Федеральное агентство связи

Государственное образовательное учреждение
высшего профессионального образования

ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ

ЭЛЕКТРОННАЯ БИБЛИОТЕЧНАЯ СИСТЕМА

Самара

Поволжский государственный университет
телекоммуникаций и информатики

Акчурин Э.А.
Ильин А.М.

Программирование на языке C#
ЛР в ИСР Visual C# 2010 Express

Для студентов направления
«Информатика и вычислительная техника»

Самара
2011


Факультет информационных систем и технологий
Кафедра «Информатика и вычислительная техника»

Автор - д.т.н., профессор Акчурин Э.А.

Другие материалы по дисциплине Вы найдете на сайте
www.ivt.psati.ru

Оглавление

Список литературы

1. Троелсен Э. Язык программирования C# 2008 и платформа .NET 3.5, 4-е изд. : Пер. с англ. - М. : "Вильямс", 2010. 1344 с.
2. Нэш Т. C# 2010. Ускоренный курс для профессионалов. Пер. с англ. - М: “Вильямс»,.2010, 592с.
3. Макки А. Введение в .NET 4.0 и Visual Studio 2010 для профессионалов. Пер. с англ. - М.: "Вильямс", 2010. 412с.
4. Нейгел К. и др. C# 2008 и платформа .NET 3.5 для профессионалов. / Пер. с англ. - М.: "Вильямс", 2009. 1392с.
5. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 2.0 на языке C#. Пер. с англ. - М.: «Русская Редакция» ; СПб. : Питер , 2007. 656 стр.
6. Lidin S. Expert .NET 2.0 IL Assembler. Apress; 2006, 530с.
7. Макаров А. и др. CIL и системное программирование в Microsoft.NET: – М. : Интернет-УИТ, 2006. 328 с..
8. Климов Л. C#. Советы программистам. - СПб.: БХВ-Петербург, 2008. 544 с: ил. + CD-ROM.

Введение

Лабораторный цикл содержит работы по изучению программирования на языке C# в ИСР Visual C# 2010 Express Edition. Эта ИСР предназначена для разработки консольных приложений, приложений для ОС с графическим интерфейсов, DLL и др.

Цикл может использоваться в лабораторном практикуме по дисциплинам:

- "Программирование на языках высокого уровня" (специальность 230105),
- "Языки программирования высокого уровня" (специальность 230201),
- "Информатика и программирование" (специальность 080801),
- "Высокоуровневые методы информатики и программирования" (специальность 080801).
- «Программирование на языке C#».

Содержание отчета

- Название работы.
- Цель работы.
- Задание в соответствии с вариантом.
- Блок-схема алгоритма в соответствии с номером варианта.
- Листинги программы.
- Результаты выполнения программы на ПК.

Выбор варианта задания

Номер варианта задания = 1 + последняя цифра номера зачетной книжки. В некотрых работах варианты выбираются по оговоренному правилу.

Сохранение кодов

ИСР для каждого проекта создает несколько файлов. Чтобы файлы разных проектов не перепутывались, следует для каждого проекта создавать свою папку с именем, отображающим смысл (наример, WFormApplHello).

Чтобы файлы разных студентов не перепутывались друг с другом, каждому студенту следует создавать для своих файлов индивидуальные папки. Рекомендуется такая иерархия вложенных друг в друга папок:

Диск пользователя (устанавливается администратором)
Папка Users (пользователи)
Папка группы (например, PO_31)
Папка студента (например, Ivanov)
Папка проекта (например, Hello_Appl)
Внутри нее файлы проекта .

В дисплейных классах в папке проекта на каждом занятии следует сохранять файлы проекта (одно задание лабораторной работы – один проект, одна папка). Рекомендуется при выполнении работы периодически сохранять незавершен-

ные проекты в текущем состоянии, чтобы избежать потерь документов при сбоях.

Методические указания

МУ в электронном виде доступны студентам на сервере (только для чтения). Иерархия папок с документацией следующая:

Диск Method

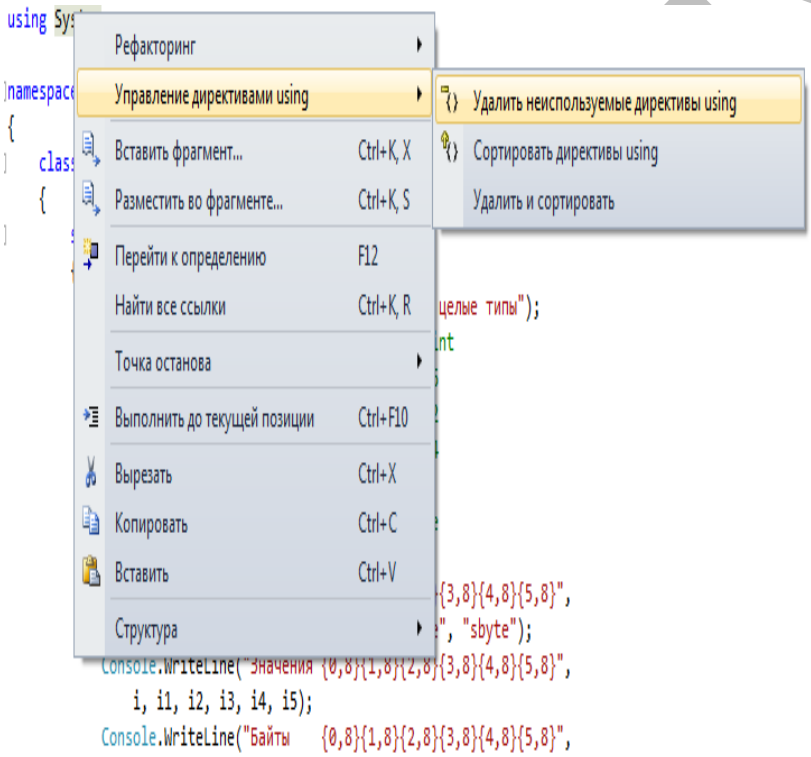
Папка преподавателя (например, Akchurin)

Папка МУ для дисциплины (например, C#)

Файлы методических указаний к отдельным работам.

Методические указания выложены и в сети Internet на сервере кафедры ИВТ ПГУТИ с именем www.ivt.psati.ru. Они находятся на Web-странице в папке «Методические руководства».

Внимание. ИСР для каждого нового проекта использует шаблон, в который нужно добавить функциональность. ИСР создает перечень доступных пространств имен директивами using по умолчанию. Часть из ни не используются. Их можно удалить. Щелчок правой кнопки по коду программы вызывает выпадающее меню, в котором нужно выбрать показанное.



1. ИСР Visual C#. Первые программы

Предмет исследования

- Активизация интегрированной среды разработки (ИСР).
- Структура главного меню и его пункты.
- Опции выпадающих меню.
- Создание простого Windows приложения.
- Создание простого консольного приложения.

Контрольные вопросы

1. Активизация ИСР и выход из среды.
2. Окна ИСР.
3. Назначение и содержание главного окна ИСР.
4. Окно Конструктора формы.
5. Окно Редактора кода.
6. Окно инструментов.
7. Главное меню ИСР. Опции пунктов.
8. Отличия проектов Приложение и Консольное приложение.
9. Использование встроенного подсказчика.
10. Создание Windows приложения.
11. Создание консольного приложения.

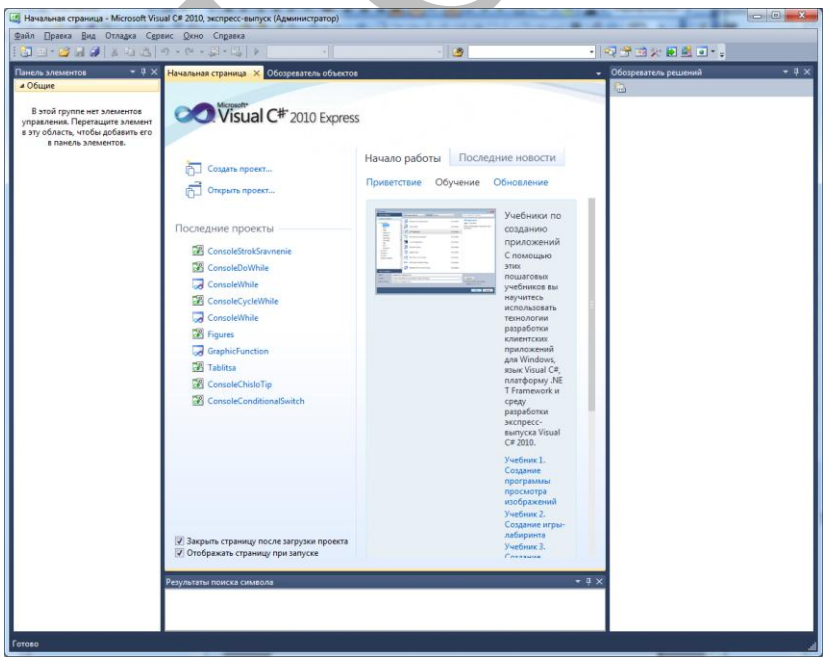
1.1. Основы ИСР

Активизировать ИСР. Ознакомиться с окнами ИСР.

Внимание. В русифицированной версии ИСР разделитель целой и дробной части числа:

- при работе с консолью – запятая.
- при наборе в редакторе кода – точка.

При старте ИСР выводятся окна ИСР.



В главном окне ИСР размещены:

- Строка заголовка с именем ИСР (Microsoft Visual C# 2010Express).
- Строка главного меню ИСР.
- Панели инструментов для быстрого выполнения часто используемых команд

В центре могут размещаться основные окна (на вкладках, если их несколько):

- Начальная страница.
- Дизайнеры.
- Редакторы кода.

Слева размещается панель инструментов с компонентами.

Справа размещаются:

- Обозреватель решений.
- Окно классов.
- Свойства.

Внизу размещается окно «Список ошибок».

Вид представления каждого окна можно изменить. Для этого вызвать выпадающее меню, в котором перечислены возможные решения. Это можно сделать двумя способами - щелчком правой кнопкой мыши по строке заголовка окна или стрелкой в строке заголовка. Меню содержит представления:

- Плавающая область. Окно автономное, может перемещаться произвольно. Опция позволяет делать снимок только этого окна.
- Закрепить. Окно докируется в основное окно и не может перемещаться автономно.
- Закрепить как вкладку. Окно размещается на вкладке в другом окне. Например, окно классов размещается на вкладке вместе с вкладкой Обозревателя решений.
- Автоматически скрывать. Окно отображается в виде узкой вертикальной полоски заголовка для экономии места. Это действие можно выполнить кнопкой в строке заголовка.
- Скрыть. Окно скрывается.

Окно «Начальная страница» содержит вложенные поля:

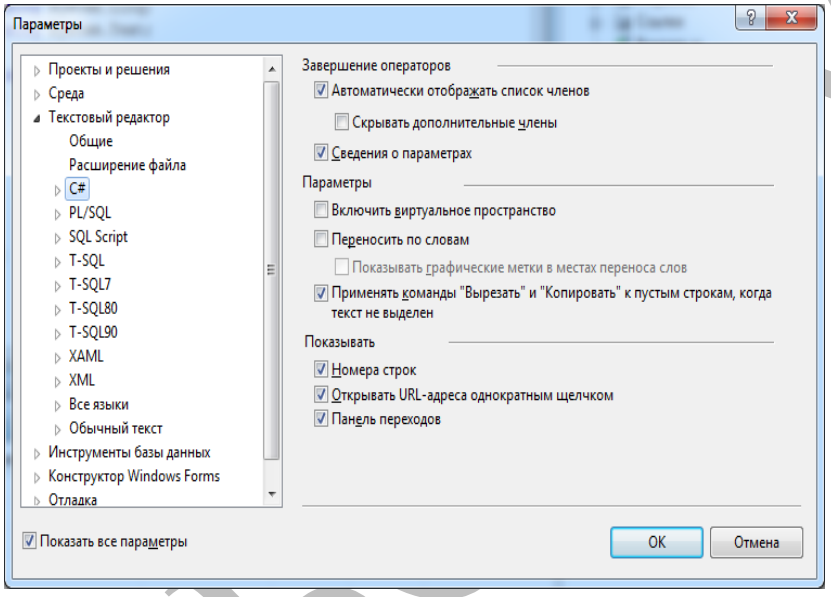
- Создать проект
- Открыть проект
- Последние проекты.
- Последние новости (есть при подключенном Интернете)..
- Начало работы.
- Приветствие.
- Обучение.
- Обновление.

Пункты главного меню ИСР в стартовом режиме:

Пункт	Назначение
Файл	Работа с файлами.
Правка	Редактирование.
Вид	Выбор, что показывать.
Отладка	
Сервис	Использование внешнего инструментария.
Окно	Перечень окон для выбора фокуса.
Справка	

В редактор кода ИСР заносит шаблон кода. В редакторе лучше отображать номера строк.

По умолчанию при первом старте ИСР отображение номеров строк выключено. Чтобы номера отображались нужно выполнить команду **Сервис=>Параметры**. Отображается окно **Параметры**. В левом нижнем углу установить флаг «Показать все параметры». В окне нужно выбрать закладку **Текстовый редактор=>C#** и в ней установить флаг **Номера строк.**



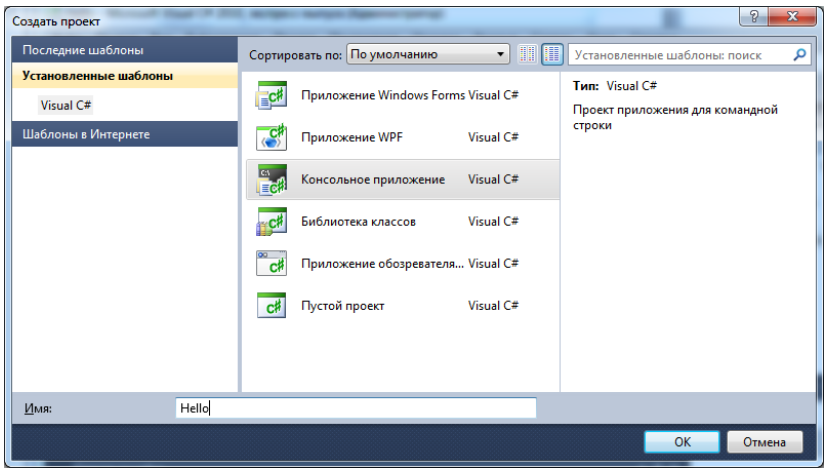
1.2. Console_Hello

Создать проект консольного приложения Hello. Оно должно в символьном режиме выводить на экран фразу «Hello, World & Россия от <Фамилия студента>».

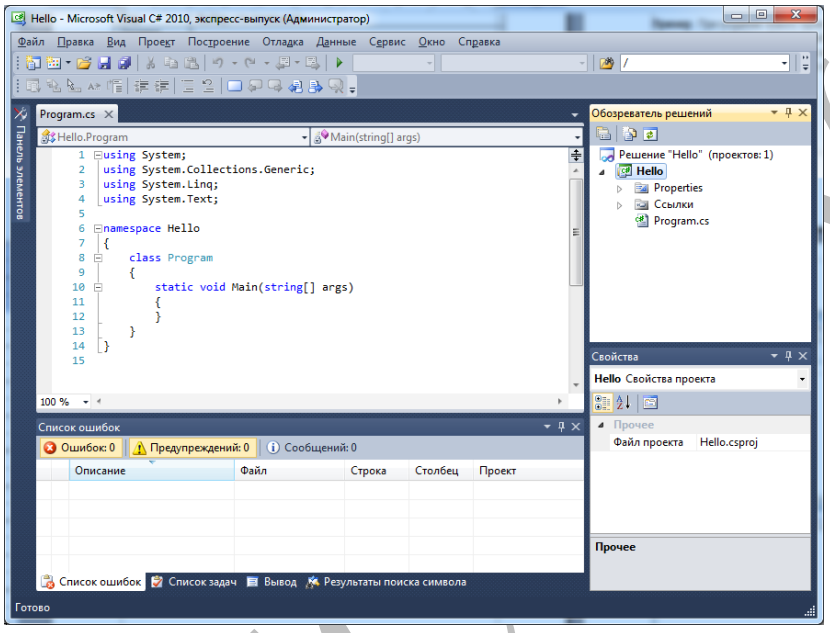
Варианты к заданию. Индивидуальные данные студента.

Пример. При создании нового проекта из вкладки **Последние проекты** или из меню командой **Файл=>Создать=>Проект** вызывается окно выбора типа проекта с набором шаблонов и полем имени проекта. По умолчанию имя совпадает с типом проекта с добавлением номера по порядку. Лучше задать имя проекта, отражающее его смысл.

Выбираем **Консольное приложение** с именем Hello.



Вид ИСР изменяется. Отображается множество окон, которые можно реконфигурировать, меняя размеры и положение. Вот итог:



Вверху слева отображается редактор кода программы в закладке с именем **programm.cs**. Под ним окно **Список ошибок**. Окна браузера проекта обновляется.

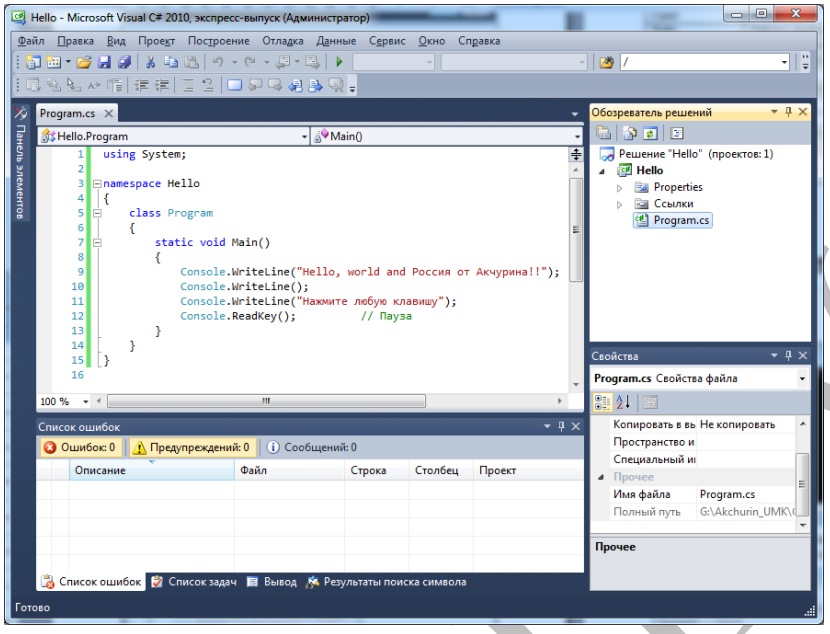
Пункты главного меню ИСР в режиме редактора кода:

Пункт	Назначение
Файл	Работа с файлами.
Правка	Редактирование.
Вид	Выбор, что показывать.
Проект	
Построение	Компоновка исполняемого файла.
Отладка	Отладка
Данные	Данные
Сервис	Использование внешнего инструментария.
Окно	Перечень окон для выбора фокуса.
Справка	Справка

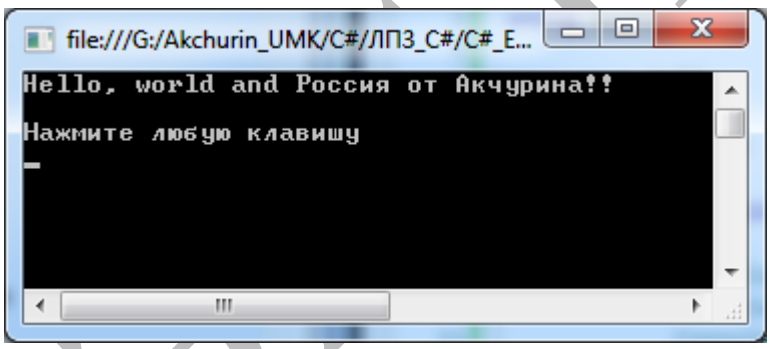
В код программы программист должен добавить функциональность. В шаблон кода, начиная с позиции курсора, нужно ввести инструкции. В консоль выводим строку “Hello, world and Россия от Акчурина ”. Для этого вводим код

```
Console.WriteLine("Hello World and Россия от Акчурина!!");
Console.ReadLine(); // ожидание Enter
```

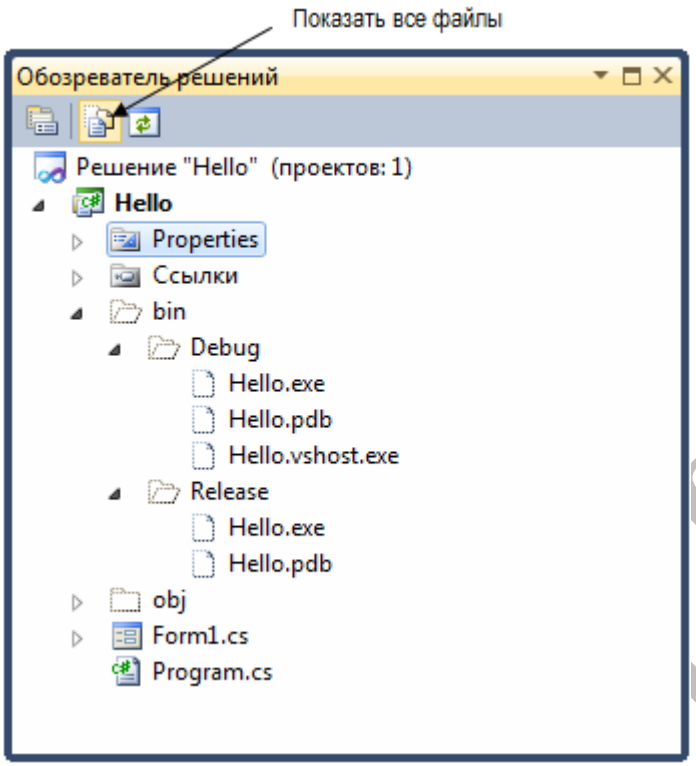
При наборе кода полезен интерактивный подсказчик, который выводит список выбора. Если вы увидите в нем нужное, то его можно перенести в код двойным щелчком или клавишей Enter.



Проект готов, проверим его командой **Отладка=>Начать отладку**.



Проект правилен, сохраняем его командой **Файл=>Сохранить все**. В диалоговом окне задаем имя проекта и его расположение. В результате проект сохраняется в структуре папок:



Решение включает:

Решение Hello	Решение
Hello	Проект
Properties	Свойства
Ссылки	Ссылки
bin	Двоичные файлы
Debug	Файлы отладки
Hello.exe	Управляемый испол-
Hello.pdb	няемый файл
Hel-	База данных для JIT
lo.vshost.exe	компилятора
Release	Служебный файл
obj	Файлы выпуска
	Объектные файлы

Исполняемые (bin) и объектные (obj) файлы образуются при компиляции (построении). Возможны два режима:

- Команда **Построение=>Построить решение**. Построение в режиме отладки, в компонуемые файлы включаются символы отладки и режим оптимизации исключается. Это может увеличить размеры файлов. Файлы размещаются в папках Debug.
- Команда **Построение=>Перестроить решение**. Построение отлаженного проекта, когда в компонуемые файлы символы отладки не включаются и компилятор использует режим оптимизации кода (например, исключает не использованные переменные). Это может уменьшить размеры файлов. Файлы размещаются в папках Release.

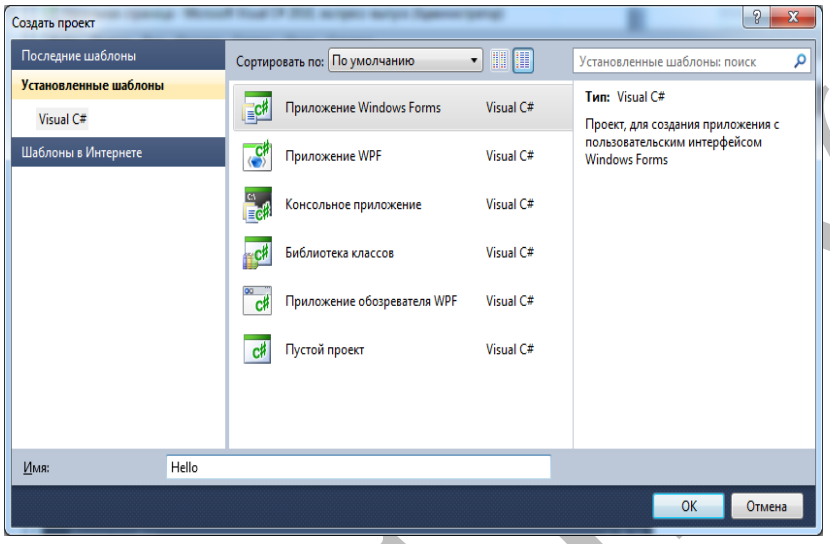
1.3. Windows Forms Hello

Создать проект Windows Forms приложения Hello. Оно должно при нажатии в форме кнопки «Нажми» выводить на экран фразу «Hello, World and Россия <Фамилия студента>».

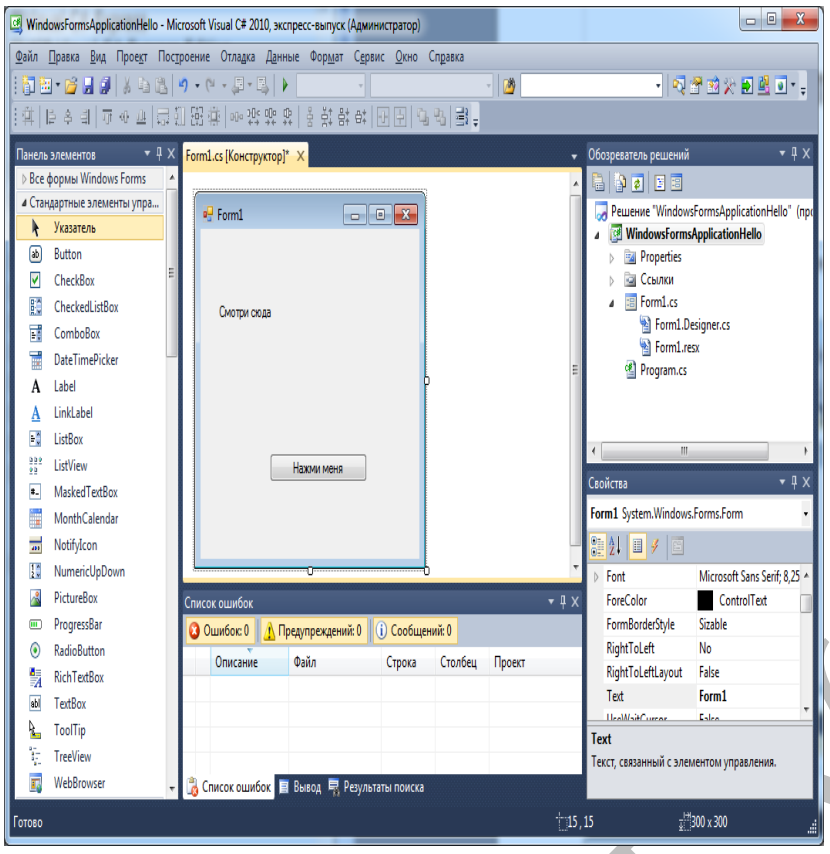
Варианты к заданию. Индивидуальные данные студента.

Пример. Создать проект Windows Forms приложения Hello. Оно должно при нажатии в форме кнопки «Нажми меня» выводить на экран фразу «Hello, World and Россия от Акчурина!».

Активизировать ИСР. В главном меню выбирается команда File=>New Project. Вызывается окно выбора типа проекта с набором шаблонов. В нем выбираем Приложение WindowsForms. Задаем имя проекта Hello.



Вид ИСР меняется.



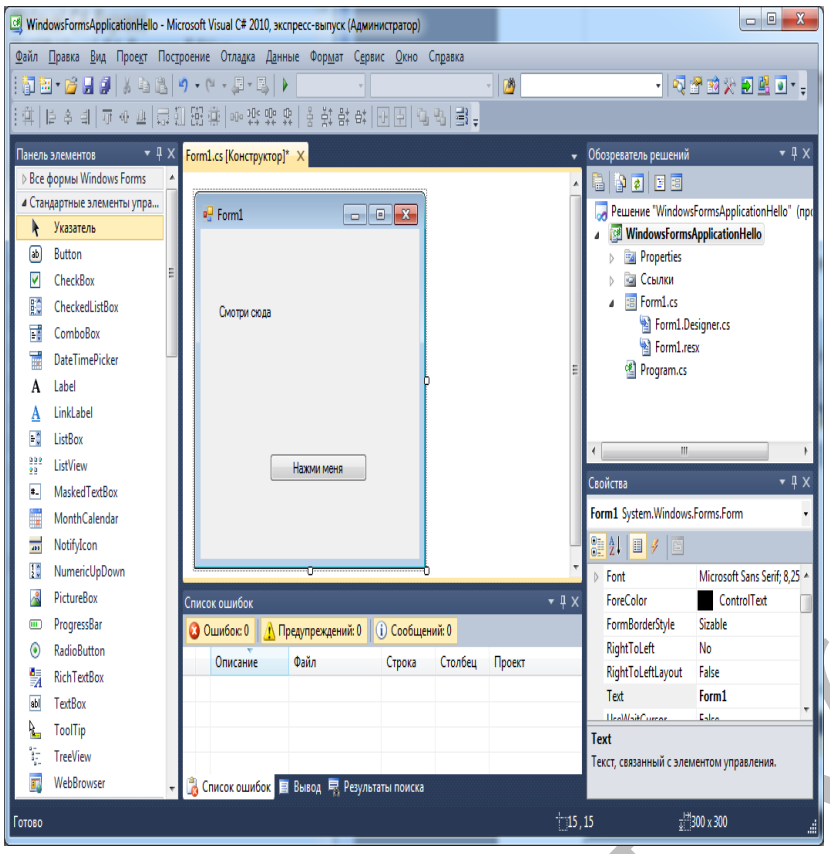
В центре в закладке **Form1.cs[Конструктор**] отображается окно **Конструктора** формы. Окно **Панель элементов** заполняется элементами для выбора. Окна **Обозреватель решений** содержит описание решения. **Конструктор** формы отображается по автоматически создаваемому коду (при желании его можно посмотреть двойным щелчком по **Form1.cs => Program.cs** в **Обозревателе решений**). **Редактор** кода модуля формы отображается командой **Перейти к коду**, которая находится в меню, выпадающем при щелчке по форме в конструкторе правой кнопкой мыши. Редактор отображается в закладке с именем **Form1.cs**.

Большая часть кода в **Редакторе** ИСР сделала автоматически. Нужно добавить функциональность.

Окна **Конструктора** и **Редактора** можно переключать кнопками в заголовках их закладок.

