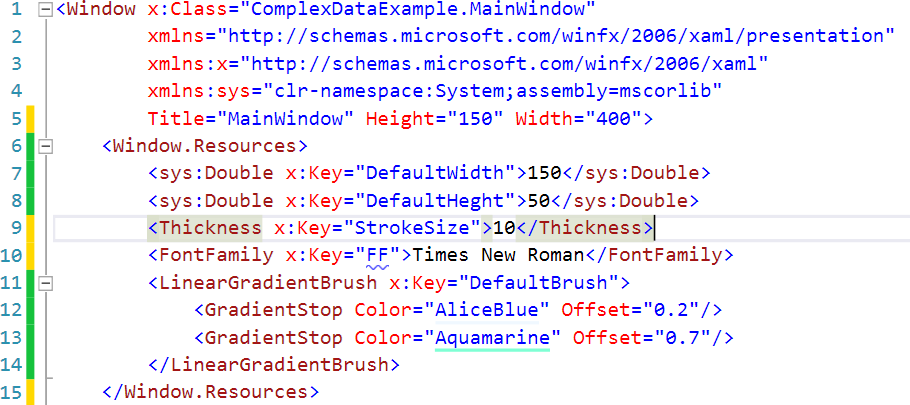


Рисунок 19.3 – Определение внешнего вида элементов управления с использованием ресурсов

Результат подобной разметки представлен на рис. 19.4.

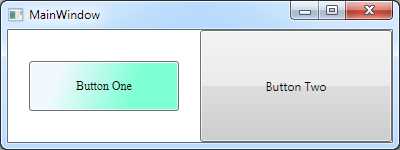


Рисунок 19.4 – Внешний вид элементов управления при использовании ресурсов

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.



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">

* + 1. 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.



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).



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.

* 1. **Методика и порядок выполнения работы**
     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.



Рисунок 19.8 – Разметка XAML для каркаса приложения

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

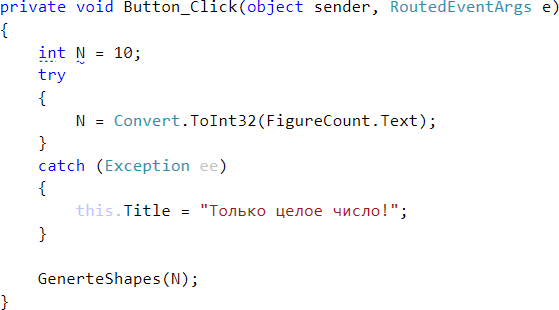


Рисунок 19.9 – Обработчик события нажатия кнопки «Generate Shapes»

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

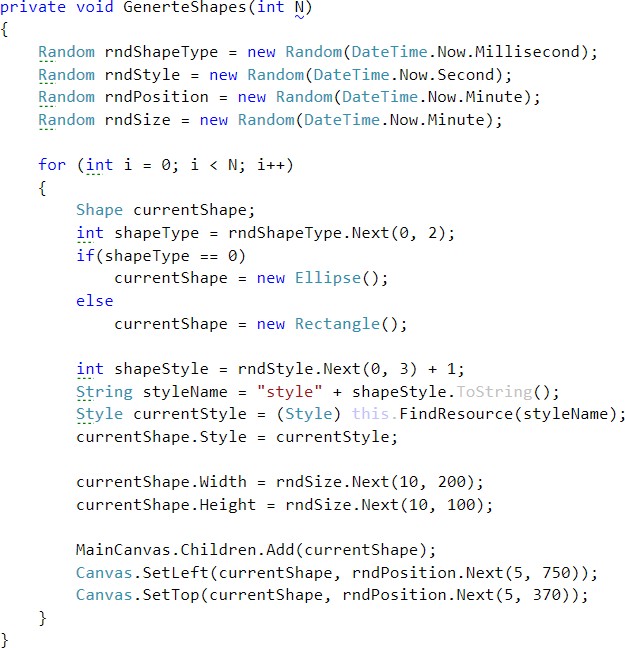


Рисунок 19.10 – Реализация функции GenerateShapes(int N) Результат (скриншот) работающего приложения показан на рис 19.11.

