

СОДЕРЖАНИЕ

Введение.....	4
Общие требования к выполнению лабораторных работ.....	6
Общие сведения о MS VISUAL STUDIO.....	8
Лабораторная работа № 1.....	15
Лабораторная работа № 2.....	39
Лабораторная работа № 3.....	58
Лабораторная работа № 4.....	67
Лабораторная работа № 5.....	75
Библиографический список.....	82

Введение

Настоящий сборник содержит задания для лабораторного практикума по дисциплине "Компонентное программирование".

Компонентным (компонентно-ориентированным) программированием называется отрасль разработки программных систем, подразумевающая использование гетерогенных в общем случае компонентов в рамках единого программного комплекса. Иными словами, отдельные программные компоненты (модули) могут разрабатываться в различных средах и с использованием различных языков программирования, а затем использоваться совместно, интегрированно. Такого рода проблематика не надуманна, а является прямым следствием усложнения программных систем, необходимости масштабного повторного использования разработок, гетерогенности и распределенности самих социальных, профессиональных и организационных структур разработчиков.

Реализация интероперабельности и интеграции гетерогенных компонентов возможна при соблюдении принципов и технологий компонентного программирования. Рядом мировых лидеров рынка инструментальных средств были разработаны свои подходы и комплексы технологий для поддержки принципов компонентной разработки – это комплекс Java-технологий и конкретно концепция JavaBeans от компании Sun, модель CORBA, линейка технологий Microsoft DDE/OLE/COM/DCOM/MTS/ActiveX/ADO/COM+/.NET. Последний подход, а именно разработка на платформе Microsoft .NET в среде MS Visual Studio, предлагается к изучению в настоящем пособии.

Компонентно-ориентированное программирование (КОП) основывается на объектно-ориентированном программировании (ООП). Вообще, для современного разработчика программных систем недостаток навыков использования ООП, как минимум, существенно сужает сферу его профессионального применения. На принципах ООП основаны все основные технологии и средств-

ва не только разработки, но и вообще поддержки жизненного цикла информационных систем (работа с требованиями, конфигурациями, тестами, документацией и т.п.), организационные аспекты формирования этого жизненного цикла, принципы формирования проектной документации.

Поэтому предлагаемый комплекс лабораторных работ построен достаточно универсально. Он ориентирован на студентов, начинающих овладевать языком C# и средой разработки MS Visual Studio. Поэтому первая работа посвящена изучению основных программных синтаксических конструкций этого языка. Программирование в C# – это всегда программирование в классах, в концепции ООП. Однако в первой работе акцента на объектно-ориентированную разработку не делается.

Вторая работа предназначена для накопления опыта решения штатных задач (к которым здесь отнесена работа с файлами и строками) в C#. Третья работа посвящена изучению концепции структур, логически предшествующей концепции классов.

ООП как работа с иерархиями классов рассматривается в четвертой работе, в рамках которой предполагаются создание классов и практическое рассмотрение свойств полиморфизма, инкапсуляции и наследования.

Наконец, пятая работа посвящена рассмотрению простейшей реализации КОП. Она предполагает создание динамически связываемой библиотеки (DLL), а затем использование ее.

Общие требования к выполнению лабораторных работ

Выполнение лабораторной работы включает 3 этапа:

1) самостоятельное выполнение студентом задания на лабораторную работу в соответствии с вариантом, выданным преподавателем, и формирование отчета;

2) демонстрация выполненной работы и отчета преподавателю;

3) защита лабораторной работы.

Защита подразумевает выполнение дополнительного задания по модификации полученной конфигурации, а также ответы на контрольные вопросы, предложенные преподавателем.

В случае невыполнения студентом дополнительного задания в течение лабораторной работы, на которой это задание было выдано, на следующей лабораторной работе студенту предлагается выполнить новое дополнительное задание.

В случае повторного неудовлетворительного выполнения студентом дополнительного задания или неудовлетворительных ответов на контрольные вопросы преподаватель может принять решение о замене выданного варианта задания на лабораторную работу на новое.

Требования к отчету:

1) отчет должен содержать: титульный лист; цель работы; задание на лабораторную работу в соответствии с вариантом; листинг всего написанного программного кода; экранные формы созданных объектов и печатные формы полученных отчетов с отображением на них результатов работы, если таковые имеются; выводы;

2) отчет должен быть выполнен в печатном виде;

3) отчет составляется на листах А4 в одностороннем формате, т.е. одна сторона листа всегда должна быть чистой (свободная сторона отводится для письменных ответов на дополнительные вопросы);

4) отчет должен быть выполнен чисто и аккуратно; текст должен быть легко читаемым, лаконичным, не должен содержать разговорных слов и выражений;

5) листинг должен являться копией кода, используемого во время демонстрации результатов работы преподавателю;

6) листинг не должен содержать закомментированного кода, а также кода, не имеющего отношения к данной лабораторной работе и не влияющего на результаты работы;

7) листинг кода каждой функции должен начинаться с закомментированного описания назначения этой функции.

ОБЩИЕ СВЕДЕНИЯ О MS VISUAL STUDIO

Visual Studio 2013 – созданная компанией Microsoft интегрированная среда разработки (IDE) для создания, запуска и отладки приложений на различных языках программирования NET.

Создание нового проекта

Разработка приложения в Visual C# начинается с создания нового или открытия существующего проекта. Чтобы создать новый проект, выполните команду Файл->Создать проект, а чтобы открыть существующий – команду Файл->Открыть проект... Проект представляет собой группу взаимосвязанных файлов. В Visual C# приложения организуются в проекты (projects) и решения (solutions), содержащие один или несколько проектов. Решения используются при создании крупномасштабных приложений.

Диалоговое окно New Project и шаблоны проектов

При выполнении команды Файл->Создать проект на начальной странице открывается диалоговое окно Создать проект (рис. 1.1). Visual Studio предоставляет разработчику несколько шаблонов (см. рис. 1.1) – типов проектов, которые могут создаваться в Visual C# и других языках. В их число входят шаблоны приложений Windows Forms, приложений WPF и многих других – полная версия Visual Studio содержит много дополнительных шаблонов. Приложение Windows Forms выполняется в операционной системе Windows (например, Windows 7 или Windows 8) и обычно обладает графическим интерфейсом (GUI), с которым взаимодействуют пользователи. По умолчанию Visual Studio присваивает новому проекту и решению на базе шаблона Windows Forms Application имя WindowsFormsApplication1 (см. рис. 1.1).

Выберите шаблон Windows Forms и щелкните на кнопке ОК, чтобы перевести IDE в режим конструирования (рис 1.2). Функции этого режима предназначены для создания графиче-

ского интерфейса приложения. Не забудьте переименовать проект и указать его расположение.

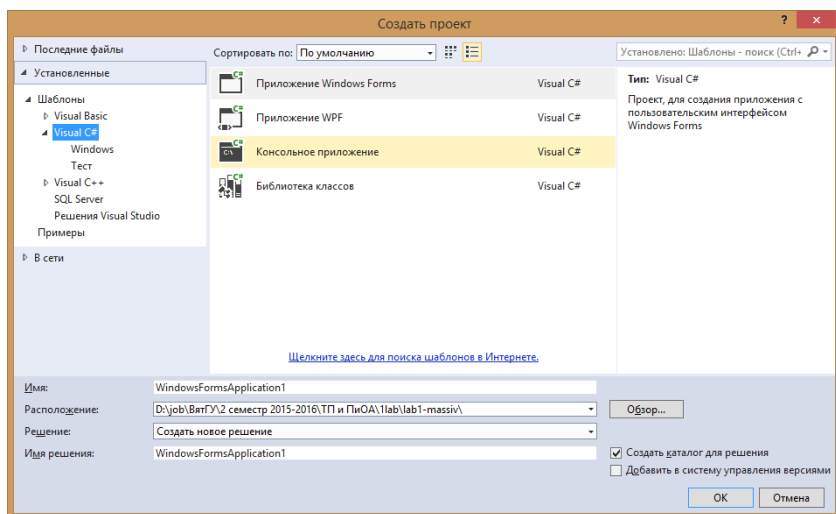


Рис. 1.1. Диалоговое окно Создать проект

Формы и элементы управления

Прямоугольник с заголовком Form1 (форма) представляет главное окно создаваемого приложения Windows Forms. Приложения Visual C# могут содержать несколько форм (окон). Формы можно изменять, добавляя к ним элементы управления, например надпись (Label) и графическое поле (PictureBox). Надписи обычно содержат текст с описанием, а в графическом поле выводится изображение. Visual Studio содержит много готовых элементов управления и других компонентов, которые могут использоваться для создания и настройки поведения приложений.

Разместив элемент на форме, разработчик может изменить его Свойства либо программно, либо вручную, выбрав соответствующую позицию в панели свойств и задав требуе-

мые настройки. Форма и элементы управления образуют графический интерфейс приложения. Пользователь набирает данные на клавиатуре, щелкает кнопками мыши или вводит их другими способами. Графический интерфейс используется для отображения инструкций и другой информации, предназначенной для пользователя. Имя каждого открытого документа выводится на корешке вкладки. Чтобы просмотреть документ при наличии нескольких открытых документов, щелкните на соответствующем корешке. Активная вкладка (вкладка с документом, отображаемым в настоящее время) выделена синим цветом (например, Form1.cs [Конструктор] на рисунке 1.2.).

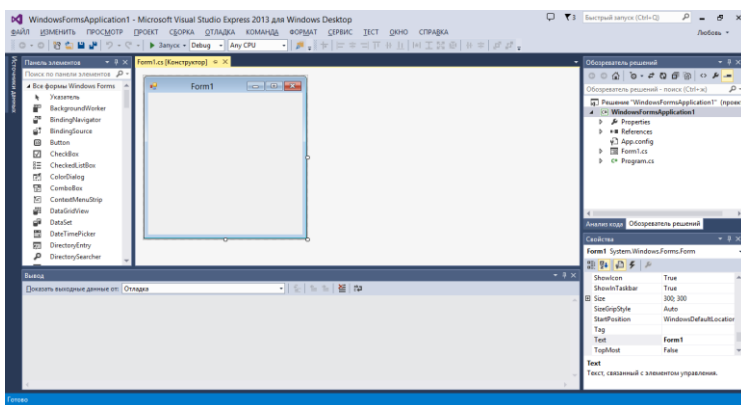


Рис. 1.2. Режим конструирования (Design view) в IDE

Перемещение в Visual Studio IDE

IDE предоставляет окна для работы с файлами проекта и настройки элементов управления. В этом разделе описаны некоторые окна, часто используемые при разработке приложений Visual C#. Чтобы обратиться к любому из этих окон, выберите его имя в меню Просмотр.

Solution Explorer

Окно Solution Explorer (рисунок 1.3) предоставляет доступ ко всем файлам приложения. Если оно не отображается в IDE, выполните команду Просмотр->Обозреватель решений. Когда вы открываете новое или существующее решение, его содержимое выводится в окне Solution Explorer.

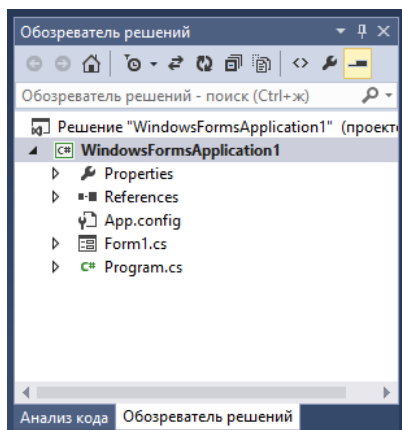


Рис. 1.3. Окно Solution Explorer с открытым проектом

Для решений с одним проектом стартовый проект является единственным (в данном случае Windows Forms Application1). Имя стартового проекта выделяется жирным шрифтом в окне Solution Explorer. При первом создании приложения окно Solution Explorer выглядит так, как показано на рисунке 1.3. Файл Visual C#, соответствующий форме называется Form1.cs. Файлы Visual C# используют расширение .cs. По умолчанию в IDE отображаются только те файлы, которые вам, возможно, придется редактировать; другие файлы, сгенерированные IDE, остаются скрытыми. В окне Solution Explorer присутствует панель инструментов с несколькими кнопками. Кнопка Show All Files отображает все файлы решения, включая сгенерированные IDE. Щелчок на стрелке слева от узла разворачивает или сворачивает этот узел. Эта схема также используется в других окнах Visual Studio.

Панель элементов

Чтобы вызвать окно панели элементов (Toolbox), выполните команду Просмотр->Панель элементов. В нем содержатся элементы, используемые для модификации форм (рис. 1.4). В методологии визуального программирования разработчик «перетаскивает» элементы управления на форму, а IDE генерирует код создания этих элементов. Так код создается проще и быстрее, чем при самостоятельном написании.

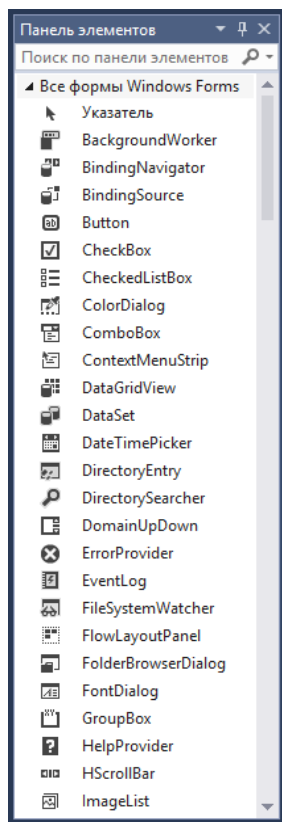


Рис. 1.4. Окно «Панель элементов» с элементами управления

Готовые элементы управления на панели элементов сгруппированы по категориям.

