Министерство образования и науки Российской Федерации
Псковский государственный университет

Кабаченко В.В., Хмылко О.Н.

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА C#. КОНСОЛЬНЫЕ ПРИЛОЖЕНИЯ

Учебно-методическое пособие

Печатается по решению кафедры прикладной информатики в образовании Псковского государственного университета

ПСКОВ
Издательство ПсковГУ
2015

УДК 002.001.8
ББК 32.973.26-018.1я73
К12

Печатается по решению кафедры прикладной информатики в образовании Псковского государственного университета

Рецензенты:
- Моркин Сергей Александрович, доцент кафедры информационных технологий Новгородского государственного университета им. Ярослава Мудрого
- В.Г. Соловьев, профессор кафедры физики ПсковГУ, д. ф.-м. наук

Кабаченко **В.В, Хмылко О.Н.**

К12 Основы программирования на C#. Консольные приложения: Учебно-методическое пособие. – Псков: Издательство ПсковГУ, 2015.- 180с.

Учебно-методическое пособие предназначено для студентов физико-математического факультета и служит методической поддержкой курсов: «Программирование», «Технологии программирования и работы на ЭВМ», «Информатика и программирование». В пособии рассмотренны теоретические основы програмированнния и представленны 10 лабораторных работ по созданию консольных приложений на языке программирования C#.

УДК 002.001.8
ББК 32.973.26-018.1я73

Содержание

Лабораторная работа № 1

Краткое описание:

- знакомство с Microsoft Visual C#
- создание консольного приложения
- ввод и вывод на Консоль

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение. Рис.1

Рисунок 1. Начало работы.

Основные компоненты стандартного запуска

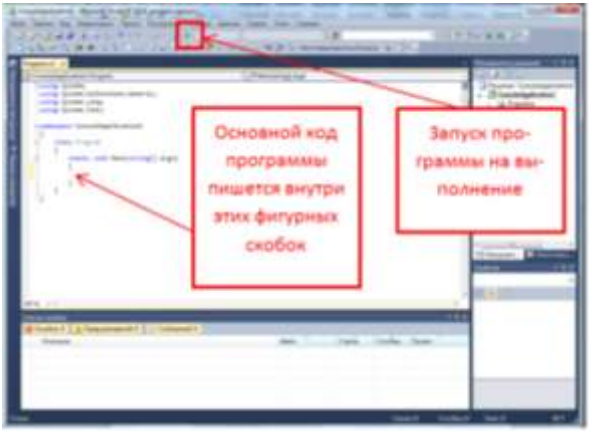


Рисунок 2. Автоматически сгенерированный стартовый код

Ввод данных с Консоли, необходимых для работы программы:

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

Одновременно с вводом данные присваиваются переменной:
```
n = Console.ReadLine();
```

Все данные, получаемые с Консоли строкового типа
```
string n = Console.ReadLine();
```

Вывод данных на Консоль:
Вывод одной переменной:
```
Console.WriteLine(n);
```

Вывод текстовой строки:
```
Console.WriteLine("Текст");
```

Вывод текстовой строки и переменной n:
```
Console.WriteLine("Переменная n="+n);
```

Задание 1: Введите код программы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, изменяя переменные и строки вывода.

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

namespace ConsoleApplication1
{
/// <summary>
/// Проект - Приветствие
/// </summary>
```

```
   class Program
   {
/// <summary>
/// Точка входа.
///Запрашивает имя пользователя и выдает приветствие
/// </summary>
/// <param name="args"></param>
      static void Main(string[] args)
      {
/* Так записываются
* многострочные
* комментарии*/
/// Вывод сообщения на Консоль
      Console.Write("Введите ваше имя ");
/// Переменной name типа string присваивается значение, считанное с Консоли
      string name = Console.ReadLine();
///Проверяется условие, если name = пустое множество, то выводится на Консоль "Здравствуй, Мир"
      if (name == "")
      Console.WriteLine("Здравствуй, мир");
///В случае если в переменную name введено Имя, то выводится "Здравствуй, + Имя"
      else
      Console.WriteLine("Здравствуй,     "     +
name);

///Ожидание дополнительного ввода
      Console.ReadLine();
      }
   }
}
```

Если мы вводим с Консоли число, то необходима конвертация

```
string n = Console.ReadLine();
```

конвертируем введенное n в целое 32-битное число
```
int n1=Convert.ToInt32(n);
```

конвертируем введенное n в действительное число типа double
```
double n2 = Convert.ToDouble(n);
```

Задание 2: Рассмотрим простую задачу, записать трехзначное число в обратном порядке.
Например: число 345, записать как 543

Создайте новый проект, наберите код программы (комментарии не обязательно), запустите на выполнение, используя различные исходные данные.

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

namespace ConsoleApplication1
{
  class Program
  {
/// <summary>
/// Пример простейшей программы с решением "напрямую", без классов
/// Ввести число до 1000 и вывести число с цифрами наоборот: 345 -> 543
/// </summary>
/// <param name="args"></param>
    static void Main(string[] args)
    {
    Console.Write("Введите натуральное число до 1000 ");

    string s = Console.ReadLine();
```

```
    int n = Convert.ToInt32(s);
```