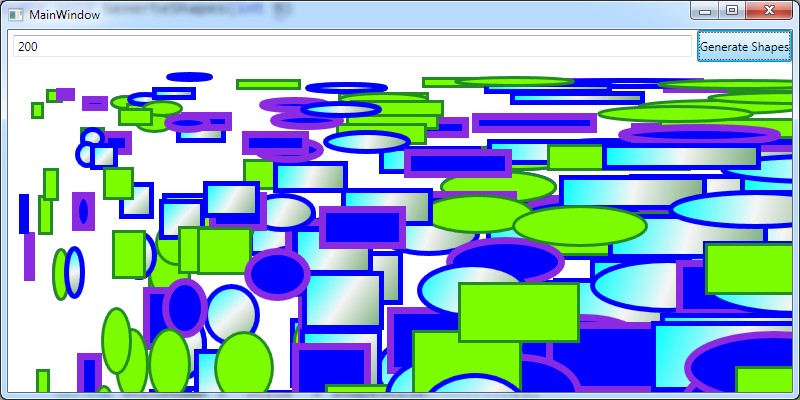


Рисунок 19.11 – Скриншот работающего приложения

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

* + 1. 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.

* 1. **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

* 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.
  1. **Теоретическое обоснование**
     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.



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#.

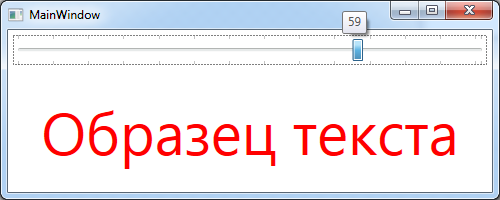


Рисунок 20.3 – Результат применения привязки

* + 1. 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.



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».

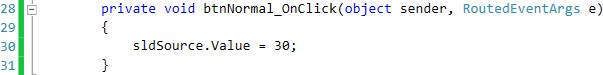


Рисунок 20.5 – Обработчик нажатия кнопки «Нормальный размер», устанавливающий значение Value ползунка sldSource

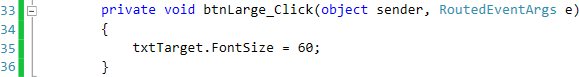


Рисунок 20.6 – Обработчик нажатия кнопки «Крупный размер», устанавливающий значение FontSize элемента txtTarget.

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):

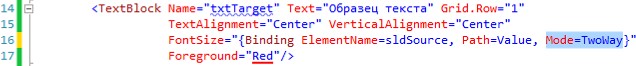


Рисунок 20.7 – Явное указание режима привязки.

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.

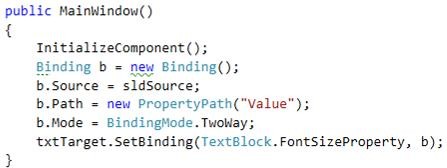


Рисунок 20.9 – Реализация привязки на языке C#

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).

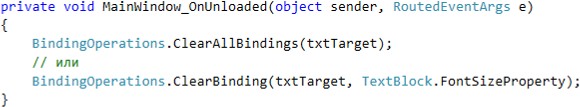


Рисунок 20.10 – Удаление привязки в коде C#

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.
   * 1. 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.



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.

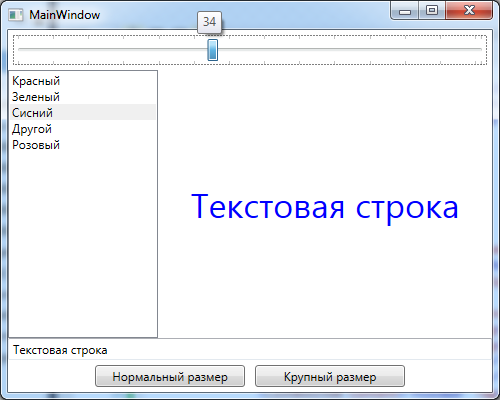


Рисунок 20.12 – Применение множественной привязки

* 1. **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.
  1. **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

* 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.
  1. **Теоретическое обоснование**
     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).