Окно свойств

Если окно свойств (Properties) не отображается под окном Solution Explorer, выполните команду Просмотр->Окно свойств. В окне свойств (рис. 1.5) выводятся свойства текущей выделенной формы, элемента управления или файла. Свойства описывают характеристики формы или элемента управления: размер, цвет, позицию и т. д. Разные формы и элементы управления обладают разными наборами свойств – описание свойства выводится в нижней части окна свойств при выделении этого свойства.

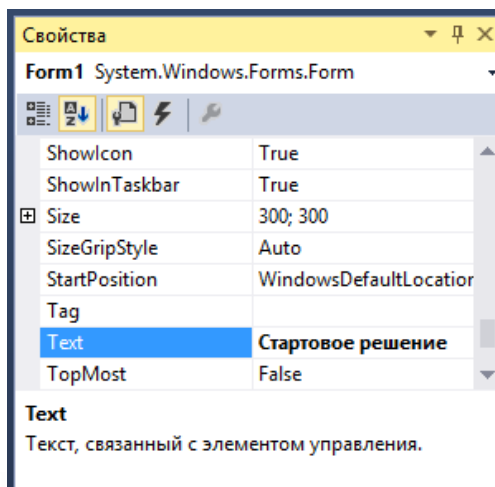


Рис. 1.5. Окно свойств

На рисунке 1.5 изображено окно свойств формы Form1. В левом столбце перечислены свойства формы, а в правом выводятся текущие значения каждого свойства. Список свойств можно отсортировать либо по алфавиту (кнопка Alphabetical), либо по категориям (кнопка Categorized). В зависимости от размера окна свойств некоторые свойства могут не поместиться на экране. Чтобы прокрутить список свойств, используйте полосу прокрутки или кнопки со стрелками у верхнего и нижнего края полосы. В окне свойств выводится краткое описание

выделенного свойства, помогающее понять его предназначение. Свойство можно быстро задать в этом окне, без написания какого-либо кода. В верхней части окна свойств расположен раскрывающийся список для выбора компонентов. В нем можно выбрать форму или элемент управления, свойства которых вы хотите вывести в окне свойств, без выделения этой формы/элемента управления в графическом интерфейсе.

ЛАБОРАТОРНАЯ РАБОТА №1

ФОРМИРОВАНИЕ И ОБРАБОТКА ЧИСЛОВЫХ МАССИВОВ

ОДНОМЕРНЫЕ МАССИВЫ

Массивы занимают место в памяти. Так как они являются объектами, обычно они создаются ключевым словом `new`. При создании объекта массива тип и количество его элементов указываются в выражении создания массива, использующем ключевое слово `new`. Такое выражение возвращает ссылку, которая сохраняется в переменной массива.

Следующее объявление и выражение создания массива создает объект массива с 12 элементами и сохраняет ссылку на него в переменной `c`:

```
int[] c = new int[ 12 ];
```

Инициализация элементов массива.

```
int[] c; // Объявление переменной массива
c = new int[ 12 ]; // Создание массива; присваивание
переменной массива
```

В этом объявлении квадратные скобки за типом `int` указывают, что переменная `c` будет ссылаться на массив с элементами `int` (то есть в `c` будет храниться ссылка на объект массива). Во второй команде переменной массива `c` присваивается ссылка на новый объект массива с 12 элементами `int`. Количество элементов также может задаваться выражением, вычисляемым на стадии выполнения. При создании массива каждому элементу присваивается значение по умолчанию – 0 для простых числовых типов, `false` для элементов `bool` и `null` для ссылок.

Массивы имеют фиксированную длину, вы можете использовать статический метод `Resize` класса `Array` с двумя аргументами (массив и новая длина) для создания нового массива заданной длины. Метод копирует содержимое старого массива в новый массив и задает переменной, полученной в первом аргументе, ссылку на новый массив.

```
int[] newArray = new int[ 5 ];  
Array.Resize( ref newArray, 10 );
```

Переменная `newArray` изначально содержит ссылку на массив из 5 элементов. В результате вызова метода `Resize` переменная `newArray` начинает указывать на новый массив из 10 элементов. Если новый массив меньше старого, то все данные, не поместившиеся в новый массив, усекаются без каких-либо предупреждений.

Передача массивов и элементов массивов методам

Чтобы передать методу **аргумент-массив**, укажите имя массива без квадратных скобок. Допустим, массив `abracadabra` объявляется в следующем виде:

```
double[ ] abracadabra = new double [ 24 ];
```

Вызов метода

```
ModifyAbracadabra (abracadabra);
```

передает ссылку на массив `abracadabra` методу `ModifyAbracadabra`. Каждый объект массива «знает» свою длину (и предоставляет доступ к ней в свойстве `Length`). Таким образом, при передаче методу ссылки.

Чтобы метод получал **ссылку на массив** при вызове метода, в списке параметров метода должен быть указан параметр-массив. Например, заголовок метода `ModifyAbracadabra` должен быть записан в виде

```
void ModifyAbracadabra (double [ ] b)
```

Он показывает, что `ModifyAbracadabra` получает в параметре `b` ссылку на массив с элементами `double`. Вызов метода передает ссылку на массив `abracadabra`, так что при использовании переменной `b` в вызванном методе она ссылается на тот же объект массива, что и переменная `abracadabra` в вызывающем методе.

ДВУМЕРНЫЕ МАССИВЫ (МАТРИЦЫ)

Двумерные массивы часто используются для представления таблиц с информацией, упорядоченной по строкам и столбцам. Конкретный элемент таблицы идентифицируется двумя индексами. Обычно первый индекс определяет строку, а второй – столбец, на пересечении которых находится элемент. Массивы, у которых элемент определяется двумя индексами, называются двумерными массивами. В C# поддерживаются два типа двумерных массивов – прямоугольные и ступенчатые.

Прямоугольные массивы

Прямоугольные массивы используются для представления таблиц, организованных в форме строк и столбцов, у которых каждая строка содержит одинаковое число столбцов. Обычно двумерный массив с *t* строк и *p* столбцов называется массивом *t* x *p*.

Каждый элемент массива идентифицируется выражением *a* [строка, столбец];, где *a* – имя массива, *a* строка и столбец – индексы, однозначно идентифицирующие каждый элемент массива *a* номерами строки и столбца.

Инициализатор двумерного прямоугольного массива

Многомерные массивы (как и одномерные) могут инициализироваться при объявлении.

Прямоугольный массив *b* из двух строк и двух столбцов объявляется и инициализируется вложенными инициализаторами массивов:

```
int [ , ] b = { { 1, 2 } , { 3, 4 } } ;
```

Прямоугольный массив может создаваться традиционным выражением создания массива. Например, следующий фрагмент объявляет переменную *b* и присваивает ей ссылку на прямоугольный массив 3 x 4:

```
int[ , ] b;  
b = new int[ 3, 4 ];
```

В этом случае количества строк и столбцов задаются литералами 3 и 4, но это не обязательно – размеры массивов могут также задаваться переменными и выражениями.

Как и в случае с одномерными массивами, элементы прямоугольного массива инициализируются при создании объекта массива.

Ступенчатые массивы

Ступенчатый массив хранится как одномерный массив, в котором каждый элемент ссылается на одномерный массив. Этот способ представления ступенчатых массивов делает их чрезвычайно гибкими, потому что длины строк массива могут быть разными.

Задания

1. В одномерном массиве нулевые элементы удалить, положительные элементы расставить по убыванию, отрицательные – по возрастанию. Получить зависимость затрат машинного времени от размера массива.

2. В матрице удалить строки с последними отрицательными элементами, а затем добавить строку из сумм элементов по столбцам.

Проектирование приложения. Выбор, размещение и задание свойств компонентов. Коды обработчиков событий и функций

1. Запустите VS.
2. Создайте новый проект.
3. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Text** впишите *МАССИВЫ*□

4. Вначале перенесите на форму многостраничную панель – компонент **tabControl1**. Для размещения остальных компонентов приложения достаточно использовать две страницы компонента – по заданиям 1 и 2 соответственно.

5. Щелкните на компоненте **tabControl1** правой кнопкой мыши и во всплывшем меню используйте команду *Добавить вкладку*. В свойство **Text** первой страницы впишите *массив*, второй – *матрица*.

6. Перенесите на первую страницу (*массив*) компонент **tabControl2** и, как и в **tabControl1**, создайте две страницы, с надписями *тестирование и использование* и *графики* соответственно.

7. На страницу *тестирование и использование* (рис. 2.1) перенесите семь меток **Label** (страница *Стандарт*), в свойство **Text** (надпись) которых соответственно впишите *размер массива, начальный, конечный, шаг, диапазон чисел, макс, мин*; пять компонентов ввода целых чисел – **numericUpDown1** для задания параметров формируемых массивов. В свойствах **Minimum** и **Maximum** укажите необходимый вам диапазон значений.

8. Для разрешения или запрещения вывода на экран исходных и результирующих массивов, а также графиков полученных зависимостей, в правую верхнюю часть страницы перенесите два компонента-индикатора **checkBox**, в свойство **Text** которых впишите соответственно *в таблицу* и *графики*.

9. Ниже индикаторов **checkBox** поместите панель **Label** для вывода в нее сообщений: *Макс значение не м.б. меньше мин значения!, В массиве только нули!, Кол-во сравнений = <число> Кол-во обменов = <число>*. Очистите свойство **Text** у панели□

10. Для вывода массивов на экран при тестировании перенесите на страницу компонент **dataGridView1**.

11. Разместите 2 кнопки с надписями соответственно: **Button1** – *ПУСК*, **Button2** – *СБРОС*.

12. Для отображения процесса обработки массива ниже компонента **dataGridView1** поместите компонент **progressBar1**.

13. Далее поместите компонент **statusStrip1**. Щелкните левой клавишей мышки и в выпадающем меню на компоненте и выберите **StatusLabel**. В свойстве **Text** элемента **toolStripStatusLabel1** введите *«Нули удалить, положитель-*

ные элементы расставить по убыванию, а отрицательные по возрастанию».

14. Затраты машинного времени на обработку массива оценивают косвенно – по количествам сравнений и обменов. Для вывода на экран зависимостей количеств сравнений и обменов от размера массива на страницу *графики* (рис. 2.2) компонента **tabControl2** перенесите компоненты **chart1** и **chart2**.

15. Задайте свойства компонентов самостоятельно.

16. После проектирования формы будут иметь следующий вид (рис. 2.1 – 2.2):

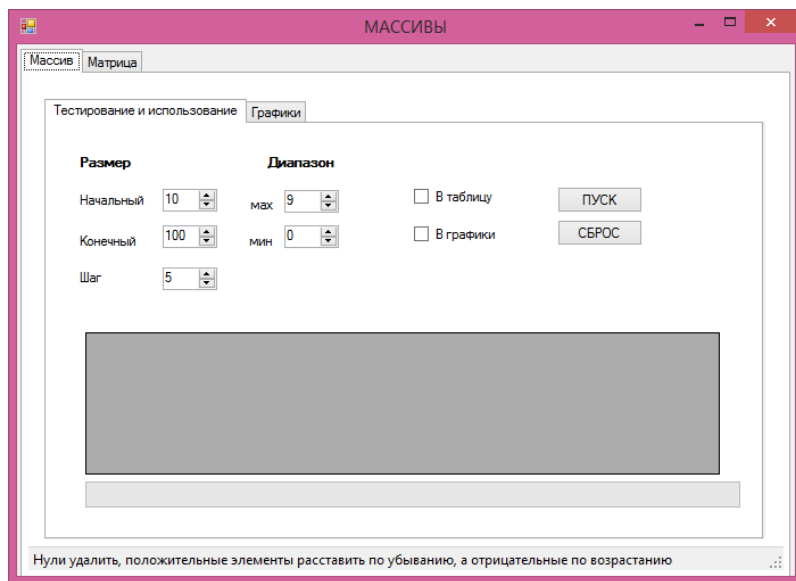


Рис. 2.1. Форма по окончании проектирования (вид 1)

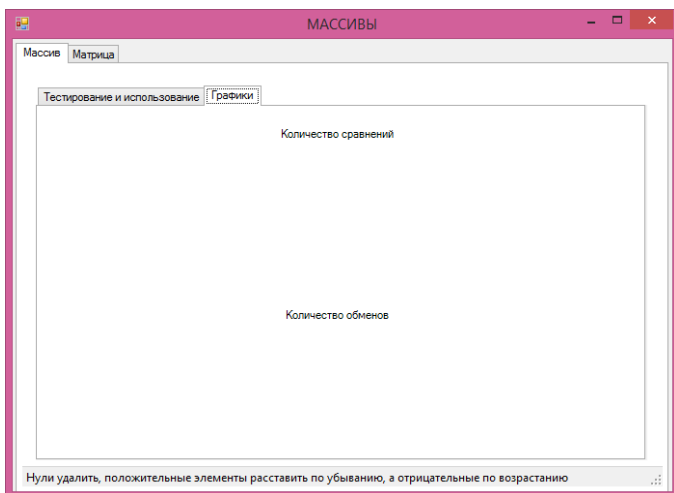


Рис. 2.2. Форма по окончании проектирования (вид 2)