/*c1 присваиваем значение последней цифры исходного числа n, вычисляя остаток от деления на 10*/

```
    int c1 = n % 10;
```

/*c3 присваиваем значение первой цифры исходного числа n, выполняя целочисленное деление на 100 */

```
    int c3 = n / 100;
```

/*c2 сначала делим n на 10, а затем вычисляем остаток от деления c2 на 10, получая среднюю цифру числа*/

```
    int c2 = n / 10;
    c2 = c2 % 10;
```

// получаем новое число n

```
    n = c1 * 100 + c2 * 10 + c3;

    Console.WriteLine("Перевернутое    число
{0}", n);
    Console.Read();
    }
  }
}
```

Другой вариант вывода на консоль.

```
Console.WriteLine("Перевернутое  число  {0}",
n);
```

В фигурных скобках перечисляются переменные, которые нужно вывести, по очереди, начиная с 0, после запитой перечисляются все нужные переменные

```
Console.WriteLine("Перевернутое          число
{0},{1},{2},{3}", n, c1, c2, c3);
```

Внутри текстового блока переменные можно выстраивать в нужном порядке

```
Console.WriteLine ("Перевернутое число {0}, цифры числа {1},{2},{3}, из них получается число {0}", n, c1, c2, c3);
```

Данный вывод можно было осуществить и так:

```
Console.WriteLine("Перевернутое число "+ n +" , цифры числа "+ c1 +", "+ c2 + ", "+ c3 + ", из них получается число "+ n);
```

Задание 3. Самостоятельная работа - написать программу, которая спрашивает у пользователя Имя и сколько ему лет и выводит на экран сообщение «Здравствуй, *Петя*. Тебе *100* лет!))» Вместо Петя и 100 соответствующие введенные данные. Предусмотреть вариант, когда пользователь не вводит имя и возраст.

Задание 4. Самостоятельная работа - написать программу, которая по введеному значению, выводит само значение, значение увеличенное на 1, значение увеличенное на 2, и сумму всех трех значений.

Написать отчет по лабораторной работе. Образец отчета см. Приложение.1

Лабораторная работа № 2

Краткое описание:

- основные типы данных
- основные операторы
- математические операторы
- операторы условия

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Основные типы данных

Таблица 1. Типы данных

Логический тип			
Имя типа	**Системный тип**	**Значения**	**Размер**
Bool	System.Boolean	true, false	8 бит
Арифметические целочисленные типы			
Имя типа	**Системный тип**	**Диапазон**	**Размер**
Sbyte	System.SByte	-128 — 127	Знаковое, 8 Бит
Byte	System.Byte	0 — 255	Беззнаковое, 8 Бит
Short	System.Short	-32768 —32767	Знаковое, 16 Бит
Ushort	System.UShort	0 — 65535	Беззнаковое, 16 Бит
Int	System.Int32	≈(-2*10^9 — 2*10^9)	Знаковое, 32 Бит
Uint	System.UInt32	≈(0 — 4*10^9)	Беззнаковое, 32 Бит
Long	System.Int64	≈(-9*10^18— 9*10^18)	Знаковое, 64 Бит
Ulong	System.UInt64	≈(0 — 18*10^18)	Беззнаковое, 64 Бит
Арифметический тип с плавающей точкой			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
Float	System.Single	+1.5*10^-45 - +3.4*10^38	7 цифр
Double	System.Double	+5.0*10^-324 -	15-16 цифр

		+1.7*10^308	
Арифметический тип с фиксированной точкой			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
Decimal	System.Decimal	+1.0*10^-28 - +7.9*10^28	28-29 значащих цифр
Символьные типы			
Имя типа	**Системный тип**	**Диапазон**	**Точность**
Char	System.Char	U+0000 - U+ffff	16 бит Unicode символ
String	System.String	Строка из символов Unicode	
Объектный тип			
Имя типа	**Системный тип**	**Примечание**	
Object	System.Object	Прародитель всех *встроенных* и пользовательских типов	

Основные арифметические операции

Оператор	Действие
+	Сложение
-	Вычитание, унарный минус
*	Умножение
/	Деление
%	Деление по модулю
--	Декремент
++	Инкремент
= *= /= %= += -= <<= >>= &= ^= |=	Присваивание

Использование математических функций

Math - математические функции
Класс Math из пространства имен System содержит статические методы для работы с тригонометрическими, логарифмическими и другими математическими функциями. Практически все функции перегружены и принимают различные типы параметров, например: double, float, int и т.д.
Рассмотри наиболее часто используемые:
Abs Метод возвращает модуль переданного в параметр числа
Acos Возвращает косинус угла
Asin Возвращает синус угла
Ceiling() Возвращает наименьшее целое число больше переданного в параметре. Например при передачи 2.1 будет возвращено 3
Exp Возвращает число E в степени (Степень передается в качестве параметра)
Log Возвращает логарифм. Можно указать основание логарифма.
Log10 Возвращает 10-ный логарифм
Max Возвращает максимум между двумя переданными параметрами
Min Возвращает минимум между двумя переданными параметрами
Pow Возвращает степень числа. Первый параметр число, второй - степень.
Round Округляет число.
Sqrt Возвращает корень числа переданного в качестве параметра.