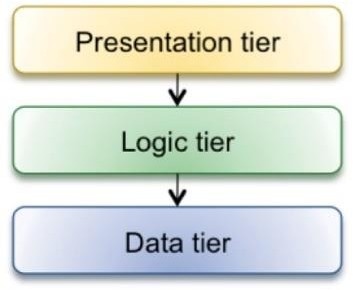


Рисунок 21.1 – Multy tier Application

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

* + 1. 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, основываяь 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.

* 1. **Методика и порядок выполнения работы**
     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).

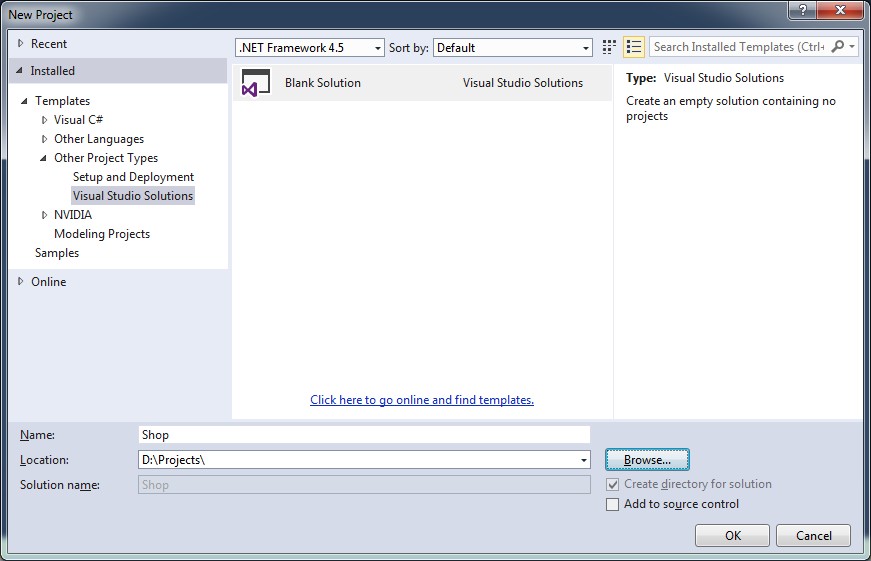


Рисунок 21.1 – Создание пустого решения Visual Studio

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

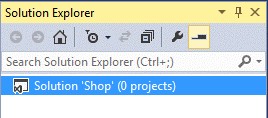


Рисунок 21.2 – Вид обозревателя проекто после создания пустого решения

«Shop»

1. 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.

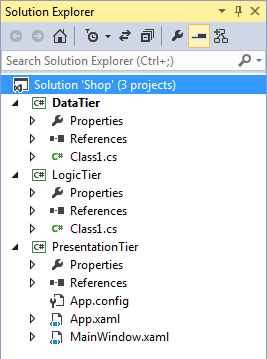


Рисунок 21.3 – Вид обозревателя проектов после добавления отдельных проектов

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.

1. Программирование уровня LogicTier. Цель данного слоя приложения – связать уровни DataTier и PresentationTier.
2. 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».

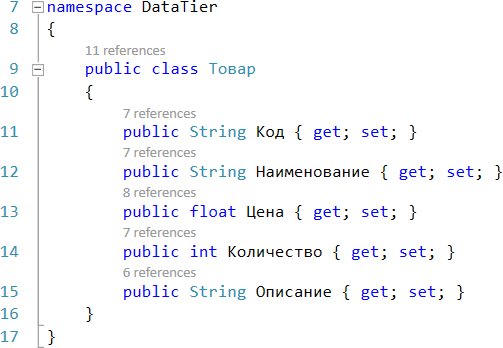


Рисунок 21.4 – Определение класса «Товар»

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.