17. Перейдем к обработчикам событий – щелчков на кнопках ПУСК1 и СБРОС. В первом обработчике размещен алгоритм формирования и обработки массива по заданию 1, а также вывод сообщений и результатов в виде таблицы и графиков, а во втором – подготовка компонентов к выводу новых сообщений и новых результатов. Перед, после и в обработчике щелчка на кнопке ПУСК напишите:

```
int i = 0;
private void button1_Click(object sender, EventArgs e)
{
    button2.Enabled = false;
    label8.Text = "";
    if (numericUpDown4.Value < numericUpDown5.Value)
    { label8.Text = "Макс значение не м.б. меньше мин
значения!"; return; }
    int count, current = 0;
    count = (Convert.ToInt32(numericUpDown2.Value) -
Convert.ToInt32(numericUpDown1.Value)) /
Convert.ToInt32(numericUpDown3.Value) + 1;
    for (int n = Convert.ToInt32(numericUpDown1.Value);
n <= Convert.ToInt32 (numericUpDown2.Value); n +=
Convert.ToInt32 (numericUpDown3.Value))
```

```

    {
        int[] vptr=new int[n];
        Random rand = new Random();
        for(int j=0;j<n;j++)
        {
            vptr[j] = rand. Next(Convert.ToInt32
(numericUpDown5.Value), Convert. ToInt32(numericUpDown4.
Value));
        }
        if(checkBox1.Checked)
        {
            dataGridView1.ColumnCount = n+1;
            dataGridView1.Rows.Add();
            dataGridView1.Rows[i].Cells[0].Value = "Ис-
ходный массив";
            for (int j = 0; j < n; j++)
            { dataGridView1.Rows[i].Cells[j +
1].Value = vptr[j]; }
            i++;
        }
        sort(vptr,n);
        current += 1;
        progressBar1.Value = 100 * current / count;
    }
}

private void sort(int[] p, int n)
{
    int k = 0, sr = 0, obm = 0, m = 0;
    for (int j = 0; j < n; j++)
    {
        if (p[j] == 0) k++;
        else p[j - k] = p[j];
    }
    n -= k;
    sr += n;
    if (n == 0) { label8.Text = "В массиве одни ну-
ли"; return; }
    for (m = 0; m < n - 1; m++)
        for (int j = m + 1; j < n; j++)
        {
            if (p[m] > 0 && p[j] > 0 && p[m] < p[j])
            { swap(ref p[m], ref p[j]); obm++; }
            if (p[m] < 0 && p[j] < 0 && p[m] > p[j])
            { swap(ref p[m], ref p[j]); obm++; }
            sr += 6;
        }
    }
}

```

```

    }
    if (checkBox1.Checked)
    {
        dataGridView1.AutoSizeColumns();
        dataGridView1.Rows.Add();
        dataGridView1.Rows[i].Cells[0].Value = "По-
лучен массив";
        for (int j = 0; j < n; j++)
        { dataGridView1.Rows[i].Cells[j + 1].Value =
p[j]; }
            i++;
        }
        if (Convert.ToInt32(numericUpDown1.Value) ==
Convert.ToInt32(numericUpDown2.Value))
        { label8.Text = "Количество сравнений=" +
Convert.ToString(sr) + " Количество обменов=" +
Convert.ToString(obm); }
        if (checkBox2.Checked)
        {
            chart1.Series[0].Points.AddXY(n, sr);
            chart2.Series[0].Points.AddXY(n, obm);
        }
    }

void swap(ref int x, ref int y)
{ int z = x; x = y; y = z; }

```

18. В обработчике щелчка на кнопке СБРОС напишите:

```

private void button1_Click(object sender, EventArgs e)
{
    chart1.Series[0].Points.Clear();
    chart2.Series[0].Points.Clear();
    dataGridView1.Rows.Clear();
    dataGridView1.Columns.Clear();
    i = 0;
    button2.Enabled = true;
}

```

19. Перенесите на вторую страницу (*матрица*) компонента **tabControl1** (рис. 2.3) восемь меток **Label**, в свойство **Text** (надпись) которых впишите значения *размерность, диапазон чисел, макс, мин, исходная матрица, матрица-результат*; три компонента ввода целых чисел – **numeric UpDown** для задания параметров формируемых матриц.

20. Перенесите также две кнопки **Button** с надписями *ПУСК, СБРОС*.

21. Сюда же поместите панель **Label** для вывода в панель сообщений: *Макс не м.б. меньше мин!, В матрице удалены все строки!, В матрице нет удаленных строк!, В матрице удалено <кол-во> строк(и)!*. Очистите свойство **Text** у панели ☐

22. Кроме того, на этой странице разместите две таблицы – компоненты **dataGridView** – для вывода матриц – исходной и матрицы-результата.

23. Далее поместите компонент **statusStrip2**. Щелкните левой клавишей мышки и в выпадающем меню на компоненте и выберите **StatusLabel**. В свойстве **Text** элемента **toolStripStatusLabel2** введите «*Строки с "-" на конце - удалить, а затем добавить строку из сумм по столбцам*».

24. После проектирования формы будут иметь следующий вид (рис. 2.3):

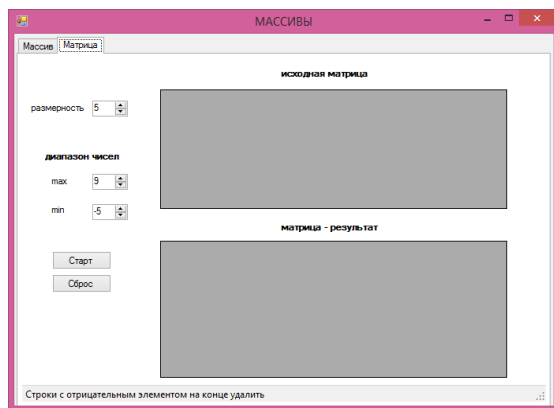


Рис. 2.3. Форма по окончании проектирования (вид 2)

25. В обработчике щелчка на кнопке *ПУСК* находится алгоритм формирования и обработки динамической матрицы согласно заданию 2 с выводом результатов и сообщений, а в обработчике щелчка на кнопке *СБРОС* – зачистка выведенных результатов и сообщений:

```
int n, m;
private void button3_Click(object sender, EventArgs e)
{
    int i, j, k, q;
    button3.Enabled = false;
    if (numericUpDown8.Value < numericUpDown9.Value)
    { label9.Text = "Макс значение не м.б. меньше
мин значения!"; return; }
    n = Convert.ToInt32(numericUpDown6.Value);
    m = Convert.ToInt32(numericUpDown6.Value);
    int[,] ptr;
    ptr = new int[m, n];
    Random rand = new Random();
    dataGridView2.AutoSizeColumnsMode();
    dataGridView2.ColumnCount = n;
    for (i = 0; i < n; i++)
    {
        dataGridView2.Rows.Add();
        for (j = 0; j < m; j++)
        {
            ptr[i,j] =
rand.Next(Convert.ToInt32(numericUpDown9.Value),
Convert.ToInt32(numericUpDown8.Value));
            dataGridView2.Rows[i].Cells[j].Value =
ptr[i,j];
        }
    }
    q = 0;
    k = 0;
    if (ptr[n - 1, m - 1] < 0) k++;
    for(q=0;q<n-1;q++)
    {
        if (ptr[q, m - 1] < 0)
        {
            k++;
            for (i = q; i < n - 1; i++)
            {
```

```

        for (j = 0; j < m; j++) ptr[i, j] =
ptr[i + 1, j];

        }
        q--;
    }
    if (k + q+1 == n) { break; }
}

if (k == n) { label9.Text = "В матрице удалены все
строки"; return; }
if (k == 0) { label9.Text = "В матрице нет удален-
ных строк"; return; }
label9.Text = "В матрице удалено " + k + " строк(и)";
for (j = 0; j < m; j++)
{
    ptr[n - k, j] = 0;
    for (i = 0; i < n - k; i++)
    {
        ptr[n-k, j] += ptr[i, j];
    }
}

dataGridView3.AutoSizeColumns();
dataGridView3.ColumnCount = n;
for (i = 0; i <= n-k; i++)
{
    dataGridView3.Rows.Add();
    for (j = 0; j < m; j++)
    {
        dataGridView3.Rows[i].Cells[j].Value =
ptr[i, j];
    }
}

}

private void button4_Click(object sender, EventArgs e)
{
    dataGridView2.Rows.Clear();
    dataGridView2.Columns.Clear();
    dataGridView3.Rows.Clear();
    dataGridView3.Columns.Clear();
}

```



```

        button3.Enabled = true;
        label9.Text = "";
    }

```

26. По окончании проектирования файл LR_1.cpp будет выглядеть так:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
namespace massiv
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        int i = 0;
        private void button1_Click(object sender, EventArgs e)
        {
            button2.Enabled = false;
            label8.Text = "";
            if (numericUpDown4.Value < numericUpDown5.Value)
            { label8.Text = "Макс значение не м.б. меньше мин
значения!"; return; }
            int count, current = 0;
            count = (Convert.ToInt32(numericUpDown2.Value) -
Convert.ToInt32(numericUpDown1.Value)) /
Convert.ToInt32(numericUpDown3.Value) + 1;
            for (int n =
Convert.ToInt32(numericUpDown1.Value); n <=
Convert.ToInt32(numericUpDown2.Value); n +=
Convert.ToInt32(numericUpDown3.Value))

```

```

        {
            int[] vptr=new int[n];
            Random rand = new Random();
            for(int j=0;j<n;j++)
            {
                vptr[j] =
rand.Next(Convert.ToInt32(numericUpDown5.Value),
Convert.ToInt32(numericUpDown4.Value));
            }
            if(checkBox1.Checked)
            {
                dataGridView1.ColumnCount = n+1;
                dataGridView1.Rows.Add();
                dataGridView1.Rows[i].Cells[0].Value =
"Исходный массив";
                for (int j = 0; j < n; j++)
                { dataGridView1.Rows[i].Cells[j +
1].Value = vptr[j]; }
                i++;
            }
            sort(vptr,n);
            current += 1;
            progressBar1.Value = 100 * current / count;
        }
    }

    private void sort(int[] p, int n)
    {
        int k = 0, sr = 0, obm = 0, m = 0;
        for (int j = 0; j < n; j++)
        {
            if (p[j] == 0) k++;
            else p[j - k] = p[j];
        }
        n -= k;
        sr += n;
        if (n == 0) { label8.Text = "В массиве одни ну-
ли"; return; }
        for (m = 0; m < n - 1; m++)
            for (int j = m + 1; j < n; j++)
            {
                if (p[m] > 0 && p[j] > 0 && p[m] < p[j])
                { swap(ref p[m], ref p[j]); obm++; }
                if (p[m] < 0 && p[j] < 0 && p[m] > p[j])
                { swap(ref p[m], ref p[j]); obm++; }
                sr += 6;
            }
        if (checkBox1.Checked)
        {
            dataGridView1.AutoResizeColumns();
            dataGridView1.Rows.Add();
        }
    }

```

```

        dataGridView1.Rows[i].Cells[0].Value = "По-
лучен массив";
        for (int j = 0; j < n; j++)
        { dataGridView1.Rows[i].Cells[j + 1].Value =
p[j]; }
        i++;
    }
    if (Convert.ToInt32(numericUpDown1.Value) ==
Convert.ToInt32(numericUpDown2.Value))
    { label8.Text = "Количество сравнений=" +
Convert.ToString(sr) + " Количество обменов=" +
Convert.ToString(obm); }
    if (checkBox2.Checked)
    {
        chart1.Series[0].Points.AddXY(n, sr);
        chart2.Series[0].Points.AddXY(n, obm);
    }
}

```

```

void swap(ref int x, ref int y)
{ int z = x; x = y; y = z; }

```

```

private void button2_Click(object sender, EventArgs e)
{
    chart1.Series[0].Points.Clear();
    chart2.Series[0].Points.Clear();
    dataGridView1.Rows.Clear();
    dataGridView1.Columns.Clear();
    i = 0;
    button2.Enabled = true;
}
int n, m;
private void button3_Click(object sender, EventArgs e)
{
    int i, j, k, q;
    button3.Enabled = false;
    if (numericUpDown8.Value < numericUpDown9.Value)
    { label9.Text = "Макс значение не м.б. меньше
мин значения!"; return; }
    n = Convert.ToInt32(numericUpDown6.Value);
    m = Convert.ToInt32(numericUpDown6.Value);
    int[, ] ptr;

```

```

ptr = new int[m, n];
Random rand = new Random();
dataGridView2.AutoSizeColumns();
dataGridView2.ColumnCount = n;
for (i = 0; i < n; i++)
{
    dataGridView2.Rows.Add();
    for (j = 0; j < m; j++)
    {
        ptr[i, j] =
rand.Next(Convert.ToInt32(numericUpDown9.Value),
Convert.ToInt32(numericUpDown8.Value));
        dataGridView2.Rows[i].Cells[j].Value =
ptr[i, j];
    }
}
q = 0;
k = 0;
if (ptr[n - 1, m - 1] < 0) k++;
for(q=0;q<n-1;q++)

{
    if (ptr[q, m - 1] < 0)
    {
        k++;
        for (i = q; i < n - 1; i++)
        {
            for (j = 0; j < m; j++) ptr[i, j] =
ptr[i + 1, j];

        }
        q--;
    }
    if (k + q + 1 == n) { break; }
}

if (k == n) { label9.Text = "В матрице удалены все
строки"; return; }
if (k == 0) { label9.Text = "В матрице нет удален-
ных строк"; return; }
label9.Text = "В матрице удалено " + k + " строк(и)";
for (j = 0; j < m; j++)
{
    ptr[n - k, j] = 0;
    for (i = 0; i < n - k; i++)

```

```

        {
            ptr[n-k, j] += ptr[i, j];
        }
    }

    dataGridView3.AutoSizeColumns();
    dataGridView3.ColumnCount = n;
    for (i = 0; i <= n-k; i++)
    {
        dataGridView3.Rows.Add();
        for (j = 0; j < m; j++)
        {
            dataGridView3.Rows[i].Cells[j].Value =
ptr[i, j];
        }
    }

}

private void button4_Click(object sender, EventArgs e)
{
    dataGridView2.Rows.Clear();
    dataGridView2.Columns.Clear();
    dataGridView3.Rows.Clear();
    dataGridView3.Columns.Clear();
    button3.Enabled = true;
    label9.Text = "";
}