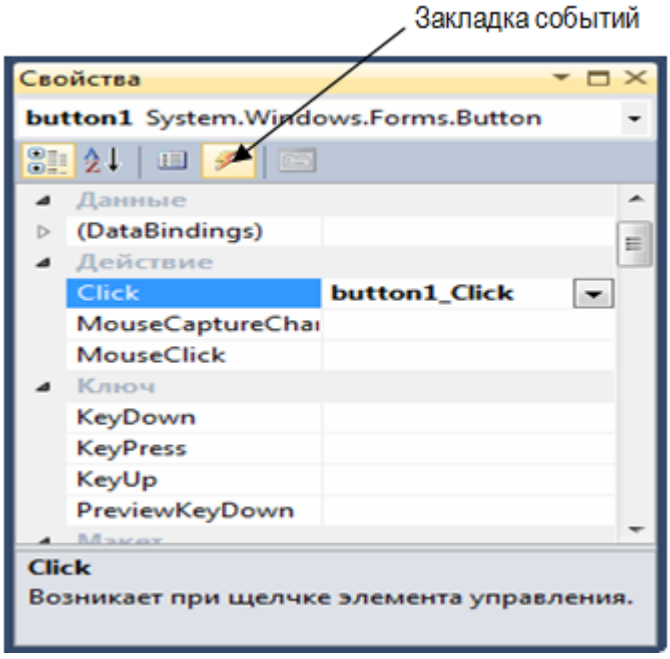
Теперь приступаем к проектированию в **Конструкторе**. Из окна Панели элементов перетаскиваем в форму объекты

- button1 – кнопка для запуска обработчика события. Выделяем объект, в окне свойств отображаются свойства кнопки. Свойству Text присваиваем значение - Нажми меня.
- label1 – метка, поле для отображения сообщения. Свойству Text присваиваем значение – Смотри сюда.



Для создания обработчика события щелчка по кнопке дважды щелкаем по кнопке в форме. Автоматически отображается окно **Редактора**, в котором в код добавлен шаблон обработчика события button1_Click, но без функциональности. Курсор устанавливается в место ввода кода, который будет задавать функциональность проекта.

Чтобы обработчик события срабатывал, нужно в окне свойств кнопки button1 в закладке событий выбрать реакцию на щелчок по кнопке из списка:



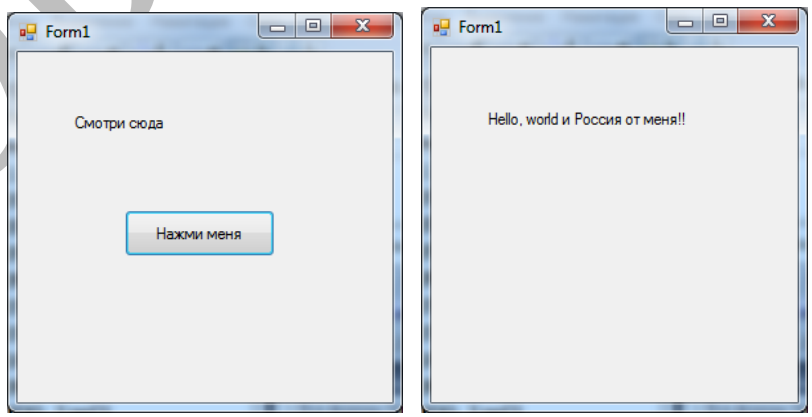
В шаблон кода, начиная с позиции курсора, нужно ввести инструкции. В примере свойству Text объекта label1 нужно присвоить строку “Hello, world and Россия от меня!!”. Чтобы исключить повторный доступ к кнопке, сделаем ее после вывода текста невидимой. Для этого вводим код

```
label1.Text= "Hello, world и Россия от меня!!";
button1.Visible = false;
```

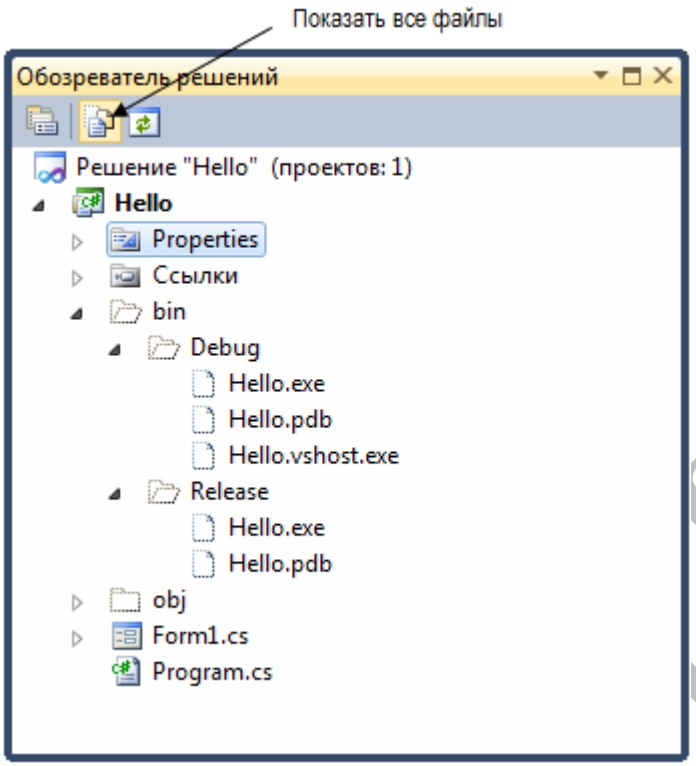
Листинг программы

```
using System;
using System.Windows.Forms;
namespace Hello
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         label1.Text = "Hello, world и Россия от меня!!";
         button1.Visible = false;
      }

      private void Form1_Load(object sender, EventArgs e)
      {
      }
   }
}
```

Проект готов, проверим его командой **Отладка=>Запуск без отладки**. Получим окно приложения.



Проект готов, сохраняем его командой **Файл=>Сохранить все** в папке Hello. В результате проект сохраняется в структуре папок (чтобы увидеть все файлы, нужно активизировать кнопку, показанную на рисунке):



Решение включает:

Решение Hello	Решение
Hello	Проект
Properties	Свойства
Ссылки	Ссылки
bin	Двоичные файлы
Debug	Файлы отладки
Hello.exe	Управляемый
Hello.pdb	исполняемый файл
Hel-lo.vshost.exe	База данных для JIT компилятора
Release	Служебный файл
obj	Файлы выпуска
	Объектные файлы

Исполняемые (bin) и объектные (obj) файлы образуются при компиляции (построении). Возможны два режима:

- Команда **Построение=>Построить решение**. Построение в режиме отладки, в компонуемые файлы включаются символы отладки и режим оптимизации исключается. Это может увеличить размеры файлов. Файлы размещаются в папках Debug.
- Команда **Построение=>Перестроить решение**. Построение отлаженного проекта, когда в компонуемые файлы символы отладки не включаются и компилятор использует режим оптимизации кода (например, исключает не использованные переменные). Это может уменьшить размеры файлов. Файлы размещаются в папках Release.

2. Численные типы в языке C#

Предмет исследований

- Структура программы на языкеC#.
- Задание констант, переменных.
- Типы численных данных и ошибки при преобразованиях типов.
- Организация простейшего ввода-вывода данных.
- Программа решения задачи в виде консольного приложения.

Контрольные вопросы

1. Алфавит языка C#, операции, идентификаторы.
2. Структура программы.
3. Переменные. Их объявление.
4. Форматы представления чисел (с фиксированной и плавающей точкой).
5. Типы целых чисел без знака: Byte.
6. Типы целых чисел со знаком: int, Int16, Int32, Int64, sbyte.
7. Типы вещественных чисел (с плавающей точкой): double,Single.
8. Тип чисел decimal.
9. Совместимость типов при присвоениях.

Задание. Создать консольную программу взаимных преобразований численных типов данных в соответствии с вариантом. В программе должны быть:

- Преобразования тип unt d другие целочисленные типы.
- Взаимные преобразования типа int и типов с плавающей точкой.
- Преобразование типа louble в типы Single и int.

При преобразованиях нужно проверять получаемые значения и размкр в байтах (метод sizeof).

Данные в консоль выводятся инструкцией Console.WriteLine(), в скобках строка. Перед выводом численных данных они должны быть преобразованы в строку методом Convert.ToString(). Для вывда нескольких данных WriteLine() можно использовать двумя способами:

- С конкатенацией строк - WriteLine(s1+s2+s3),
- С форматным выводом - WriteLine(”{0,L1} {1,L2} {3,L3} “,s1,s2,s3). Имеетя одна строка, в которую вставляются форматы вставки подстрок, список имен которых следует за строкой. Формат заключается в фигурные скобки и включает через запятую номер ввода (начиная с нуля) и число пробелов L до вводимой подстроки. В коде прогаммы значения L нужно подобрать, чтобы получит красивую картинку в консоли.

Варианты к заданию

№	Значение вещественного числа	Значение целого числа
1.	123.45	150
2.	23.456	160

3.	34.567	170
4.	456.78	180
5.	567.89	190
6.	67.890	200
7.	78.901	210
8.	890.12	220
9.	91.012	230
10.	109.876	240

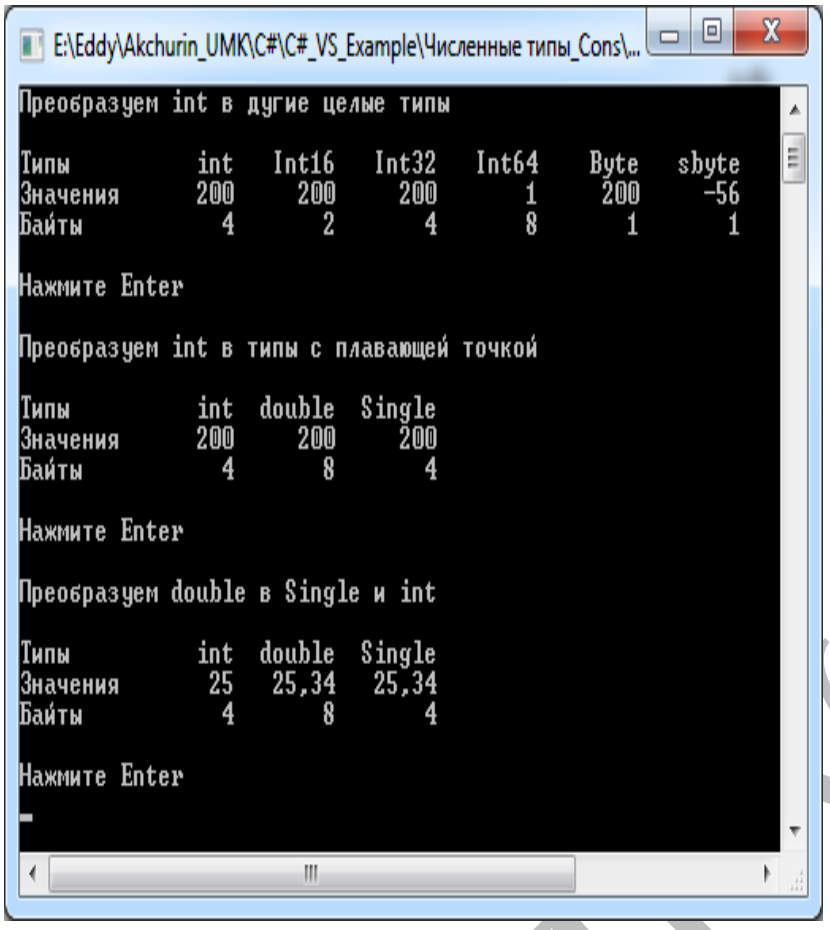
Пример. Создать программу взаимных преобразований типов численных данных в соответствии с вариантом.

Листинг программы

```
using System;

namespace ConsoleChisloTip
{
   class Program
   {
      static void Main()
      {
         Console.WriteLine("Преобразуем int в дугие целые типы");
         int i = 200;              // Целое типа int
         Int16 i1 = (Int16)i;      // int => int16
         Int32 i2 = i;             // int => int32
         Int64 i3 = i;             // int => int64
         Byte i4 = (Byte)i;        // int => Byte
         sbyte i5 = (sbyte)i;      // int => sbyte
         Console.WriteLine();
         Console.WriteLine("Типы     {0,8}{1,8}{2,8}{3,8}{4,8}{5,8}",
            "int", "Int16", "Int32", "Int64", "Byte", "sbyte");
         Console.WriteLine("Значения {0,8}{1,8}{2,8}{3,8}{4,8}{5,8}",
            i, i1, i2, i3, i4, i5);
         Console.WriteLine("Байты    {0,8}{1,8}{2,8}{3,8}{4,8}{5,8}",
            sizeof(int), sizeof(Int16), sizeof(Int32), sizeof(Int64),
            sizeof(byte), sizeof(sbyte));
         Console.WriteLine();              // Пропуск строки
         Console.WriteLine("Нажмите любую клавишу");
         Console.WriteLine();
         Console.ReadKey();                // Пауза
         Console.WriteLine();
         Console.WriteLine("Преобразуем int в типы с плавающей точкой");
         Console.WriteLine();
         double d = i;                  // int => double
         Single s = i;                  // int => Single
```

```
            Console.WriteLine("Типы     {0,8}{1,8}{2,8}",
               "int", "double", "Single");
            Console.WriteLine("Значения {0,8}{1,8}{2,8}",
               i, d, s);
            Console.WriteLine("Байты    {0,8}{1,8}{2,8}",
               sizeof(int), sizeof(double), sizeof(Single));
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
            Console.WriteLine();
            Console.WriteLine("Преобразуем double в Single и int");
            d = 25.34;                    // Число типа double
            Console.WriteLine();
            i = (int)d;                   // double => int
            s = (Single)d;                // double => Single
            Console.WriteLine("Типы     {0,8}{1,8}{2,8}",
               "int", "double", "Single");
            Console.WriteLine("Значения {0,8}{1,8}{2,8}",
               i, d, s);
            Console.WriteLine("Байты    {0,8}{1,8}{2,8}",
               sizeof(int), sizeof(double), sizeof(Single));
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Стартовые числа Start. Значение int_1 выбрано таким, чтобы оно умещалось в формате выбранного типа, но выходило за пределы типа sbyte. Объявлены переменные для каждого вида чисел.


E:\Eddy\Akchurin_UMK\C#\C#_VS_Example\Численные типы_Cons\...
Преобразуем int в дугие целые типы
Типы int Int16 Int32 Int64 Byte sbyte
Значения 200 200 200 1 200 -56
Байты 4 2 4 8 1 1
Нажмите Enter
Преобразуем int в типы с плавающей точкой
Типы int double Single
Значения 200 200 200
Байты 4 8 4
Нажмите Enter
Преобразуем double в Single и int
Типы int double Single
Значения 25 25,34 25,34
Байты 4 8 4
Нажмите Enter

3. Строковые и символьные типы в языке C#

Предмет исследований

- Структура программы на языкеC#.
- Задание констант, переменных.
- Типы строкых и символьных типов.
- Организация ввода-вывода строковых данных.

Контрольные вопросы

1. Алфавит языка C#, операции, идентификаторы.
2. Структура программы.
3. Переменные. Их объявление.
4. Символьный тип: char.
5. Строковыq тип string.
6. Совместимость типов при присвоениях.
7. Назначение форматирования чисел.
8. Стандартные форматы. G формат – общий. F формат - с фиксированной точкой. N формат – числовой. E формат – научный. C формат – денежный. P формат – процентный. D формат – десятичный.
9. Нестандартные форматы. Форматы с символами E+0 и E-0. формат с символом 0. формат с символом #. Формат с символом точка. Формат с символом запятая. Формат с символом %.

3.1. Взаимные преобразования

Создать консольную программу взаимных преобразований строковых и символьных типов. Она должна отображать фамилию студента строкой и добавлять к строке символ.

Варианты к заданию

№	Значение символа
1.	@
2.	#
3.	$
4.	%
5.	*
6.	+
7.	?
8.	^
9.	&
10.	/

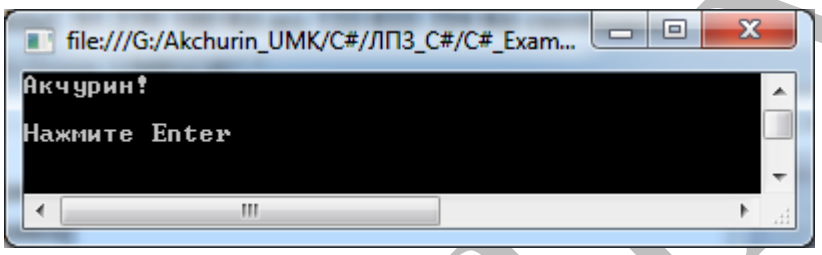
Пример. Создать программу взаимных преобразований строковых и символьных типов. Она должна отображать строку Фамилия строками разного типа и добавлять к строке символ восклицательного знака (!).

Листинг программы

```
using System;

namespace ConsoleChar
```

```
{
   class Program
   {
      static void Main()
      {
         string Фамилия;
         char Символ;
         Фамилия = "Акчурин";                    // тип string
         Символ = '!';                           // тип char
         Console.WriteLine(Фамилия + Символ);
         Console.WriteLine();                    // Пропуск строки
         Console.WriteLine("Нажмите Enter");
         Console.ReadLine();                     // Пауза
      }
   }
}
```

Это результат его прогона:



3.2. Стандартное форматирование

Создать консольную программу вывода целого и вещественного чисел в стандартных форматах. **Варианты к заданию.** Целое число – номер зачетной точки, вещественное – номер зачетной книжки с 4-мя последними цифрами в дробной части.

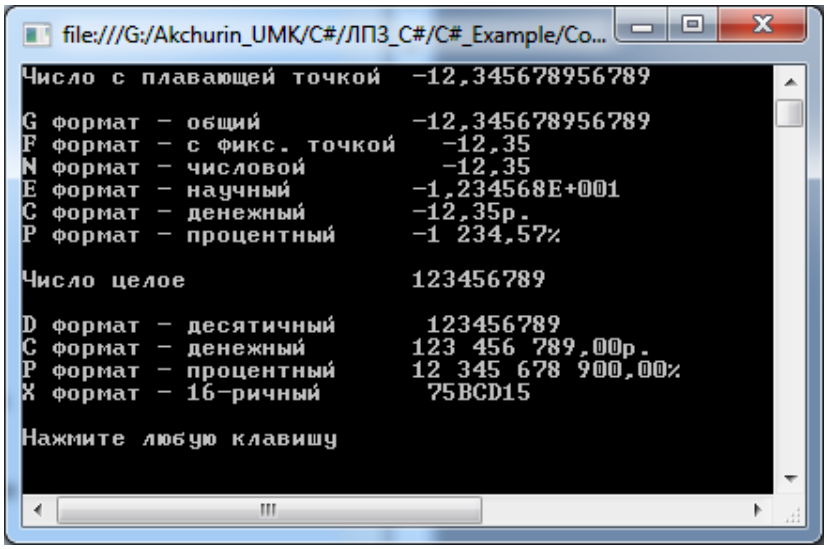
Пример. Создать консольную программу вывода целого 123456789 и вещественного -12.345678956789 чисел в стандартных форматах.

Листинг программы

```
using System;
namespace ChislaFormatStand
{
   class Program
   {
      static void Main()
      {
         double d = -12.345678956789;
         int i = 123456789;
         Console.Write("Число с плавающей точкой  ");
         Console.WriteLine(d);
```

```
            Console.WriteLine();
            Console.Write("G формат - общий          ");
            Console.WriteLine("{0,8:G}", d);
            Console.Write("F формат - с фикс. точкой ");
            Console.WriteLine("{0,8:F}", d);
            Console.Write("N формат - числовой       ");
            Console.WriteLine("{0,8:N}", d);
            Console.Write("E формат - научный        ");
            Console.WriteLine("{0,8:E}", d);
            Console.Write("C формат - денежный       ");
            Console.WriteLine("{0,8:C}", d);
            Console.Write("P формат - процентный     ");
            Console.WriteLine("{0,8:P}", d);
            Console.WriteLine();
            Console.Write("Число целое               ");
            Console.WriteLine(i);
            Console.WriteLine();
            Console.Write("D формат - десятичный     ");
            Console.WriteLine("{0,8:D}", i);
            Console.Write("C формат - денежный       ");
            Console.WriteLine("{0,8:C}", i);
            Console.Write("P формат - процентный     ");
            Console.WriteLine("{0,8:P}", i);
            Console.Write("X формат - 16-ричный      ");
            Console.WriteLine("{0,8:X}", i);
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Это результат его прогона:



3.3. Нестандартное форматирование

Создать консольную программу вывода вещественного числа в нестандартных форматах. **Варианты к заданию.** Вещественное число со знаком минус, целая и дробная части – номер зачетной книжки.

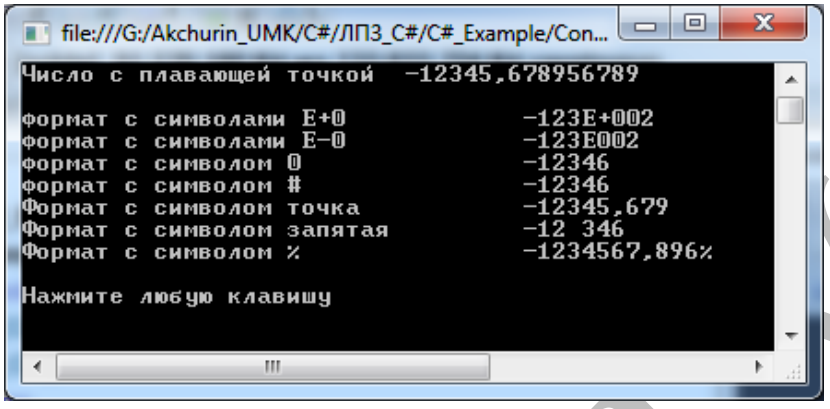
Пример. Создать консольную программу вывода в нестандартных форматах числа (-12345.678956789).

Листинг программы

```
using System;

namespace ChislaFormatNestand
{
   class Program
   {
      static void Main()
      {
         double d = -12345.678956789;
         Console.Write("Число с плавающей точкой  ");
         Console.WriteLine(d);
         Console.WriteLine();
         Console.Write("формат с символами E+0          ");
         Console.WriteLine("{0:000E+000}", d);
         Console.Write("формат с символами E-0          ");
         Console.WriteLine("{0:000E-000}", d);
         Console.Write("формат с символом 0             ");
         Console.WriteLine("{0:000}", d);
         Console.Write("формат с символом #             ");
         Console.WriteLine("{0:###}", d);
         Console.Write("Формат с символом точка         ");
         Console.WriteLine("{0:###.000}", d);
         Console.Write("Формат с символом запятая       ");
```

```
            Console.WriteLine("{0:###,000}", d);
            Console.Write("Формат с символом %              ");
            Console.WriteLine("{0:###.000%}", d);
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Это результат его прогона:



4. Тип DateTime в языке C#

Предмет исследований

- Структура программы на языкеC#.
- Задание констант, переменных.
- Тип DateTime.
- Компонент DateTimePicker.
- Программа решения задачи в виде консольного приложения.
- Программа решения задачи в виде Windows приложения.

Контрольные вопросы

1. Алфавит языка C#, операции, идентификаторы.
2. Структура программы.
3. Переменные. Их объявление.
4. Тип DateTime. Назначение
5. Компонент DateTimePicker.
6. Совместимость типов при присвоениях.

4.1. Консольное приложение

Создать консольное приложение для работы с типом DateTime. Оно должно запрашивать дату и время рождения студента, выводить текущие дату/время, запрашивать желаемое время дальнейшего обучения, выводить дату/время и день недели конца обучения.

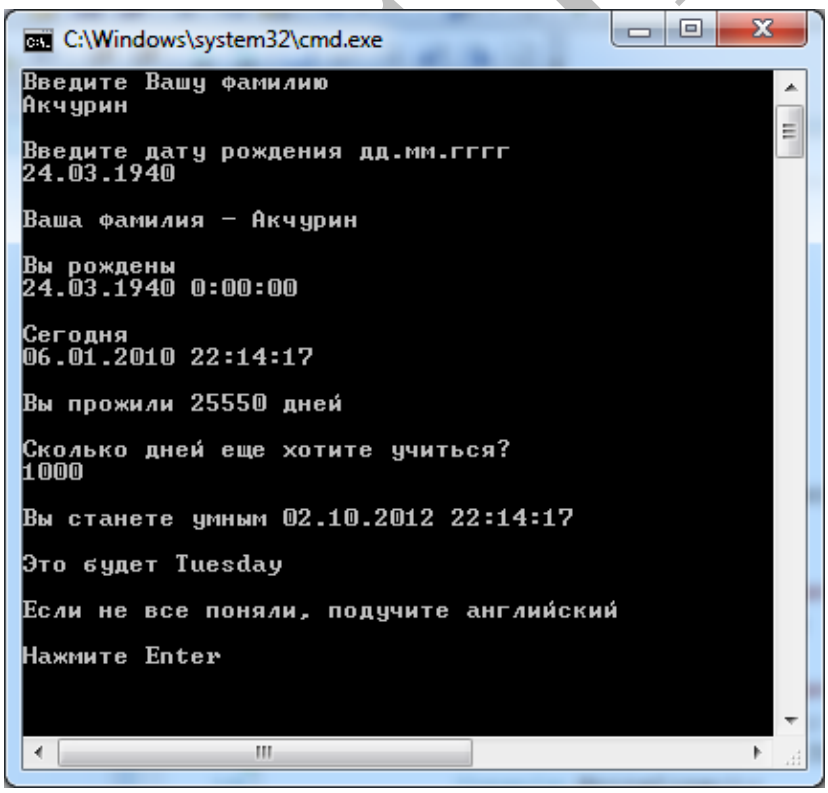
Варианты к заданиям. Индивидуальные данные студента.

Пример.

Листинг программы

```
using System;
namespace ConsoleDateTime
{
   class Program
   {
     static void Main()
     {
         string d1, s, f;
         Console.WriteLine("Введите Вашу фамилию");
         f = Console.ReadLine();
         Console.WriteLine();
         Console.WriteLine("Введите дату рождения дд.мм.гггг");
         d1 = Console.ReadLine();
         DateTime dt1 = Convert.ToDateTime(d1);
         Console.WriteLine();
         Console.WriteLine("Ваша фамилия - " + f);
         Console.WriteLine();
```

```
            Console.WriteLine("Вы рождены");
            Console.WriteLine(dt1);
            DateTime dt2 = DateTime.Now;
            Console.WriteLine();
            Console.WriteLine("Сегодня");
            Console.WriteLine(dt2);
            Console.WriteLine();
            int i = (dt2.Year - dt1.Year) * 365;
            Console.WriteLine("Вы прожили " + i + " дней");
            Console.WriteLine();
            Console.WriteLine("Сколько дней еще хотите учиться?");
            s = Console.ReadLine();
            dt2 = dt2.AddDays(Convert.ToInt32(s));
            Console.WriteLine();
            Console.WriteLine("Вы станете умным " + dt2);
            Console.WriteLine();
            Console.WriteLine("Это будет " + dt2.DayOfWeek);
            Console.WriteLine();
            Console.WriteLine("Если не все поняли, подучите английский");
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```



4.2. Windows Forms приложение

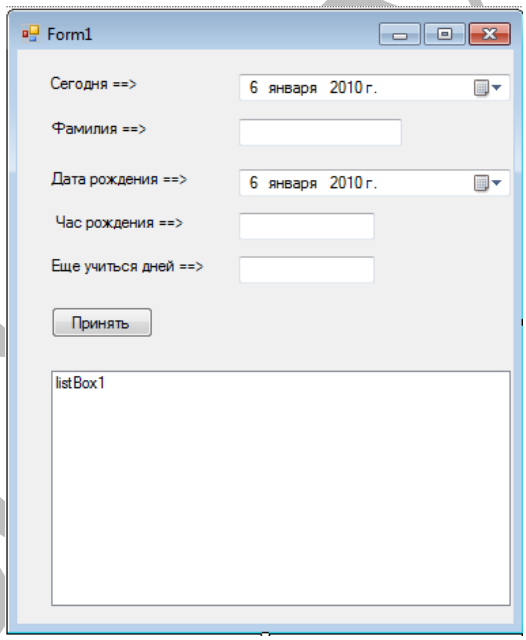
Создать Windows Forms приложение для работы с типом DateTime. Оно должно запрашивать дату и время рождения студента, выводить текущие дату/время, запрашивать желаемое время дальнейшего обучения, выводить дату/время и день недели конца обучения.

Варианты к заданиям. Индивидуальные данные студента.

Пример. Активизируем ИСР, выбираем создание Windows приложения. В форму заносим компоненты:

- 2 компонента dateTimePicker, один для отображения текущей даты, второй для редпктирования даты рождения.
- 5 компонентов label для размещения поясняющих текстов.
- 2 компонента TextBox для ввода данных.
- Компонент listBox для вывода данныхю
- Компонент button (кнопка) для запуска обработчика события.

Компонентам назначаем свойства, используя окно свойств ИСР. Итог:



Двойным щелчком по кнопке инициируем в редакторе кода шаблон обработчика события, в который заносим программу.

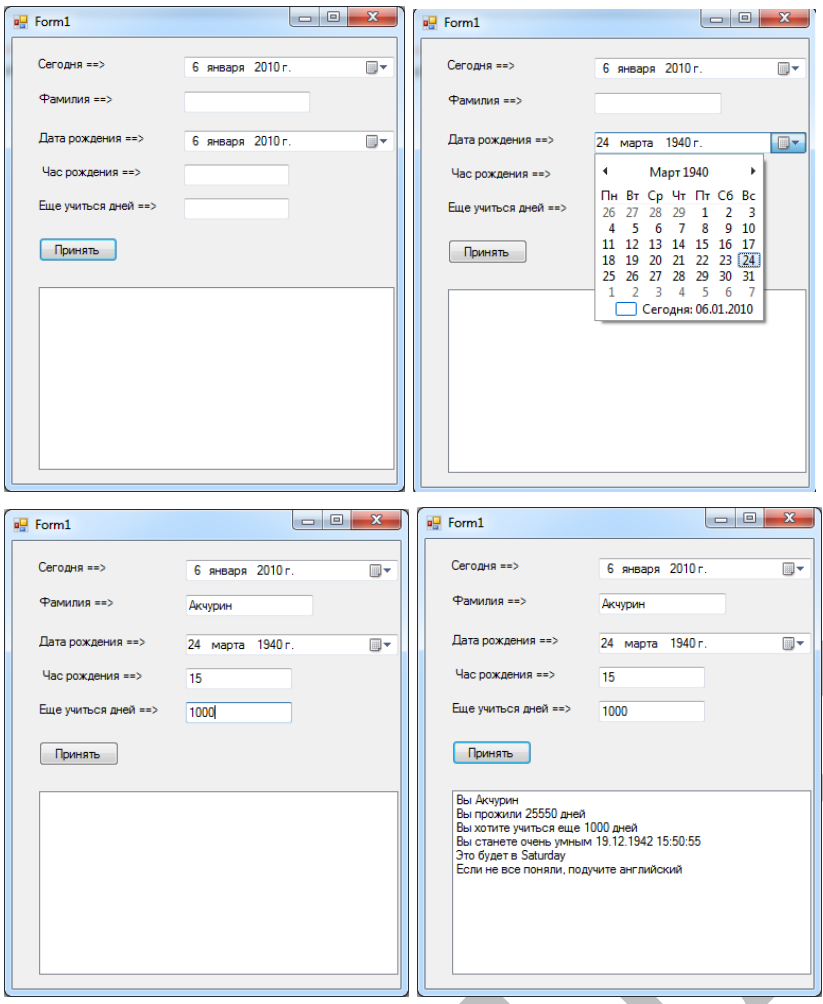
Листинг программы

```
using System;
using System.Windows.Forms;
namespace WindowsFormsDateTime
{
   public partial class Form1 : Form
   {
      public Form1()
      {
```

```
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        string f = textBox1.Text;
        int c = Convert.ToInt32(textBox2.Text);
        int c1 = Convert.ToInt32(textBox3.Text);
        dateTimePicker2.MaxDate = DateTime.Now;
        DateTime dt1 = dateTimePicker2.Value;
        DateTime dt2 = DateTime.Now;
        int d = (dt2.Year - dt1.Year)*365;
        string s = "Вы " + f;
        listBox1.Items.Add(s);
        s = Convert.ToString(d);
        s = "Вы прожили " + s + " дней";
        listBox1.Items.Add(s);
        s = Convert.ToString(c1);
        s = "Вы хотите учиться еще " + s + " дней";
        listBox1.Items.Add(s);
        dt1 = dt1.AddDays(c1);
        s = "Вы станете очень умным " + dt1;
        listBox1.Items.Add(s);
        s = Convert.ToString(dt1.DayOfWeek);
        s = "Это будет в " + s;
        listBox1.Items.Add(s);
        s = "Если не все поняли, подучите английский";
        listBox1.Items.Add(s);
    }
  }
}
```

Далее окна формы на этапах прогона:

- Сначала стартовое Окно.
- Затем окно при редактирваии компонента DateTimePicker2. В нем нужно установить год, Затем из списка вызвать коллекцию календарей месяцев, в ней выбрать календарь нужного месяца, в котором выбрать день.
- Затем заполнить поля формы. Час рождения должен быть от 0 до 24.
- Нажатие кнопки Принять приводит к обработке данных и формирование итогов в компоненте ListBox1.