В классе также есть два свойства возвращающие константы для числа **E** и числа **PI**

```
double x, y, z;
x=-10.2;
z=0.75;
```

```
y=Math.Abs(x); //Возвращает модуль числа значение
Console.WriteLine("Модуль числа =
{0,10:f3}",y);
y=Math.Acos(z);  // Возвращает косинус угла
Console.WriteLine("Косинус угла в радианах =
{0,10:f3} ", y);
y=Math.Asin(z); //  Возвращает синус угла
Console.WriteLine("Синус угла в радианах =
{0,10:f3}" , y);
y=Math.Ceiling(x);  //Возвращает наименьшее целое
число больше переданного в параметре
Console.WriteLine("Наименьшее, больше задан-
ного =  {0,10:f3}" ,y);
y=Math.Exp(x); //  Возвращает число E в степени (Сте-
пень передается в качествепараметра)
Console.WriteLine("e в степени x =
{0,10:f3}", y);
y=Math.Log(z);  //Возвращает логарифм. Можно ука-
зать основание логарифма
Console.WriteLine("Логарифм Z = {0,10:f3}"
,y);
y=Math.Log10(z);  // Возвращает 10-ный логарифм
Console.WriteLine("Десятичный логарифм Z =
{0,10:f3}" ,y);
y=Math.Max(x,z); //  Возвращает максимум между дву-
мя переданными параметрами
Console.WriteLine("Максимальное из двух чи-
сел =  {0,10:f3} " , y);
y=Math.Min(x,z); //  Возвращает минимум между дву-
мя переданными параметрами
Console.WriteLine("Минимальное из двух чисел
=  {0,10:f3}" , y);
y=Math.Pow(x,3);  // Возвращает степень числа. Пер-
вый параметр число, второй – степень.
Console.WriteLine("x в кубе =  {0,10:f3}",
y);
y=Math.Round(x); // Округляет число
```

```
Console.WriteLine("Округлить до целого = {0,10:f3}" , y);
```
y=Math.Sqrt(z); // Возвращает корень числа переданного в качестве параметра.
```
Console.WriteLine("Корень числа ={0,10:f3}" ,y);
```

Алгоритмы ветвления

1. Одиночный выбор:

Если дождь, то взять зонт и надеть сапоги

```
if (x=="дождь")
{
Console.Write("берем зонт ");
Console.WriteLine("и обуваем сапоги, т.к. на улице {0}", x);
}
```

2. Двойной выбор.

Если переменная больше 5, то увеличить ее в 10 раз, иначе уменьшить втрое.

```
if (y > 5)
y *= 10;
else
y /= 3;
```

3. Множественный выбор

В зависимости от введенного знака операции, выполнить сложение, вычитание, умножение или деление

```
switch (k)
{
case "*":
z = x * y;
Console.WriteLine("Умножение - Ответ: "+z);
break;
case "/":
z = x / y;
```

```
Console.WriteLine("деление - Ответ: "+z);
break;
case "-":
z = x - y;
Console.WriteLine("вычитание- Ответ: "+z);
break;
case "+":
z = x + y;
Console.WriteLine("Сложение - Ответ: "+z);
break;
default:
Console.WriteLine("Ошибка");
break;
}
```

Проверка условий, условные операторы.

Таблица 2. Условные операторы

Оператор	Значение		Оператор	Значение
= =	Равно		&	И
!=	Не равно		|	ИЛИ
>	Больше		^	Исключающее ИЛИ
<	Меньше		&&	Укороченное И
>=	Больше или равно		||	Укороченное ИЛИ
<=	Меньше или равно		!	НЕ

Задание 5. Решение квадратного уравнения (a≠0) без применения Классов.
Создайте новый проект, введите код программы, запустите на выполнения и проверьте работу программы для разных входных данных.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    // коэффициенты уравнения
    double a, b, c;
a=Convert.ToDouble(Console.ReadLine());
b=Convert.ToDouble(Console.ReadLine());
c=Convert.ToDouble(Console.ReadLine());
    //  вычисление дискриминанта
    double D;
    D=Math.Pow(b, 2) - 4 * a * c ;
    // метод: вычисление количества корней
    if (D > 0){
    Console.WriteLine("2 корня");
    double d = Math.Sqrt(D);
    double x1 = (-b + d) / 2 / a;
    double x2 = (-b - d) / 2 / a;
    Console.WriteLine("x1 = " +
    x1.ToString() + "  x2 = " +
    x2.ToString());
    }
    else
      if (D == 0){
```