}
}

```

Тестирование и использование приложения

1. Запустите приложение на выполнение, нажав быстрые кнопки *Сохранить все* и *Запуск*.

2. Подготовьте приложение к тестированию задания 1, щелкнув на закладке *массив*, а затем *тестирование и использование* (рис. 2.1). Включите индикатор *вывод в таблицу*.

3. Пользуясь *ПУСК* и *СБРОС*, убедитесь в работоспособности приложения с параметрами массива, заданными по умолчанию (рис. 2.4).

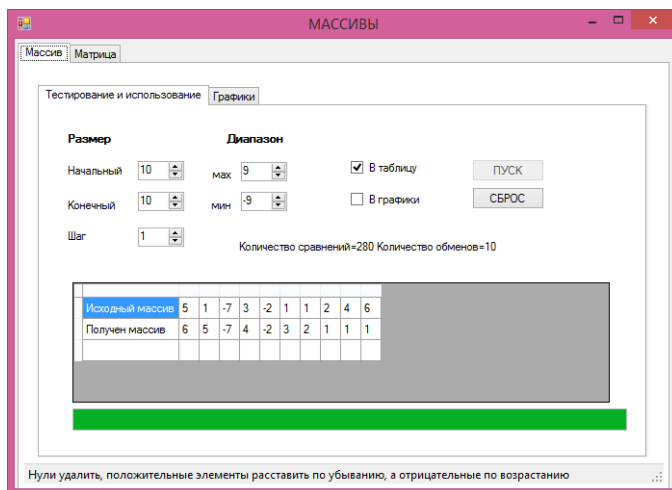


Рис. 2.4. Результаты тестирования по заданию 1

4. Изменяя диапазон значений элементов массива, убедитесь в работоспособности приложения в случаях: а) максимальное значение меньше минимального, б) в массиве только нули, в) в массиве только положительные элементы, г) в массиве только отрицательные элементы, д) массив состоит из равных по величине элементов (положительных и отрицательных).

5. Перейдите к использованию приложения для построения графиков зависимостей затрат машинного времени от размера массива при разных диапазонах значений элементов массива. Включите индикатор *графики*. При заданных по умолчанию остальных параметрах задайте конечный размер массива – 1000 и нажмите *ПУСК*. По окончании обработки массива получите на вкладке *графики* результат, представленный на рис. 2.5.

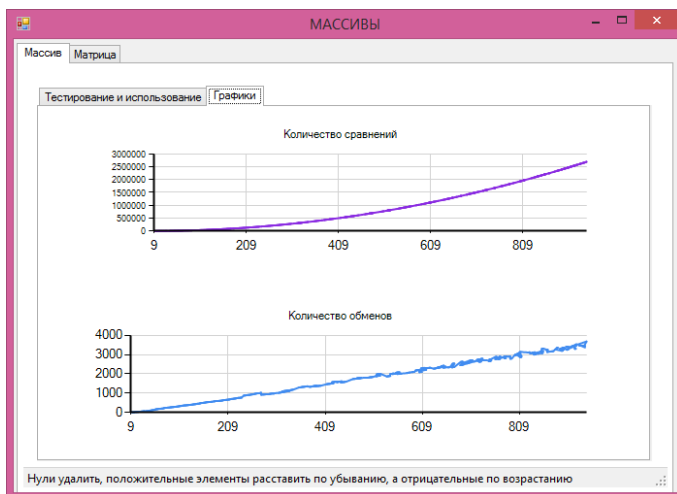


Рис. 2.5. Зависимости затрат машинного времени от размера массива

6. Установите влияние диапазона значений элементов и размера массива на ход указанных зависимостей. В частности, установите параметры диапазона такими, чтобы количество обменов было нулевым при любом размере массива.

7. Подготовьте приложение к тестированию и выполнению задания 2, щелкнув на закладке *матрица* (рис. 2.3).

8. Протестируйте приложение для заданных по умолчанию параметров матрицы, щелкая на кнопках *ПУСК* и *СБРОС* (рис. 2.6).

9. Изменяя диапазон значений элементов матрицы, убедитесь в работоспособности приложения в случаях: а) максимальное значение меньше минимального, б) удалены все строки, в) нет удаленных строк. Установите влияние диапазона на среднее количество удаленных строк.

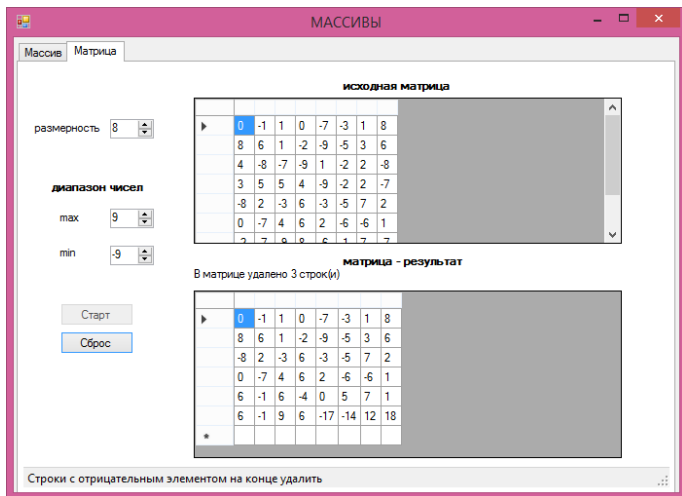


Рис. 2.6. Результаты тестирования по заданию 2

10. Прodelайте то же самое, варьируя размерами матрицы.
11. Для завершения работы щелкните на кнопке формы “Закреть” и выйдите из среды VS.

Контрольные вопросы

1. Как еще можно использовать свойство **Text** (надпись) формы? Приведите примеры фрагментов кода.
2. Какие свойства компонента **tab Control** используются в приложении? Как установить значения этих свойств?
3. Поясните назначение компонентов **data Grid View** в приложении. Значения каких свойств использованы по умолчанию, а у каких значения по умолчанию пришлось изменить?
4. Расскажите о порядке установки значений свойств компонента **chart**. Каким свойствам можно задать другие значения без ущерба для качества представления результатов?
5. Какую функцию выполняют в приложении компоненты **check Box**? Для ответа воспользуйтесь кодом.

6. Как задаются размеры массива и матрицы? Поясните по коду.

7. Используя код, объясните, как задается диапазон значений элементов массива.

8. Используя код, объясните, как задается диапазон значений элементов матрицы.

9. Используя код, объясните, как задаются значения элементов массива и матрицы.

10. Поясните назначение компонентов **status Strip**. Укажите и объясните фрагменты кода, относящиеся к этим компонентам. Как вывести сообщение в компоненты во время выполнения приложения?

11. Поясните назначение компонента **pogress Bar**. Укажите и объясните фрагменты кода, относящиеся к этому компоненту. Изобразите закон изменения переменной *current* по времени.

12. Используя код, объясните, как удаляются из массива нули. Представьте алгоритм.

13. Используя код, объясните, как сортируются в массиве положительные элементы. Представьте алгоритм.

14. Используя код, объясните, как сортируются в массиве отрицательные элементы. Представьте алгоритм.

15. Как выводятся на экран исходный массив и массив, полученный в результате сортировки? Для ответа используйте код.

16. Как строятся графики зависимостей? Для объяснения воспользуйтесь кодом. Объясните вид зависимостей.

17. Как выводятся на экран исходная матрица и матрица, полученная после обработки? Для ответа обратитесь к коду.

18. Используя код, объясните, как выделяется динамическая память под формируемые массив и матрицу и как она освобождается. Представьте алгоритмы.

19. Какие операции с динамической памятью выполняются во время обработки матрицы?

20. Представьте алгоритм удаления из матрицы строк с последними отрицательными элементами.

21. Представьте алгоритм добавления строки в матрицу согласно коду.

22. Что происходит при щелчке на кнопках *ПУСК* и *ПУСК* в первой вкладке?

23. Что происходит при щелчке на кнопках *СБРОС* и *СБРОС* во второй вкладке?

Задания

1. В матрице удалить строки, содержащие нули, а затем добавить строку, элементы которой равны произведениям элементов в соответствующих столбцах.

2. В матрице удалить столбцы с нулевыми элементами ниже главной диагонали, а затем добавить столбец, элементы которого равны суммам элементов в соответствующих строках.

3. В матрице удалить столбцы, в которых количество отрицательных элементов превышает заданное, а затем в качестве первого добавить столбец с максимальными элементами по строкам.

4. В матрице удалить строки с нулевыми элементами выше главной диагонали, а затем в качестве третьей добавить строку, элементы которой равны разностям соответствующих элементов первой и второй строк

5. В матрице удалить столбцы с положительными суммами элементов, а затем в качестве первого вставить столбец из минимальных элементов соответствующих строк.

6. В матрице удалить строку с минимальным произведением элементов, а затем в качестве второй добавить строку, элементы которой равны разностям элементов первой и последней строк.

7. В матрице удалить строки, последние элементы которых отрицательны, а затем в качестве первой добавить строку из элементов заданного массива.

8. В матрице удалить первую и последнюю строки, а затем добавить строку из максимальных элементов соответствующих столбцов.

9. В матрице удалить столбцы с отрицательной суммой элементов, а затем добавить столбец из минимальных элементов соответствующих строк.

10. В матрице удалить столбцы с максимальным и минимальным элементами матрицы, а затем на место первого добавить столбец из произведений элементов соответствующих строк.

11. В матрице удалить строки с элементами на главной диагонали, превышающими заданную величину, а затем в качестве первой вставить строку из максимальных элементов соответствующих столбцов.

12. В матрице удалить столбцы с положительными третьими элементами, а затем добавить столбец из элементов заданного массива.

13. Вычислить m -норму матрицы.

14. Вычислить l -норму матрицы.

15. Вычислить k -норму матрицы.

16. В матрице удалить строки с положительными последними элементами, а затем добавить строку из минимальных элементов по соответствующим столбцам.

17. Сформировать массивы из элементов в седловых точках матриц.

Примечание: в седловой точке элемент является минимальным в строке и максимальным в столбце.

18. Приняв за характеристику столбца матрицы сумму модулей его отрицательных нечетных элементов, расположить столбцы матриц в соответствии с ростом характеристик.

19. Выполнить операции сглаживания матриц. Операция сглаживания дает матрицу того же размера, каждый элемент которой получается как среднее арифметическое соседей соответствующего элемента матрицы. Соседями элемента a_{ij} в матрице называют элементы a_{kl} с $i-1 \leq k \leq i+1, j-1 \leq l \leq j+1$,

$(k, l) \neq (i, j)$.

20. Элемент матрицы называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Сформировать массив из количеств локальных минимумов обработанных матриц.

21. Осуществить циклический сдвиг элементов прямоугольных матриц на n элементов вправо или вниз (в зависимости от указанного режима); n может быть больше количества элементов в строке или столбце матрицы.

22. Осуществить циклический сдвиг элементов квадратных матриц вправо на k элементов таким образом: элементы первой строки сдвигаются в последний столбец сверху вниз, из него – в последнюю строку справа налево, из неё – в первый столбец снизу вверх, из него – в первую строку; для остальных элементов – аналогично.

23. Сортировать матрицы следующим образом: помещать элементы на диагонали, начиная с главной диагонали, в порядке убывания.

24. В матрице удалить строку с минимальным элементом матрицы, а затем добавить отсортированную по возрастанию предыдущую строку.

25. В матрицах проверять указанную строку на ортогональность остальным строкам.

26. В матрице удалить строку с максимальным по модулю элементом матрицы, а затем в качестве первой добавить строку, элементы которой равны суммам модулей элементов в соответствующих столбцах.

27. Проверять матрицы на попарную ортогональность строк; обработку каждой матрицы завершать выводом списка попарно ортогональных строк.

28. Характеристикой строки матрицы назовём сумму её отрицательных четных элементов. Расположить строки в соответствии с убыванием характеристик.

29. В матрице удалить столбец с максимальным элементом матрицы, а затем вставить заданный столбец перед столбцом с минимальным элементом полученной матрицы.

30. В матрицу добавить строку из элементов заданного массива, а затем удалить строки с положительной суммой элементов.

ЛАБОРАТОРНАЯ РАБОТА № 2

ТЕКСТОВЫЕ ФАЙЛЫ, СИМВОЛЫ, СТРОКИ

Задание

Создать текстовый файл, содержащий текст.

Программа должна выводить текст из файла на экран, сохранять измененный текст, позволять задавать два слова, как первое и второе, которые нужно найти и выделить в тексте, а затем там же поменять их местами.

Проектирование приложения

Для работы с файлами воспользуемся возможностями пространства имен System.IO, которое содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок. Для работы также необходимы компоненты openFileDialog и saveFileDialog, предоставляемые средой.

Выводить текст, имея в виду последующую работу с текстом, целесообразно в многострочное окно редактирования – компонент richTextBox, который работает с текстом в обогащенном формате RTF.

Для регистрации хода выполнения программы воспользуемся компонентом textBox, имеющим свойство Multiline, позволяющее выводить многострочные данные.

Контекстное меню будет создаваться программно, посредством функционала класса ContextMenu. Всплывающее меню должно содержать команды Сохранить и Сохранить как и Открыть.

Первое и второе слова, которые необходимо вписать в окнах textBox, а затем там же поменять их местами.


1. Запустите VS.
2. Создайте новый проект.

3. Выделите форму, щелкнув на ней левой кнопкой мыши, и в свойство **Text** впишите *Текстовые файлы, символы и строки* □

4. Перенесите на форму компоненты `openFileDialog1` и `saveFileDialog1`. Все диалоги являются невизуальными компонентами. При обращении к этим компонентам вызываются стандартные диалоги.

5. Перенесите на форму 3 компонента `textBox` и один `richTextBox`. Для одного из компонентов `textBox`, который будет отвечать за вывод информации о ходе работы программы, установите свойство `Multiline` в позицию `true`.

6. Для реализации контекстного меню необходимо создать обработчик события `Form1_Load`. Для этого перейдите на

вкладку События (). Найдите событие `Load` и создайте функцию-обработчик, выбрав из выпадающего списка `Form1_Load`.

7. Добавьте 4 компонента `button`. Первая кнопка будет отвечать за очистку всех полей, вторая – за поиск первого слова, третья – за поиск второго, четвертая – осуществит замену слов.

8. По окончании проектирования форма примет вид, представленный на рисунке 3.1.

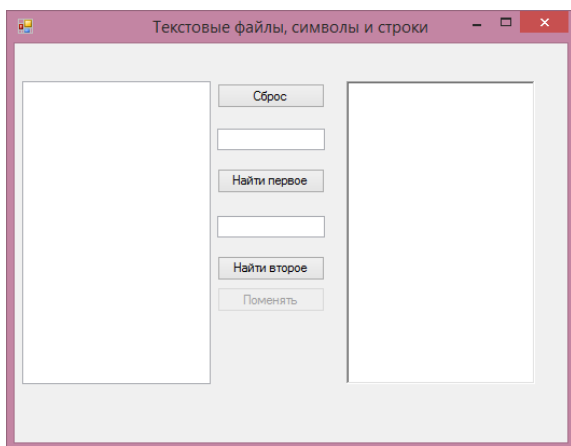


Рис. 3.1. Форма по окончании проектирования

9. В `public Form1()` введите следующий код:
`button4.Enabled = false;`

10. В обработчике события `Form1_Load`, отвечающего за создание контекстного меню, при загрузке формы введите следующий код:

```
private void Form1_Load(object sender, EventArgs e)
{
    //создание контекстного меню
    System.Windows.Forms.ContextMenu contextMenu1;
    contextMenu1 = new Sys-
tem.Windows.Forms.ContextMenu();

    System.Windows.Forms.MenuItem menuItem1;
    menuItem1 = new System.Windows.Forms.MenuItem();
    System.Windows.Forms.MenuItem menuItem2;
    menuItem2 = new System.Windows.Forms.MenuItem();

    System.Windows.Forms.MenuItem menuItem3;
    menuItem3 = new System.Windows.Forms.MenuItem();

    contextMenu1.MenuItems.AddRange(new Sys-
tem.Windows.Forms.MenuItem[] { menuItem1, menuItem2, menuI-
tem3 });

    menuItem1.Index = 0;
    menuItem1.Text = "Открыть";
    menuItem2.Index = 1;
    menuItem2.Text = "Сохранить";
    menuItem3.Index = 2;
    menuItem3.Text = "Сохранить как";

    richTextBox1.ContextMenu = contextMenu1;
    menuItem1.Click += new Sys-
tem.EventHandler(this.menuItem1_Click);
    menuItem2.Click += new Sys-
tem.EventHandler(this.menuItem2_Click);
    menuItem3.Click += new Sys-
tem.EventHandler(this.menuItem3_Click);
}
```

11. Введите код, отвечающий за обработку событий выбора позиций в контекстном меню. Необходимо, также вне

функций, объявить переменную MyFName и добавить пространство имен **using** System.IO.

```
string MyFName = "";
    private void menuItem1_Click(object sender, System.EventArgs e)
    {
        openFileDialog1.Filter = "Текстовые файлы (*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            MyFName = openFileDialog1.FileName;
            richTextBox1.LoadFile(MyFName);
        }
    }

    private void menuItem2_Click(object sender, EventArgs e)
    {
        if (MyFName != "")
        {
            richTextBox1.SaveFile(MyFName);
        }
        else
        {
            saveFileDialog1.Filter = "Текстовые файлы (*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                MyFName = saveFileDialog1.FileName;
                richTextBox1.SaveFile(MyFName);
            }
        }
    }

    private void menuItem3_Click(object sender, System.EventArgs e)
    {
        saveFileDialog1.Filter = "Текстовые файлы (*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
```



```

        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
        {
            MyFName = saveFileDialog1.FileName;
            richTextBox1.SaveFile(MyFName);
        }
    }

```

12. Код события, отвечающего за очистку всех полей формы:

```

private void button1_Click(object sender, EventArgs e)
{
    textBox1.Clear();
    textBox2.Clear();
    textBox3.Clear();
    richTextBox1.Clear();
    button2.Enabled = true;
    button3.Enabled = true;
    button4.Enabled = true;
}

```

13. Код события, отвечающего за нахождение и выделение первого слова:

```

int result1, result2;
private void button2_Click(object sender, EventArgs e)
{
    int LenText;
    textBox3.Text += "Поиск первого слова" + Environment.NewLine;
    String FWord = textBox1.Text.ToString();
    LenText = richTextBox1.Text.Length;
    result1 = FindWord(FWord, LenText);
    if(result1 != -1)
    {
        textBox3.Text += "Позиция первого слова: " + (result1+1) + Environment.NewLine + Environment.NewLine;
        richTextBox1.SelectionStart = result1;
        richTextBox1.SelectionLength = FWord.Length;
        richTextBox1.SelectionBackColor = Color.Red;
        button2.Enabled = false;
    }
}

```

```

        if (button3.Enabled == false)
        { button4.Enabled = true; }

    }
    else
    {
        textBox3.Text += "Слово не найдено " +
Environment.NewLine + Environment.NewLine;
    }

}

```

14. Код события, отвечающего за нахождение и выделение второго слова:

```

private void button3_Click(object sender, EventArgs e)
{
    int LenText;

    textBox3.Text += "Поиск второго слова" +
Environment.NewLine;
    String FWord = textBox2.Text.ToString();
    LenText = richTextBox1.Text.Length;
    result2 = FindWord(FWord, LenText);
    if (result2 != -1)
    {
        textBox3.Text += "Позиция второго слова: " +
(result2+1) + Environment.NewLine + Environment.NewLine;
        richTextBox1.SelectionStart = result2;
        richTextBox1.SelectionLength = FWord.Length;
        richTextBox1.SelectionBackColor =
Color.Green;

        button3.Enabled = false;
        if (button2.Enabled == false )
        { button4.Enabled = true; }
    }
    else
    {
        textBox3.Text += "Слово не найдено " +
Environment.NewLine + Environment.NewLine;
    }

}

```

15. Код события, отвечающего за меню слов местами:

```
private void button4_Click(object sender, EventArgs e)
{
    if (result1 < result2)
    {
        richTextBox1.Select(result2,
textBox2.Text.Length);
        richTextBox1.SelectedText =
textBox1.Text.ToString();
        richTextBox1.Select(result1,
textBox1.Text.Length);
        richTextBox1.SelectedText =
textBox2.Text.ToString();
        textBox3.Text += "Произошла замена
слов";
        button4.Enabled = false;
    }
    else
    {
        richTextBox1.Select(result1,
textBox1.Text.Length);
        richTextBox1.SelectedText =
textBox2.Text.ToString();
        richTextBox1.Select(result2,
textBox2.Text.Length);
        richTextBox1.SelectedText =
textBox1.Text.ToString();
        textBox3.Text += "Произошла замена
слов";
        button4.Enabled = false;
    }
}
```

16. Код функции FindWord, которая осуществляет поиск слова и возвращает номер позиции начала слова, в случае неудачи функция вернет -1. Аргументами, передаваемыми в функцию, является слово для поиска и длина текста.

```
int FindWord(String FWord, int n)
{
    int LenWord;
    String ComparText;
```

```

        LenWord = FWord.Length;
        for (int i = 0; i <= n - LenWord; i++)
        {
            ComparText = richTextBox1.Text.Substring(i,
LenWord);
            if (ComparText == FWord)
            {
                return i;
            }
        }
        return -1;
    }
}

```

17. По окончании проектирования код программы будет выглядеть следующим образом:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TextFiles
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            button4.Enabled = false;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            //создание контекстного меню
            System.Windows.Forms.ContextMenu contextMenu1;
            contextMenu1 = new
System.Windows.Forms.ContextMenu();

            System.Windows.Forms.MenuItem menuItem1;
            menuItem1 = new System.Windows.Forms.
MenuItem();

```

```

        System.Windows.Forms.MenuItem menuItem2;
        menuItem2 = new System.Windows.Forms.MenuItem();
        System.Windows.Forms.MenuItem menuItem3;
        menuItem3 = new System.Windows.Forms.MenuItem();

        contextMenu1.MenuItems.AddRange(new
System.Windows. Forms.MenuItem[] { menuItem1, menuItem2,
menuItem3 });
        menuItem1.Index = 0;
        menuItem1.Text = "Открыть";
        menuItem2.Index = 1;
        menuItem2.Text = "Сохранить";
        menuItem3.Index = 2;
        menuItem3.Text = "Сохранить как";

        richTextBox1.ContextMenu = contextMenu1;
        menuItem1.Click += new
System.EventHandler(this.menuItem1_Click);
        menuItem2.Click += new
System.EventHandler(this.menuItem2_Click);
        menuItem3.Click += new
System.EventHandler(this.menuItem3_Click);

    }

    string MyFName = "";

    private void menuItem1_Click(object sender,
System.EventArgs e)
    {
        openFileDialog1.Filter = "Текстовые файлы
(*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
        if (openFileDialog1.ShowDialog() ==
DialogResult.OK)
        {
            MyFName = openFileDialog1.FileName;
            richTextBox1.LoadFile(MyFName);
        }
    }
}

```

```

        private void menuItem2_Click(object sender,
EventArgs e)
        {
            if (MyFName != "")
            {
                richTextBox1.SaveFile(MyFName);
            }
            else
            {
                saveFileDialog1.Filter = "Текстовые файлы
(*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
                if (saveFileDialog1.ShowDialog() ==
DialogResult.OK)
                {
                    MyFName = saveFileDialog1.FileName;
                    richTextBox1.SaveFile(MyFName);
                }
            }
        }

        private void menuItem3_Click(object sender,
System.EventArgs e)
        {
            saveFileDialog1.Filter = "Текстовые файлы
(*.rtf; *.txt; *.dat) | *.rtf; *.txt; *.dat";
            if (saveFileDialog1.ShowDialog() ==
DialogResult.OK)
            {
                MyFName = saveFileDialog1.FileName;
                richTextBox1.SaveFile(MyFName);
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Clear();
            textBox2.Clear();
            textBox3.Clear();
            richTextBox1.Clear();
            button2.Enabled = true;
            button3.Enabled = true;
            button4.Enabled = true;
        }

```

```

    }
    int result1, result2;
    private void button2_Click(object sender, EventArgs e)
    {
        int LenText;
        textBox3.Text += "Поиск первого слова" +
Environment.NewLine;
        String FWord = textBox1.Text.ToString();
        LenText = richTextBox1.Text.Length;
        result1 = FindWord(FWord, LenText);
        if(result1 != -1)
        {
            textBox3.Text += "Позиция первого слова: " +
(result1+1) + Environment.NewLine + Environment.NewLine;
            richTextBox1.SelectionStart = result1;
            richTextBox1.SelectionLength = FWord.Length;
            richTextBox1.SelectionBackColor = Color.Red;
            button2.Enabled = false;
            if (button3.Enabled == false)
            { button4.Enabled = true; }
        }
        else
        {
            textBox3.Text += "Слово не найдено " +
Environment.NewLine + Environment.NewLine;
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        int LenText;

        textBox3.Text += "Поиск второго слова" +
Environment.NewLine;
        String FWord = textBox2.Text.ToString();
        LenText = richTextBox1.Text.Length;
        result2 = FindWord(FWord, LenText);
        if (result2 != -1)

```

```

        {
            textBox3.Text += "Позиция второго слова: " +
(result2+1) + Environment.NewLine + Environment.NewLine;
            richTextBox1.SelectionStart = result2;
            richTextBox1.SelectionLength = FWord.Length;
            richTextBox1.SelectionBackColor =
Color.Green;

            button3.Enabled = false;
            if (button2.Enabled == false )
                { button4.Enabled = true; }
        }
        else
        {
            textBox3.Text += "Слово не найдено " +
Environment.NewLine + Environment.NewLine;
        }

    }
    int FindWord(String FWord, int n)
    {
        int LenWord;
        String ComparText;
        LenWord = FWord.Length;
        for (int i = 0; i <= n - LenWord; i++)
        {
            ComparText = richTextBox1.Text.Substring(i,
LenWord);
            if (ComparText == FWord)
            {
                return i;
            }
        }
        return -1;
    }

    private void button4_Click(object sender, EventArgs e)
    {
        if (result1 < result2)
        {
            richTextBox1.Select(result2,
textBox2.Text.Length);
            richTextBox1.SelectedText =
textBox1.Text.ToString();

```



```

        richTextBox1.Select(result1,
textBox1.Text.Length);
        richTextBox1.SelectedText =
textBox2.Text.ToString();
        textBox3.Text += "Произошла замена
слов";
        button4.Enabled = false;
    }
    else
    {
        richTextBox1.Select(result1,
textBox1.Text.Length);
        richTextBox1.SelectedText =
textBox2.Text.ToString();
        richTextBox1.Select(result2,
textBox2.Text.Length);
        richTextBox1.SelectedText =
textBox1.Text.ToString();
        textBox3.Text += "Произошла замена
слов";
        button4.Enabled = false;
    }
}
}
}

```

Тестирование и использование приложения

Пример выполнения приложения представлен на рисунках 3.2, 3.3.

1. Запустите приложение на выполнение, нажав быстрые кнопки Сохранить все и Запуск. Нажмите кнопку СБРОС.

2. Наберите текст в окне richTextBox для вывода в файл. Правой кнопкой мыши щелкните на окне и во всплывшем меню выберите нужную команду (сохранить или сохранить как). Сохраните текст в файле с расширением .dat, .txt, .rtf.

3. Сотрите текст, щелкнув на кнопке СБРОС. Щелкните правой кнопкой мыши на окне richTextBox и прочитайте файл.

4. В окно первое слово введите слово из текста и нажмите кнопку найти первое слово.