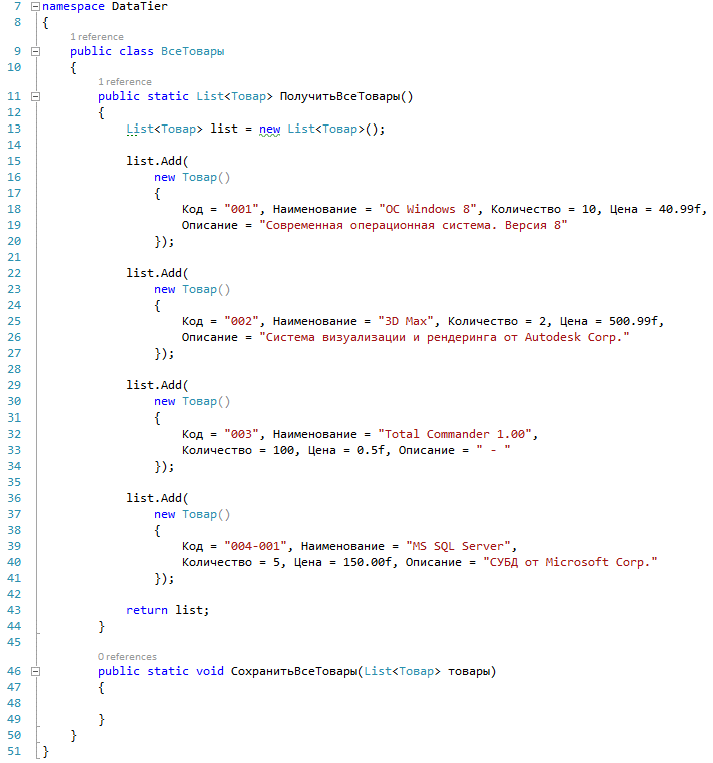


Рисунок 21.5 – Определение класса «ВсеТовары»

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.

1. 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.

1. 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.

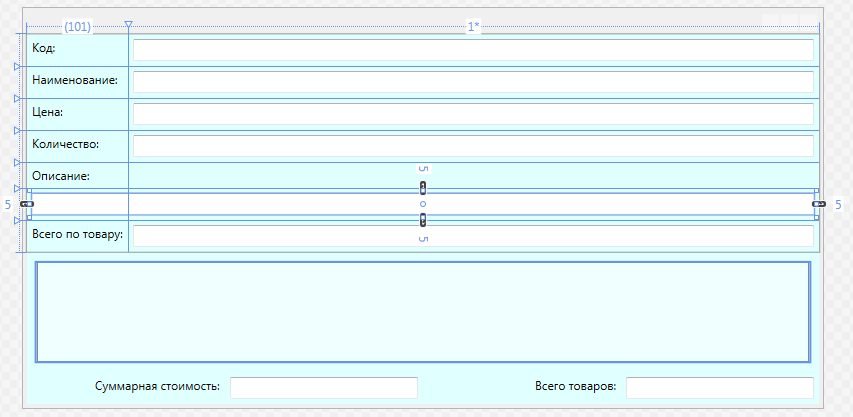


Рисунок 21.6 – Дизайн MainWindow.xaml Рисунок 21.7 иерархию элементов управления в главном окне.

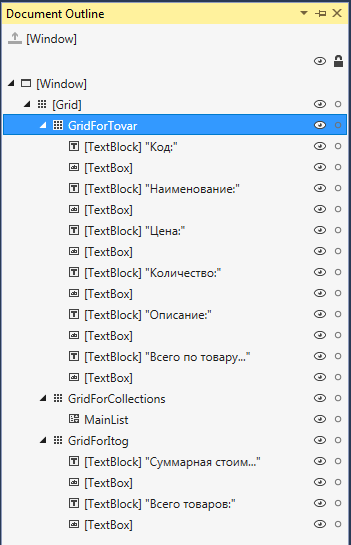


Рисунок 21.7 – Иерархическая структура элементов в окне MainWindow.xaml

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.



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.

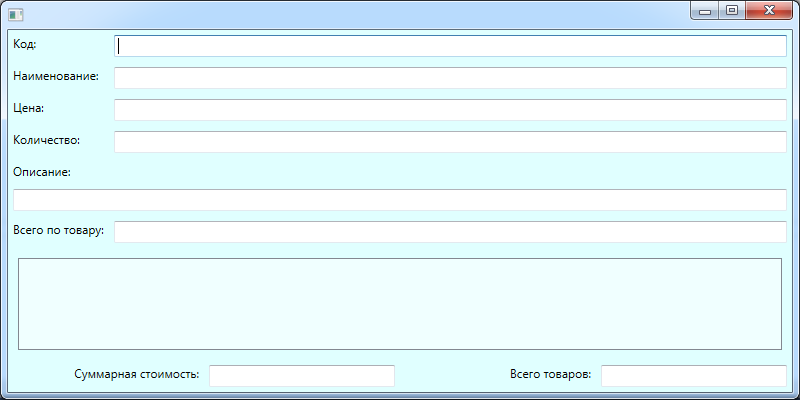


Рисунок 21.9 – Главное окно приложения в режиме выполнения

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.