```
        Console.WriteLine("1 корень");
        Console.WriteLine("x = " + (-b /
        2 / a).ToString());
        }
        else Console.WriteLine("нет
         корней");

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

Самостоятельная работа

Задачи по теме: Условия

1. Решить линейное уравнение ax + b =0. Рассмотреть все возможные случаи: 1) a#0 2) a=0, b=0 3) a=0, b#0
2. Решить квадратное уравнение (a может быть равно 0, b#0)
3. Решить квадратное уравнение (рассмотреть все возможные случаи)
4. Даны координаты точки на плоскости (x,y). Определить, в каком квадранте находится точка
5. Дано натуральное число n < 1000. Определить количество цифр в числе
6. Даны стороны треугольника a, b, c. Определить, вид треугольника: а) равносторонний б) равнобедренный в) разносторонний
7. Определить, поместится ли квадрат со стороной a в круг с радиусом r
8. Даны три точки с координатами (x1, y1), (x2, y2), (x3, y3). Определить, какая из них расположена ближе к началу координат
9. Введены два числа a и b и одно из действий (+, —, *, /). Выполнить указанное действие над введенными числами.

10. Даны стороны треугольника a, b, c. Определить вид треугольника: а) остроугольный б) прямоугольный в) тупоугольный. Указание: Первоначально необходимо найти максимальную из сторон. Затем проверить для нее выполнение соотношения c2 > a2 + b2. Если оно выполняется, то треугольник тупоугольный. Если это равенство, прямоугольный. В ином случае - остроугольный.
11. Дано натуральное число n < 1000. Получить из него новое число такое, чтобы цифры в нем располагались по возрастанию. Указание: первоначально необходимо выделить все цифры числа. Рассмотреть случаи однозначных, двухзначных и трехзначных чисел
12. Введен символ. Определить, какой это символ: буква, цифра, знак операции, другое.

13. Введено целое число. Если это цифра, вывести ее словесное обозначение (т.е. если введено 9, должно быть выведено девять). В противном случае вывести число в неизменном виде.

14. Даны числа A, B, C. Удвоить эти числа если A>B>=C. И заменить их абсолютными значениями если это не так.

15. Даны x и y. Если x<0 и y<0 , то каждое значение заменить модулем, если отрицательное только одно из них, то оба значения увеличить на 0,5, если оба не отрицательны и ни одно из них не принадлежит интервалу (0,5; 2), то оба значения уменьшить в 10 раз. В остальных случаях x и y без изменений

Написать отчет по лабораторной работе.
Образец отчета см. Приложение.1

Лабораторная работа № 3

Краткое описание:

- Понятие цикла
- Виды циклов

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение
Цикл – многократное повторение однотипных действий, событий или состояний.
Циклы могут повторяться фиксированное число раз, или завершаться при выполнении определенного условия.

1. Циклы с предусловием

Цикл с предусловием — цикл, который выполняется пока истинно некоторое условие, указанное перед его началом. Это условие проверяется до выполнения тела цикла, поэтому тело может быть не выполнено ни разу (если условие с самого начала ложно). В большинстве процедурных языков программирования реализуется оператором while, отсюда его второе название — while-цикл.

```
while (Условие)
{

Тело цикла

}
```

```
while (x<0)
{
  y*=4;
  z += 10;
  x += 10;
}
```

2. Циклы с постусловием

Цикл с постусловием — цикл, в котором условие проверяется после выполнения тела цикла. Отсюда следует, что тело всегда выполняется хотя бы один раз.

```
do
{
Тело цикла

} while (Условие);
```

```
do  {
   y *= 4;
   z += 10;
   x += 10;
} while (x < 0);
```

3. Циклы со счетчиком

Цикл со счётчиком — цикл, в котором некоторая переменная изменяет своё значение от заданного начального значения до конечного значения с некоторым шагом, и для каждого значения этой переменной тело цикла выполняется один раз. В большинстве процедурных языков программирования реализуется оператором for, в котором указывается счётчик (так называемая «переменная цикла»), требуемое количество проходов (или граничное значение счётчика) и, возможно, шаг, с которым изменяется счётчик.

```
for(начальное значение; условие; прира-
щение)
{
Тело цикла
}
```

```
for (int i = 25; i < 125; i+=5)
{
    s = s + i;
    skv = skv + Math.Pow(i, 2);
}
```

4. Вложенные циклы

Существует возможность организовать цикл внутри тела другого цикла. Такой цикл будет называться вложенным циклом. Вложенный цикл по отношению к циклу в тело которого он вложен будет именоваться внутренним циклом, и наоборот цикл в теле которого существует вложенный цикл будет именоваться внешним по отношению к вложенному. Внутри вложенного цикла в свою очередь может быть вложен еще один цикл, образуя следующий уровень вложенности и так далее. Количество уровней вложенности, как правило, не ограничивается.