5. В окно второе слово введите слово из текста и нажмите кнопку найти второе слово. Результат представлен на рис. 3.2.

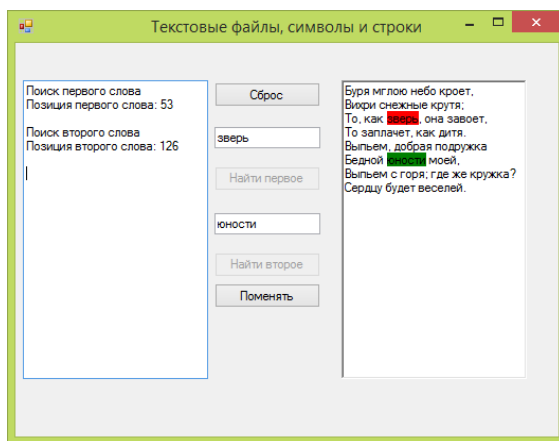


Рис. 3.2. Заданные первое и второе слова найдены и выделены в тексте

6. Нажатием кнопки поменять завершаем выполнение задания (рис. 3.3).

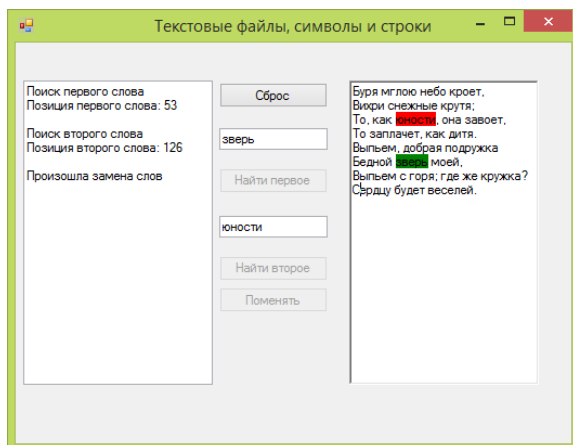


Рис. 3.3. Найденные слова переставлены местами

Контрольные вопросы

1. Поясните назначение и использование компонентов “Сохранить файл” и “Открыть файл”.
2. Поясните реализацию контекстного всплывающего меню. Как осуществляется связь этого компонента с окном richTextBox?
3. Как используются компоненты richTextBox и textBox при выполнении задания?
4. Представьте блок-схему алгоритма, реализованного функцией FindWord(). Расскажите по алгоритму, как осуществляется поиск слова.
5. Как, когда и где вызывается функция FindWord()?
6. Объясните, как переставляются местами найденные в тексте слова. Какие методы при этом используются?
7. Поясните назначение глобальных переменных.
8. Как сохранить в файле текст, представленный в richTextBox?
9. Как изменить размер шрифта и цвет выделенных в тексте слов?

Задания

Написать программу, которая:

- а) выводит текст на экран дисплея и сохраняет его;
- б) далее – по варианту.

1. По нажатию произвольной клавиши поочередно выделяет каждое предложение текста; определяет количество предложений в тексте.

2. По нажатию произвольной клавиши поочередно выделяет каждое слово текста; определяет количество слов в тексте.

3. По нажатию произвольной клавиши поочередно выделяет каждое слово текста, оканчивающееся на гласную букву; определяет количество таких слов в тексте.

4. По нажатию произвольной клавиши поочередно выделяет каждое предложение текста в последовательности 2, 1, 3.

5. По нажатию произвольной клавиши поочередно выделяет каждое из слов текста, у которых первый и последний символы совпадают; определяет количество таких слов в тексте.

6. По нажатию произвольной клавиши поочередно выделяет каждое слово текста, начинающееся на гласную букву; определяет количество таких слов в тексте.

7. Определяет количество символов в самом длинном слове; по нажатию произвольной клавиши поочередно выделяет каждое слово текста, содержащее максимальное количество символов.

8. Определяет количество символов в самом коротком слове; по нажатию произвольной клавиши поочередно выделяет каждое слово текста, содержащее минимальное количество символов.

9. Определяет в каждом предложении текста количество символов, отличных от букв и пробела; по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении – поочередно все символы, отличные от букв и пробела.

10. Определяет количество предложений текста и количество слов в каждом предложении; по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении - поочередно все слова.

11. Определяет количество букв 'а' в последнем слове текста; по нажатию произвольной клавиши выделяет последнее слово текста, а в выделенном слове – поочередно все буквы 'а'.

12. Определяет самую длинную последовательность цифр в тексте (считать, что любое количество пробелов между двумя цифрами не прерывает последовательности цифр); по нажатию произвольной клавиши поочередно выделяет каждую последовательность цифр, содержащую максимальное количество символов.

13. Определяет порядковый номер заданного слова в каждом предложении текста (заданное слово вводится с клавиатуры); по нажатию произвольной клавиши поочередно выделяет каждое предложение текста, а в выделенном предложении – заданное слово.

14. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводить с клавиатуры); выводит текст на экран дисплея ещё раз, выкидывая из него заданное слово и удаляя лишние пробелы.

15. По нажатию произвольной клавиши поочередно выделяет в тексте заданные слова, которые нужно поменять местами (заданные слова вводить с клавиатуры); выводит текст на экран дисплея ещё раз, меняя в нём местами заданные слова и удаляя лишние пробелы.

16. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводить с клавиатуры); выводит текст на экран дисплея ещё раз, заключая заданное слово в кавычки, и поочередно выделяет заданное слово вместе с кавычками.

17. Выводит текст на экран дисплея ещё раз, вставляя в каждое предложение в качестве последнего заданное слово, введенное с клавиатуры в качестве исходных данных; по на-

жатию произвольной клавиши поочередно выделяет в тексте вставленное слово.

18. По нажатию произвольной клавиши поочередно выделяет в тексте лишние пробелы между словами; выводит текст на экран дисплея ещё раз, удаляя лишние пробелы между словами и начиная каждое предложение с новой строки.

19. По нажатию произвольной клавиши поочередно выделяет в тексте заданное слово (заданное слово вводится с клавиатуры); выводит текст на экран дисплея ещё раз, заменяя в заданном слове строчные буквы прописными.

20. Определяет наибольшее количество подряд идущих пробелов в тексте; по нажатию произвольной клавиши поочередно выделяет каждую из последовательностей пробелов максимальной длины.

21. Определяет в каждой строке текста количество прописных букв; по нажатию произвольной клавиши поочередно выделяет каждое слово, начинающееся с прописной буквы, а в выделенном слове – прописные буквы.

22. По нажатию произвольной клавиши поочередно выделяет в тексте слово с заданной буквой; выводит на экран дисплея ещё раз те слова, в которых заданная буква встречается более одного раза.

23. По нажатию произвольной клавиши поочередно выводит фрагменты текста, отделенные знаками препинания; выводит на экран дисплея сведения о знаках препинания по строкам в виде: знак препинания – количество.

24. По нажатию произвольной клавиши поочередно выводит построчно фрагменты текста, разделенные символом горизонтальной табуляции; выводит на экран дисплея общее количество символов табуляции в тексте.

25. Выводит текст на экран дисплея ещё раз, разделяя знаками переноса каждое слово на слоги; по нажатию произвольной клавиши поочередно выделяет в каждой строке текста слово с наибольшим количеством слогов.

26. По нажатию произвольной клавиши поочередно выделяет в тексте слова, после которых стоит знак препинания; выводит текст на экран ещё раз, выделяя знаки препинания.

27. По нажатию произвольной клавиши выводит количество десятичных чисел по строкам; выводит текст на экран дисплея ещё раз, заменяя десятичные числа на шестнадцатеричные.

28. По нажатию произвольной клавиши поочередно выделяет каждое число в тексте; выводит текст на экран дисплея ещё раз, заменяя числа пробелами.

29. По нажатию произвольной клавиши поочередно выделяет в тексте слова с заданной буквой (вводится с клавиатуры); выводит на экран дисплея ещё раз те слова, в которых нет заданной буквы.

30. По нажатию произвольной клавиши поочередно выделяет в тексте каждые первое и второе слова с первыми строчными гласными буквами.

ЛАБОРАТОРНАЯ РАБОТА № 3

СТРУКТУРЫ

Задание

Описать структуру согласно варианту задания. Написать функции, позволяющие:

- 1) вводить с клавиатуры данные;
- 2) реализовать запрос согласно варианту задания.

Вариант задания отдел кадров:

- ФИО работника;
- должность;
- дата рождения;
- ученая степень;
- стаж работы.

Запрос. Вывести на экран работников, стаж которых превышает заданный. Вывести на экран работников выбранной должности.

Проектирование приложения

1. Запустите VS.
2. Создайте новый проект.
3. Добавьте компоненты:

textBox – ввод ФИО, стажа работника и данных для запроса,

label – формирование вспомогательных надписей,

groupBox – разделение группы элементов ввода данных и группы запросов к структуре,

menuStrip – меню программы,

dataGridView – таблицы для отображения данных структуры и результатов запроса,

radioButton – выбор варианта запроса,

comboBox – выбор должности и ученой степени из выпадающего списка,

button – кнопки для обработки событий,

dateTimePicker – календарь для выбора даты рождения.

По окончании проектирования форма примет вид, представленный на рисунках 4.1 – 4.2. Настройка всех элементов, кроме свойства Name и компонента меню, выполнена программно.

Рис. 4.1. Форма по окончании проектирования

Рис. 4.2. Меню программы

4. Пункты меню *Добавить данные* и *Выполнить запрос* дублируют кнопки *Добавить* и *Найти* соответственно. По нажатию кнопки *Выход* происходит закрытие программы.

5. Создаем события обработки нажатия кнопок, пунктов меню и событие загрузки формы. Код файла Form1.cs представлен ниже.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Struct
{
    struct Employee
    {
        public string FIO;//ФИО
        public string Post;// Должность
        public string Date_of_Birth;// Дата рождения
        public string Degree; //Ученая степень
        public int Experience;//Стаж работы
        public Employee(string f, string p, string d, string
deg, int e)//конструктор
        {
            FIO = f;
            Post = p;
            Date_of_Birth = d;
            Degree = deg;
            Experience = e;
        }
    }

    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            comboBox1.Items.Add("Преподаватель");
            comboBox1.Items.Add("Ст. преподаватель");
            comboBox1.Items.Add("Доцент");
            comboBox1.Items.Add("Профессор");

            comboBox2.Items.AddRange(new object[] { "Без уч.
```

```

степени", "Кандидат наук", "Доктор наук"}));

        dataGridView1.RowHeadersVisible = false;
        dataGridView1.ColumnCount = 5;
        dataGridView1.Columns[0].HeaderText = "ФИО";
        dataGridView1.Columns[1].HeaderText = "Долж-
ность";
        dataGridView1.Columns[2].HeaderText = "Дата рож-
дения";
        dataGridView1.Columns[3].HeaderText = "Ученая
степень";
        dataGridView1.Columns[4].HeaderText = "Стаж";

        dataGridView2.RowHeadersVisible = false;
        dataGridView2.ColumnCount = 5;
        dataGridView2.Columns[0].HeaderText = "ФИО";
        dataGridView2.Columns[1].HeaderText = "Долж-
ность";
        dataGridView2.Columns[2].HeaderText = "Дата рож-
дения";
        dataGridView2.Columns[3].HeaderText = "Ученая
степень";
        dataGridView2.Columns[4].HeaderText = "Стаж";
        dataGridView2.RowHeadersVisible = false;
    }

    Employee[] worker = new Employee[10];
    int cout = 0;

    private void button1_Click(object sender, EventArgs e)
    {
        worker[cout].FIO = textBox1.Text;
        worker[cout].Post = comboBox1.Text;
        worker[cout].Date_of_Birth =
dateTimePicker1.Value.ToString("dd.MM.yyyy");
        worker[cout].Degree = comboBox2.Text;
        worker[cout].Experience =
Convert.ToInt32(textBox2.Text);
        dataGridView1.Rows.Add(worker[cout].FIO,
worker[cout].Post, worker[cout].Date_of_Birth,
worker[cout].Degree, worker[cout].Experience.ToString());
        cout++;
    }

    private void button2_Click(object sender, EventArgs e)
    {
        if (radioButton1.Checked == true)

```

```

        {
            dataGridView2.Rows.Clear();
            int select1 =
Convert.ToInt32(textBox3.Text);
            foreach (Employee wSel in worker)
            {
                if (wSel.Experience >= select1)
                    dataGridView2.Rows.Add(wSel.FIO,
wSel.Post, wSel.Date_of_Birth, wSel.Degree,
wSel.Experience.ToString());
            }

            if (radioButton2.Checked == true)
            {
                dataGridView2.Rows.Clear();
                string select2 = textBox3.Text;
                foreach (Employee wSel in worker)
                {
                    if (wSel.Post == select2)
                        dataGridView2.Rows.Add(wSel.FIO,
wSel.Post, wSel.Date_of_Birth, wSel.Degree, wSel.Experience.
ToString());
                }
            }