1. 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».

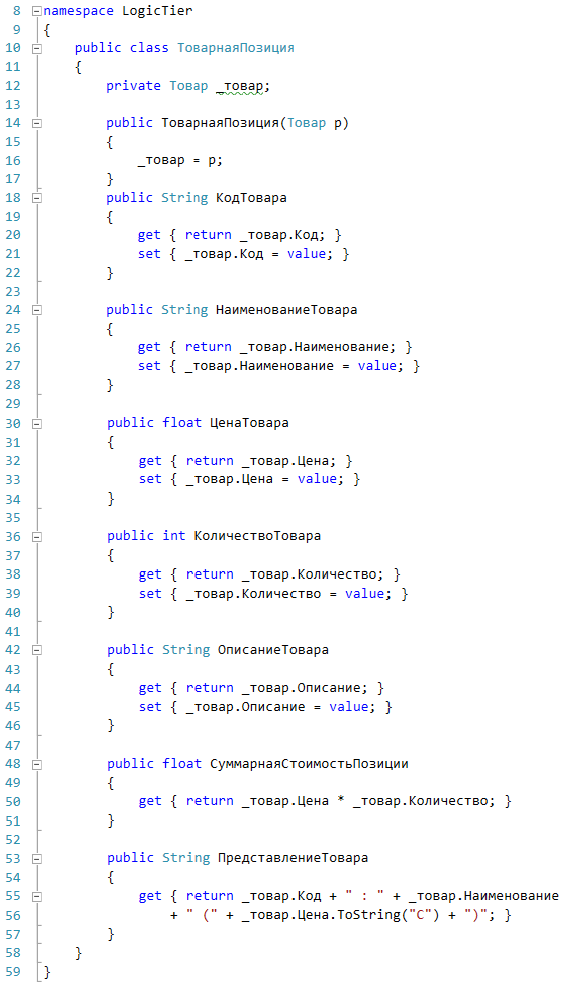


Рисунок 21.10 – Код класса «ТоварнаяПозиция»

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.



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.

1. 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.

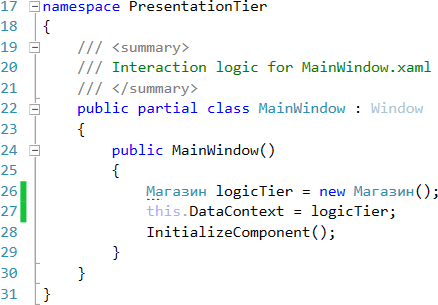


Рисунок 21.12 – Конструктор MainWindow

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.

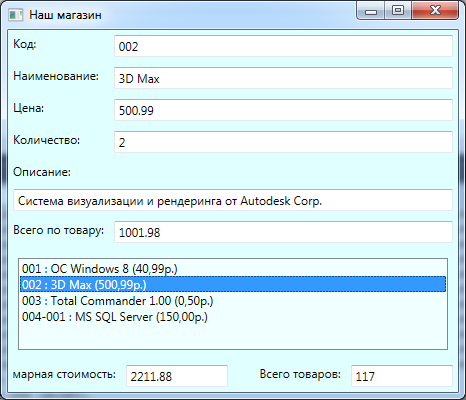


Рисунок 21.13 – Результат решения задачи

1. Выполните задание в соответствии с индивидуальным вариантом.
   * 1. 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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  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):  Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой. |

* 1. **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?

Учебное пособие (лабораторный практикум) по дисциплине

«Технологии программирования» Часть 1 (лабораторные работы 1-11) для студентов направления

09.03.02 «Информационные системы и технологии» Составитель: Е.И. Николаев