Form1
Сегодня ==>
6 января 2010 г.
Фамилия ==>
Дата рождения ==>
Час рождения ==>
Еще учиться дней ==>
Принять
Март 1940
Пн Вт Ср Чт Пт Сб Вс
Сегодня: 06.01.2010
Акчурин
24 марта 1940 г.
15
1000
Вы Акчурин
Вы прожили 25550 дней
Вы хотите учиться еще 1000 дней
Вы станете очень умным 19.12.1942 15:50:55
Это будет в Saturday
Если не все поняли, подучите английский

5. Линейные структуры

Предмет исследований

- Запись констант, переменных, стандартных функций.
- Правила записи арифметических выражений.
- Арифметические операторы присваивания.
- Разработка алгоритма решения в соответствии с заданием.
- Составление программы решения задачи в виде консольного приложения.

Контрольные вопросы

1. Алфавит языка C#.
2. Операции.
3. Идентификаторы.
4. Типы данных.
5. Структура программы консольного приложения.
6. Где описываются константы, переменные и типы данных?
7. Стандартные функции.
8. Операторы присваивания.
9. Пустая и составная инструкция.
10. Процедуры ввода Read и ReadLine.
11. Процедуры вывода Write и WriteLine.
12. Последовательность действий при выполнении оператора присваивания.
13. Приоритетность выполнения операций в выражениях.
14. Как организовать пропуск одной, двух строк при выводе?

Задание. Вычислить значения переменных в соответствии с вариантами задания. Вывести значения вводимых исходных данных и результаты, сопровождая их вывод именами выводимых переменных. Задание выполнить в виде консольного приложения.

Варианты заданий

№	Расчетные формулы	Данные
1.	$s = \left\lvert x^{y/x} - \sqrt{\frac{y}{x}} \right\rvert;$ $w = (y - x)\frac{y - \frac{z}{y - x}}{1 + (y - x)^2}$	x = 1.82 y = 18 z = -3.29
2.	$s = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!};$ $w = x\left(\sin(x) + \cos(y)\right)$	x = 0.33 y = 0.02
3.	$y = e^{-bt}\sin(at + b) - \sqrt{\lvert bt + a \rvert};$ $s = b\sin\left(at^2\cos(at)\right) - 1$	a = -0.5 b = 1.7 t = 0.44
4.	$w = \sqrt{x^2 + b} - \frac{b^2\sin^3(x + a)}{x};$ $y = \cos^2(x^3) - x/\sqrt{a^2 + b^2}$	a = -0.5 b = 15.5 x = -2.9
5.	$s = x^3 tg^2\left((x + b)^2\right) + \frac{a}{\sqrt{x + b}};$ $g = \frac{bx^2 - a}{e^{ax} - 1}$	a = 16.5 b = 3.4 x = 0.61
6.	$r = \frac{x^2(x + 1)}{b} - \sin^2(x + a);$ $s = \sqrt{\frac{xb}{a}} + \cos\left((x + b)^3\right)$	a = 0.7 b = 0.05 x = 0.5
7.	$y = \sin^3\left(\left(x^2 + a\right)^2\right) - \sqrt{\frac{x}{b}};$ $z = \frac{x^2}{a} + \cos\left((x + b)^3\right)$	a = 1.1 b = 0.04 x = 0.2
8.	$y = b \cdot tg^2 x - \frac{a}{\sin^2(x/a)};$ $d = a \cdot e^{\sqrt{a}} \cdot \cos(bx/a)$	a = 3.2 b = 17.5 x = -4.8
9.	$f = \ln\left(a + x^2\right) + \sin^2(x/b);$ $z = e^{-\alpha} \cdot \frac{x + \sqrt{x + a}}{x - \ln(\lvert x - b \rvert)}$	a = 10.2 x = 2.2 b = 9.2 c = 0.5
10.	$y = \frac{a^{2x} + b^{-x} \cdot \cos((a + b)x)}{x + 1};$ $r = \sqrt{x^2 + b} - b^2\sin(x + a)/x$	a = 0.3 b = 0.9 x = 0.61

Пример. Вычислить при x = 2.1, y = 0.59, z = -4.8 значения a и b, используя формулы:

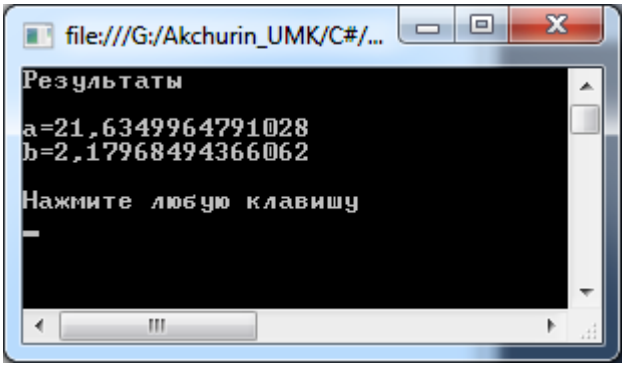
$$a = y \cdot tg^3(x^2) + \sqrt{\frac{z^2}{y^2 + x^2}};$$

$$b = \ln\left(y + x^2\right) + \sin^2(z/x)$$

Листинг программы

```
using System;
namespace ConsoleLineStr
{
   class Program
   {
      static void Main()
      {
         double x =2.1, y =0.59, z = -4.8, a =0, b = 0;  // Переменные типа double
         a = y * Math.Pow(Math.Tan(x * x), 3);// Вычисляем a
         a += Math.Sqrt(z * z / (y * y + x * x));
         Console.WriteLine("Результаты");
         Console.WriteLine();
         Console.Write("a=");              // Вывод a
         Console.WriteLine(a.ToString());
         b = Math.Log(y + x * x);          // Вычисляем b
         b += Math.Pow(Math.Sin(z / x), 2);
         Console.Write("b=");              // Вывод b
         Console.WriteLine(b.ToString());
         Console.WriteLine();
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();                // Пауза
      }
   }
}
```

Внимание. При вводе данных в консоли разделитель целой и дробнй части вещественного числа – запятая.



6. Ветвления

Предмет исследований

- Условная и безусловная передача управления;
- Вычислительные процессы с разветвляющейся структурой.
- Разработать алгоритмы решения в соответствии с заданием.
- Составить программы решения задач.

Контрольные вопросы

1. Какие структуры вычислительных процессов Вы знаете?
2. Как организовать разветвление вычислений?
3. Ветвление if... else.
4. Вложенные ветвления.
5. Инструкция выбора switch.
6. Фраза case.
7. Зачем во фразе case применяется оператор break?

6.1. Ветвление if; else

Вычислить значения функции по варианту задания. Вывести значения исходных данных и полученные результаты, сопровождая их именами переменных. Значения аргумента взять из указанного диапазона так, чтобы протестировать все ветви программы. Проект – консольное приложение.

Варианты заданий

№	Функции и условия	Данные	Диапазон
1.	$y = \begin{cases} 1 \cdots при \cdots t\langle 1 \\ at^2 \ln t \cdots при \cdots 1 \le t \le 2 \\ e^{at} \cos bt \cdots при \cdots t\rangle 2 \end{cases}$	a = -0.5 b = 2	t = [0 .. 3] шаг 0.25
2.	$y = \begin{cases} \pi x^2 - 7 / x^2 \cdots при \cdots x\langle 1.3 \\ ax^3 + 7\sqrt{x} \cdots\cdots при \cdots x = 1.3 \\ tg\left(x + 7\sqrt{x}\right) \cdots\cdots при \cdots x\rangle 1.3 \end{cases}$	a = 1.5	x=[0.8..2.8] шаг 0.25
3.	$y = \begin{cases} ax^2 + bx + c \cdots при \cdots x\langle 1.2 \\ a / x + \sqrt{x^2 - 1} \cdots при \cdots x = 1.2 \\ (a + bx) / \sqrt{x^2 + 1} \cdots при \cdots x\rangle 1.2 \end{cases}$	a = 2.8 b = -0.3 c = 4	x = [1 .. 2] шаг 0.25
4.	$y = \begin{cases} \pi x^2 - 7 / x^2 \cdots при \cdots x\langle 1.4 \\ ax^3 + 7\sqrt{x^2 - 1} \cdots при \cdots x = 1.4 \\ (a + bx) / \sqrt{x^2 + 1} \cdots при \cdots x\rangle 1.4 \end{cases}$	a = 1.65 b = 1.1	x = [0.5 .. 2] шаг 0.25
5.	$y = \begin{cases} 1.5 \cos^2 x \cdots при \cdots x \le 1 \\ (x-2)^2 + 6 \cdots при \cdots 1\langle x\langle 2 \\ 3tgx \cdots при \cdots x\rangle 2 \end{cases}$	a = 2.3	x = [0.2 .. 3] шаг 0.4
6.	$q = \begin{cases} bx - \lg bx \cdots при \cdots bx\langle 1 \\ 1 \cdots при \cdots bx = 1 \\ bx + \ln bx \cdots при \cdots bx\rangle 1 \end{cases}$	a = 2.5	x =[1 .. 2.5] шаг 0.25
7.	$q = \begin{cases} bx - \lg bx \cdots при \cdots bx\langle 1 \\ 1 \cdots при \cdots bx = 1 \\ bx + \ln bx \cdots при \cdots bx\rangle 1 \end{cases}$	b = 1.5	x= [0.5 .. 3] шаг 0.25
8.	$q = \begin{cases} \sin(\ln x) \cdots при \cdots x\rangle 3.5 \\ \cos^2 x \cdots при \cdots x \le 3.5 \end{cases}$		x = [2 .. 5] шаг 0.25
9.	$f = \begin{cases} \ln(x+1) \cdots при \cdots x\rangle 1 \\ \sin^2\left(\sqrt{\lvert ax \rvert}\right) \cdots при \cdots x \le 1 \end{cases}$	a = 20.3	x= [0.5 .. 2] шаг 0.25
10.	$y = \begin{cases} a \cdot \ln x + \sqrt{\lvert x \rvert} \cdots при \cdots x\rangle 1 \\ 2a \cdot \cos x + 3x^2 \cdots при \cdots x \le 1 \end{cases}$	a = 0.9	x= [0.5 .. 2] шаг 0.5

Пример. Вычислить при y=1.3, x=[0.. 2.1] с шагом 0.3 значения функции a. Результат вывести в виде таблицы. Проект – консольное приложение.

$$a = \begin{cases} yx + 1 \cdots при \cdots yx > 1 \\ \cos(yx) \cdots при \cdots yx = 1 \\ e^{-yx} \cos(yx) \cdots при \cdots yx < 1 \end{cases}$$

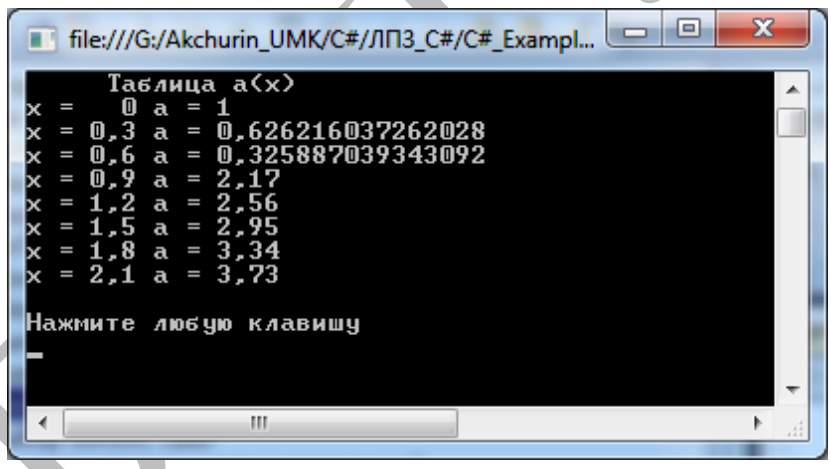
Блок-схема алгоритма представлена на рисунке. Для организации цикла введены следующие переменные: xs - начальное значение, xk - конечное значение dx - шаг изменения аргумента x.

Листинг программы

```
using System;

namespace ConsoleConditional_If
{
   class Program
```

```
{
 static void Main()
 {
  // Переменные типа double
 double a = 0, x = 0, xs = 0, xk = 2.1, dx = 0.3, y = 1.3;
 Console.WriteLine("     Таблица a(x)");   // Заголовок
 for (x = xs; (x <= xk); x = x + dx)                    // Начало цикла
     {
        if (y * x < 1)                    // Первое ветвление
            a = Math.Exp(-y * x) * Math.Cos(y * x);
        else if (y * x == 1)               // Вложенное ветвление
                a = Math.Cos(y * x + 1);
            else if (y * x > 1)           // Вложенное ветвление
                a = y * x + 1;
        Console.WriteLine("x = {0,3} a = {1}",x,a);
       }
       Console.WriteLine();
       Console.WriteLine("Нажмите любую клавишу");
       Console.ReadKey();                          // Пауза
```

Консоль перед закрытием программы:



6.2. Выбор switch; case

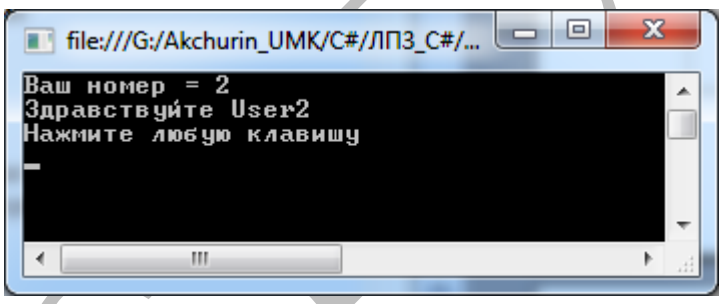
Вывести сообщения для подтверждения имени пользователя с использованием инструкции выбора switch. Проект – консольное приложение.

Пример. Вывести сообщения о номере пользователя с номерами от 1 до 3. Проект – консольное приложение. Пользователя просят ввести свой номер. Для каждого из известных пользователей выводится подтвердение его имени. Для пользователя с неизвестным номером сообщается, что он новичок.

Листинг программы

```
using System;
namespace ConsoleConditionalSwitch
{
```

```
    class Program
    {
        static void Main()
        {
            int user = 1;
            Console.Write("Ваш номер = ");
            user = Convert.ToInt32(Console.ReadLine());
            switch (user)
            {
                case 1: Console.WriteLine("Здравствуйте Userl"); break;
                case 2: Console.WriteLine("Здравствуйте User2"); break;
                case 3: Console.WriteLine("Здравствуйте User3"); break;
                default: Console.WriteLine("Здравствуйте новичок"); break;
            }
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();                  // Пауза
        }
    }
}
```



7. Циклы с неизвестным числом повторений

Предмет исследований

- Организация циклов с неизвестным числом повторений.
- Инструкции циклов while и do…while.
- Вычисление суммы членов бесконечного ряда.
- Разработать алгоритмы решения задачи.
- Составить программы решения задачи.

Контрольные вопросы

1. Циклический процесс с неизвестным числом повторений.
2. Его отличия от цикла с заданным числом повторений.
3. Инструкции языка C# для организации таких циклов. Их сравнение.
4. Синтаксис инструкции while.
5. Как выполнить группу операторов в цикле while?
6. Синтаксис инструкции do…while.
7. Синтаксис инструкции foreach.
8. Прямое вычисление суммы членов бесконечного ряда.
9. Вычисление суммы членов бесконечного ряда по рекуррентной формуле.
10. Условие выхода из цикла при вычислении суммы членов бесконечного ряда.

7.1. Цикл while

Вычислить значение суммы членов бесконечного ряда с заданной точностью E с использованием инструкции цикла while. На печать вывести значение суммы и число членов ряда, вошедших в сумму. Проект – консольное приложение.

Варианты заданий

№	Сумма членов ряда	x	Точность
1.	$s = -\frac{(2x)^2}{2!} + \frac{(2x)^4}{4!} + (-1)^n \frac{(2x)^{2n}}{(2n)!} + \cdots$	0.1	0.00001
2.	$s = x - \frac{x^3}{3} + \ldots + (-1)^{n-1} \frac{x^{2n-1}}{2n-1} + \ldots$	0.1	0.00005
3.	$s = \frac{x^3}{5} - \frac{x^5}{17} + (-1)^{n+1} \frac{x^{2n+1}}{4n^2+1} + \ldots$	0.15	0.001
4.	$s = 1 + \frac{x}{1!}\cos\frac{\pi}{4} + \cdots + \frac{x^n}{n!}\cos(\frac{\pi}{4}n) + \cdots$	0.12	0.0001
5.	$ch(x) = s = 1 + \frac{x^2}{2!} + \cdots + \frac{x^{2n}}{(2n)!} + \cdots$	0.7	0.0001
6.	$sh(x) = s = x + \frac{x^3}{3!} + \cdots + \frac{x^{2n+1}}{(2n+1)!} + \cdots$	1.7	0.001
7.	$arctg(x) = s = \frac{1}{x} + \cdots + (-1)^n \frac{1}{(2n+1)x^{2n+1}} + \cdots$	1.5	0.0005
8.	$\pi = s = 4\left(1 - \frac{1}{3} + \cdots + (-1)^n \frac{1}{2n+1} + \cdots\right)$		0.0001
9.	$\cos\frac{\pi}{6} = s = 1 - \frac{(\pi/6)^2}{2!} + \cdots + (-1)^n \frac{(\pi/6)^{2n}}{(2n)!} + \cdots$		0.00005
10.	$\sin\frac{\pi}{3} = s = \frac{\pi}{3} - \frac{(\pi/3)^3}{3!} + \cdots + (-1)^n \frac{(\pi/3)^{2n+1}}{(2n+1)!} + \cdots$		0.00005

Пример. Вычислить значение суммы членов бесконечного ряда

$$s = x - \frac{x^3}{3!} + \frac{X^5}{5!} + \cdots + (-1)^n \frac{x^{2n+1}}{(2n+1)!} + \cdots$$

при x = 0.1 с точностью до члена ряда с модулем, меньшим E=0.00001.

Для вычисления очередного члена ряда будем использовать рекуррентное соотношение, связывающее его с предыдущим членом a(n+1) = q*a(n). Применение рекуррентных формул позволяет избежать вычисления факториала и возведения в произвольную степень. Рекуррентный коэффициент q найдем из выражений для текущего и следующего членов ряда

$$a(n) = (-1)^n \frac{x^{2n+1}}{(2n+1)!} \quad \text{и} \quad a(n+1) = (-1)^{n+1} \frac{x^{2(n+1)+1}}{(2(n+1)+1)!}$$

Деля второе выражение на первое, получим

$$q = \frac{a(n+1)}{a(n)} = \frac{-x^2}{(2n+2)(2n+3)}$$

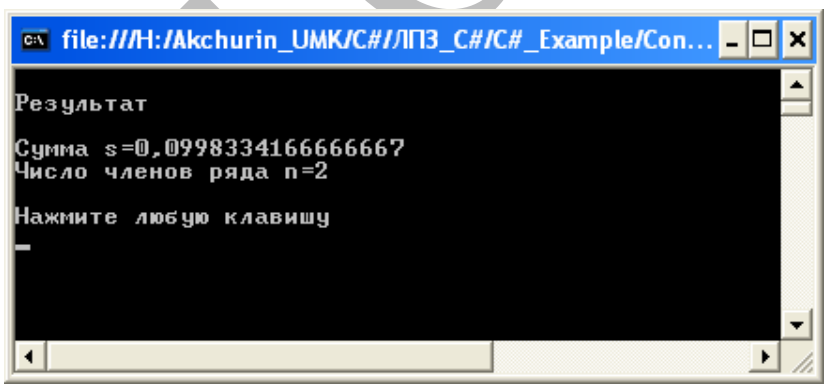
Значение начального члена ряда задаем до цикла путем прямого присваивания (номер начального члена n в разных вариантах равен 0 или 1, правильное значение определяется по формуле текущего члена). В нашем задании n=0, a=x.

Листинг программы

```
using System;
namespace ConsoleWhile
```

```
{
   class Program
   {
      static void Main()
      {
         double a=0, e=0.00001, q=0, s=0, x=0.1;
         int n = 0;
         a = x;                          // Инициализация цикла
         s = a;
         while (Math.Abs(a) > e)         // Цикл
         {
            q = -x * x / (2 * n + 2) / (2 * n + 3);
            a *= q;
            s += a;
            n++;
         }
         Console.WriteLine();
         Console.WriteLine("Результат");
         Console.WriteLine();
         Console.WriteLine("Сумма s=" + Convert.ToString(s));
         Console.WriteLine("Число членов ряда n=" + Convert.ToString(n));
         Console.WriteLine();
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();                          // Пауза

      }
   }
}
```

Консоль перед закрытием программы:



7.2. Цикл do…while

Выполнить ту же задачу с применением инструкции цикла do…while. Проект – консольное приложение.

Пример.

Листинг программы

```
using System;

namespace DoWhile
{
   class Program
   {
      static void Main()
      {
         double a = 0, e = 0.00001, q = 0, s = 0, x = 0.1;
         int n = 0;
         a = x;                    // Инициализация цикла
         s = a;
         do                        // Тело цикла
         {
            q = -x * x / (2 * n + 2) / (2 * n + 3);
            a *= q;
            s += a;
            n++;
         }
         while (Math.Abs(a) > e);    // Цикл повторять
         Console.WriteLine();
         Console.WriteLine("Результат");
         Console.WriteLine();
         Console.WriteLine("Сумма s = {0}",s);
         Console.WriteLine("Число членов ряда n = {0}",n);
         Console.WriteLine();
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();              // Пауза
      }
   }
}
```

Результат работы программы такой же, как для задания 1.

8. Циклы с заданным числом повторений

Предмет исследований

- Организация циклов с известным числом пвторений.
- Инструкция циклов for.
- Разработать алгоритмы решения задач.
- Составить программы решения задач.

Контрольные вопросы

1. Преимущества использования инструкций циклов в программе.
2. Инструкция цикла for.
3. Как организовать цикл с нарастанием индекса?
4. Как организовать цикл с убыванием индекса?
5. Организация вычисления суммы.
6. Организация вычисления произведения.

8.1. Команда Goto и метки

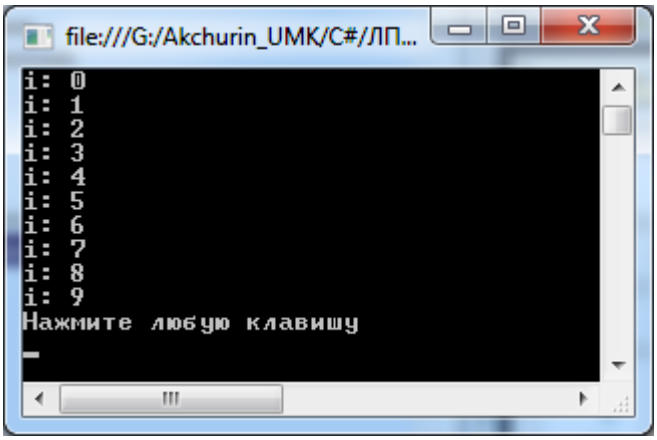
Вывести последовательность чисел (их число две последние цифры в номере зачетной книжки) с использованием инструкции goto и метки. Проект – консольное приложение.

Пример. Вычислить последовательность 10 чисел с использованием инструкции goto и метки. Проект – консольное приложение.

Листинг программы

```
using System;
namespace Goto
{
   class Program
   {
      static void Main()
      {
         int i = 0;
         M: Console.WriteLine("i: {0 } ", i);
         i = i + 1;
         if (i < 10) goto M;
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();
      }
   }
}
```

Результат прогона.



8.2. Цикл for

Вычислить значения функции с использованием инструкции цикла for. Проект – консольное приложение.

Варианты заданий 2 и 3

№	Задача 1	Задача 2
	a = 2.8; x = 5.1; c = 1.8; b = 0.81	
1.	$t=\sum_{n=2}^{5}\cos^2(na)$	$t=\sum_{n=1}^{10}\prod_{a=1}^{10}\cos(na)$
2.	$z=\prod_{k=1}^{12}(\cos(k)+\sin(k))$	$t=\prod_{a=1}^{5}\sum_{b=2}^{10}\sin(ab)$
3.	$p=\prod_{n=1}^{15}\cos(nbcx)$	$t=\prod_{b=1}^{10}\sum_{c=2}^{10}tg(bc)$
4.	$z=\sum_{k=1}^{18}(\cos(ak)+\sin(c))$	$t=\prod_{a=1}^{5}\sum_{c=1}^{10}\ln(ac)$
5.	$z=\prod_{n=1}^{15}(a^2+\ln(nc))$	$t=\prod_{a=2}^{10}\sum_{b=1}^{10}\cos(a+b)$
6.	$z=b\cdot\prod_{n=1}^{13}\cos(n^2b)$	$t=\prod_{b=2}^{5}\sum_{c=1}^{10}bc$
7.	$z=\prod_{n=1}^{40}(\cos(an)+bn)$	$t=\prod_{a=1}^{5}\sum_{c=1}^{3}(a^2\sqrt{c})$
8.	$t=\sum_{n=1}^{12}(x\cdot\sin(na))$	$t=\prod_{b=2}^{8}\sum_{c=1}^{10}(\sqrt{b}+c)$
9.	$u=\sum_{n=1}^{17}(n\cdot\sin^2(n^2a))$	$t=\prod_{a=2}^{5}\sum_{b=1}^{5}b^2a$
10.	$t=\sum_{n=2}^{10}5\cdot\lg(abn)$	$t=\prod_{c=2}^{7}\sum_{b=1}^{5}cb^2$

Пример. Вычислить значения функции с использованием инструкции цикла for. Проект – консольное приложение.

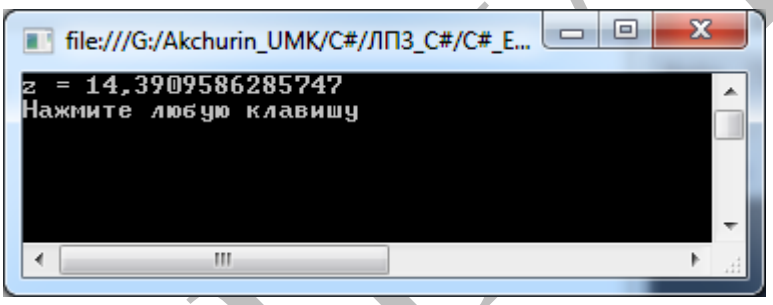
$$z=\sum_{i=1}^{20}x^2/i$$

Листинг программы

```
using System;
```

```
namespace For1
{
   class Program
   {
      static void Main()
      {
         double z = 0, x = 2;
         for (int i = 1; i < 21; i++) z += x * x / i;
         Console.WriteLine("z = {0}", z);
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();
      }
   }
}
```

Консоль перед закрытием программы:



8.3. Вложенные циклы for

Вычислить значения функции с использованием вложенных инструкций цикла for. Проект – консольное приложение.

Пример. Вычислить

$$t = \prod_{i=1}^{10} \sum_{j=1}^{10} \cos(ijx)$$

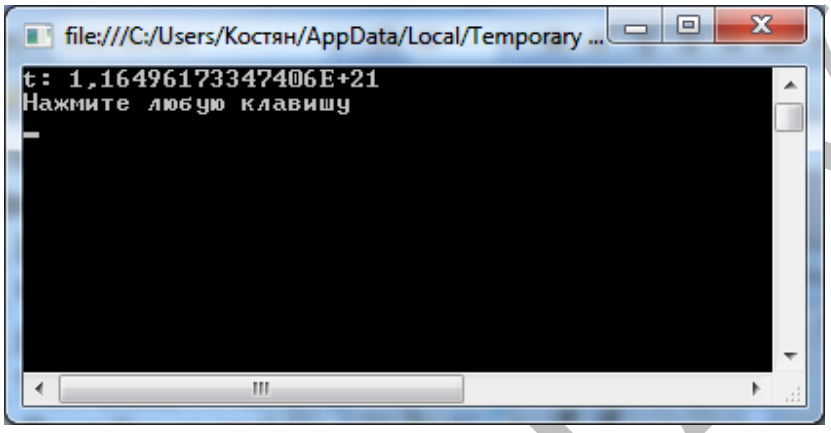
В алгоритме для разнообразия один цикл реализован инструкцией for нарастанием индекса, а другой - инструкцией for с убыванием индекса.

Листинг программы

```
using System;
namespace For2
{
   class Program
   {
      static void Main()
      {
         double t = 1, s = 0, x = 2;
         for (int i = 1; i < 11; i++)
```

```
        {
            for (int j = 10; j > 0; j--)
            {
                s += Math.Cos(i * j * x);
                t *= s;
            }
        }
        Console.WriteLine("t: {0 } ", t);
        Console.WriteLine("Нажмите любую клавишу");
        Console.ReadKey();
    }
  }
}
```

Консоль перед закрытием программы:



8.4. Команда break

Создать программу с использованием команды **break**. Проект – консольное приложение. В программе ищется первое не простое число из последовательности чисел от N до 1. Варианты заданий – N = две последние цифры номера зачетной книжки.