        }

        private void добавить_ДанныеToolStrip MenuItem_Click
(object sender, EventArgs e)
        {
            button1_Click(sender, e);
        }

        private void
выполнитьЗапросToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            button2_Click(sender, e);
        }

        private void выходToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Close();
        }

```

Тестирование и использование приложения

Пример выполнения приложения представлен на рисунках 4.3 – 4.4.

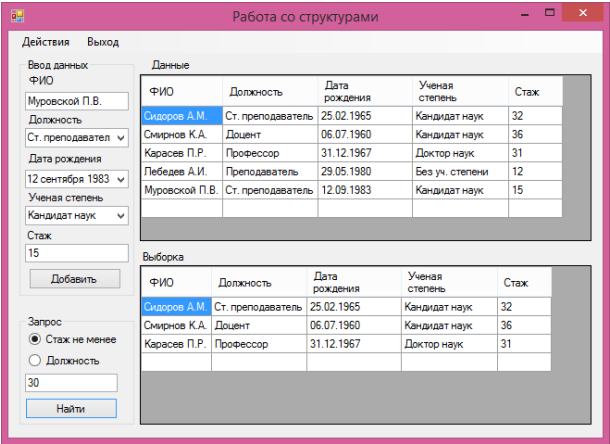


Рис. 4.3. Выборка по стажу

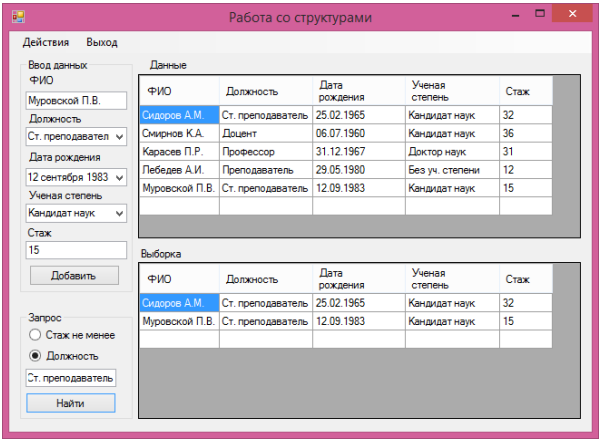


Рис. 4.4. Выборка по должности

1. Запустите приложение на выполнение.
2. Заполните структуры 4 – 5 наборами данных.
3. Выполните поиск сотрудников со стажем выше заданного порога.
4. Выполните поиск сотрудников соответствующей должности.
5. Нажмите кнопки меню, проверив их работоспособность.

Задания

1. Каталог книг:

- название;
- автор;
- количество страниц;
- год издания.

Запрос. Вывести на экран все книги данного автора.

2. Каталог газет:

- название газеты;
- номер;
- дата выхода;
- количество страниц.

Запрос. Вывести на экран все газеты, выходившие в определенном месяце.

3. Перечень факультетов:

- название факультета;
- ФИО декана;
- Телефон;
- адрес.

Запрос. Вывести на экран декана данного факультета.

4. Перечень кафедр:

- название кафедры;
- ФИО заведующего кафедрой;
- количество преподавателей;
- адрес.

Запрос. Вывести на экран кафедры, где количество преподавателей превышает заданное.

5. Перечень студентов:

- ФИО студента;
- дата рождения;
- адрес;
- телефон.

Запрос. Вывести на экран студентов, имеющих одинаковую дату рождения.

6. Рейтинг успеваемости студентов:

- ФИО студента;
- группа;
- средний бал;
- размер стипендии.

Запрос. Вывести на экран студентов, средний балл которых превышает заданный.

7. Перечень основных дисциплин:

- название дисциплины;
- кафедра, на которой читается дисциплина;
- ФИО преподавателя, читающего лекции;
- ФИО преподавателя, ведущего лабораторные

занятия.

Запрос. Вывести на экран дисциплины, которые читаются преподавателями заданной кафедры.

8. Список дисциплин кафедры:

- Название;
- ФИО преподавателя;
- семестр, в котором читается дисциплина;
- группа.

Запрос. Вывести на экран дисциплины, отсортированные по преподавателям.

9. Расписание преподавателя:

- дата;
- день недели;
- предмет;

- группа;
- аудитория.

Запрос. Вывести на экран все пары преподавателя с заданной группой.

10. Список выданных книг:

- код книги;
- номер читательского билета;
- дата выдачи;
- срок выдачи.

Запрос. Вывести на экран книги, которые читатели не сдали вовремя в соответствии с заданным числом.

11. Список товаров:

- наименования товара;
- единица измерения;
- количество на складе;
- цена за единицу.

Запрос. Рассчитать общую стоимость каждого товара на складе.

12. Список продаж:

- наименование товара;
- наименование покупателя;
- дата продажи;
- количество;
- стоимость.

Запрос. Вывести на экран суммарную стоимость покупок каждого покупателя.

13. Справочник улиц города:

- наименование улицы;
- длина;
- история;
- район.

Запрос. Вывести на экран улицы, отсортированные по районам.

14. Справочник пропусков студентов.

- ФИО студента;
- Группа;
- количество пропусков;
- количество неаттестаций.

Запрос. Вывести на экран студентов, количество пропусков которых превышает заданное.

ЛАБОРАТОРНАЯ РАБОТА № 4

ИЕРАРХИЯ: ТОЧКА, КРУГ, ЦИЛИНДР, КОНУС

Задание

Реализовать иерархию (точка, круг, цилиндр, конус), используя наследование. Найти площадь и объем конуса и цилиндра, используя абстрактные или виртуальные методы. Реализовать доступ к полям класса с помощью `get` и `set` аксессоров.

Проектирование приложения

Классы созданной иерархии будут вынесены в отдельный файл `class.cs`.

1. Запустите VS.
2. Создайте новый проект.
3. Спроектируйте форму согласно рисунку 5.1. Лог работы программы реализован с помощью компонента `textBox`. Для возможности многострочного вывода поставьте свойство `Multiline` в позицию `true`.

4. Создайте файл с классом `class.cs`. Процесс добавления нового файла класса описан в методическом указании к лабораторной работе №3.

ИЕРАХИЯ: ТОЧКА, КРУГ, ЦИЛИНДР, КОНУС

Точка

Координаты

X = -1.3 Y = 2.4

Круг

Координаты

X = -3.5 Y = 4.2

Радиус R = 0.8

Цилиндр

Координаты

X = 2.6 Y = 5.7

Радиус основания R = 2.1

Высота цилиндра H = 1.9

Конус

Координаты

X = 2.6 Y = 5.7

Радиус основания R = 2.1

Высота конуса H = 1.9

Расчет Очистка Выход

Рис. 5.1. Форма по окончании проектирования

5. В файле class.cs введите следующий код класса:

```
class Point
{
    protected double x, y;
    public double pointX
    {
        get { return x; }
        set { x = value; }
    }

    public double pointY
    {
        get { return y; }
        set { y = value; }
    }
}
```

```

abstract class Circle : Point
{
    public Circle(double X, double Y)
    { x = X; y = Y; }
    protected double radius;
    public double Rad
    {
        get { return radius; }
        set { radius = value; }
    }
    public abstract double volume();
    public abstract double squareFull();
    public double square() { return Math.PI * radius *
radius; }
}

```

```

class Cylinder : Circle
{
    public Cylinder(double X, double Y, double R)
        : base(X, Y)
    {
        radius = R;
    }
    double height;
    public double Hei
    {
        get { return height; }
        set { height = value; }
    }

    public override double volume()
    {
        return square() * height;
    }
    public override double squareFull()
    {
        return square() + 2 * Math.PI * radius * height;
    }
}

```

```

class Cone : Circle
{
    public Cone(double X, double Y, double R)
        : base(X, Y)

```

```

    {
        radius = R;
    }
    double height;
    public double Height
    {
        get { return height; }
        set { height = value; }
    }
    double squareSide()
    {
        return Math.PI * radius *
Math.Sqrt(Math.Pow(radius, 2) + Math.Pow(height, 2));
    }
    public override double volume()
    {
        return square() * height * (1.0 / 3);
    }

    public override double squareFull()
    {
        return square() + squareSide();
    }
}

```

6. Создаем события обработки нажатия кнопок. Код файла Form1.cs представлен ниже. Кнопка button1 отвечает за расчетную часть, button2 – за очистку поля, button3 – за выход из программы.

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace polimorf
{
    public partial class Form1 : Form
    {
        public Form1()

```

```

    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        button1.Enabled = false;
        double x1, y1, x2, y2, r2, x3, y3, r3, h3, x4,
y4, r4, h4;
        x1 = Convert.ToDouble(textBox1.Text);
        y1 = Convert.ToDouble(textBox2.Text);
        x2 = Convert.ToDouble(textBox3.Text);
        y2 = Convert.ToDouble(textBox4.Text);
        r2 = Convert.ToDouble(textBox5.Text);
        x3 = Convert.ToDouble(textBox6.Text);
        y3 = Convert.ToDouble(textBox7.Text);
        r3 = Convert.ToDouble(textBox8.Text);
        h3 = Convert.ToDouble(textBox9.Text);
        x4 = Convert.ToDouble(textBox10.Text);
        y4 = Convert.ToDouble(textBox11.Text);
        r4 = Convert.ToDouble(textBox12.Text);
        h4 = Convert.ToDouble(textBox13.Text);

        Point tochka = new Point();
        tochka.pointX = x1;
        tochka.pointY = y1;
        textBox14.Text += "Точка" + Environment.NewLine;
        textBox14.Text += "x= " + tochka.pointX + "\t";
        textBox14.Text += "y= " + tochka.pointY +
Environment.NewLine;
        textBox14.Text += Environment.NewLine +
Environment.NewLine;

        textBox14.Text += "Круг" + Environment.NewLine;
        textBox14.Text += "Невозможно создать экземпля
абстрактного класса Circle " +Environment.NewLine;
        textBox14.Text += Environment.NewLine +
Environment.NewLine;
        Cylinder roll = new Cylinder(x3, y3, r3);
        roll.Hei = h3;
        textBox14.Text += "Цилиндр" +
Environment.NewLine;
        textBox14.Text += "x= " + roll.pointX + "\t";
        textBox14.Text += "y= " + roll.pointY + "\t";
        textBox14.Text += "r= " + roll.Rad + "\t";
        textBox14.Text += "h= " + roll.Hei +
Environment.NewLine;

```

```

        textBox14.Text += "Площадь основания цилиндра = " +
        string.Format("{0:F3}", roll.square()) +
        Environment.NewLine;
        textBox14.Text += "Объем цилиндра = " +
        string.Format("{0:F3}", roll.volume()) +
        Environment.NewLine;
        textBox14.Text += "Площадь поверхности цилиндра = " +
        string.Format("{0:F3}", roll.squareFull()) +
        Environment.NewLine;
        textBox14.Text += Environment.NewLine +
        Environment.NewLine;

        Cone bell = new Cone(x4, y4, r4);
        bell.Hei = h4;
        textBox14.Text += "Конус" + Environment.NewLine;
        textBox14.Text += "x= " + bell.pointX + "\t";
        textBox14.Text += "x= " + bell.pointY + "\t";
        textBox14.Text += "r= " + bell.Rad + "\t";
        textBox14.Text += "h= " + bell.Hei +
        Environment.NewLine;
        textBox14.Text += "Площадь основания конуса = " +
        string.Format("{0:F3}", bell.square()) +
        Environment.NewLine;
        textBox14.Text += "Объем конуса = " +
        string.Format("{0:F3}", bell.volume()) +
        Environment.NewLine;
        textBox14.Text += "Площадь поверхности конуса = " +
        string.Format("{0:F3}", bell.squareFull()) +
        Environment.NewLine;

    }

    private void button2_Click(object sender, EventArgs e)
    {
        textBox14.Clear();
        button1.Enabled = true;
    }

    private void button3_Click(object sender, EventArgs e)
    {
        Close();
    }

}
}

```

7. По окончании проектирования приложения Обозреватель решений будет выглядеть следующим образом.

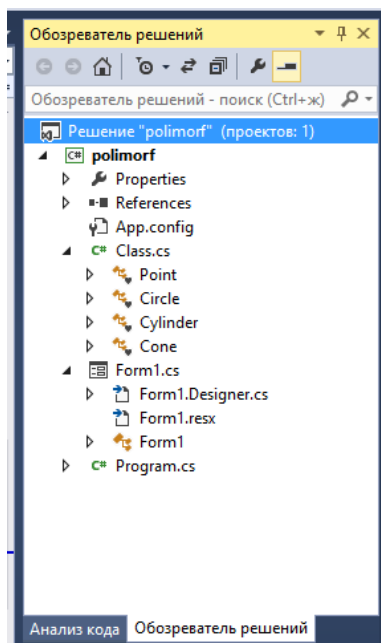


Рис. 5.2. Обозреватель решений по окончании проектирования

Тестирование и использование приложения

Пример выполнения приложения представлен на рисунке 5.3.

1. Запустите приложение на выполнение.
2. Выполните настройку параметров фигур.
3. Нажмите на кнопку Расчет.
4. Нажмите на кнопки Очистка и Выход, проверив их работоспособность.

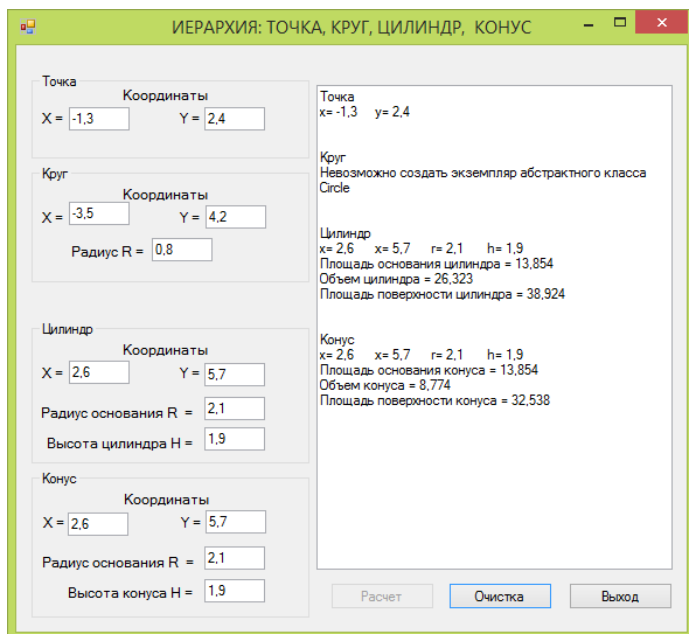


Рис. 5.3. Экранная форма приложения

Задания

1. Форма: прямоугольник: параллелепипед (периметр, площадь, площадь поверхности, объем).
2. Точка: окружность: дуга: сегмент (длина, периметр, площадь).
3. Форма: круг: прямой круговой конус: усеченный прямой круговой конус (площадь, площадь поверхности, объем).
4. Двумерная форма: квадрат: ромб (периметр, площадь).
5. Числовая переменная: массив: матрица (сложение, умножение, сортировка).