Задание 6. Пример программы, в которой использованы все циклы.

Создать новый проект, набрать код программы, запустить программу на выполнение, используя разные варианты входных данных.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    int x, z, f;
    double y,k,s, skv;
```

//Посчитать факториал числа n

```
    Console.WriteLine("Введите     число
для определения факториала");
Int n=Convert.ToInt32 (Console.ReadLine());
    f = 1;
    x = 1;
    while (x<=n)
    {
      f=f*x;
      x +=1;
    }
    Console.WriteLine("Факторил    числа
{0} равен {1}",n,f);
    Console.ReadLine();
```

//Сумма кубов четных чисел меньших половины заданного z

```
    Console.WriteLine("Введите z");
  z = Convert.ToInt32(Console.ReadLine());
    k = 2;
    s = 0;
    int b = 0;
    do
    {
      if (k%2==0) {
      y = Math.Pow(k,3);
      s=s+y;
      b += 1;
      }
      k += 1;
    } while (k < z/2);

    Console.WriteLine("Сумма кубов пер-
вых {0} четных чисел меньших {1} равна
{2}", b, z/2, s);
    Console.ReadLine();
```

//Сумма чисел от 25 до 125 кратных 5, и сумма квадратов этих чисел

```
    s = 0;
    skv = 0;
    for (int i = 25; i < 125; i+=5)
    {
       s = s + i;
       skv = skv + Math.Pow(i, 2);
    }
    Console.WriteLine("Сумма чисел от
25 до 125 кратных 5 равна {0}, \nсумма
квадратов равна  {1}", s, skv);
    Console.ReadLine();
```

/*Даны натуральные числа n, m. Получить все меньшие n натуральные числа, сумма цифр которых равен m. */

```
    Console.WriteLine("Введите n");
n=Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("Введите m");
int m=Convert.ToInt32(Console.ReadLine());
    int u, v;

    for (int i = 0; i < n; i++) {
      s = 0;
      u=i;
      while (u>0)
      {
        v=u%10;
        s=s+v;
        u = u / 10;
      }
      if(s==m) Console.WriteLine("Число
      {0}, сумма цифр равна {1}",i, m);
    }