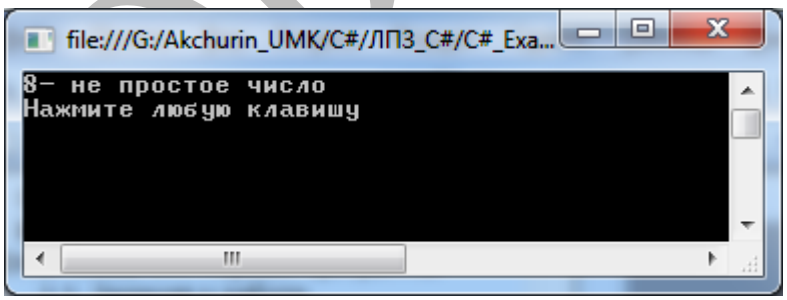
Пример. Создать программу с использованием команды **break**. Проект – консольное приложение. В программе ищется первое не простое число из последовательности чисел от i = 8 до 1. Число не простое, если при его делении на целые числа, меньшие его, получается нулевой остаток. Во внешнем цикле перебираются числа делимые от i = 8 до 1. Во внутреннем цикле перебираются делители от j = i -1 до 1. При обнаружении нулевого остатка сбрасывается флаг простого числа, по которму команда break прекращает итерации, так как ответ уже найден.

Листинг программы

```
using System;
namespace Break
{
   class Program
```

```
    {
        static void Main()
        {
            // объявляем флаг с именем bool для обозначения простых чисел
            bool IsPrimeNumber = true;
            for (int i = 8; i > 1; i--)
            {
                // устанавливаем флаг
                //IsPrimeNumber = true;
                for (int j = i - 1; j > 1; j--)
                {
                    // если существует делитель с нулевым остатком
                    // сбрасываем флаг
                    if (i % j == 0)
                    {
                        IsPrimeNumber = false;
                        // дальнейшая проверка бессмысленна
                        // если с нулевым остатком - то число простое
                        if (IsPrimeNumber == true)
                            Console.WriteLine("{0}— простое число", i);
                        else Console.WriteLine("{0}— не простое число", i);
                        Console.WriteLine("Нажмите любую клавишу");
                        Console.ReadKey();
                    }
                    if (IsPrimeNumber == false) break;
                }
            }
        }
    }
```

Консоль перед закрытием программы:



8.5. Команда continue

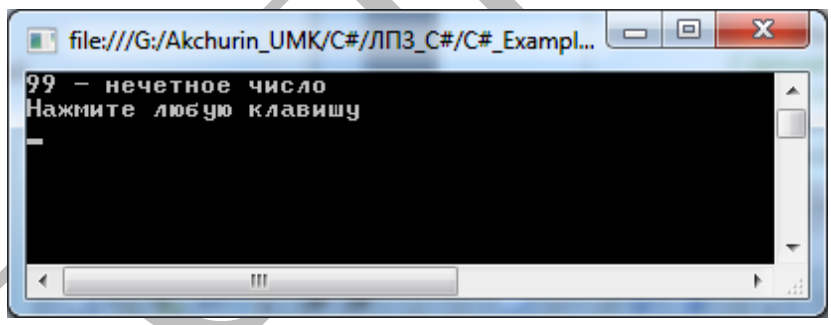
Создать программу с использованием команды **continue**. Проект – консольное приложение. В программе ищется первое нечетное число из последовательности чисел от N до 1. Варианты заданий – N = две последние цифры номера зачетной книжки.

Пример. В программе ищется первое нечетное число из последовательности чисел от N до 1. Признак нечетности – остаток от деления на 2 не равен нулю. Если он равен нулю, то команда **continue** прерывает текущую итерацию и переъодит к следующей.

Листинг программы

```
using System;
class PoiskNechet
{
   static void Main()
    {
        for ( int i = 100; i > 0; i--)
        {
            if ( i%2 ==0 )
            continue;
            Console.WriteLine("{0} - нечетное число", i);
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Консоль перед закрытием программы:



9. Логические операции

Предмет исследований

- Логические операции в C#.
- Разработать алгоритмы решения в соответствии с заданием.
- Составить программы решения задач.

Контрольные вопросы

1. Операция НЕ – Not.
2. Операция ИЛИ – OR.
3. Операция И – AND.
4. Операция исключающее ИЛИ – XOR.
5. Логические сдвиги.

Выполнить над операндами i1 и i2 операции НЕ – Not, ИЛИ – OR, И – AND,, исключающее ИЛИ – XOR. Выполнить над операндом i1 логические двиги влево и вправо на j разрядов. Проект – консольное приложение.

Варианты задания

- Число i1– первая с конца пара цифр в номере зачетной книжки.
- Число i2 – вторая с конца пара цифр в номере зачетной книжки.
- Число j – третья с конца пара цифр в номере зачетной книжки.

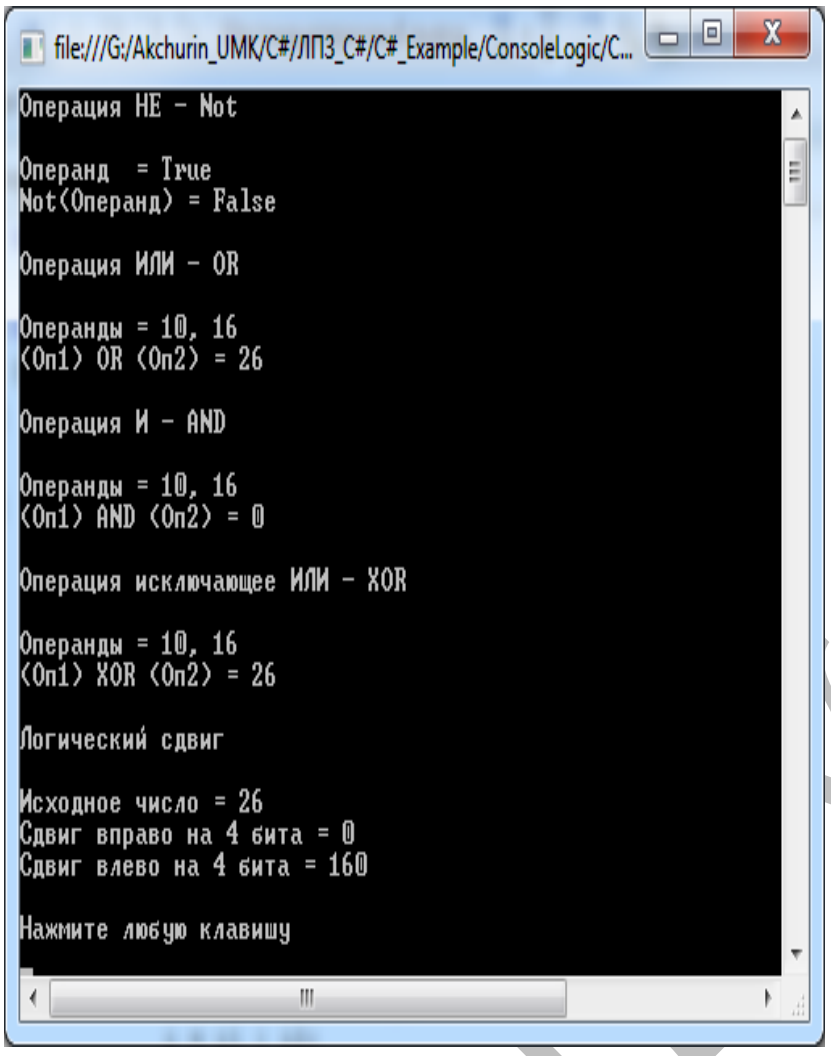
Пример. Выполнить над операндами i1 = 10 и i2 = 16 операции НЕ – Not, ИЛИ – OR, И – AND, исключающее ИЛИ – XOR. Выполнить над операндом i1 = 10 логическис двиги влево и вправо на j = 4 разряда. Проект – консольное приложение.

Листинг программы

```
using System;
namespace ConsoleLogic
{
   class Program
   {
     static void Main()
     {
        int i = -16, i1 = 10, i2 = 16;
        bool b = true;
        int j = 4;                  // Размер свига
        Console.WriteLine("Операция НЕ - Not");
        Console.WriteLine();
        Console.WriteLine("Операнд  = {0}",b);
        Console.WriteLine("Not(Операнд) = {0}", !b);
        Console.WriteLine();
        Console.WriteLine("Операция ИЛИ - OR");
        Console.WriteLine();
```

```
            i = i1 | i2;
            Console.WriteLine("Операнды = {0}, {1}",i1 ,i2);
            Console.WriteLine("(Оп1) OR (Оп2) = {0}", i);
            Console.WriteLine();
            Console.WriteLine("Операция И - AND");
            Console.WriteLine();
            i = i1 & i2;
            Console.WriteLine("Операнды = {0}, {1}", i1, i2);
            Console.WriteLine("(Оп1) AND (Оп2) = {0}", i );
            Console.WriteLine();
            Console.WriteLine("Операция исключающее ИЛИ - XOR");
            Console.WriteLine();
            i = i1 ^ i2;
            Console.WriteLine("Операнды = {0}, {1}", i1, i2);
            Console.WriteLine("(Оп1) XOR (Оп2) = {0}", i);
            Console.WriteLine();
            Console.WriteLine("Логический сдвиг");
            Console.WriteLine();
            Console.WriteLine("Исходное число = {0}", i);
            i = i1 >> j;
            Console.WriteLine("Сдвиг вправо на 4 бита = {0}", i);
            i = i1 << j;
            Console.WriteLine("Сдвиг влево на 4 бита = {0}", i);
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Консоль перед закрытием программы:


file:///G:/Akchurin_UMK/C#/ЛП3_C#/C#_Example/ConsoleLogic/C...
Операция НЕ - Not
Операнд = True
Not(Операнд) = False
Операция ИЛИ - OR
Операнды = 10, 16
(On1) OR (On2) = 26
Операция И - AND
Операнды = 10, 16
(On1) AND (On2) = 0
Операция исключающее ИЛИ - XOR
Операнды = 10, 16
(On1) XOR (On2) = 26
Логический сдвиг
Исходное число = 26
Сдвиг вправо на 4 бита = 0
Сдвиг влево на 4 бита = 160
Нажмите любую клавишу

10. Массивы

Предмет исследований

- Способы описания размеров массивов.
- Способы ввода и вывода массивов.
- Реализация приемов накопления суммы или произведения элементов массивов, запоминания результатов, нахождения наибольшего и наименьшего.

Контрольные вопросы

1. Что такое массив?
2. Описание типа - массив.
3. Какие операторы языка можно использовать для описания массивов?
4. Особенности организации цикла при обработке массивов?
5. Особенности программирования при обработке массивов?
6. Особенность ввода и вывода массивов?
7. Представление строковых переменных типа String, как одномерных массивов.
8. Вложенные массивы.

10.1. Одномерный массив

Обработка одномерного массива. Проект – консольное приложение

Варианты задания 1

№	Массив	Задание
1.	X[100]	Вычислить сумму и количество элементов массива X, удовлетворяющих условию 0 ≤ X[I] ≤ 1.
2.	A[80]	Вычислить среднее арифметическое значение элементов массива A, удовлетворяющих условию A[I [> 0.
3.	X[70]	Переписать элементы, удовлетворяющих условию -1 ≤ X[I] ≤ 1 из массива X в массив Y и подсчитать их количество.
4.	B[50]	Определить максимальный положительный элемент массива B и его порядковый номер.
5.	C[40]	Вычислить минимальный отрицательный элемент массива C и его номер.
6.	D[80]	Найти максимальный и минимальный элементы массива D и поменять их местами.
7.	Y[20]	Найти сумму элементов массива. Разделить каждый элемент исходного массива на полученное значение.
8.	A[4]	Найти сумму элементов массива. Разделить каждый элемент исходного массива на полученное значение.
9.	N[50]	Определить сумму элементов массива N, кратных трем. Условие кратности: INT(N(i)/3)*3=N(i).
10.	X[N]	Вычислить сумму и количество элементов массива X. Условие $X(i)\rangle 0, N \le 30$

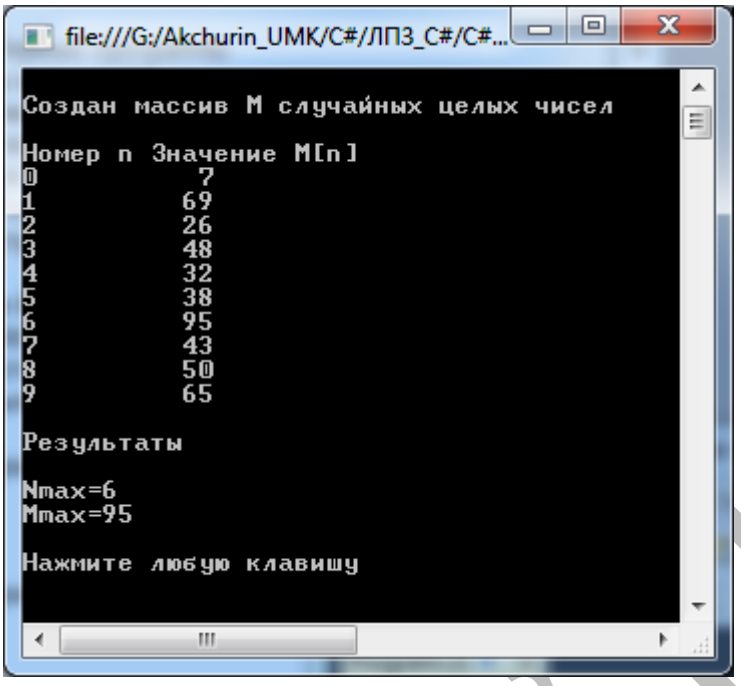
Пример. Вычислить наибольший элемент Xmax массива X и его порядковый номер Nmax. Размерность массива N=10. Результат – консольное приложение. В программе элементы массива создаются генератором случайных чисел.

Листинг программы

```
using System;
```

```
namespace Massiv
{
   class Program
   {
      static void Main()
      {
         int n = 0, N=10, Nmax=0, Mmax=0;       // Переменные типа int
         Console.WriteLine();
         Random r = new Random();           // Для случайных чисел
         int[] M = new int[N];              // Массив типа int
         Console.WriteLine("Создан массив M случайных целых чисел");
         Console.WriteLine();
         Console.Write("Номер n" + " Значение M[n]");
         Console.WriteLine();
         for (n = 0; n < N; n++)
         {
            int Mm = r.Next(0, 100);        // Генерация случайного числа
            M[n] = Mm;
            if (Mm > Mmax)                  // Обнаружение максимума
            {
               Mmax = Mm;
               Nmax = n;
            }
            Console.WriteLine("{0} {1,10}",n, Mm);
         }
         Console.WriteLine();
         Console.WriteLine("Результаты");
         Console.WriteLine();
         Console.Write("Nmax=");            // Вывод Nmax
         Console.WriteLine(Nmax.ToString());
         Console.Write("Mmax=");            // Вывод Mmax
         Console.WriteLine(Mmax.ToString());
         Console.WriteLine();
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();                 // Пауза
      }
   }
}
```

Консоль перед закрытием программы:



10.2. Матрица

Обработка матрицы. Проект – консольное приложение

Варианты задания 2

№	Задание
1.	Вычислить сумму элементов побочной диагонали матрицы.
2.	Найти сумму элементов второй строки матрицы.
3.	Найти сумму элементов второго столбца матрицы.
4.	Вывести на печать элементы матрицы 2≤ A[i, j] ≤5 .
5.	Вывести на печать отрицательные элементы матрицы.
6.	Вывести на печать положительные элементы матрицы.
7.	Вычислить сумму элементов матрицы.
8.	Найти сумму элементов побочной диагонали матрицы.
9.	Вычислить сумму элементов двух первых столбцов матрицы.
10.	Вычислить среднее арифметическое матрицы.

Пример. Вывести квадратную матрицу X размером RxC в виде таблицы. Найти максимальное значание элемента и его ккординаты: номер строки R и номер столбца C.

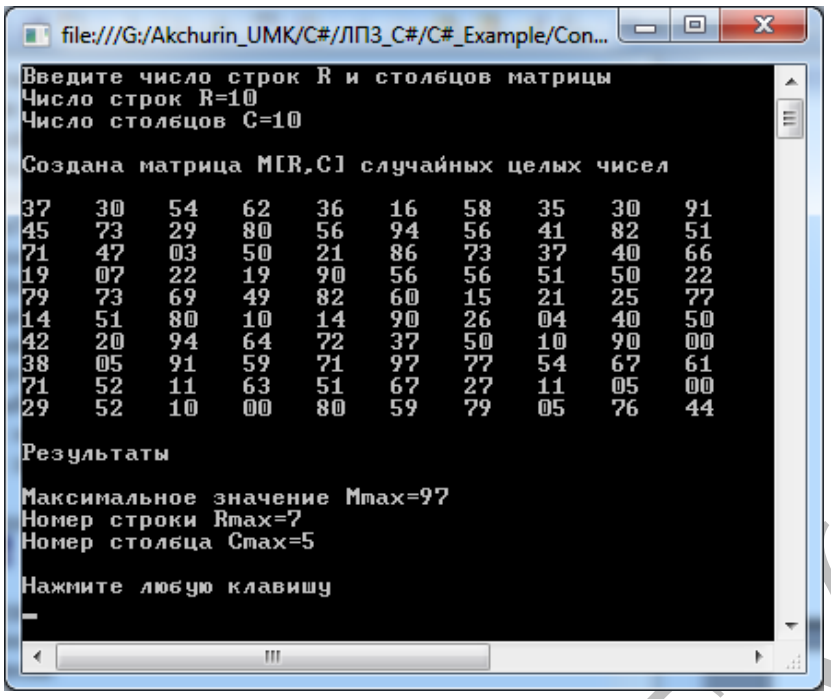
Листинг программы

```
using System;
namespace Matrix
{
   class Program
   {
      static void Main()
      {
         int r=0,R=0,Rmax=0,c=0,C=0,Cmax=0,Mmax=0;   // Переменные типа int
         Console.WriteLine("Введите число строк R и столбцов матрицы");
         Console.Write("Число строк R=");           // Введите R
         R = Convert.ToInt32(Console.ReadLine());
```

```
        Console.Write("Число столбцов C=");        // Введите C
        C = Convert.ToInt32(Console.ReadLine());
        Console.WriteLine();
        Random rnd = new Random();          // Для случайных чисел
        int[,] M = new int[R, C];           // Матрица M[R,C] типа int
        for (r = 0; r < R; r++)
        {
           for (c = 0; c < C; c++)
           {
              int Mm = rnd.Next(0, 100);  // Генерация случайного числа
              M[r,c] = Mm;
              if (Mm > Mmax)              // Обнаружение максимума
              {
                 Mmax = Mm;               // Значение
                 Rmax = r;               // Номер строки
                 Cmax = c;               // Номер столбца
              }
           }
        }
        // Вывод матрицы
        Console.WriteLine("Создана матрица M[R,C] случайных целых чисел");
        Console.WriteLine();
        for (r = 0; r < R; r++)
        {
           for (c = 0; c < C; c++)
           {
              Console.Write(M[r, c].ToString("D2")+"   ");
           }
           Console.WriteLine();
        }
        Console.WriteLine();
        Console.WriteLine("Результаты");
        Console.WriteLine();
        Console.Write("Максимальное значение Mmax=");   // Вывод Mmax
        Console.WriteLine(Mmax.ToString());
        Console.Write("Номер строки Rmax=");           // Вывод Rmax
        Console.WriteLine(Rmax.ToString());
        Console.Write("Номер столбца Cmax=");          // Вывод Cmax
        Console.WriteLine(Cmax.ToString());
        Console.WriteLine();
        Console.WriteLine("Нажмите любую клавишу");
        Console.ReadKey();                          // Пауза
    }
}
```

}

Консоль перед закрытием программы:



По нему нужно проверить правильность исполнения алгоритма.

11. Файлы

Предмет исследований

- Типы файлов, определенные в C#.
- Связь с дисковыми файлами.
- Операции с типизированными файлами.
- Операции с текстовыми файлами.
- Операции с нетипизированными файлами.

Контрольные вопросы

1. Типы файлов, определенные в C#..
2. Способы связи с файлами.
3. Директива using System.IO
4. Тип FileStream для связи с файлами.
5. Метод FileMode.Create.
6. Метод FileMode.Append.
7. Тип StreamWrite для связи с типом FileStream.
8. Файловая переменная.
9. Как связать файловую переменную с дисковым файлом?
10. Процедуры создания, открытия, закрытия файлов.

Задание. Создать программу создания и дополнения текстового файла. Сначала создать текстовый файл из заданного числа строк и посмотреть его содержимое. Затем добавить в этот файл строки и проверить содержимое итогового файла.

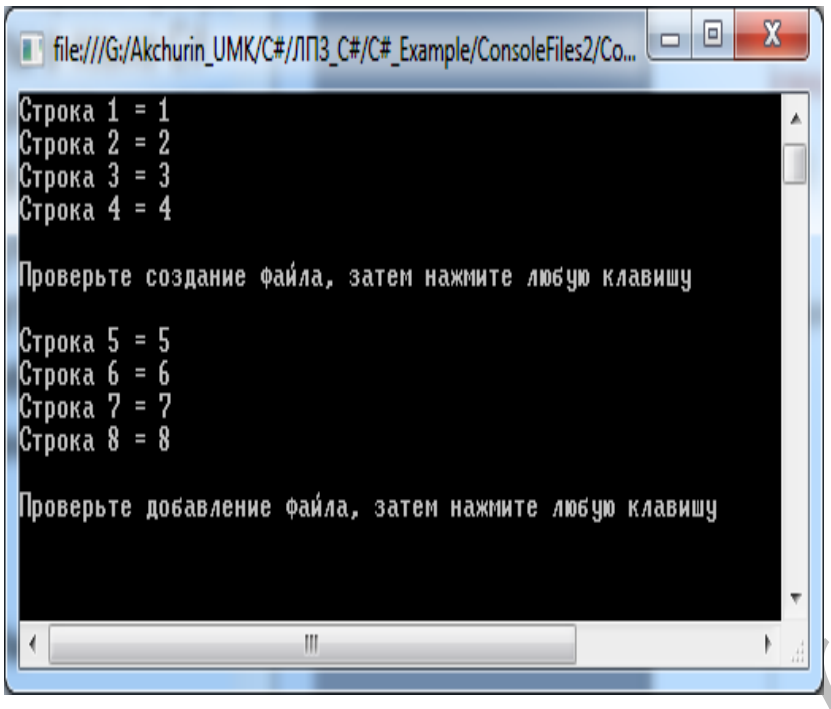
Пример.

Листинг программы

```
using System;
using System.IO;
namespace Files
{
   class Program
   {
      static void Main()
      {
         string ПутьКФайлу = "e:\\";
         string ИмяФайла = "data.txt";
         string ПолноеИмяФайла = ПутьКФайлу + ИмяФайла;
         FileStream f = new FileStream(ПолноеИмяФайла, FileMode.Create);
         StreamWriter str = new StreamWriter(f);
         int ЧислоСтрокБлока = 4;
         // Запись в файл блока строк
         for (int i = 1; i <= ЧислоСтрокБлока; i++)
         {
```

```
            Console.Write("Строка {0} = ", i);
            string s = Console.ReadLine();
            str.WriteLine(s);
          }
          str.Close();
          f.Close();
          Console.WriteLine();
          Console.WriteLine("Проверьте создание файла, затем нажмите
        любую клавишу");
          Console.ReadKey();
          Console.WriteLine();
          FileStream f1 = new FileStream(ПолноеИмяФайла, FileMode.Append);
          StreamWriter str1 = new StreamWriter(f1);
          // Добавление в файл блока строк
          for (int i = 1; i <= ЧислоСтрокБлока; i++)
          {
            Console.Write("Строка {0} = ", i + ЧислоСтрокБлока);
            string s = Console.ReadLine();
            str1.WriteLine(s);
          }
          str1.Close();
          f1.Close();
          Console.WriteLine();
          Console.WriteLine("Проверьте добавление файла, затем нажмите
            любую клавишу");
          Console.ReadKey();
        }
    }
}
```

Консоль перед закрытием программы:


file:///G:/Akchurin_UMK/C#/ЛП3_C#/C#_Example/ConsoleFiles2/Co...
Строка 1 = 1
Строка 2 = 2
Строка 3 = 3
Строка 4 = 4
Проверьте создание файла, затем нажмите любую клавишу
Строка 5 = 5
Строка 6 = 6
Строка 7 = 7
Строка 8 = 8
Проверьте добавление файла, затем нажмите любую клавишу

12. Подпрограммы

Предмет исследований

- Правила объявления подпрограмм.
- Связь формальных и фактических параметров.
- Способы передачи фактических параметров в подпрограмму.
- Обращение к функциям.

Контрольные вопросы

1. Что такое подпрограмма? Ее назначение.
2. Правила объявления функций в программе.
3. Правила обращения к функции в программе.
4. Формальные и фактические параметры.

Задание. Создать программу, использующую 2 подпрограммы-функции:

- Первая функция должна возвращать наибольший общий делитель NOD двух целых чисел.
- Вторая функция должна возвращать наименьшее общее делимое NOK двух целых чисел.

Обе функции в главной программе доложны использоваться с тремя разными целыми числами. Проект – консольное приложение.

Варианты задания

- Число 1 – первая с конца пара цифр в номере зачетной книжки.
- Число 2 – вторая с конца пара цифр в номере зачетной книжки.
- Число 3 – третья с конца пара цифр в номере зачетной книжки.

Пример. Создать программу, использующую 2 подпрограммы-функции:

- Первая функция должна возвращать наибольший общий делитель NOD двух целых чисел.
- Вторая функция должна возвращать наименьшее общее делимое NOK двух целых чисел.

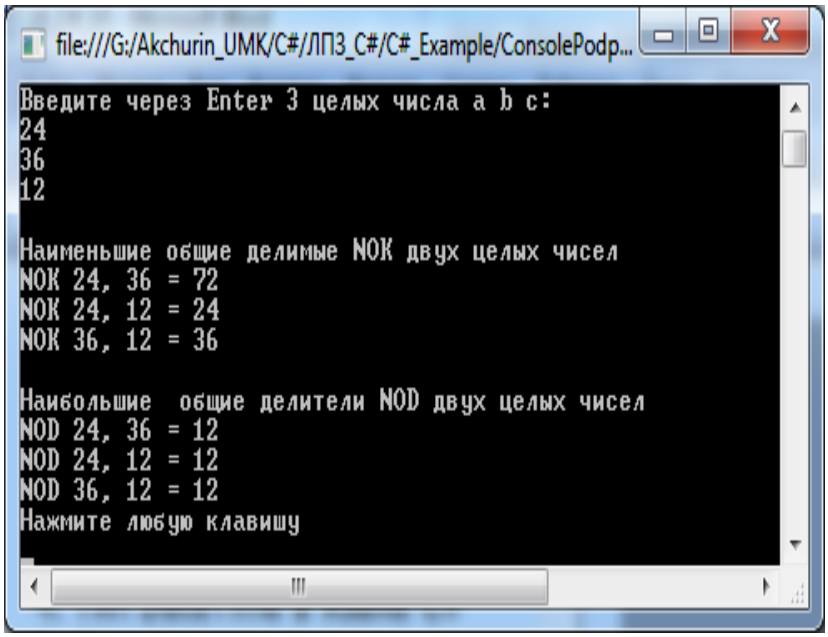
Обе функции в главной программе доложны использоваться с тремя разными целыми числами, вводимыми с клавиатуры. Проект – консольное приложение.

Листинг программы

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

class PodProgram
{
   // Функция NOD определения наибольшего общего делителя
   static int NOD(int x, int y)        // Функция NOD
```

```
    {
        if (x != 0)
            return NOD((y % x), x);     // Рекурсивный вызов
        else return y;
    }
    // Функция NOK определения наименьшего общего делимого
    static int NOK(int x, int y)        // Функция NOK
    {
        return (x / NOD(x, y)) * y;
    }
    // Главная программа
    static void Main()
    {
        int a, b, c;
        Console.WriteLine("Введите через Enter 3 целых числа a b c:");
        a = Int32.Parse(Console.ReadLine());
        b = Int32.Parse(Console.ReadLine());
        c = Int32.Parse(Console.ReadLine());
        Console.WriteLine();
        Console.WriteLine("Наименьшие общие делимые NOK двух целых чисел");
        Console.WriteLine("NOK {0}, {1} = {2}", a, b, NOK(a, b));
        Console.WriteLine("NOK {0}, {1} = {2}", a, c, NOK(a, c));
        Console.WriteLine("NOK {0}, {1} = {2}", b, c, NOK(b, c));
        Console.WriteLine();
        Console.WriteLine("Наибольшие  общие делители NOD двух целых
чисел");
        Console.WriteLine("NOD {0}, {1} = {2}", a, b, NOD(a, b));
        Console.WriteLine("NOD {0}, {1} = {2}", a, c, NOD(a, c));
        Console.WriteLine("NOD {0}, {1} = {2}", b, c, NOD(b, c));
        Console.WriteLine("Нажмите любую клавишу");
        Console.ReadKey();
    }
}
```

Консоль перед закрытием программы:


file:///G:/Akchurin_UMK/C#/ЛП3_C#/C#_Example/ConsolePodp...
Введите через Enter 3 целых числа a b c:
24
36
12
Наименьшие общие делимые NOK двух целых чисел
NOK 24, 36 = 72
NOK 24, 12 = 24
NOK 36, 12 = 36
Наибольшие общие делители NOD двух целых чисел
NOD 24, 36 = 12
NOD 24, 12 = 12
NOD 36, 12 = 12
Нажмите любую клавишу

13. Операции со строками

Предмет исследований

- Строки и их обработка.
- Методы работы состроками..
- Разработать алгоритмы решения задач.
- Составить программы решения задач.

Контрольные вопросы

1. Класс String.
2. Объявление строковой переменной
3. Сравнение строк.
4. Объдинение строк
5. Разбиение строк

Варианты заданий

Варианты задания используют строки, набранные из Фамилии, Имени и Отчества студента.

13.1. Сравнение строк

Создать программу сравнения строк. Проект – консольное приложение.

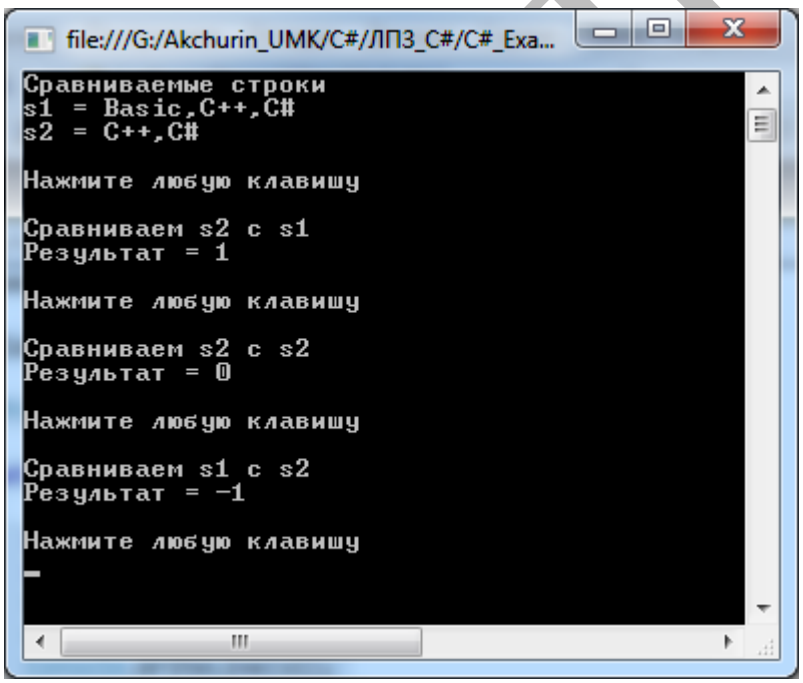
Пример.

Листинг программы

```
using System;
namespace StrokSravnenie
{
   class Program
   {
      static void Main()
      {
         string s1 = "Basic,C++,C#";
         string s2 = "C++,C#";
         Console.WriteLine("Сравниваемые строки");
         Console.Write("s1 = ");
         Console.WriteLine(s1);
         Console.Write("s2 = ");
         Console.WriteLine(s2);
         Console.WriteLine();
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();
         Console.WriteLine();
         Console.WriteLine("Сравниваем s2 с s1");
         int r = s2.CompareTo(s1);
         Console.WriteLine("Результат = " + r.ToString());
```

```
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
            Console.WriteLine();
            Console.WriteLine("Сравниваем s2 с s2");
            r = s2.CompareTo(s2);
            Console.WriteLine("Результат = " + r.ToString());
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
            Console.WriteLine();
            Console.WriteLine("Сравниваем s1 с s2");
            r = s1.CompareTo(s2);
            Console.WriteLine("Результат = " + r.ToString());
            Console.WriteLine();
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Консоль перед закрытием программы:



13.2. Объединение строк

Создать программу объединения строк. Проект – консольное приложение.