ЛАБОРАТОРНАЯ РАБОТА № 5

РАЗРАБОТКА ПРОГРАММНОГО КОМПЛЕКСА С ИСПОЛЬЗОВАНИЕМ КОМПОНЕНТ DLL

Задание

Разработать серверную DLL-библиотеку в соответствии с вариантом. Разработать клиентское оконное приложение.

Проектирование приложения

Простейшим примером компонентного программирования является разработка библиотеки формата DLL (Dynamic Linking Library) и использование ее из Windows-приложения. При этом и библиотеку (выполняющую серверные функции по отношению к оконному приложению), и приложение (являющееся, соответственно, клиентом) мы будем реализовывать в среде MS Visual Studio на языке C#. Компонентный подход подразумевает, что DLL может быть реализована и другими средствами.

В качестве примера разработаем функцию, выполняющую преобразование одноразрядного десятичного числа в его строковое представление («один», «два», «три» и т.д.). Естественно, в соответствии с концепциями ООП, заложенными в C#, эта функция будет являться методом некоторого класса. Условно будем считать, что это класс, инкапсулирующий функциональность разного рода преобразований. Исходя из этого, выберем его имя и пространство имен.

Создание серверной библиотеки

Создание проекта выполняется из среды MS Visual Studio при помощи вызова пункта «Файл→Создать проект» главного меню. Выберем в навигаторе шаблонов «Visual C#→Windows», а из типов проектов выберем «Библиотека классов» (рис. 6.1). Сразу определим расположение проекта, имя и имя решения.

Созданный в результате класс также переименуем. В данном случае переименование можно выполнить «вручную»,

т.е. просто перенабрав имя, однако здесь и далее будем пользоваться функциональностью рефакторинга. Эта функциональность доступна из контекстного меню имени класса: «Выполнить рефакторинг→Переименовать». В диалоговом окне (рис. 6.2) можно ввести новое имя класса.

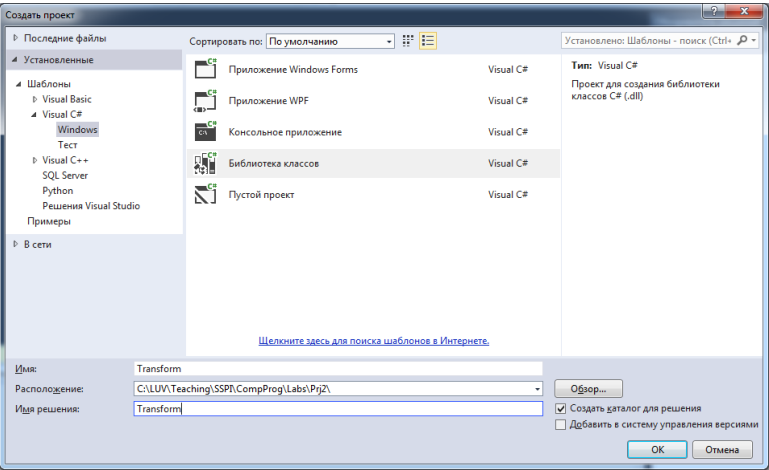


Рис. 6.1. Настройка создаваемого проекта

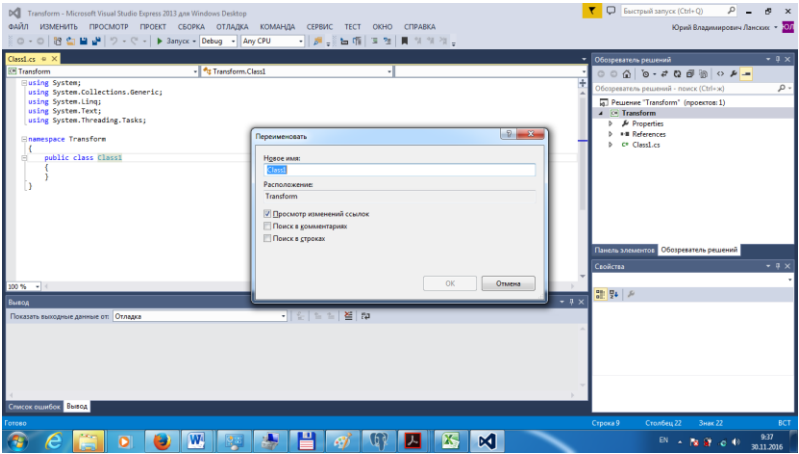


Рис. 6.2. Переименование класса

Функциональность рефакторинга необходима, когда нужно выполнить «глобальное» переименование класса, объекта, метода, интерфейса и т.п. Использование функции рефакторинга, грубо говоря, избавляет разработчика от необходимости думать «всё ли я правильно сделал» и «не забыл ли я чего-то». Обеспечение корректности переименования берет при этом на себя среда разработки. Имя класса также пусть будет Transform.

Функция рефакторинга демонстрирует изменения, которые должны быть выполнены в результате (рис. 6.3). Прием изменения нажатием кнопки «Применить».

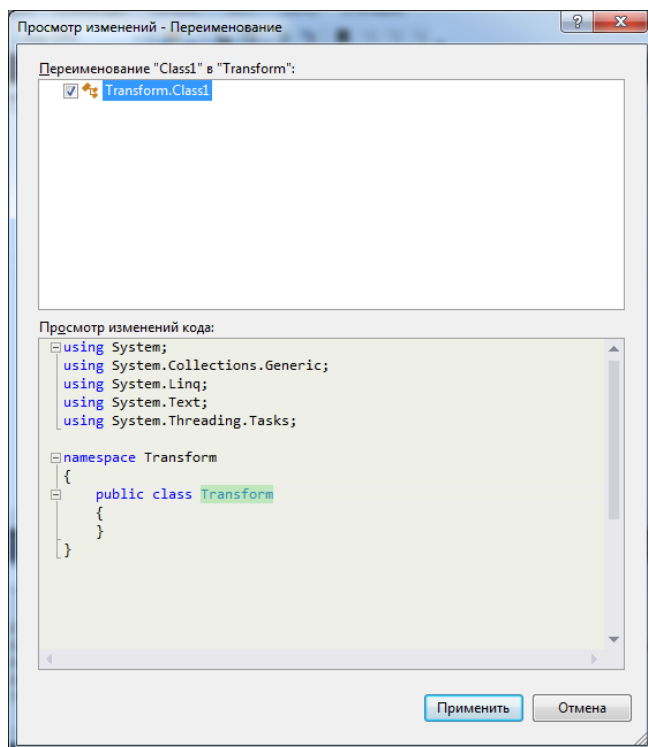


Рис. 6.3. Просмотр изменений в результате рефакторинга

Класс будет сервисный, реализующий всего одну функцию преобразования, поэтому создадим в нем публичный статический метод:

```
namespace Transform
{
    public class Transform
    {
        public static String num2Str(int num)
        {
        }
    }
}
```

И реализуем функциональность простейшим образом:

```
namespace Transform
{
    public class Transform
    {
        public static String num2Str(int num)
        {
            String[] strings = {
                "Ноль",
                "Один",
                "Два",
                "Три",
                "Четыре",
                "Пять",
                "Шесть",
                "Семь",
                "Восемь",
                "Девять"
            };

            return strings[num%10];
        }
    }
}
```

Воспользуемся функцией «Сборка→Собрать решение» главного меню для сборки библиотеки. Как можно убедиться, изучив папку проекта, файл DLL в результате был создан.

Если ошибки были проиндцированы средой в процессе написания кода или обнаружены в процессе сборки, необходимо их исправить. Собрав проект в полной уверенности в отсутствии ошибок, можно использовать функцию «Файл→Заккрыть решение» главного меню.

Создание клиентского приложения

Клиентское приложение также создадим, используя функцию «Файл→Создать решение» главного меню. Выберем тип приложения «Приложение Windows Forms» и введем имя (рис. 6.4).

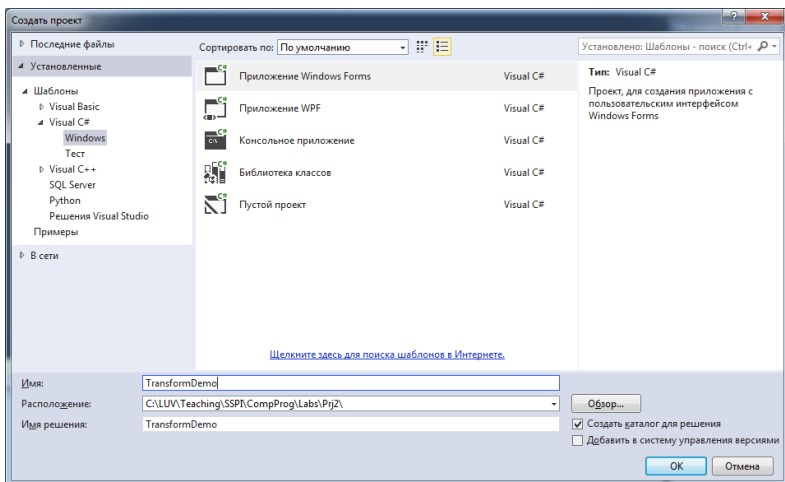


Рис. 6.4. Создание и настройка клиентского приложения

Открыв панель элементов («Просмотр→панель элементов»), сформируем интерфейс оконного приложения – поле типа TextBox для ввода числа, кнопку Button, поле для вывода текста Label. Переименуем визуальные компоненты, а также саму форму для читаемости кода (свойство Name каждого компонента). Настроим надписи (свойство Text каждого компонента).

Ссылка на подключаемую DLL добавляется в проект при помощи функции «Проект→Добавить ссылку» главного меню. Подключить библиотеку можно, выбрав ее из файловой системы при помощи кнопки «Обзор», выбрав предварительно «Обзор» в навигаторе в левой части окна менеджера ссылок (рис. 6.5).

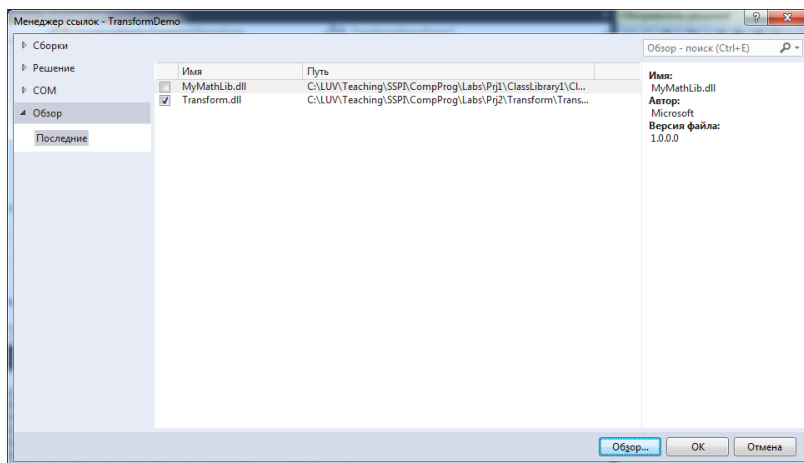


Рис. 6.5. Настройка ссылки на DLL

В результате пространство имен Transform, описанное в созданной ранее DLL, становится доступно из кода приложения:

```
namespace TransformDemo
{
    public partial class TransformDemoForm : Form
    {
        public TransformDemoForm()
        {
            InitializeComponent();
        }

        private void bDoIt_Click(object sender, EventArgs
e)
        {
            lNum.Text = Trans-
form.Transform.num2Str(Int16.Parse(tbNum.Text));
        }
    }
}
```

Запустив приложение, убеждаемся в его работоспособности (рис. 6.6).

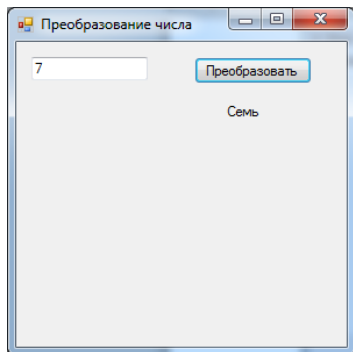


Рис. 6.6. Результаты работы приложения

Задания

1. Пересчет из сантиметров в дюймы и из дюймов в сантиметры.
2. Пересчет из радианов в градусы и из градусов в радианы.
3. Преобразование двузначного числа в его русскую текстовую запись (по аналогии с приведенным примером).
4. «Переворот» строки (изменение порядка следования букв на противоположный).
5. «Переворот» целого числа (изменение порядка следования цифр на противоположный).
6. Удаление из строки лишних (дублирующихся) пробелов и преобразование первых букв первых слов предложений к верхнему регистру.
7. Элементарный калькулятор: вычисление результата для строки, представляющей собой два целых числа и знак одного из четырех основных арифметических действий между ними.
8. Подсчет количества вхождений подстроки в строку.
9. Удаление из строки всех вхождений подстроки, превышающих заданное количество.
10. Вычисление факториала.
11. Вычисление величины C_n^m .
12. Изменение состава слов строки на противоположный.

Библиографический список

1. Дейтел П., Дейтел Х. Как программировать на Visual C# 2012. – 5-е изд. – СПб.: Питер, 2014. – 864 с. : ил. – (Библиотека программиста).
2. Ишкова Э.А. Самоучитель C#. Начала программирования. – 2-е изд. – СПб.: Наука и техника, 2013. – 496 с.: ил.
3. Осипов Н.А. Разработка приложений на C#. – СПб.: НИУ ИТМО, 2012. – 118 с.
4. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.0 на языке C#. – 3-е изд. – СПб.: Питер, 2012. – 928 с.: ил.