    Console.ReadLine();
    }
  }
```

}

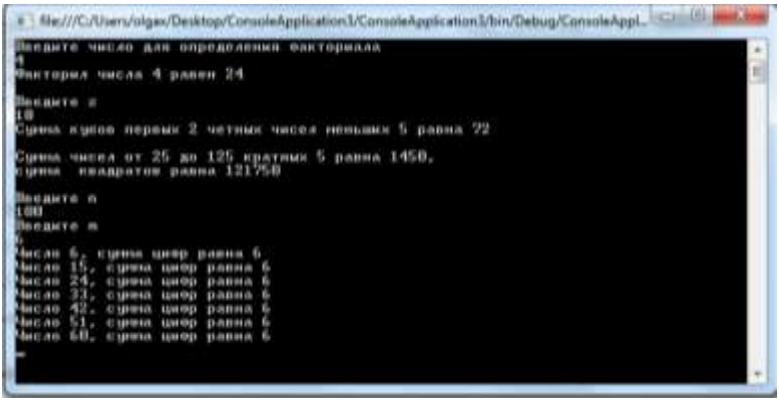


Рисунок 3. Вариант вывода результата на Консоль

Самостоятельная работа

Циклы с предусловием

1. Найти сумму цифр числа.
2. Приписать по 1 в начало и в конец записи числа n. Например, было n=3456, стало n=134561.
3. Поменять местами первую и последнюю цифры числа.
4. Поменять порядок цифр числа на обратный. Например, было 12345, стало 54321.
5. Найти количество чётных цифр целого положительного числа.
6. Найти самую большую цифру целого числа.
7. Найти сумму цифр целого числа, больших 5.
8. Сколько раз данная цифра встречается в целом числе?
9. Составить программу, проверяющую, является ли последовательность из 10 целых чисел, вводимых с клавиатуры, возрастающей.
10. Составить программу, проверяющую, является ли заданное натуральное число палиндромом, то есть

таким, десятичная запись которого читается одинаково слева направо и справа налево.

Циклы с постусловием

1. Произведение максимального количества N первых нечетных чисел не больше *p*. Вывести последний сомножитель , общее количество сомножителей и произведение.
2. Числа Фибоначчи (f_n) определяется формулами: $f_0 = f_1 = 1$; $f_n = f_n = f_{n-1} + f_{n-2}$ при $n=2, 3,...$ Составить программу определения *f* - 40-е число Фибоначчи.
3. Числа Фибоначчи (f_n) определяется формулами: $f_0 = f_1 = 1$; $f_n = f_n = f_{n-1} + f_{n-2}$ при $n=2, 3,...$ Составить программу поиска *f* - первого числа Фибоначчи, большего *m (m>1).*
4. Числа Фибоначчи (f_n) определяется формулами: $f_0 = f_1 = 1$; $f_n = f_n = f_{n-1} + f_{n-2}$ при $n=2, 3,...$ Составить программу вычисления *s* - суммы всех чисел Фибоначчи, которые не превосходят 1000
5. Составить программу, проверяющую, является ли заданное натуральное число совершенным, то есть равным сумме своих положительных делителей, кроме самого этого числа.
6. Дана непустая последовательность натуральных чисел, за которой следует 0. Вычислить сумму положительных элементов последовательности, порядковые номера которых нечетны.
7. Составить программу, проверяющую, является ли заданное натуральное число палиндромом, то есть таким, десятичная запись которого читается одинаково слева направо и справа налево.
8. Найти количество нечётных цифр целого положительного числа.
9. Найти первую цифру числа.
10. Подсчитать сумму квадратов первых нечетных чисел меньших заданного Z.

Циклы со счетчиком

1. Составить программу возведения натурального числа в квадрат, используя следующую закономерность:

   $1^2 = 1$
   $2^2 = 1 + 3$
   $3^2 = 1 + 3 + 5$
   $4^2 = 1 + 3 + 5 + 7$
   ....
   $n^2 = 1 + 3 + 5 + 7 + 9 + \mathbf{...} + 2n\text{-}1$
2. Определить количество трехзначных натуральных чисел, сумма цифр которых равна заданному числу N.
3. Составить программу вычисления суммы кубов чисел от 25 до 125.
4. Среди двузначных чисел найти те, сумма квадратов цифр которых делится на 13. Ответ: .
5. Написать программу поиска двузначных чисел, таких, что если к сумме цифр этого числа прибавить квадрат этой суммы, то получится это число.
6. Квадрат трехзначного числа оканчивается тремя цифрами, которые как раз и составляют это число.
7. Написать программу поиска четырехзначного числа, которое при делении на 133 дает в остатке 125, а при делении на 134 дает в остатке 111. Ответ: 1987.
8. Найти сумму положительных нечетных чисел, меньших 100.
9. Найти сумму целых положительных чисел из промежутка от A до B, кратных 4 (значения переменных A и B вводятся с клавиатуры).
10. Найти сумму целых положительных чисел, больших 20, меньших 100, кратных 3 и заканчивающихся на 2, 4 или 8.

Вложенные циклы

1. Исходное данное - натуральное число q, выражающее площадь. Написать программу для нахождения всех таких прямоугольников, площадь ко-

торых равна *q* и стороны выражены натуральными числами.

2. Составить программу для графического изображения делимости чисел от *1* до *n* (*n* - исходное данное). В каждой строке надо печатать число и сколько плюсов, сколько делителей у этого числа. Например, если исходное данное - число 4, то на экране должно быть напечатано:

   1+
   2++
   3++
   4+++

3. Составить программу получения всех совершенных чисел, меньших заданного числа *n*. Число называется **совершенным**, если равно сумме всех своих положительных делителей, кроме самого этого числа. Например, 28 - совершенно, так как 28=1+2+4+7+14.
4. *Из истории*. Грекам были известны первые четыре совершенных числа: 6, 28, 496, 8128. Эти числа высоко ценились. Даже в XII веке церковь утверждала, что для спасения души необходимо найти пятое совершенное число. Это число было найдено только в XV веке. До сих пор совершенные числа полностью не исследованы - не известно, имеется ли конечное число совершенных чисел или их число бесконечно, кроме того, неизвестно ни одного нечётного совершенного числа, но и не доказано, что таких чисел нет.
5. Дано натуральное число *n*. Можно его представить в виде суммы трёх квадратов натуральных чисел? Если можно, то:

a. указать тройку *x, y, z,* таких натуральных чисел, что $x^2 + y^2 + z^2 = n$;
b. указать все тройки *x, y, z* таких натуральных чисел, что $x^2 + y^2 + z^2 = n$.

6. Найти натуральное число от 1 до 10000 с максимальной суммой делителей.

7. Даны натуральные числа *a, b* (a<b). Получить все простые числа *p*, удовлетворяющие неравенствам: a<=p<=b.
8. Даны натуральные числа *n, m*. Получить все меньшие *n* натуральные числа, квадрат суммы цифр которых равен *m*.
9. Даны натуральные числа *n* и *m*. Найти все пары дружественных чисел, лежащих в диапазоне от *n* до *m*. Два числа называются **дружественными**, если каждое из них равно сумме всех делителей другого (само число в качестве делителя не рассматривается).
10. В данном натуральном числе переставить цифры таким образом, чтобы образовалось наименьшее число, записанное этими же цифрами.
11. Составить программу, печатающую для данного натурального числа *k*-ю цифру последовательности:

a. 12345678910..., в которой выписаны подряд все натуральные числа;

b. 14916253649..., в которой выписаны подряд квадраты всех натуральных чисел;

c. 1123581321..., в которой выписаны подряд все числа Фибоначчи.

12. Составить программу возведения заданного числа в третью степень, используя следующую закономерность:

$1^3 = 1$
$2^3 = 3 + 5$
$3^3 = 7 + 9 + 11$
$4^3 = 13 + 15 + 17 + 19$
$5^3 = 21 + 23 + 25 + 27 + 29$

13. Составить программу для нахождения всех натуральных решений уравнения $n^2 + m^2 = k^2$ в интервале [1, 10].

Примечание. Решения, которые получаются перестановкой *n* и *m*, считать совпадающими.

Написать отчет по лабораторной работе.

Лабораторная работа № 4

Краткое описание:

- понятие Класса
- создание простого Класса
- ввод и вывод на Консоль

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Классы – понятие

Если программисту надо передать много взаимосвязанных частей данных функции, может быть сложным использовать только переменные. Например, каждый раз, когда нам нужно передать адрес, было бы утомительным передавать имя, улицу, дом, квартиру, город, страну и регион. Лучше задать структуру данных, которая будет содержать всю информацию вместе, а затем присваивать данным имена, такие как *Address.*

Задание 7. Создать новый проект и набрать код программы.
Для начала нужно добавить класс.

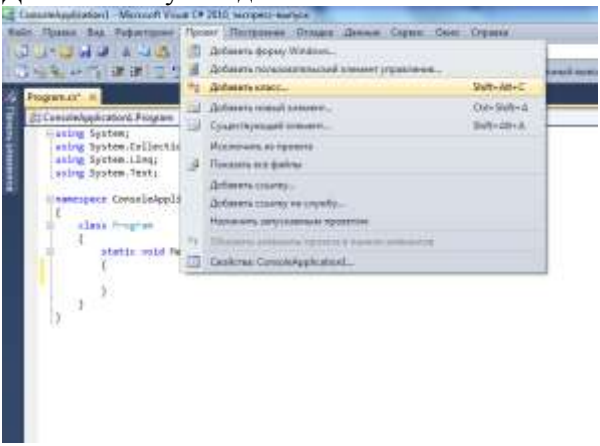


Рисунок 4. Добавление нового класса

В новом окне меняем имя класса на Adress и описываем поля.