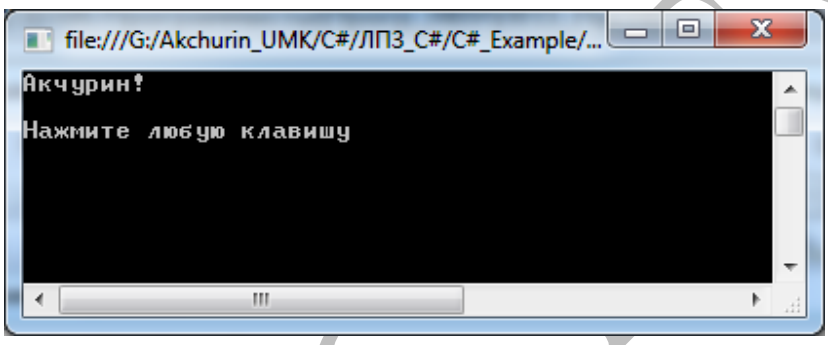
Пример.

Листинг программы

```
using System;
```

```
namespace StrokObyedinenie
{
   class Program
   {
      static void Main()
      {
         string Фамилия;
         char Символ;
         Фамилия = "Акчурин";              // тип string
         Символ = '!';                     // тип char
         Console.WriteLine(Фамилия + Символ);
         Console.WriteLine();              // Пропуск строки
         Console.WriteLine("Нажмите любую клавишу");
         Console.ReadKey();                // Пауза
      }
   }
}
```

Консоль перед закрытием программы:



13.3. Разбиение строк

Создать программу разбиения строк. Проект – консольное приложение.

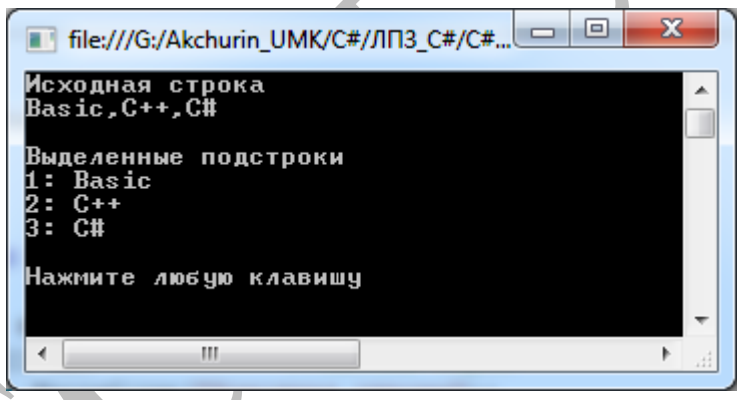
Пример.

Листинг программы

```
using System;
namespace StrokRazbienie
{
   class Program
   {
      static void Main()
      {
         string s1 = "Basic,C++,C#";
         string[] s2;          //Массив строк
         Console.WriteLine("Исходная строка");
         Console.WriteLine(s1);
         Console.WriteLine();
```

```
            s2 = s1.Split(',');     // символ раэделения в массиве
            // Вывод подстрок
            string output = "";
            int ctr = 1;
            foreach (string substring in s2)
            {
               output += ctr++;
               output += ": ";
               output += substring;
               output += "\n";
            }
            Console.WriteLine("Выделенные подстроки");
            Console.WriteLine(output);
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
        }
    }
}
```

Консоль перед закрытием программы:



14. Исключения

Предмет исследований

- Класс исключений и их обработка.
- Инструкции try, catch, finally.
- Разработать алгоритмы решения задач.
- Составить программы решения задач.

Контрольные вопросы

1. Что такое исключение?
2. Классы исключений и их свойства.
3. Инструкция try,catcht. Назначение, структура и применение.
4. Инструкция try, catch,finally. Назначение, структура и применение.
5. Исключение EZeroDivide для целых чисел.
6. Исключение EZeroDivide для вещественных чисел.
7. Исключение ERangeError.
8. Как включить отработку исключения ERangeError.
9. Как отключить перехват исключения системой.

14.1. Операции с вещественными числами

Задание. Создать программу, обрабатывающую исключения при работе с вещественными числами с использованием инструкции try… catch…finally. Проект – консольное приложение.

В задании надо рассчитывать массив f [x] = 1 / (x – N) при x =0…k.

Варианты заданий. N – последняя цифра номера зачетной книжки плюс 5.

Необходимо отслеживать два исключения:

- Деление на 0.
- Выход за пределы диапазона.

При работе с программой менять k, чтобы получать разные условия возможного возникновения исключений.

Пример. Вариант задания: f[x] = 1 / (x-5). N=10/

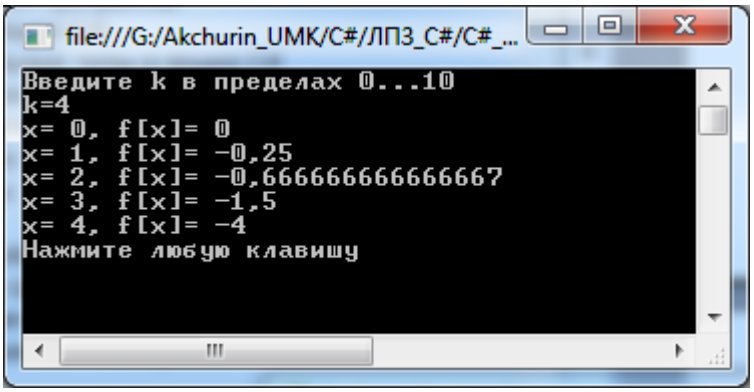
Листинг программы

```
using System;
namespace TryCatchDouble
{
   class Program
   {
      static void Main()
      {
         int k=0;
         Console.WriteLine("Введите k в пределах 0...10");
         Console.Write("  k=");       // Приглашение ввода
```

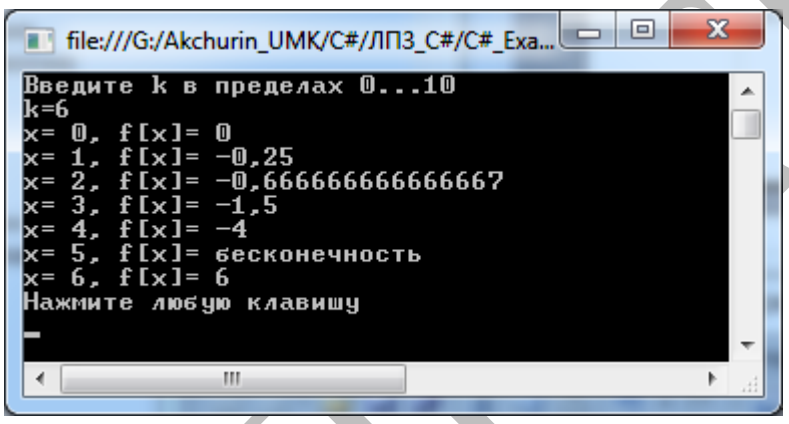
```
            string v = Console.ReadLine();  // Ввод строки
            k = Convert.ToInt32(v);
            double[] f = new double[10];     // Определен массив размером 10
            try                              // Инициализация исключения
            {
               for (int x = 0; xi < k + 1;x ++)
               {
                  f[x] = (double) 1 / (x - 5); // Вычисление элемента массива
                  Console.WriteLine("x= {0}, f[x]= {1}", x, f[x]);
               }
            }
            // Перехватить попытку деления на 0
            catch (DivideByZeroException e)
            {
               Console.WriteLine("Попытка деления на 0");
               Console.WriteLine(e.ToString());
            }
            // Перехватить выход за пределы диапазона
            catch (IndexOutOfRangeException e)
            {
               Console.WriteLine("Выход за пределы диапазона");
               Console.WriteLine(e.ToString());
            }
            // Перехватить другие исключения
            catch (Exception e)
            {
               Console.WriteLine("Фатальная ошибка");
               Console.WriteLine(e.ToString());
            }finally
            {
               Console.WriteLine("Нажмите любую клавишу");
               Console.ReadKey();
            }
         }
      }
   }
```

Консоль перед закрытием программы (3 варианта):

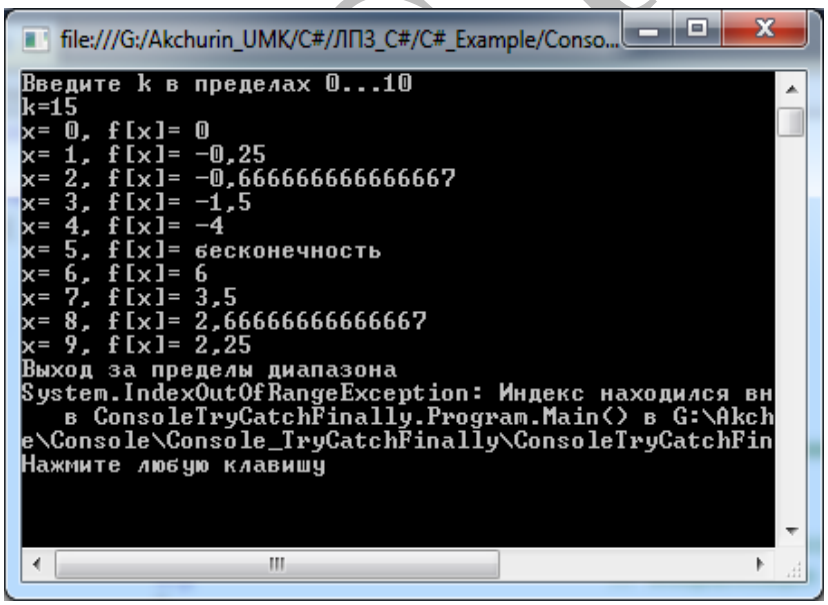
- Исключений нет. Задаем k = 4.



- Ошибка деления на 0. Задаем k = 6. Ожидается деление на 0 при x = 5. В языке C# эта ошибка нефатальна. Результат деления на 0 интерпретируется как бесконечность.



- Ошибка диапазона. Задаем k = 15. Ожидается выход за пределы диапазона массива при x = 10. Эта ошибка фатальна. Она обрабатывается в секции catch.



14.2. Операции с целыми числами

Задание. Создать программу, обрабатывающую исключения при работе с целыми числами с использованием инструкции try… catch…finally. Проект – консольное приложение.

В задании надо рассчитывать массив f [x] = 1 / (x – N) при x =0…k.

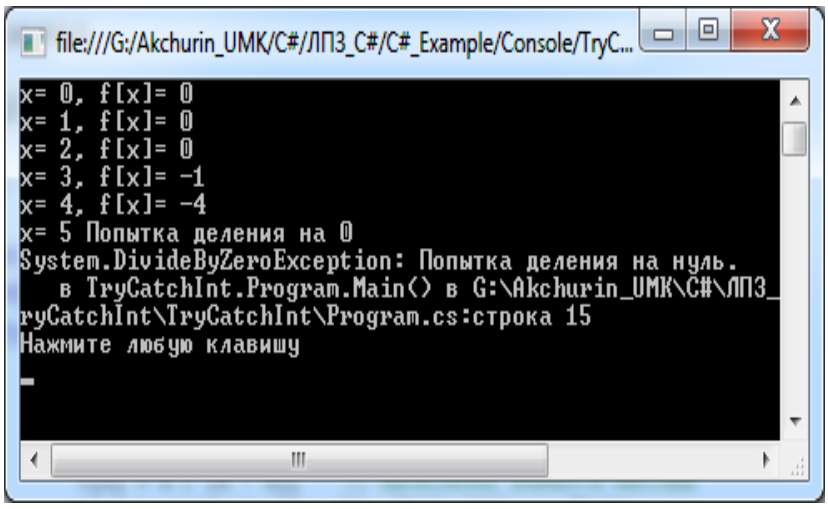
Варианты заданий. N – последняя цифра номера зачетной книжки плюс 5.

Необходимо отслеживать исключение деление на 0.

Пример. N=10, k = 5. Ожидается ошибка деления на ноль.

Листинг программы

```
using System;
namespace TryCatchInt
{
   class Program
   {
      static void Main()
      {
         int k = 5;
         int[] f = new int[10];         // Определен массив размером 10
         try                            // Инициализация исключения
         {
            for (int x = 0; x < k + 1; x++)
            {
               f[x] = x / (x - 5);    // Вычисление элемента массива
               Console.WriteLine("x= {0}, f[x]= {1}", x, f[x]);
            }
         }
         // Перехватить попытку деления на 0
         catch (DivideByZeroException e)
         {
            Console.WriteLine("x= {0} Попытка деления на 0", k);
            Console.WriteLine(e.ToString());
         }

         finally
         {
            Console.WriteLine("Нажмите любую клавишу");
            Console.ReadKey();
         }
      }
   }
}
```


file:///G:/Akchurin_UMK/C#/ЛП3_C#/C#_Example/Console/TryC...
x= 0, f[x]= 0
x= 1, f[x]= 0
x= 2, f[x]= 0
x= 3, f[x]= -1
x= 4, f[x]= -4
x= 5 Попытка деления на 0
System.DivideByZeroException: Попытка деления на нуль.
в TryCatchInt.Program.Main() в G:\Akchurin_UMK\C#\ЛП3_
ryCatchInt\TryCatchInt\Program.cs:строка 15
Нажмите любую клавишу

15. Работы с классами и объектами

Предмет исследований

- Исследование объектов.
- Работа с объектами
- Использование своих объектов в программах.
- Объекты среды .NET Framework

Контрольные вопросы

1. Классы в программах
2. Объекты в программах.
3. Создание пользовательских классов в C#.
4. Использование встроенных классов среды .NET Framework.
5. Работа с созданными классами.
6. Преимущества ОПП.

Задание. Напишите программу, в которой вы создаете собственный класс и используете его объекты в процессе работы программы.

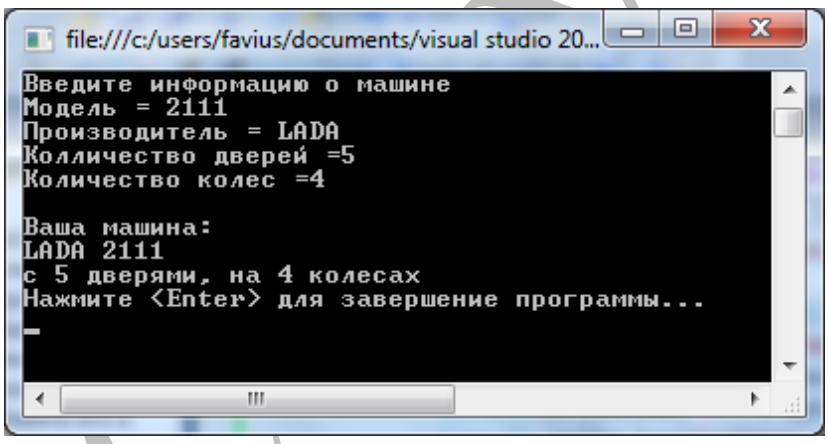
Пример. В пример представлена простая программ в которой создается класс описывающий автомобиль объекты этого класса являются параметры автомобиля. Происходит ввод параметров и вывод результата.

Листинг программы

```
using System;

namespace LabOb
{
   public class Vehicle
   {
      public string model;
      public string manufacturer;
      public int numOfDoors;
      public int numOfWheels;
   }
   class Program
   {
      static void Main()
      {
         Console.WriteLine("Введите информацию о машине");
         Vehicle myCar = new Vehicle();
         Console.Write("Модель = ");
         string s = Console.ReadLine();
         myCar.model = s;
         Console.Write("Производитель = ");
         myCar.manufacturer = Console.ReadLine();
```

```
            Console.Write("Колличество дверей =");
            s = Console.ReadLine();
            myCar.numOfDoors = Convert.ToInt32(s);
            Console.Write("Количество колес =");
            s = Console.ReadLine();
            myCar.numOfWheels = Convert.ToInt32(s);
            Console.WriteLine("\nВаша машина: ");
            Console.WriteLine(myCar.manufacturer + " " + myCar.model);
            Console.WriteLine("с " + myCar.numOfDoors + " дверями, " + "на " +
myCar.numOfWheels + " колесах");
            Console.WriteLine("Нажмите <Enter> для " + "завершение програм-
мы...");
            Console.Read();
        }
    }
}
```

Консоль перед завершением



16. Использование командной строки при запуске программ

Предмет исследований

- Задание параметров выполнения программы при запуске в командной строке.
- Задание значения переменных через параметры командной строки при запуске программы.
- Использование оператора Switch.
- Встраивание в консольную программу справки.

Контрольные вопросы

1. Использование параметров командной строки в программах.
2. Реализация параметров запуска в программах.
3. Использование Switch.
4. Встраивание справки в консольные программы.
5. Чем удобно использования функций
6. Работа с if.. else
7. Использование объекта Length.
8. Работа со "string[] args".

Задание. Напишите программу, в которой обеспечивается вызов справки, задание значения переменной через параметр в командной строке и выполнениие вычисления факториала числа, равного номеру варианта.

Пример. В примере показана программа, которая выполняет одно из двух действий в зависимости от заданных параметров командной строки: вывод справки, либо вычисление факториала заданного в параметрах значения.

$$n! = 1 \cdot 2 \cdot \ldots \cdot n = \prod_{i=1}^{n} i$$

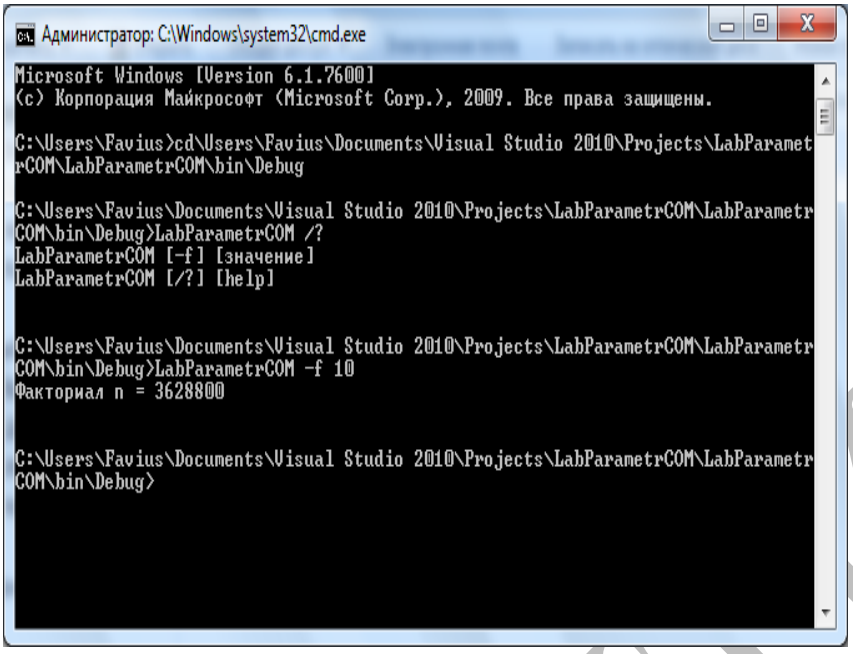
Если запустить программу без параметров, то она не выполняет никаких действий. Для вызова справки используется параметр "LabParametrCOM /?". При использовании параметра "LabParametrCOM -f 10" происходит вычисление факториала указанного значения.

Листинг программы

```
using System;

namespace LabParametrCOM
{
   class Program
   {
      static void Main(string[] args)
      {
         if (args.Length != 0)
         {
```

```
            switch (args[0])
            {
                case "-f":
                    pid(args[1]);
                    break;
                case "/?":
                    help();
                    break;
                default:
                    break;
            }
        }
    }
    static void help()
    {
        Console.WriteLine("LabParametrCOM [-f] [значение]");
        Console.WriteLine("LabParametrCOM [/?] [help]");
        Console.ReadLine();
    }
    static void pid(string b)
    {
        double n = 1;
        double d = 0;
        for (double i = 0; i < Convert.ToDouble(b); i++)
        {
            d++;
            n = n * d;
        }
        Console.WriteLine("Факториал n = {0}", n);

        Console.ReadLine();
    }
  }
}
```

Консоль перед закрытием.


Администратор: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
(c) Корпорация Майкрософт (Microsoft Corp.), 2009. Все права защищены.
C:\Users\Favius>cd\Users\Favius\Documents\Visual Studio 2010\Projects\LabParametrCOM\LabParametrCOM\bin\Debug
C:\Users\Favius\Documents\Visual Studio 2010\Projects\LabParametrCOM\LabParametrCOM\bin\Debug>LabParametrCOM /?
LabParametrCOM [-f] [значение]
LabParametrCOM [/?] [help]
C:\Users\Favius\Documents\Visual Studio 2010\Projects\LabParametrCOM\LabParametrCOM\bin\Debug>LabParametrCOM -f 10
Факториал n = 3628800
C:\Users\Favius\Documents\Visual Studio 2010\Projects\LabParametrCOM\LabParametrCOM\bin\Debug>

17. Отладка и тестирование в C#

Предмет исследований

- Отладка программ.
- Способы отладки.
- Тестирование созданных программ.

Контрольные вопросы

1. Возможности по отладке в Visual C# 2010.
2. Использование точек останова при отладке программ.
3. Проверка значения отдельных переменных в процесс отладки.
4. Настройка параметров отладки в Visual C# 2010.
5. Одновременный просмотр значения нескольких переменных в процесс отладки.

Задание. Напишите и протестируйте программу.

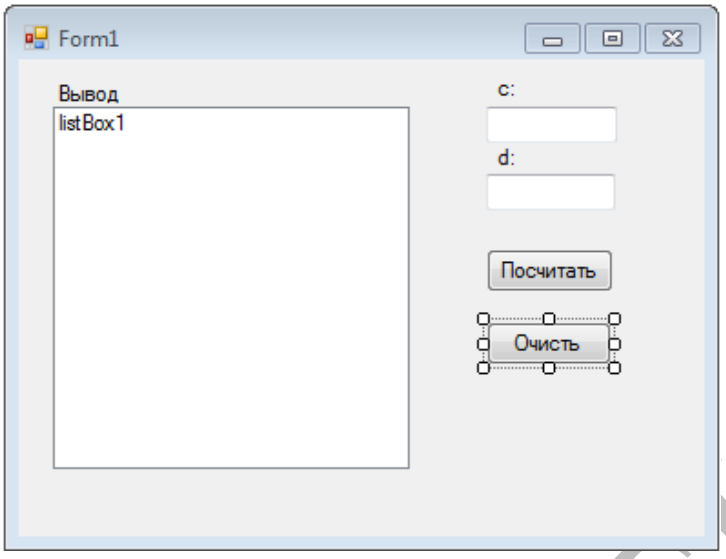
Пример. В примере представлена программа, в которой по паре вводимых чисел c, d в цикле для i=1...10 вычисляятся значения пары других параметров a=(c+d)*i, b=(c-d)*i. Функция Ample вычисляет квадратный корень из суммы квадратов целых частей от a, b, и эти значения суммируются в переменной s. По завершении цикла вычисляется среднее арифметическое sm от выходных параметров функции Ample. Функция Ample оформлена в виде подпрограммы. В программе определены метки, задающие точки останова.

Протестировать учебную программу с условиями:

- С помощью отладочной печати контролировать массив s[i].
- Переменные для окна «Список наблюдения» - a,b,s,i.
- Точки останова - m1,m2,m4.

Варианты		
№	Переменные для окна «Список наблюдения»	Строки останова
1.	a, b, f	m1,m2,m3
2.	c, d, s	m4,m5,m6
3.	a, c, s	m2,m3,m4
4.	b, d, f	m1,m5,m6
5.	a, d, c	m3,m4,m5
6.	b, a, s	m1,m2,m6
7.	c, d, f	m4,m5,m6
8.	d, c, a	m1,m2,m3
9.	f, d, b	m5,m6,m1
10.	s, f, c	m4,m2,m3
11.	a, s, f	m6,m1,m2
12.	b, a, s	m3,m4,m5
13.	c, d, f	m1,m3,m5
14.	s, d, c	m4,m5,m3
15.	a, c, f	m2,m1,m6

В программе использован графичесчкий интерфейс. Форма окна



Листинг программы

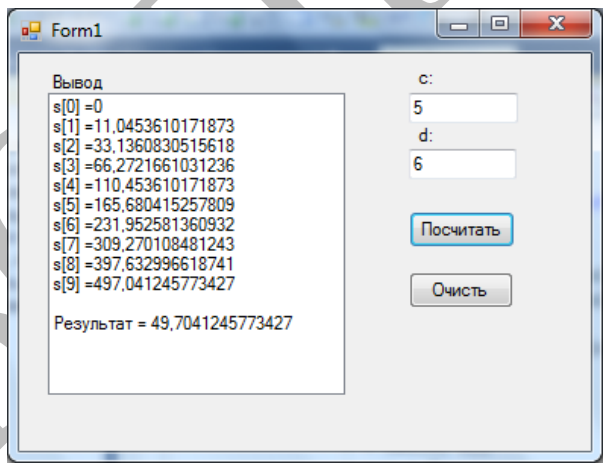
```
using System;
using System.Windows.Forms;

namespace LabDeb
{
   public partial class Form1 : Form
   {
      double a, b, c, d, s, f, sm;
      public Form1()
      {
         InitializeComponent();
      }
      public static double Aml(double x, double y)
      {
         double x1, y1;
         x1 = Math.Truncate(x);
         y1 = Math.Truncate(y);
         double result = Math.Sqrt(x1 * x1 + y1 * y1);
         return result;
      }
      private void button1_Click(object sender, EventArgs e)
      {
         string z;
         c = Convert.ToDouble(textBox1.Text);
         d = Convert.ToDouble(textBox2.Text);
         s = 0;
         for (double i = 0; i < 10; i++)
         {
            a = (c + d) * i;
            b = (c - d) * i;
```

```
            f = Aml(a, b);
            s = s + f;
            z = "s[" + Convert.ToString(i) + "] =" + Convert.ToString(s);
            listBox1.Items.Add(z);
        }
        sm = s / 10;
        listBox1.Items.Add(" ");
        z = "Результат = " + sm;
        listBox1.Items.Add(z);
    }
    private void button2_Click(object sender, EventArgs e)
    {
        listBox1.Items.Clear();
    }
}
```

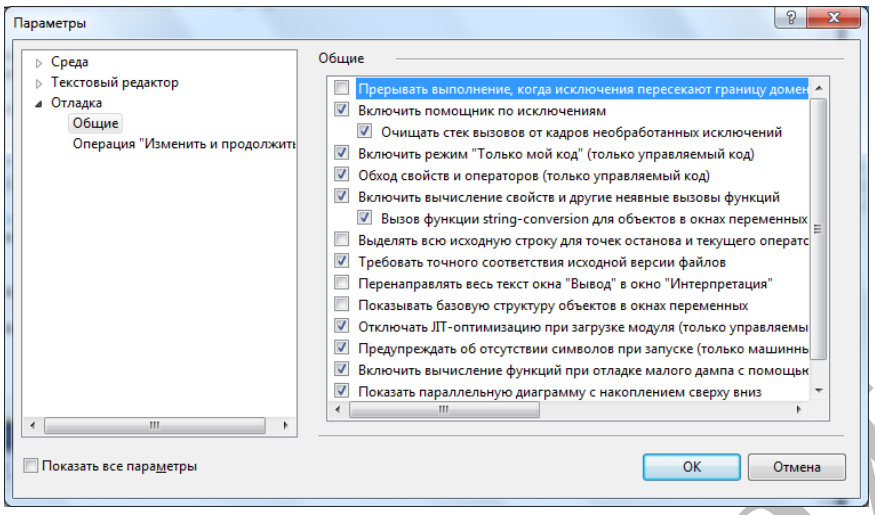
Выполнение

1. Создаем приложение Windows Form.
2. Размещаем в нем код учебной программы.

3. Проверка значений s[i] с помощью отладочной печати. Отладочная печать - вывод значений переменных в окно приложения осуществляется при прогоне программы. Ниже показано окно приложения при прогоне

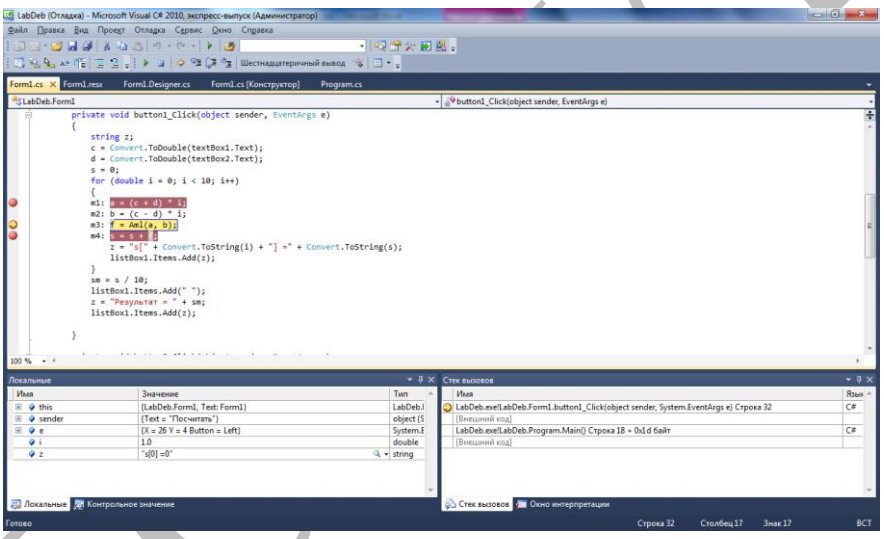


Настройка параметров отладки производится в меню "Отладка => Параметры и настройки.



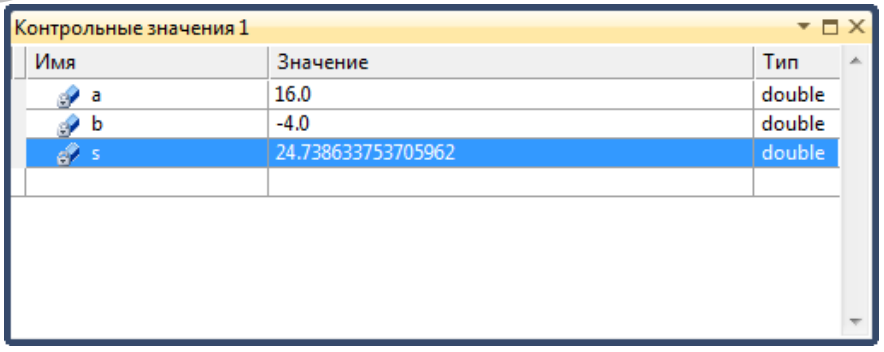
4. Создание точек останова в Visual C# производится следующим образом: Выбираем нужную строку за тем "Отладка => Точка останова" или нажать F9

Ниже показан просмотр результата в точке останова при запущенной программе.



5. Проверяем значения переменных при остановке программы в точках останова наведя на переменны курсор мыши .

6. Для проверки значения нескольких переменных в процессе выполнения программы выделяем нужную переменную щелкаем правой кнопкой мыши и "Добавить контрольное значение" после этого возможен просмотр значения переменных в момент остановки программ в точках останова.



18. Многопоточные приложения

Предмет исследований

- Преимущества многопоточности.
- Реализация многопоточности.
- Применение в программах.

Контрольные вопросы

1. Типы многопоточности в программах.
2. Создание дополнительных потоков в программах.
3. Обмен данными между потоками.
4. Применение делегатов при создание многопоточных приложений в C#/
5. Преимущества многопоточных приложений на современных ПК.
6. Использование метода "Thread.Sleep".

Задание. Напишите программу реализующую создание второго потока при ее запуске и обмен данными между потоками.

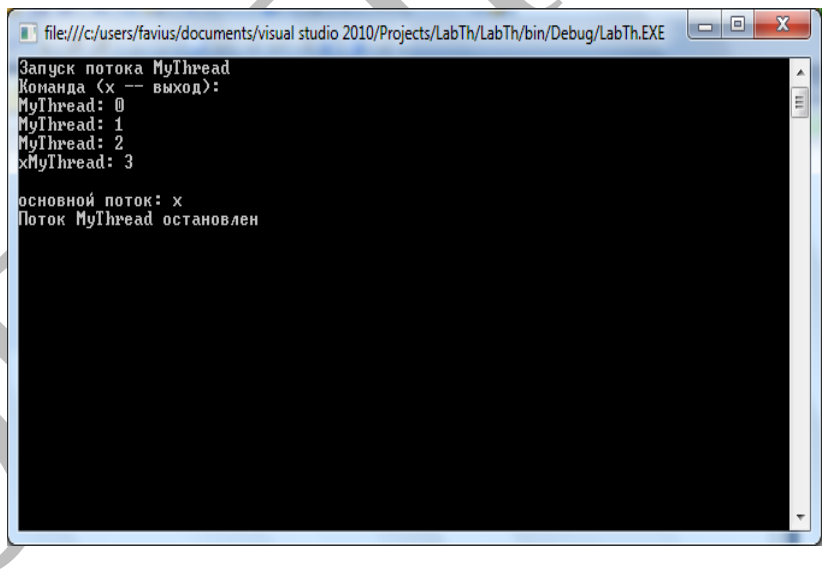
Пример. В примере показана программ в которой из основного потока создается еще один поток. Второй поток исполняется до того момента пока в первом переменной str не будет присвоено значение x. После этого глобальная переменная stopThread примет значение true и цикл во втором потоке завершится, что привет к завершения работы самого потока. В данном примере представлен обмен данными между потоками с помощью глобальных переменных.

Листинг программы

```
using System;
using System.Threading;

namespace LabTh
{
   class Program
   {
      static bool stopThread;
      public static void MyThread()
      {
         for (int i = 0; ; i++)
         {
            if (stopThread)
               break;
            Console.WriteLine("MyThread: {0}", i);
            Thread.Sleep(2000);
         }
         Console.WriteLine("Поток MyThread остановлен");
      }
      [STAThread]
```

```
        static void Main(string[] args)
        {
            ThreadStart myThreadDelegate = new ThreadStart(MyThread);
            Thread thr = new Thread(myThreadDelegate);
            Console.WriteLine("Запуск потока MyThread");
            stopThread = false;
            thr.Start();
            string str;
            do
            {
                Console.WriteLine("Команда (x -- выход): ");
                str = Console.ReadLine();
                Console.WriteLine("основной поток: {0}", str);
            }
            while (str != "x");
            stopThread = true;
            Console.ReadLine();
        }
    }
}
```

При работе программы видно, что при присвоении заданной переменной значения "x" происходит прекращение работы потока.



19. Работа с папками

Предмет исследований

- Свойство компонентов label, button, textBox, listbox.
- Просмотр содержимого папки.
- Создание папки в указанном месте.
- Удаление указанной папки

Контрольные вопросы

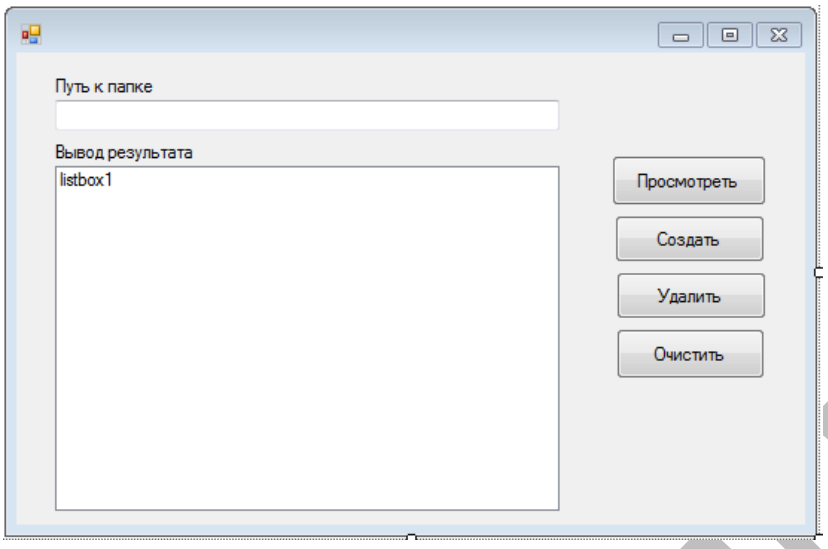
1. Просмотр содержимого папками.
2. Создание новых папками и назначение имени.
3. Удаление заданных папок.
4. Использование listbox для вывода содержимого папок.
5. Использование textbox для работы с папками.
6. События onClick.
7. Права доступа к папкам.

Задание. Создать программу, позволяющую создавать и удалять, а также просматривать их содержимое. Программа должна содержать графический интерфейс.

Пример. В примере представлена программа, позволяющая создавать, удалять и просматривать папки на жестком диске. В программе реализован графический интерфейс. При помощи элемента интерфейса textbox указывается, какую папку создавать, удалять или просматривать. Выбор действий производится с помощь элементов button**.** Ввод результата производится с помощью элемента интерфейса listbox**.**

В поле с "Путь к папке" указывается папка. При помощи кнопки "Просмотреть" выводится содержимое указанной папки. Кнопки "Создать" и "Удалить" используются для создания либо удаление по заданному пути. В поле "Вывод результата выводится содержимое папки либо сообщение об успешности операций создания или удаления папки. Кнопка "Очистить используется для очистки поля "Вывод результат".

Окно формы



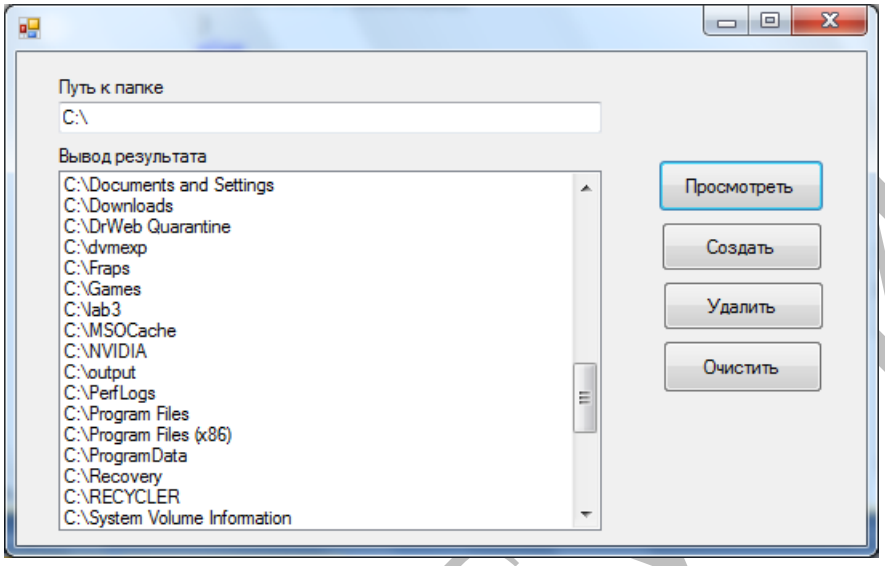
Листинг программы

```
using System;
using System.Windows.Forms;
using System.IO;
namespace LabDirector
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         string a = textBox1.Text;
         string[] fl;
         string[] dr;
         if(a != (null))
         {
            fl = Directory.GetFiles(a);
            dr = Directory.GetDirectories(a);
            foreach (string h in fl) listbox1.Items.Add(h);
            foreach (string s in dr) listbox1.Items.Add(s);
         }
      }
      private void button2_Click(object sender, EventArgs e)
      {
         string a = textBox1.Text;
         string q;
         if (a != (null))
         {
```

```
            Directory.CreateDirectory(a);
            if (Directory.Exists(a))
            {
                q = String.Concat(a, " Успешное создания!");
                listbox1.Items.Add(q);
            }
            else
            {
                q = String.Concat(a, " Ошибка создания!");
                listbox1.Items.Add(q);
            }
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        string a = textBox1.Text;
        string q;
        if (a != (null))
        {
            if (Directory.Exists(a))
            {
                Directory.Delete(a);
            }
            else
            {
                q = String.Concat(a, " Каталога не существует!");
                listbox1.Items.Add(q);
            }
            if (Directory.Exists(a))
            {
                q = String.Concat(a, " Ошибка удаление!");
                listbox1.Items.Add(q);
            }
            else
            {
                q = String.Concat(a, " Успешно удален!");
                listbox1.Items.Add(q);
            }
        }
    }
    private void button4_Click(object sender, EventArgs e)
    {
        listbox1.Items.Clear();
```

```
        }
    }
}
```

При запуске программы мы указываем путь и выбираем одно из возможных действий. На данной картинке выбран просмотр содержимого папки:



20. Многооконные приложения

Предмет исследований

- Исследование использование нескольких окон форм (Form) в одном приложении.
- Возможность вывода сообщений при работе с приложениями Windows Form.
- Открытие и закрытие новых окон в приложениях.
- Создание парольного доступа к окнам приложения.

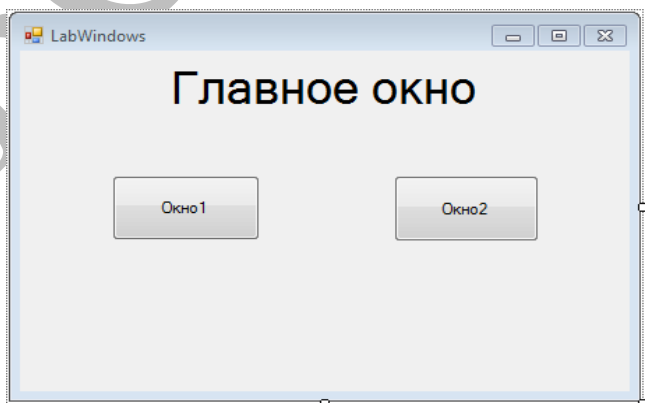
Контрольные вопросы

1. Открытие новых окон в приложениях.
2. Организация доступа к окнам.
3. Работа с MessageBox.
4. Организация завершение приложения.
5. Интеграция изображений в форму.
6. Закрытие окно в приложении.
7. Работа с событием FormClosing.
8. Объекты Show или Showdialog.

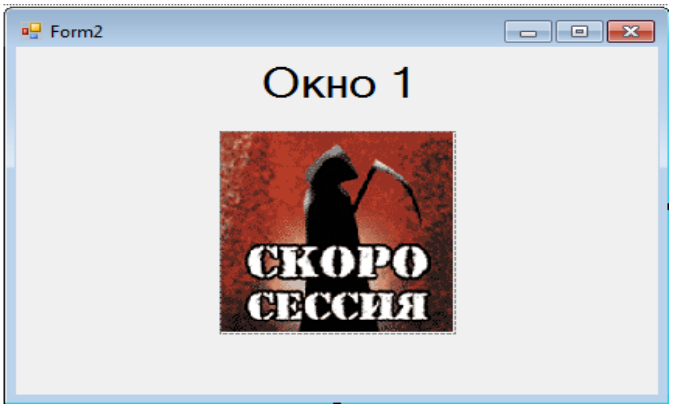
Задание. Создайте программу, в которой предусмотрена работа с несколькими окнами, организация парольного доступа к скрытым окном и вывод сообщений об ошибках через MessageBox.

Пример. В примере показана программа, демонстрирующая работу с многооконными программами и организацию парольного доступа к окнам а так же работу с MessageBox. В программе используются компоненты Button, TextBox и PictureBox. Все эти компоненты можно найти в "Панели элементов" вкладка "Стандартные элементы управления".

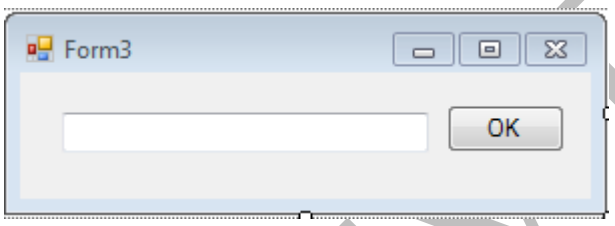
При запуске программы открывается "Главное окно" (форма 1) с двумя кнопками.



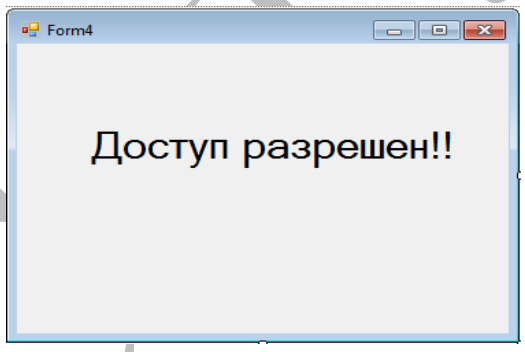
При нажатии на кнопку с именем "Окно 1" открывается “Окно 1” (форма 2) с картинкой. Например



Если пользователь нажимает кнопку "Окно 2", то вызывается форма 4, окно которой не отображается, а из нее вызывается форма 3. Появляется “Окно 3” (форма 3) с запросом пароля.



Если пароль введен правильно, то пользователь получает доступ к “Окну 4” (форма 4).



Окно с запросом пароля автоматически закрываетчя. Если пользователь закрывает окно ввода пароля, то приложение завершается.

Листинг программы (Форма 1)

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

namespace LabWindows
{
   public partial class Form1: Form
   {
      public Form1()
      {
         InitializeComponent();/        // Показать окно 1
      }
      private void button1_Click(object sender, EventArgs e)
```

```
        {
            Form2 f2 = new Form2();
            f2.Show();                  // Показать окно 2
        }
        private void button2_Click(object sender, EventArgs e)
        {
            Form4 f4 = new Form4();
            f4.Show();                  // Показать окно 4
        }
    }
}
```

Листинг программы (Форма 2)

```
using System.Windows.Forms;

namespace LabWindows
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();      // Показать окно 2
        }
    }
}
```

Отображается окно запроса пароля.

Листинг программы (Форма 3)

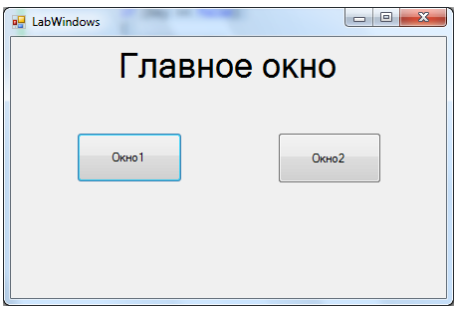
```
using System;
using System.Windows.Forms;

namespace LabWindows
{
    public partial class Form3 : Form
    {
        bool key = false;
        public Form3()
        {
            InitializeComponent();      // Показать окно 3
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string pass = "123";        // Это пароль
```

```
            if (pass == textBox1.Text)
            {
                key = true;
                Form3.ActiveForm.Close();          // Закрыть окно 3
            }
            else
                MessageBox.Show("Ошибка!!! Пароля!!");              // Вывод сообщения
        }
        private void Form3_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (key == false)
            {
                int a = 0;
               Environment.Exit(a);        // Закрыть приложение
            }
        }
    }
}
```

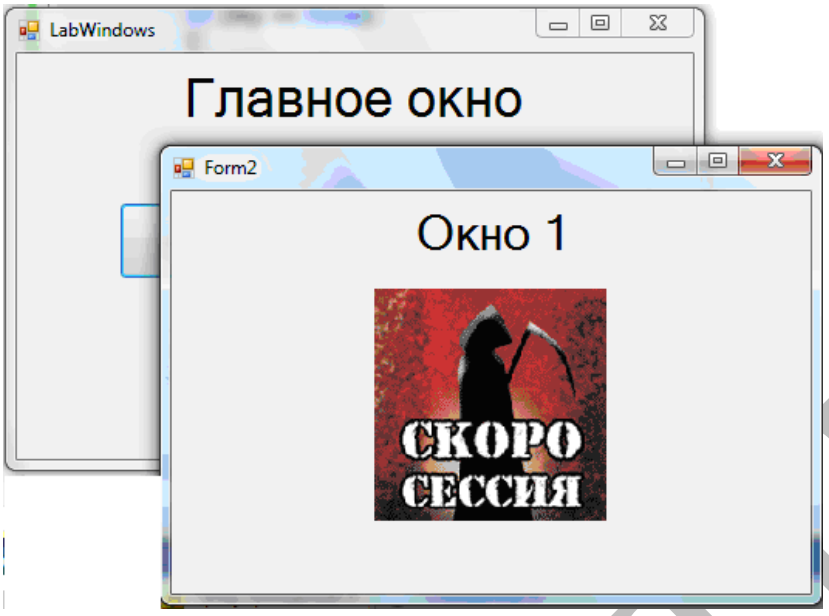
Листинг программы (Форма 4)

```
using System.Windows.Forms;

namespace LabWindows
{
    public partial class Form4 :Form
    {
        public Form4()
        {
            Form3 f3 = new Form3();
            f3.ShowDialog();            // Показать окно 4
            InitializeComponent();
        }
    }
}
```

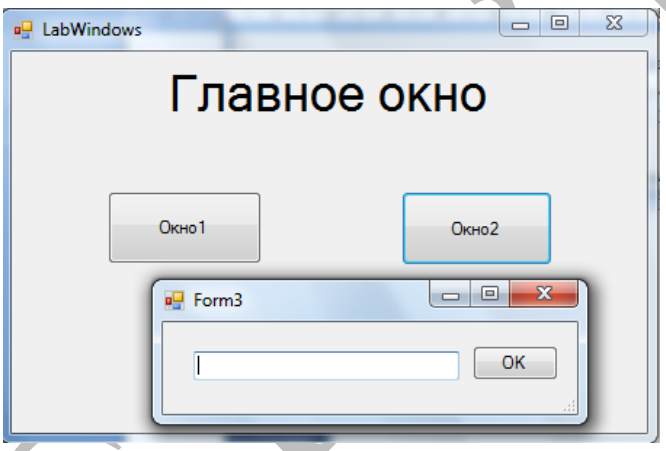
При запуске программы мы видим следующее:



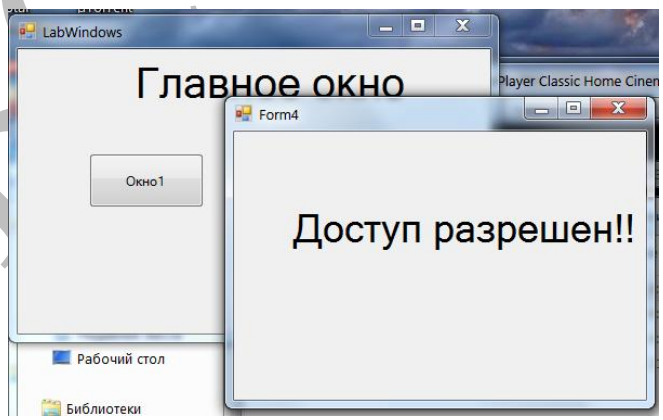
Если нажата кнопка «Окно 1», то



Если нажата кнопка «Окно 2», то



Если пароль правильный, то



21. Компонент ProgressBar

Предмет исследований

- Использование таймера и его режимов.
- Свойства компонента ProgressBar.
- Компонент statusStrip и его варианты вывода им информации.
- Работа с подкомпонентом StatusLabel компонента statusStrip.

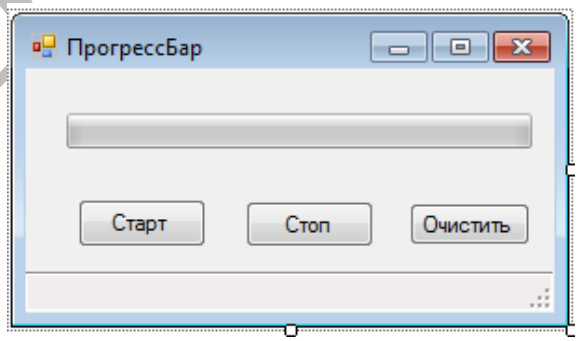
Контрольные вопросы

1. Варианты использования компонента ProgressBar в программах.
2. Задание параметров ProgressBar.
3. Работа с таймером.
4. Варианты использование Таймера в программах.
5. Компонент statusStrip и его подкомпоненты.
6. Задание параметров подкомпонентам statusStrip.
7. События onClick.

Задание. Создать программу, в которой будут использоваться следующие компоненты: ProgressBar, statusStrip (и один из его подкомпонентов). Для работы с ProgressBar, statusStrip используйте таймер.

Пример. Программа, демонстрирующая работу с компонентами ProgressBar, statusStrip (и один из его подкомпонентов). Их мы добавляем в форму из "Панели элементов", вкладка "Стандартные элементы управления". В программе также использованы компоненты Button для управления, они в "Панели элементов".

При нажатии кнопки "Старт" полоса прогресса в компоненте ProgressBar начинает заполняться слева направо, внизу в компоненте statusBar показывается процент выполнения. При помощи кнопки "Стоп" полосу можно остановить. Кнопка "Очистка" используется для перевода ProgressBar и statusStrip в изначальное состояние.

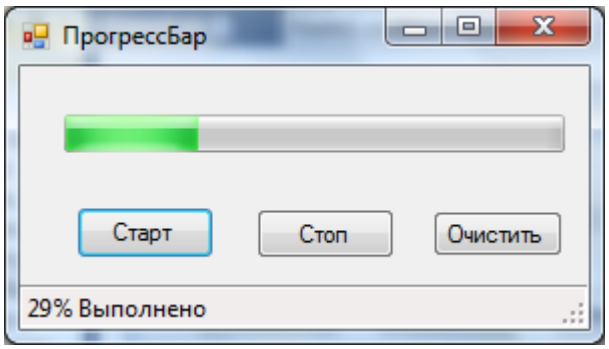


Листинг программы

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

namespace LabProgress
```

```
{
   public partial class Form1 : Form
   {
      Timer time = new Timer();
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         time.Interval = 250;
         time.Tick += new EventHandler(IncreaseProgressBar);
         time.Start();
      }
      private void IncreaseProgressBar(object sender, EventArgs e)
      {
        progressBar1.Increment(1);
        toolStripStatusLabel1.Text = progressBar1.Value.ToString() +
           "% Выполнено";
        if (progressBar1.Value == progressBar1.Maximum)
        {
           time.Stop();
           progressBar1.Value = 0;
        }
      }
      private void button2_Click(object sender, EventArgs e)
      {
         time.Stop();
      }

      private void button3_Click(object sender, EventArgs e)
      {
         progressBar1.Value = 0;
         toolStripStatusLabel1.Text = "0";
      }
   }
}
```

При запуске нажимаем кнопку "Старт" и смотрим результат выполнения.



22. Списки

Предмет исследований

Приложение Windows с компонентами списков:

- Список listBox.
- Комбинированный список comboBox. Это комбинация простого списка с однострочным редактором textBox.

Контрольные вопросы

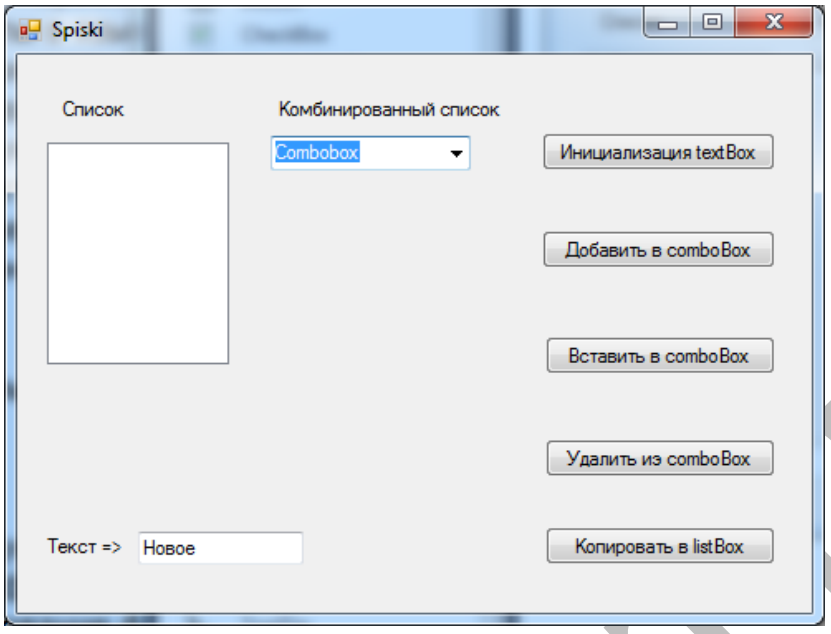
1. Списки, их назначение.
2. Компоненты списков listBox (список) и ComboBox (комбинированный список) в ИСР. Их назначение и сравнение.
3. Создание в форме компонентов списков.
4. Свойство Items (элементы). Нумерация элементов списков.
5. Метод SelectedIndex.
6. Предварительное редактирование списков.
7. Свойство Text (текст) компонента comboBox.
8. Программное редактирование строк компонентов списков.
9. Метод добавить элемент Add(Item) и его использование.
10. Метод ввести элемент Insert(Index,Item) и его использование.
11. Метод удалить элемент RemoveAt(Index) и его использование.

Задание. Создать проект работы со списками.

Пример. Создать проект работы со списками. В проекте предусмотреть:

- Инициализацию поля ввода.
- Добавление строки из поля ввода в конец комбинированного списка.
- Вставку строки из поля ввода в комбинированный список в выделенное место.
- Удаление строки из комбинированного списка в выделенном месте.
- Копирование строки из комбинированного списка в выделенном месте в список

Рекомендуемая форма программы:



Форма содержит следующие компоненты:

- listBox1 - список.
- label1 – метка для заголовка (Список) компонента listBox1.
- comboBox1 - комбинированный список.
- label2 – метка для заголовка (Комбинированный список) компонента comboBox1.
- textBox1 - однострочный редактор для ввода текста. В него вводится строка текста (по умолчанию - Новое).
- label3 – метка для заголовка (Текст = >) компонента textBox1.
- button1– кнопка "Инициализация textBox". Позволяет в textBox1 очистить поле Text и установить там курсор.
- button2 – кнопка "Добавить в comboBox" для заполнения компонента comboBox1.
- button3 – кнопка "Вставить в comboBox". Позволяет вставить строку в выделенное место в списке.
- Button4 – кнопка "Удалить из comboBox". Позволяет удаплить выделенную строку в списке.
- button5 – кнопка "Копировать в listBox". Позволяет копировать в listBox строку, выделяемую в comboBox.

Выполнение

Активизируем ИСР. В главном меню исполняем команду **File=>New Proect.** Активизируется окно выбора типа проекта. В нем нужно выбрать WindowsFormApplication. Отображается окно выбора скрытности кода, в котором выбираем обычныой загрузки. Теперь отображаются окна проекта:

- Form1.cs [Design]. Конструктор формы, там пустая форма, которую нужно заполнить компонентами.
- Toolbox - компоненты.

- Properties - свойства.
- Form1.cs. Редактор кода с шаблоном кода модуля формы. Если окна нет, то правый щелчок по окну конструктора вызывает контекстное меню, в котором нужно исполнить команду View Code.
- Error List. Сообщения об ошибках.

Задание свойств формы. Выбираем объект Form1 и подбираем его размеры и положение. В окне **Properties** назначаем свойства формы:

Свойство		Значение
English	Перевод	English
Name	Имя	Form1
Text	Текст	Списки

Заполнение формы. При заполнении формы в палитре компонент **Toolbox** выбирается нужный блок и переносится в форму. Там подбираются его положение и размеры. Затем в окне **Properties** назначаются свойства компонента. Большая часть свойств устанавливается автоматически, их редактировать не следует.

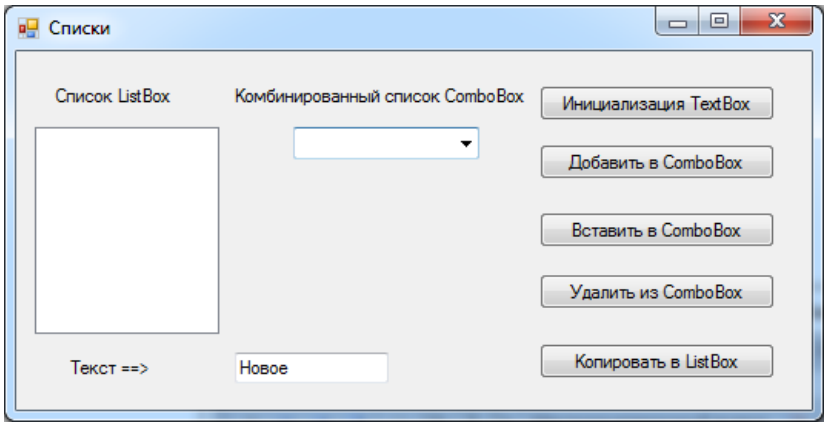
Для кнопок следует изменить текст, показав выполняемое действие. Также необходимо добавить функциональность кнопки, написав код обработчика события нажатия. При двойном щелчке по кнопке ИСР автоматически объявляет подпрограмму обработчика и в код автоматически заносит ее шаблон. Остается для каждого обработчика записать исполняемые команды.