```
class Adress
{
public string FIO="";
public string Strana="";
public string Region="";
public string Gorod="";
public string ulitsa = "";
public int dom = 0;
public int kv = 0;
}
```

В коде, приведённом выше, Address является *именем класса*. Переменные в классе, такие, как FIO и ulitsa, называются *атрибутами* или *полями*. Данный код только определят класс.

Необходимо привязать класс к определенным объектам. Класс и объект — это разные вещи, хотя в некоторых случаях они взаимозаменяемы. Класс определяет тип объекта, но не сам объект. Объект — это конкретная сущность, основанная на классе и иногда называемая экземпляром класса.

В **основной программе** необходимо создать новый экземпляр класса Adress:

```
//Определяем тип переменной adr1
Adress adr1;
// Создаем экземпляр класса
adr1=new Adress();
```

Далее заполняем поля определенными значениями, присущими экземпляру adr1

```
adr1.FIO="Иванов И.И.";
adr1.Strana = "Россия";
adr1.Region = "СЗ";
```

```
adr1.Gorod="Псков";
adr1.ulitsa="Ленина";
adr1.dom=2;
adr1.kv=101;
```

Создадим еще один экземпляр adr2

```
Adress adr2; //Определяем тип переменной adr2
adr2=new Adress(); // Создаем экземпляр класса

adr2.FIO="Петров П.П.";
adr2.Strana = "Белорусь";
adr2.Region = "Ц";
adr2.Gorod="Минск";
adr2.ulitsa="Ленина";
adr2.dom=2;
adr2.kv=101;
```

Обработаем событие, по набранной фамилии, выводим адрес либо Иванова, либо Петрова