Листинг программы

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

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

      private void Form1_Load(object sender, EventArgs e)
      {

      }

      private void button1_Click(object sender, EventArgs e)
      {
         textBox1.Text = "";
         textBox1.Focus();
```

```
        }

        private void button2_Click(object sender, EventArgs e)
        {
            comboBox1.Items.Add(textBox1.Text);
            textBox1.Text = "";
            textBox1.Focus();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            int i = comboBox1.SelectedIndex;
            comboBox1.Items.Insert(i, textBox1.Text);
            textBox1.Text = "";
            textBox1.Focus();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            int i = comboBox1.SelectedIndex;
            comboBox1.Items.RemoveAt(i);
        }

        private void button5_Click(object sender, EventArgs e)
        {
            int i = comboBox1.SelectedIndex;
            listBox1.Items.Add(comboBox1.Items[i]);
        }
    }
}
```

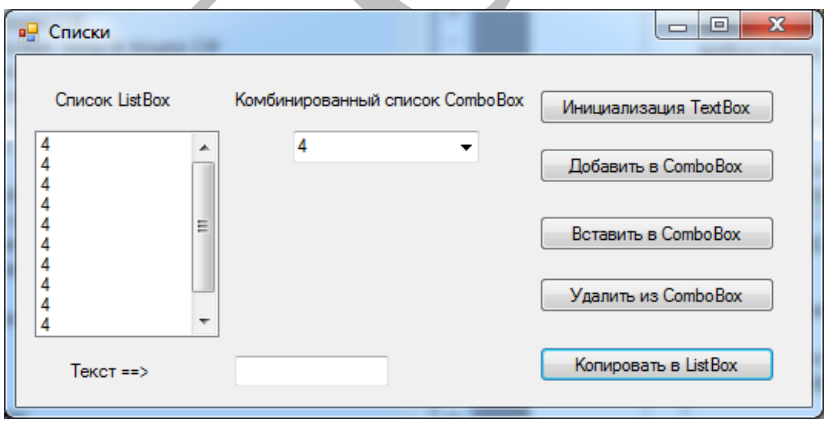
Проверим программу в работе сначала без компиляции. Запускаем программу командой **Отладка=>Запуск без отладки**. Если допущены ошибки, то программа не исполняется, выводится окно с сообщением об этом. В окне ошибок все ошибки перечислены, их нужно исправить и посторно запустить пронрамму. Если ошибок нет, то программа запускается.



Теперь надо проверить правильность логики обработчиков.

- Инициализация textBox. Кнопка очищает textBox1 и передает ему фокус ввода. Курсор появляется в его поле.
- Добавить в comboBox. Для каждой строки в textBox1 заносим текст, кнопкой переносим его в конец списка. Повторяем операцию нужное число раз. Каждый раз проверяем список comboBox1, убеждаясь в появлении новой строки в конце.
- Вставить в comboBox.. В textBox1 заносим строку текста для ввода в произвольно выбираемую позицию. В comboBox1 выделяем позицию для ввода. Кнопкой текст из textBox1 перемещается в выделенную строку comboBox1.
- Удалить из comboBox.. В comboBox1 выделяем позицию для удаления. Кнопкой выделенную строка удаляется из comboBox1.
- Копировать в listBox. В comboBox выделяем позицию для копирования, ее текст перемещается в верхнюю строку. Кнопкой копируем ее в конец listBox. Каждый раз наблюдаем listBox, чтобы убедится в появлении там новой строки в очередной позиции. Обратите внимание на появление в listBox линейки прокрутки, когда его поле заполнено.

Приложение перед закрытием программы:



Если все правильно, то проект компилируется и собирается. Это можно сделать двумя способами:

- Команда Построение=>Построить решение генерирует исполняемый файл в отладочном режиме. Он размещается во вложенной папке проекта bin\debug.

- Команда Построение=>Перестроить решение генерирует исполяемый файл в выпускном режиме. Он размещаеттсв во вложенной папке проекта bin\release.

23. Таблицы

Предмет исследований

- Свойства компонентов label, button, textBox, dataGridView.
- События onClick, onChange, onCreate.
- Как создать код обработчика события.

Контрольные вопросы

1. События onClick, onCreate.
2. Назначение и свойства компонента кнопка button.
3. Назначение и свойства компонент метка label.
4. Назначение и свойства компонента однострочный редактор textBox.
5. Назначение и свойства компонента dataGridView.
6. Как иници таблицу при старте приложения.
7. Как создать обработчик события onClick.
8. Как использовать строковые данные для вычислений.

Задание. Создать проект “Таблица конвертации” для конвертации рублей России в разные валюты с учетом комиссии. В таблице предусмотреть возможность редактирования содержимого ячеек, добавления и удаления строк и столбцов.

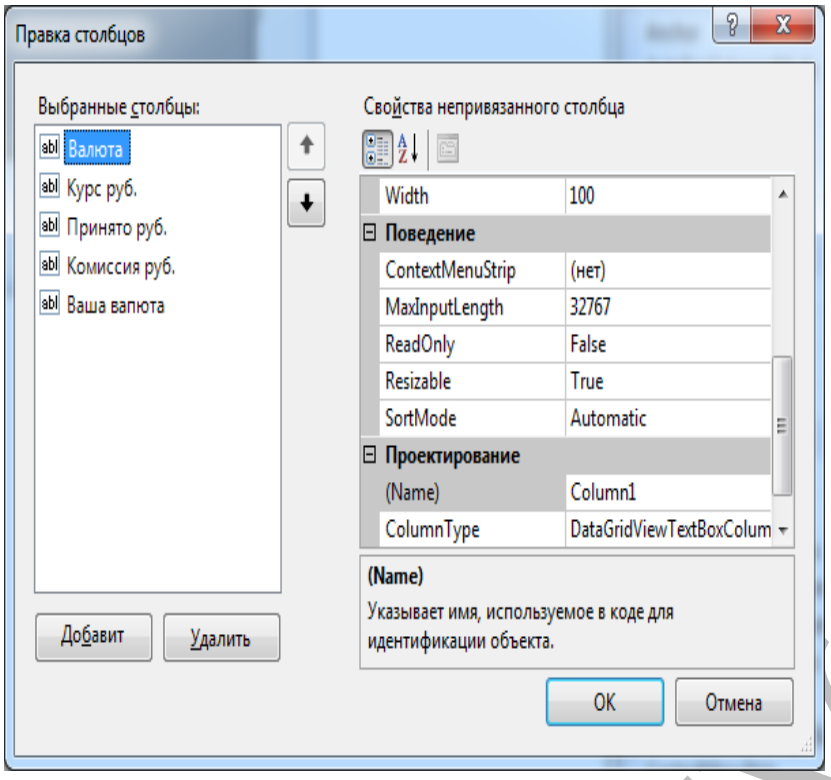
Варианты заданий

№	Валюты		
1.	Доллар США	Фунт стерлингов	Евро
2.	Фунт стерлингов	Евро	Рубль Беларуси
3.	Евро	Японская йена	Гривна Украины
4.	Японская йена	Рубль Беларуси	Японская йена
5.	Рубль Беларуси	Гривна Украины	Доллар США
6.	Гривна Украины	Фунт стерлингов	Доллар США
7.	Доллар США	Японская йена	Фунт стерлингов
8.	Фунт стерлингов	Японская йена	Евро
9.	Евро	Рубль Беларуси	Японская йена
10.	Японская йена	Гривна Украины	Рубль Беларуси

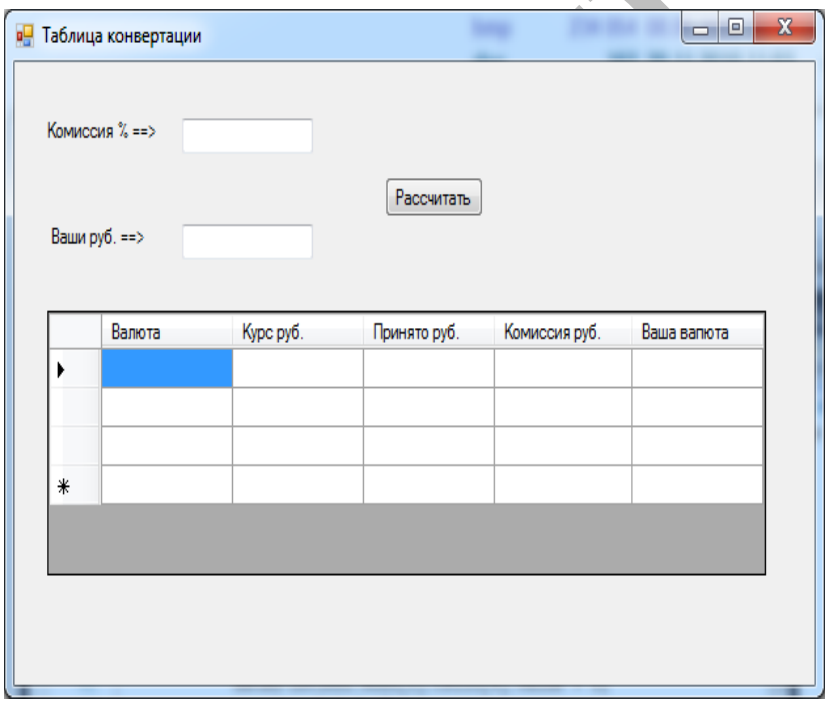
Пример.

- Запустить ИСР.
- Создать в ней новое WindowsForm приложение.
- Сохранить проект под именем WindowsFormTablitsa.
- Свойству формы text присвить значение Таблица конвертации
- Установить на форме компоненты: однострочные редакторы textBox1 и textBox1, метки label1 и label2, кнопку button1, таблицу строк dataGridView1
- В окне свойств метки label1 свойство text = Комиссия % ==>.
- В окне свойств метки label1 свойство text = Ваши руб. ==>.
- В окне свойств кнопки button1 свойство text = Рассчитать.

Правой кнопкой мыши вызвать для объекта dataGridView1 команду «Правка столбцов». Вызывается диалоговое окно



В нем задаем заголовки столбцов. После установок интерфейс проекта примет вид



Двойным щелчком по кнопке в форме создать шаблоны обработчика нажатия на кнопку.

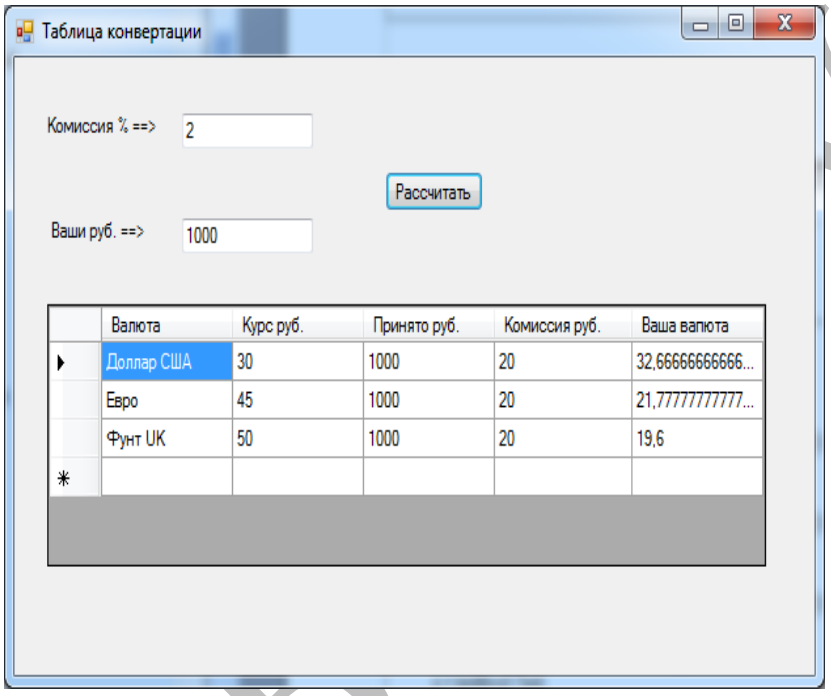
Написать коды процедуры обработчика. В ней задаются 3 строки таблицы, задается комиссионный сбор, выбираются 3 валюты, для которых заносятся текущие курсы

Листинг программы

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

namespace WindowsFormsTablitsa
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         dataGridView1.Rows.Add(3);
      }

      private void button1_Click(object sender, EventArgs e)
      {
         dataGridView1.Rows[0].Cells[0].Value = "Доллар США";
         dataGridView1.Rows[1].Cells[0].Value = "Евро";
         dataGridView1.Rows[2].Cells[0].Value = "Фунт UK";
         dataGridView1.Rows[0].Cells[1].Value = "30";
         double vd =
       System.Convert.ToDouble(dataGridView1.Rows[0].Cells[1].Value);
         dataGridView1.Rows[1].Cells[1].Value = "45";
         double ve =
       System.Convert.ToDouble(dataGridView1.Rows[1].Cells[1].Value);
         dataGridView1.Rows[2].Cells[1].Value = "50";
         double vf =
       System.Convert.ToDouble(dataGridView1.Rows[2].Cells[1].Value);
         string s = textBox1.Text;
         double k = System.Convert.ToDouble(s);
         s = textBox2.Text;
         double r = System.Convert.ToDouble(s);
         dataGridView1.Rows[0].Cells[2].Value = s;
         dataGridView1.Rows[1].Cells[2].Value = s;
         dataGridView1.Rows[2].Cells[2].Value = s;
         double kr = r*k/100;
         s = System.Convert.ToString(kr);
         dataGridView1.Rows[0].Cells[3].Value = s;
         dataGridView1.Rows[1].Cells[3].Value = s;
         dataGridView1.Rows[2].Cells[3].Value = s;
         double v = (r-kr)/vd;
         s = System.Convert.ToString(v);
         dataGridView1.Rows[0].Cells[4].Value = s;
         v = (r - kr) / ve;
         s = System.Convert.ToString(v);
```

```
            dataGridView1.Rows[1].Cells[4].Value = s;
            v = (r - kr) / vf;
            s = System.Convert.ToString(v);
            dataGridView1.Rows[2].Cells[4].Value = s;
        }

        private void dataGridView1_CellContentClick(object sender,
         DataGridViewCellEventArgs e)
        {

        }
    }
}
```

Окно формы перед завершением программы:



24. Графика. Рисуем функции

Предмет исследований

- Графические средства C#.
- Компонент Chart

Контрольные вопросы

1. Класс Graphics (графический объект).
2. Класс Font - шрифт.
3. Класс Pen - перо.
4. Класс Brush - кисть.
5. Вывод текста. Метод DrawString.
6. Вывод линии. Метод DrawLine.
7. Компонент Chart – средство отбражения диаграмм.
8. ChartAreas – области диаграммы.
9. Series – ряды диаграммы.
10. Сколько рядов надо для рисования N графиков в одной области.
11. Legendes – лененды диаграммы..
12. Задание функции для отображения в компоненте Chart.
13. Стили отображения диаграмм.
14. Использование легенды в компоненте Chart.

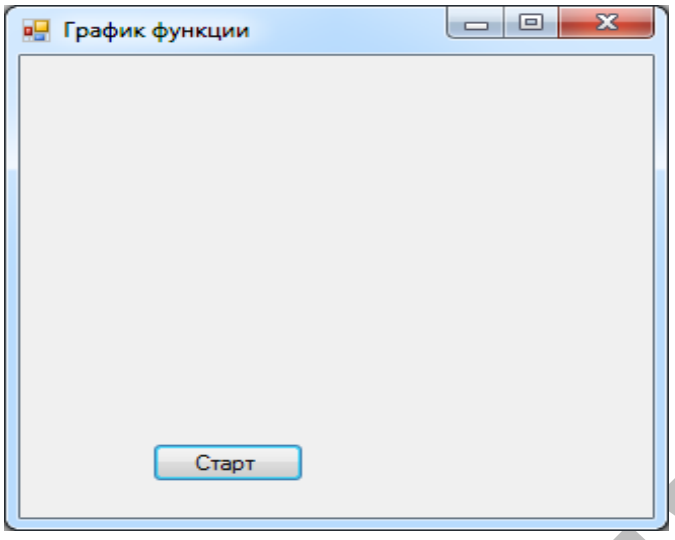
24.1. Текст и график функции в форме

Создать программу вывода текста и рисования в форме графиков функций. Проект – WindowsForm приложение.

Варианты заданий

№	График функции y(x)		
	Функция	x начальное	x конечное
1.	sin(x)	0	6π
2.	cos(x)	0	6π
3.	sin(x)+sin(2x)	0	6π
4.	sin(x)-sin(2x)	0	6π
5.	sin(x)+cos(2x)	0	6π
6.	sin(x)-cos(2x)	0	6π
7.	sin(x)*exp(x)	0	6π
8.	cos(x)*exp(x)	0	6π
9.	sin(x)*exp(-x)	0	6π
10.	cos(x)*exp(-x)	0	6π

Пример. Создать программу рисования в форме графика функции с поясняющим текстом. Проект – WindowsForms приложение. Функция - синус. Программа предусматривает рисование графика в форме Form1 линиями с помощью метода DrawLine. Над графиком с помощью метода DrawString выводится поясняющий текст. Кнопка Старт вызывает построение графиков в окне.

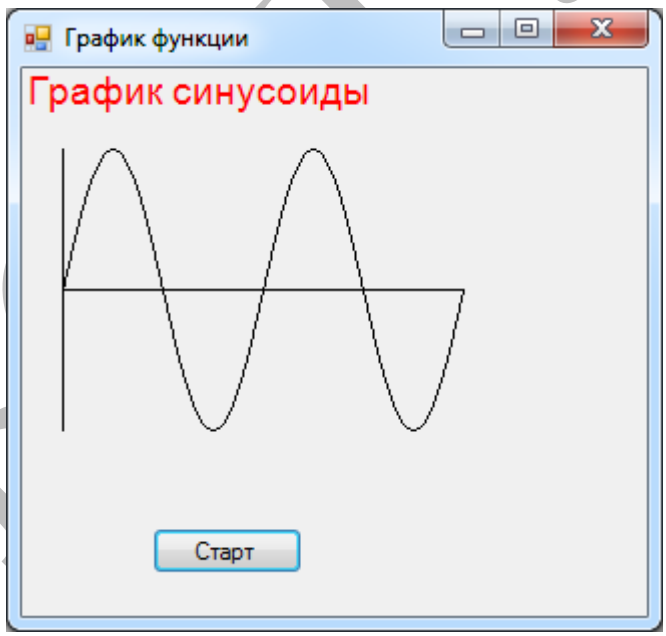


Листинг программы

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

namespace GraphicFunction
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
      private void button1_Click(object sender, EventArgs e)
      {
         int imax =100;            //число точек в периоде
         int t=2;                  //число периодов
         int amp=70;               //амплитуда
         int h = 40;               //отступ для текста
         int x0=20;                //начала координат
         int y0 = h+amp;
         double[] f = new double [imax*t+10];
         // Функция
         for (int i = 0; i < imax * t; i++)
         {
            f[i] = Math.Round(amp * Math.Sin(2 * Math.PI / imax * i));
         }
         // Инструменты рисования
         Graphics g = Graphics.FromHwnd(this.Handle);          // Где рисуем
         Pen pen = Pens.Black;                        // Чем рисуем
         // Текст заголовка
```

```
            g.DrawString("График синусоиды", new Font("Arial", 14),
              Brushes.Red, 0, 0); //Вывод текста
            //textBox1.Text = "График синусоиды";
            //Рисуем график
            g.DrawLine(pen , x0, y0, x0+imax*t, y0);       //Рисуем ось X
            g.DrawLine(pen, x0, y0-amp, x0, y0+amp);       //Рисуем ось Y
            for (int i = 0; i < imax * t; i++)             //Рисуем график
            {
               int f1 = y0 - (int)f[i];                  //Координата Y[i]
               int f2 = y0 - (int)f[i + 1];              //Координата Y[i+1]
               g.DrawLine(pen, x0+i, f1, x0+i+1, f2);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

При прогоне программы получается результат:



24.2. Компонент Chart

Создать программу отображения диаграммы двух функций с использованием компонента Chart. Проект – WindowsForm приложение.

Варианты заданий

№	График функции y(x)		
	Функция 1	Функция 2	Стиль линий
1.	sin(x)	sin(2x)	Spline
2.	sin(x)	-sin(2x)	Point
3.	cos(x)	sin(2x)	StepLine
4.	cos(x)	-sin(2x)	Line
5.	sin(x)	sin(2x)	Colomn
6.	sin(x)	-sin(2x)	Spline
7.	cos(x)	sin(2x)	Point
8.	cos(x)	-sin(2x)	StepLine
9.	sin(x)	sin(3x)	Line
10.	sin(x)	-sin(3x)	Colomn

Пример. Создать программу отображения диаграммы двух функций sin(x) и cos(x) с использованием компонента Chart. Проект – WindowsForm приложение. Стиль линий Spline.

Создаем – WindowsForm приложение. В форму заносим компоненты chart1 для отображения диаграммыи button1 для создания стартового обработчика событий.

В окне свойств Button1 задаем его свойству Text значение Старт.

В окне компонента Chart1 в раэделе Seies определяем две серии:

- Для функции sin(x) с именем Синус.
- Для функции cos(x) с именем Косинус.

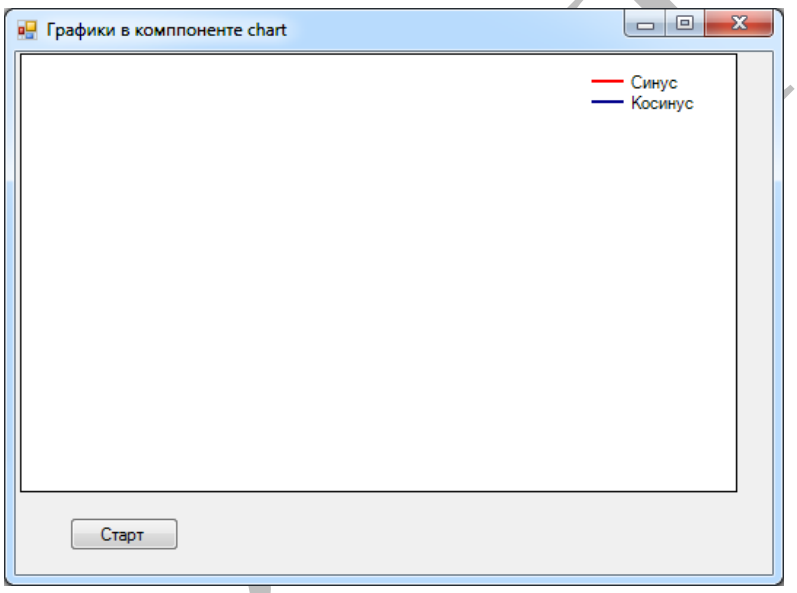
Двойным щелчком по кнопке создаем в окне кода шаблон обработчика собыия нажатия кнопки. Функциоинальная часть обработчика включает задание в цикле наборов данных для серий.

Листинг программы

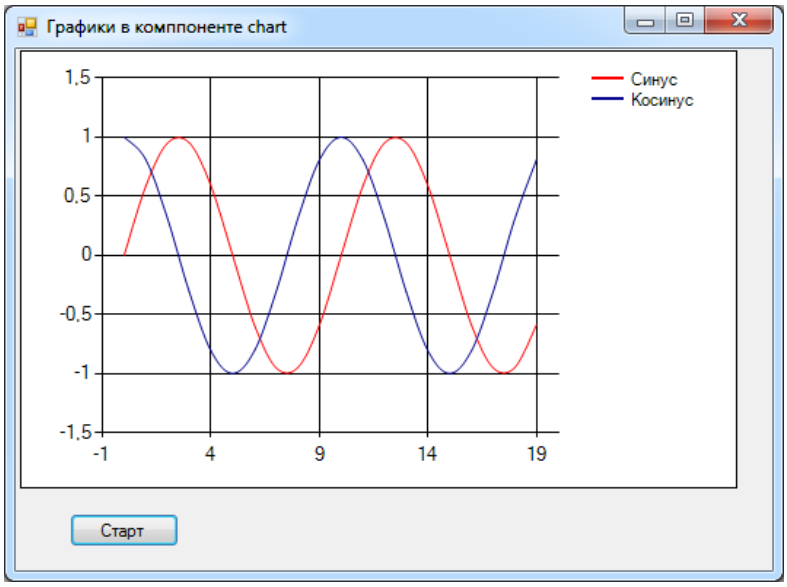
```
using System;
using System.Windows.Forms;

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

      private void chart1_Click(object sender, EventArgs e)
      {
      }

      private void button1_Click(object sender, EventArgs e)
```

```
        {
            double y = 0;
            for (int x = 0; (x <= 19); x++)
            {
                y = Math.Sin(Math.PI / 5 * x);
                chart1.Series["Синус"].Points.AddXY(x, y);
                y = Math.Cos(Math.PI / 5 * x);
                chart1.Series["Косинус"].Points.AddXY(x, y);
            }

        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

При запуске программы отображается форма, в которой .прорисовываются диаграмма с двумя поименованными сериями и кнопка старта. Самаих графиков пока нет, так как данные для них формирует обработчик.



Кнопка button1 (Старт). Она вызывает построение графиков в окне.


Графики в комппоненте chart
1,5
1
0,5
0
-0,5
-1
-1,5
-1
4
9
14
19
Синус
Косинус
Старт

25. Графика, рисование фигур

Предмет исследований

- Методы работы с графическими фигурами.
- Методы работы с залитыми графическими фигурами.

Контрольные вопросы

1. Назначение графических примитивов.
2. Кривая Безье и метод DrawBezier.
3. Кривая и метод DrawCurve.
4. Замкнутая кривая и метод DrawClosedCurve.
5. Прямоугольник и метод DrawRectangle.
6. Полигон и метод DrawPolygon.
7. Эллипс и метод DrawEllipse.
8. Дуга эллипса и метод DrawArc.
9. Торт и метод DrawPie.
10. Метод заполнения замкнутых фигур (заполнить область),
11. Стили линий.
12. Стили заливки.

Задание. Создать программу рисования в форме графических фигур без заливки и с заливкой.

Варианты заданий

№	Фигура	Стиль линий	Метод заливки	Стиль заливки
1.	Кривая Безье	Dush	FillClosedCurve	Cross
2.	Кривая	DashDot	FillRectangle	DiagonalCross
3.	Замкнутая кривая	DashDotDot	FillPolygon	ForwardDiagonal
4.	Прямоугольник	Dot	FillEllipse	BackwardDiagonal
5.	Полигон	Dush	FillPie	Cross
6.	Эллипс	DashDot	FillClosedCurve	DiagonalCross
7.	Торт	DashDotDot	FillRectangle	ForwardDiagonal
8.	Дуга эллипса	Dot	FillPolygon	BackwardDiagonal
9.	Кривая	Dush	FillEllipse	Cross
10.	Замкнутая кривая	DashDot	FillPie	DiagonalCross

Пример. Создать программу рисования в форме графических фигур без заливки и с заливкой.

В левом верхнем углу формы будут рисоваться два графика: верхний – фигура без заливки, нижний – фигура с заливкой.

В форме размещаем 4 выпадающих списка Combobox и метки Label для заголовков списков:

- Фигура - Combobox1 и Label1.
- Стиль линии – Combobox2 и Labe2.
- Фигура с заливкой – Combobox3 и Label3.
- Стиль заливки – Combobox4 и Label4.

Кнопка button1 с надписью старт для запуска обработчика события.

В окне свойств Combobox1 определим фигуры рисования:

- DrawBezier – кривая Безье.
- DrawCurve – кривая.
- DrawClosedCurve – замкнутая кривая.
- DrawRectangle – прямоугольник.
- DrawPolygon – многоугольник.
- DrawEllipse – эллипс.
- DrawArc – дуга эллипса.
- DrawPie – торт.

В окне свойств Combobox2 определим стили линий фигур:

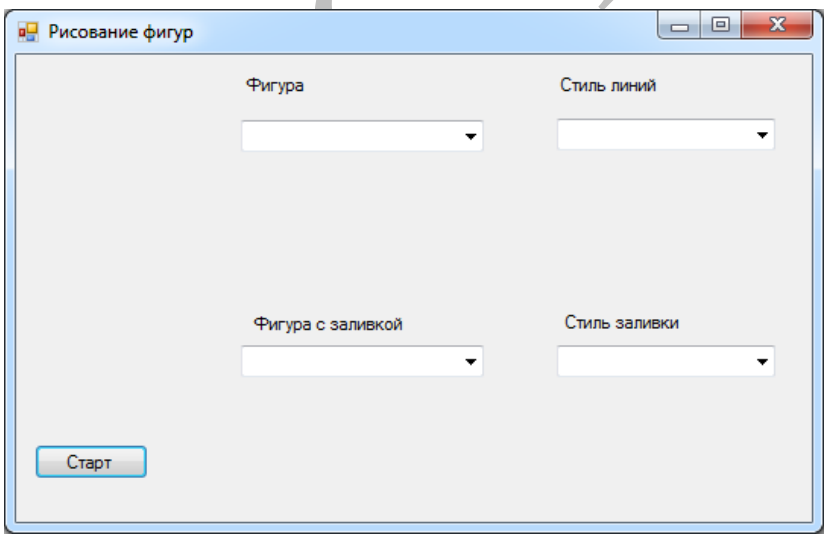
- Dash – тире.
- DashDot – тире + точки.
- DashDotDot - тире + 2 точки.
- Dot - точки.

В окне свойств Combobox3 определим фигуры с заливкой:

- FillClosedCurve - замкнутая кривая.
- FillRectangle - прямоугольник.
- FillPolygon - многоугольник.
- FillEllipse - эллипс.
- FillPie - торт.

В окне свойств Combobox4 определим стили заливки фигур:

- Cross – сетка.
- DiagonalCross – диагональная сетка.
- ForwardDiagonal – диагональ прямая.
- BackwardDiagonal – диагональ обратная.



В листинге программы дополнительно задаются размеры рисуемых фигур.

Листинг прогаммы

```
using System;
using System.Drawing;
```

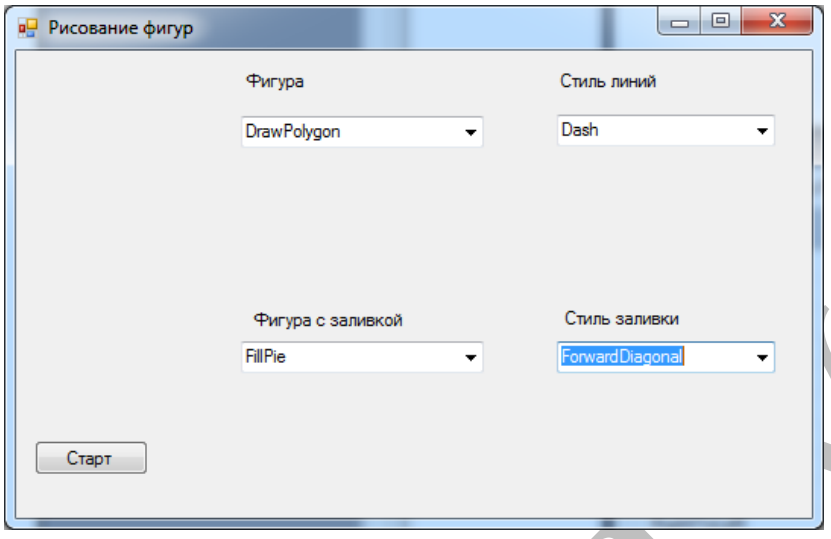
```
using System.Drawing.Drawing2D;
using System.Windows.Forms;

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

      private void button1_Click(object sender, EventArgs e)
      {
         Graphics g = Graphics.FromHwnd(this.Handle);       // Где рисуем
         Pen pen = new Pen(Color.Black);                    // Чем рисуем
         Brush brush = Brushes.White;
         Brush brush1 = Brushes.Blue;
         //stem.Drawing.Drawing2D.FillMode;
         HatchBrush brush2 = new HatchBrush(HatchStyle.Cross,
           ForeColor, BackColor);
         HatchBrush brush3 = new HatchBrush(HatchStyle.DiagonalCross,
           ForeColor, BackColor);
         HatchBrush brush4 = new HatchBrush(HatchStyle.ForwardDiagonal,
           ForeColor, BackColor);
         HatchBrush brush5 = new HatchBrush(HatchStyle.BackwardDiagonal,
           ForeColor, BackColor);
         Point[] p =
         {
            new Point(10, 0),
            new Point(80,70),
            new Point(90,50),
            new Point(50,90),
         };
         Point[] p1 =
         {
            new Point(0,110),
            new Point(80,180),
            new Point(90,160),
            new Point(50,200),
         };
         Rectangle rect = new Rectangle(0,0, 110, 220);
         Rectangle rect1 = new Rectangle(10,0, 100, 100);
         Rectangle rect2= new Rectangle(10,110, 100, 100);
```

```
g.FillRectangle(brush, rect);

if (comboBox3.SelectedItem == "Dash")
{
   pen.DashStyle = DashStyle.Dash;
}
if (comboBox3.SelectedItem == "DashDot")
{
   pen.DashStyle = DashStyle.DashDot;
}
if (comboBox3.SelectedItem == "DashDotDot")
{
   pen.DashStyle = DashStyle.DashDotDot;
}
if (comboBox3.SelectedItem == "Dot")
{
   pen.DashStyle = DashStyle.Dot;
}
if (comboBox4.SelectedItem == "Cross")
{
   brush1 = brush2;
}
if (comboBox4.SelectedItem == "DiagonalCross")
{
   brush1 = brush3;
}
if (comboBox4.SelectedItem == "ForwardDiagonal")
{
   brush1 = brush4;
}
if (comboBox4.SelectedItem == "BackwardDiagonal")
{
   brush1 = brush5;
}
if (comboBox1.SelectedItem == "DrawRectangle")
{
   g.DrawRectangle(pen, rect1);
}
if (comboBox2.SelectedItem == "FillRectangle")
{
   g.FillRectangle(brush1,10, 110, 100, 100);
}
if (comboBox1.SelectedItem == "DrawPolygon")
```

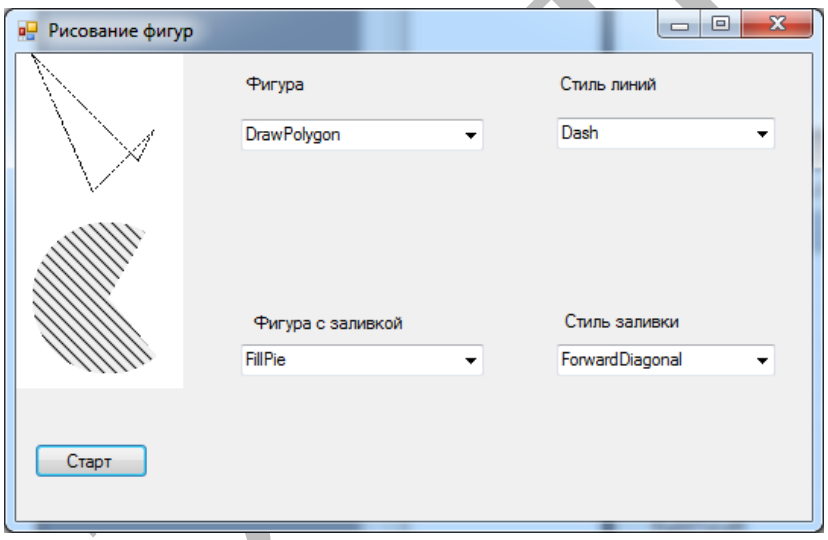
```
    {
        g.DrawPolygon(pen, p);
    }
    if (comboBox2.SelectedItem == "FillPolygon")
    {
        g.FillPolygon(brush1, p1);
    }
    if (comboBox1.SelectedItem == "DrawEllipse")
    {
        g.DrawEllipse(pen, rect1);
    }
    if (comboBox2.SelectedItem == "FillEllipse")
    {
        g.FillEllipse(brush1,10, 110, 100, 100);
    }
    if (comboBox1.SelectedItem == "DrawPie")
    {
        g.DrawPie(pen, rect1, 50, 250);
    }
    if (comboBox2.SelectedItem == "FillPie")
    {
        g.FillPie(brush1, rect2, 50, 250);
    }
    if (comboBox1.SelectedItem == "DrawCurve")
    {
        g.DrawCurve(pen, p);
    }
    if (comboBox1.SelectedItem == "DrawClosedCurve")
    {
        g.DrawClosedCurve(pen, p);
    }
    if (comboBox2.SelectedItem == "FillClosedCurve")
    {
        g.FillClosedCurve(brush1, p1);
    }
    if (comboBox1.SelectedItem == "DrawArc")
    {
        g.DrawArc(pen, rect1,50,250);
    }
    if (comboBox1.SelectedItem == "DrawBezier")
    {
        g.DrawBezier(pen, p[0], p[1], p[2], p[3]);
    }
}
```

```
        private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
        }
    }
}
```

Пример запуска.



А это результат:



26. Графика, растровые изображения

Предмет исследований

- Графические средства C#.
- Форматы графики.

Контрольные вопросы

1. BitMap изображение.
2. TIFF изображение
3. GiF изображение.
4. PNG изображение.
5. JPEG изображение.
6. Компонент PictureBox.
7. Принцип создания анимации.
8. Использование в анимационной программе фоновой картинки.
9. Использование в анимационной программе картинки образа.
10. Использование в анимационной программе буфера.

Задание. Создать программу работы с графическими файлами. Проект – WindowsForm приложение. В проекте растровая картинка *.bmp преобразуется в другие форматы графики.

Пример. Создать программу работы с графическими файлами. Проект – WindowsForm приложение. В программе используется файл растровой картинки Кувшинка.bmp, который надо загрузить в ту же папку, где находится проект приложения. Этот файл находится в папке УМД к ЛР.

В проекте растровая картинка Кувшинка.bmp преобразуется в другие форматы графики. Картинки отображаются в компонентах PictureBox1 (с заголовком BitMap в компоненте Label1) для исходника и PictureBox2 (с заголовком Результат в компоненте Label2).

Тип результата (TIFF, GIF, PNG, JPEG), выбирается из выпадающего списка combobox1 с заголовком «Выбор формата» в компоненте Label3.

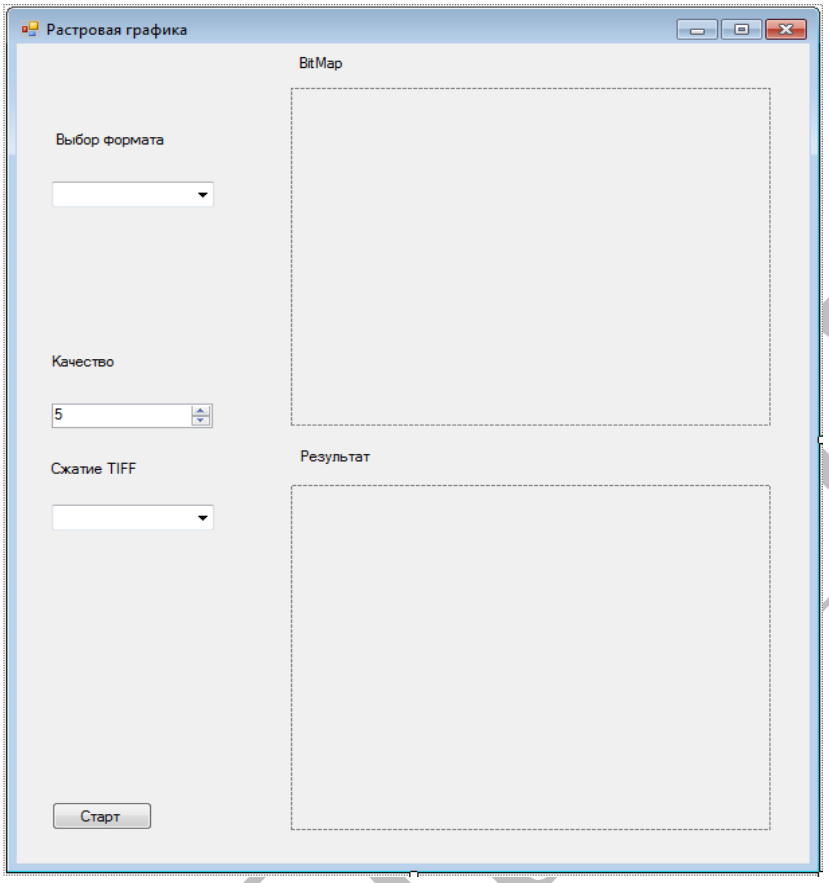
При выборе JPEG дополнительно выбирается желаемое качество результата от 5 до 100 из выпадающего списка-диапазона NumericUpDown с заголовком «Качество» в компоненте Label4.

При выборе TIFF дополнительно из выпадающего списка combobox2 с заголовком «Сжатие TIFF» в компоненте Label5 выбирается алгоритм сжатия:

- None – сжатия нет.
- Default – по умолчанию.
- Lzw – сжатие без потерь качества.
- Rle – с анализом повторов пикселей
- Zip – для архивирования.

Кнопка с надписью «Старт» запускает процесс преобразования.

Форма проекта задания 1. В окне свойств формы задаем свойство text = Растровая графика.



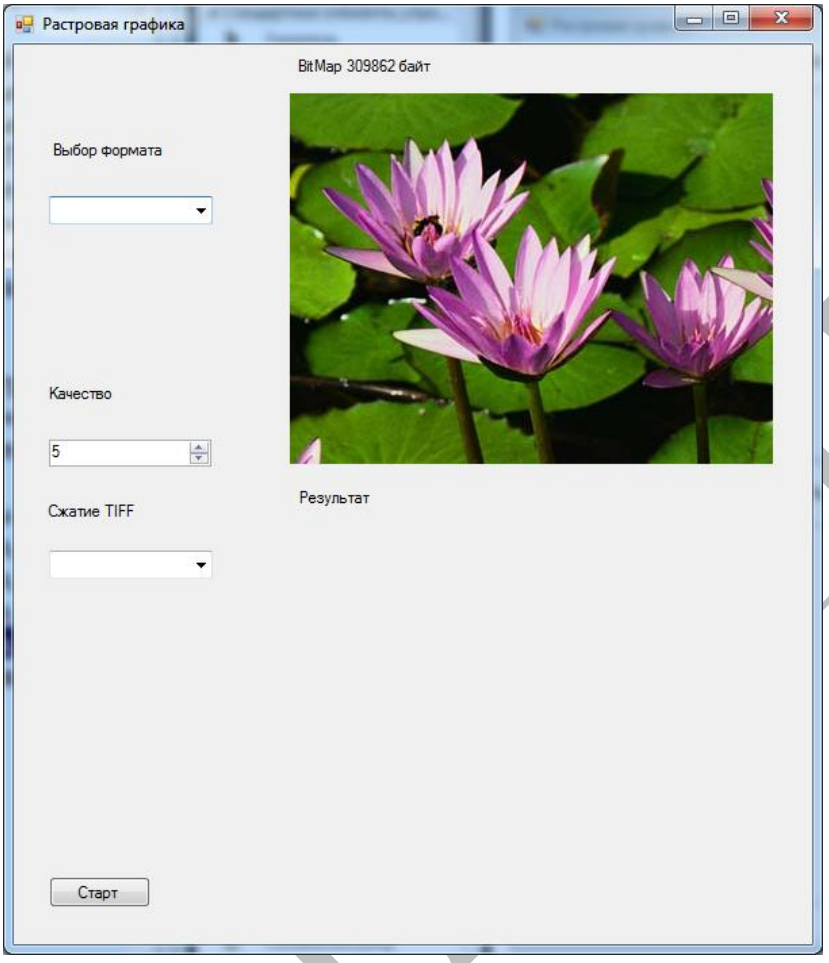
Листинг программы

```
using System;
using System.Drawing;
using System.Windows.Media.Imaging;
using System.IO;
using System.Windows.Forms;
namespace RastrGraphic
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
         // Загрузка bitmap
         FileStream fin = new FileStream(@"e:\Kuvshinka.bmp", FileMode.Open);
         pictureBox1.Image = Image.FromStream(fin);
         long Bytes = fin.Length;
         fin.Close();
         label1.Text = "BitMap " + Bytes.ToString() + " байт";
      }
```

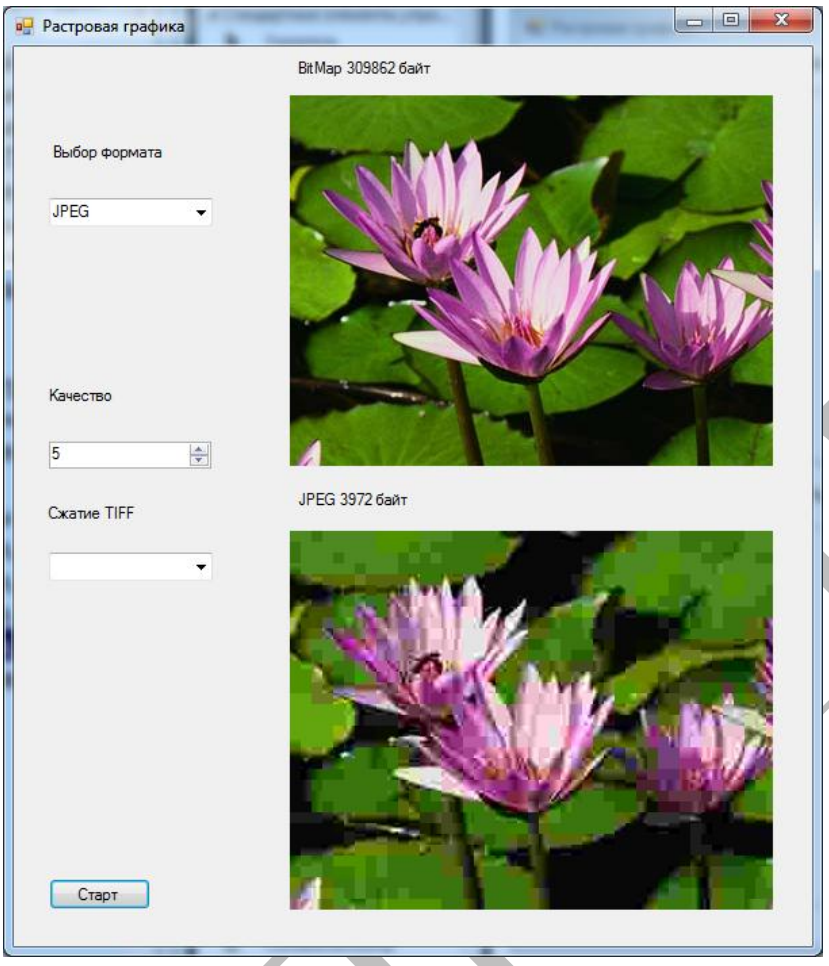
```
private void button1_Click(object sender, EventArgs e)
{
   if ( comboBox1.SelectedItem == "TIFF")
   {
      FileStream fin = new FileStream(@"e:\Kuvshinka.bmp", FileMode.Open);
      FileStream fout = new FileStream(@"e:\Kuvshinka.tiff", FileMode.Create);
      TiffBitmapEncoder encTiff = new TiffBitmapEncoder();
      if (comboBox2.SelectedItem == "Default")
      {
         encTiff.Compression = TiffCompressOption.Default;
      }
      if (comboBox2.SelectedItem == "None")
      {
         encTiff.Compression = TiffCompressOption.None;
      }
      if (comboBox2.SelectedItem == "Rle")
      {
         encTiff.Compression = TiffCompressOption.Rle;
      }
      if (comboBox2.SelectedItem == "Lzw")
      {
         encTiff.Compression = TiffCompressOption.Lzw;
      }
      if (comboBox2.SelectedItem == "Ccitt3")
      {
         encTiff.Compression = TiffCompressOption.Ccitt3;
      }
      if (comboBox2.SelectedItem == "Ccitt4")
      {
         encTiff.Compression = TiffCompressOption.Ccitt4;
      }
      if (comboBox2.SelectedItem == "Zip")
      {
         encTiff.Compression = TiffCompressOption.Zip;
      }
      encTiff.Frames.Add(BitmapFrame.Create(fin));
      encTiff.Save(fout);
      fin.Close();
      long Bytes = fout.Length;
      pictureBox2.Image = Image.FromStream(fout);
      fout.Close();
      label2.Text = "TIFF " + Bytes.ToString() + " байт";
   }
   if (comboBox1.SelectedItem == "GIF")
```

```
            {
                FileStream fin = new FileStream(@"e:\Kuvshinka.bmp", FileMode.Open);
                FileStream fout = new FileStream(@"e:\Kuvshinka.gif", FileMode.Create);
                GifBitmapEncoder encGif = new GifBitmapEncoder();
                encGif.Frames.Add(BitmapFrame.Create(fin));
                encGif.Save(fout);
                fin.Close();
                long Bytes = fout.Length;
                pictureBox2.Image = Image.FromStream(fout);
                fout.Close();
                label2.Text = "GIF " + Bytes.ToString() + " байт";
            }
            if (comboBox1.SelectedItem == "PNG")
            {
                FileStream fin = new FileStream(@"e:\Kuvshinka.bmp", FileMode.Open);
                FileStream fout = new FileStream(@"e:\Kuvshinka.png",
              FileMode.Create);
                PngBitmapEncoder encPng = new PngBitmapEncoder();
                encPng.Frames.Add(BitmapFrame.Create(fin));
                encPng.Save(fout);
                fin.Close();
                long Bytes = fout.Length;
                pictureBox2.Image = Image.FromStream(fout);
                fout.Close();
                label2.Text = "PNG " + Bytes.ToString() + " байт";
            }
            if (comboBox1.SelectedItem == "JPEG")
            {
                FileStream fin = new FileStream(@"e:\Kuvshinka.bmp", FileMode.Open);
                FileStream fout = new FileStream(@"e:\Kuvshinka.jpeg",
              FileMode.Create);
                JpegBitmapEncoder encJpeg = new JpegBitmapEncoder();
                encJpeg.QualityLevel = (int)numericUpDown1.Value;
                encJpeg.Frames.Add(BitmapFrame.Create(fin));
                encJpeg.Save(fout);
                fin.Close();
                long Bytes = fout.Length;
                pictureBox2.Image = Image.FromStream(fout);
                fout.Close();
                label2.Text = "JPEG " + Bytes.ToString() + " байт";
            }
        }
    }
}
```

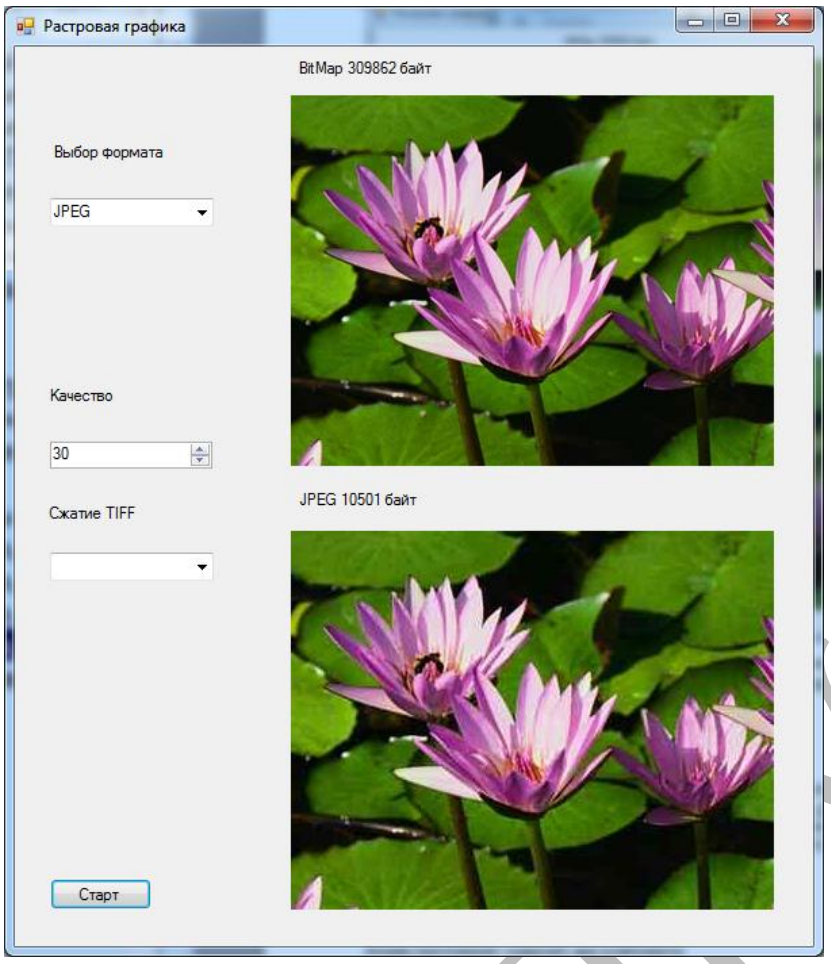
При запуске проекта на выполнение появляется окно, в котором преобразованной картики нет, а для исходника отображается тип и размер файла.



Пусть выбран тип JPEG с качество 5. При нажатии на кнопку Старт формируется результат – выводится картинка, а в поле результата прописывается тип картинки и размер файла.



Картинка получилась со сжатием почти в 100 раз, но с плохим качеством. Изменим качество на 30 и повторим преобразование. Результат с хорошим качеством, но сжатие стало меньше, около 30 раз.


Растровая графика
BitMap 309862 байт
Выбор формата
JPEG
Качество
30
JPEG 10501 байт
Сжатие TIFF
Старт

27. Графика, анимация

Предмет исследований

- Графические средства C#.
- Средства анимации.

Контрольные вопросы

1. Анимация.
2. Принцип создания движения.
3. Как стирается текущий объект.
4. Назначение компонентв Timer
5. От чего зависит скорость движения объекта

Задание. Создать программу демонстрации анимации: движение самолета на неба. Проект – WindowsForm приложение. В программе используются два файла растровых картинок, которые надо загрузить в ту же папку, где находится проект приложения:

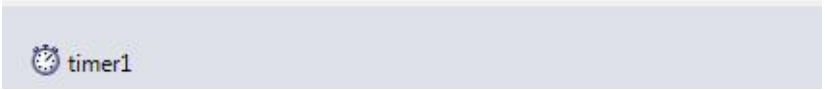
- фон - sky, файл sky.bmp,
- движущийся объект - самолет, файл plane.bmp.

Эти файлы находятся в папке УМД к ЛР.

При анимации осуществляются действия:

- В форму загружается фоновая картинка.
- В стартовой позиции накладывается самолет.
- Через интервал времени, задаваемый встроенным в форму таймером, самолет стирается.
- Вычисляется новые координаты.
- По ним накладывается самолет в новой позиции.

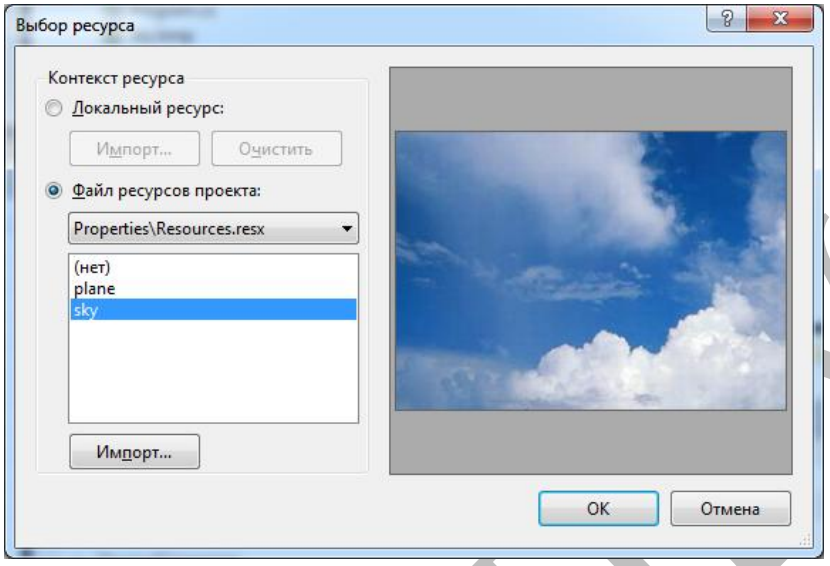
Пример. Создаем форму приложения.



Она содержит два компонента:

- Timer1 – таймер для определения времени, влияющего на скорость полета самолета. Время задается в свойстве таймера Интервал.
- Form1 – сама форма для отображения картинки. В ее свойствах задаем text = Полет. В форме будет отображаться фон – небо. В проект включаем файл sky.bmp. В окне свойств формы определим свойство BackgroundImage. Запускаем браузер выьора ресурса, в котором выберем файл sky.bmp из ресурсов проекта.

Скорость полета самолета определяем, как произведение расстояния, пройденного за один интервал таймера, на длительность интервала.



Листинг программы

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

namespace Plane
{
    public partial class Form1 : Form
    {
        Bitmap sky = new Bitmap(@"G:\sky.bmp");        // небо
        Bitmap plane = new Bitmap(@"G:\plane.bmp");  // самолет
        Graphics g;       // рабочая графическая поверхность
        int dx;    // приращение координаты X,определяет скорость полета
        Rectangle rct;    // область, в которой находится самолет
        Boolean demo = true;    // true - самолет скрывается в облаках

        public Form1()
        {
            InitializeComponent();

            plane.MakeTransparent();    // прозрачный фон у самолета
```

```
    // задать размер формы в соответствии с размером sky
    this.ClientSize = new System.Drawing.Size(
        new Point(BackgroundImage.Width,
            BackgroundImage.Height));
    // будем использовать BackgroundImage формы
    g = Graphics.FromImage(BackgroundImage);

    // исходное положение самолета
    rct.X = -40;
    rct.Y = 20;
    rct.Width = plane.Width;
    rct.Height = plane.Height;

    // скорость полета
    dx = 2;       // скорость полета - 2 пикселя/тик_таймера
    timer1.Interval = 20;
    timer1.Enabled = true;
}

private void timer1_Tick(object sender, EventArgs e)
{
    // стираем самолет копированием фона на рабочую поверхность
    g.DrawImage(sky,new Point(0,0));

    // изменяем положение самолета
    if (rct.X < this.ClientRectangle.Width) rct.X += dx;
    else
    {
        // если граница, задаем заново положение самолета
        rct.X = -40;
        rct.Y = 20;
    }

    // рисуем самолет на рабочей поверхности
    g.DrawImage(plane, rct.X, rct.Y);

    // Метод Invalidate(rct) - перерисовка области rct
    if ( ! demo ) this.Invalidate(rct); // обновить область, где самолет
    else
    {
        // если объект вне области rct, он не виден
        Rectangle reg = new Rectangle(20,20,sky.Width - 40,
            sky.Height - 40);
        // показать обновляемую область
```

```
                g.DrawRectangle(Pens.Black,reg.X ,reg.Y ,
                    reg.Width-1, reg.Height-1);
                this.Invalidate(reg);   // обновить область
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
        }
    }
}
```

При запуске видим летящий самолет.

28. Приложения

28.1. Класс System.Math

Поля класса

Вызов	Функция
Math.E	Значение свойства E примерно равно 2,718.
Math.LN10	Значение свойства LN10 примерно равно 2,302.
Math.LN2	Значение свойства LN2 примерно равно 0,693.
Math.LOG10E	Свойство LOG10E (константа) приблизительно равно 0,434.
Math.LOG2E	Значение свойства LOG2E (константа) приблизительно равно 1,442.
Math.SQRT1_2	Свойство SQRT1_2 (константа) приблизительно равно 0,707.
Math.SQRT2	Свойство SQRT2 (константа) приблизительно равно 1,414.
Math.PI	Свойство PI является константой, приблизительно равной 3,14159.

Методы класса. Имена с заглавной буквы.

Вызов	Функция
Abs(x)	Абсолютное значение
Acos(x)	Обратный косинус
Asin(x)	Обратный синус
Atan(x)	Обратный тангенс
Atan2(x,y)	Обратный тангенс. Atan(x/y)
BigMul(x,y)	Умножает два 32-битовых числа.
Ceiling(x)	Округление вверх
Cos(x)	Косинус
Cosh(x)	Косинус гиперболический
DivRem(x,y)	Остаток от x/y, числа целые
Exp(x)	Экспонента = e^x
Floor(x)	Округление вниз
IEEERemainder(x,y)	Остаток от x/y, числа вещественные
Log(x)	Натуральный логарифм
Log(x,y)	Логарифм от x по основанию y
Log10(x)	Логарифм от x по основанию 10
Max(x,y)	Максимальное из двух
Min(x,y)	Минимальное из двух
Pow(x,y)	Возводит x в любую степень y
Round(x)	Округление до ближайшего целого
Sign(x)	Знак числа
Sin(x)	Синус

Sinh(x)	Синус гиперболичгский
Sqrt(x)	Квадратный корень
Tan(x)	Тангенс
Tanh(x)	Тангенс гиперболичгский
Truncate(x)	Отсечение дробной части

28.2. Класс System.Console

Методы класса

Вызов	Функция
Beep()	Гудок в консоли
Beep(f,t)	Гудок в консоли с частотой f в течение t секунд
Clear	Стирание буфера консоли
Read()	Чтение символа
ReadLine()	Чтение строки символов
Write(String, O1,…,O4)	Вывод в консоль текстового представления объектов Ob (до 4-ех) в формате String
WriteLine(String, O1,…,O4)	То же самое и перевод строки

28.3. Класс System.String

Методы класса

Вызов	Действие
Clone()	Возвращает ссылку на экземпляр класса
Copy(str)	Копирование строки str
Concat(strA, strB)	Сцепление строки strA со строкой strB
Compare(strA, indA, StrB, indB)	Сравнивает подстроки строк strA strB в позициях indA и indB
strA.CompareTo(strB)	Сравнивает строку strA со строкой strB
Replace(strA, strB)	Заменяет строку strA на строку strB
Remove(Ind, Count)	Удаляет Count знаков после позиции Ind
strA.Insert(Ind, strB)	Вставляет строку strB в строку strA с позиции ind
Equals(strA, strB)	Проверка совпадения строк strA и strB
ToCharArray(str)	Возвращает массив символов строки str
str.GetHashCode()	Возвращает хэш-код для этой строки
str.Length	Возвращает число знаков в str
str.ToLower()	Копия str в нижнем регистре
str.ToUpper()	Копия str в верхнем регистре