```
string name = Console.ReadLine();
if (name == "Иванов И.И.")
Console.WriteLine("Адресс Иванова И.И.:
\n{0}, {1} регион, город {2}, улица
{3}, д. {4}, кв. {5}", adr1.Strana,
adr1.Region, adr1.Gorod, adr1.ulitsa,
adr1.dom, adr1.kv);
else if (name == "Петров П.П.")
Console.WriteLine("Адресс Петрова
П.П..: \n{0}, {1} регион, город {2},
улица {3}, д. {4}, кв. {5}",
adr2.Strana, adr2.Region, adr2.Gorod,
adr2.ulitsa, adr2.dom, adr2.kv);
else
Console.WriteLine("Такого человека нет
в базе");
```

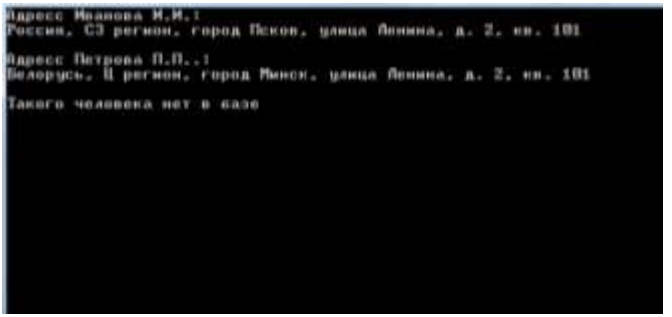


Рисунок 5. Пример работы программы

Задание 8. Пример использования Класса без конструкторов и методов.
Создать новый проект и набрать код программы, проверить программу для разных вариантов: когда тел. номер начинается на 77 или заканчивается на 5.

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

class Tel
{
  public string FIO="";
  public int nom = 0;
}

namespace ConsoleApplication1
{
    class Program
    {
      static void Main(string[] args)
      {
      //Телефонная книга.
      /*Составьте программу,  выдающую список абонентов, имеющих телефонный номер, начинающийся на 33.*/
      Tel t1; //Определяем тип переменной t1
```

```
t1=new Tel(); // Создаем экземпляр класса

t1.FIO = "Иванов";
t1.nom = 111213;
int y1 = t1.nom / 10000;
if (y1 == 33)
Console.WriteLine(t1.FIO + ",    "
+ t1.nom);

Tel t2; //Определяем тип переменной t2
t2 = new Tel(); // Создаем экземпляр
класса

t2.FIO = "Петров";
t2.nom = 222324;
int y2 = t2.nom / 10000;
if (y2 == 33)
Console.WriteLine(t2.FIO + ",    "
+ t2.nom);

Tel t3; //Определяем тип переменной adr1
t3 = new Tel(); // Создаем экземпляр
класса

t3.FIO = "Сидоров";
t3.nom = 333435;
int y3 = t3.nom / 10000;
if (y3 == 33)
Console.WriteLine(t3.FIO + ",    "
+ t3.nom);

Tel t4; //Определяем тип переменной adr1
t4 = new Tel(); // Создаем экземпляр
класса

t4.FIO = "Никитин";
t4.nom = 332324;
int y4 = t4.nom / 10000;
```

```
      if (y4 == 33)
      Console.WriteLine(t4.FIO + ",    "
      + t4.nom);

      Console.ReadLine();

      }
    }
}
```

Конструктор Класса

Для начала кратко обсудим, что такое конструктор вообще. Конструктор - это функция (метод) класса. Раз эта функция, то описываем мы ее почти точно так же, как и любую другую функцию класса - пишем параметры в круглых скобках и т. п. Когда конструктор вызывается? В момент создания переменной. При этом у класса может быть несколько конструкторов - но при этом они должны различаться или типом параметров, или их количеством.

Откройте проект, создающий Класс Adress. Добавьте конструктор:

```
class Adress
{
  public string FIO;
  public string Strana;
  public string Region;
  public string Gorod;
  public string ulitsa;
  public int dom = 0;
  public int kv = 0;

  public Adress(string FIO="", string
  Strana="", string Region="", string
  Gorod="", string ulitsa = "", int dom
  = 0, int kv = 0)
  {
```

```
    this.FIO=FIO;
    this.Strana=Strana;
    this.Region=Region;
    this.Gorod=Gorod;
    this.ulitsa = ulitsa;
    this.dom = dom;
    this.kv = kv;
  }
}
```

Обратите внимание на ряд особенностей конструктора. Во-первых, конструктор называется как класс. Раз наш класс называется Adress, значит и конструктор должен называться точно также. И во-вторых, конструктор, в отличие от других методов класса, не возвращает никакого значения (даже типа void).

Что делает наш конструктор? Он записывает передаваемые в него параметры во внутренние переменные класса. Обратите внимание, что называются они одинаково - kv и kv, FIO и FIO и т.д. Компилятор сначала ищет локальную переменную с таким именем, и, если не находит, то переменную класса. Поэтому kv (и другие) - это передаваемые в конструктор параметры. Если же нам надо сослаться на переменную класса (при существовании переменной с таким же именем, как и передаваемый в функцию параметр), то мы используем ключевое слово this. Оно всегда указывает на текущий экземпляр нашего класса. Таким образом в строчках

Теперь в основной программе заполняем поля без вызова конструктора(adr1) и вызывая конструктор(adr2)

```
Adress adr1; //Определяем тип переменной adr1
adr1=new Adress(); // Создаем экземпляр класса

adr1.FIO="Иванов И.И.";
adr1.Strana = "Россия";
adr1.Region = "СЗ";
```

```
adr1.Gorod="Псков";
adr1.ulitsa="Ленина";
adr1.dom=2;
adr1.kv=101;

//Вызываем конструктор
Adress adr2 = new Adress("Петров
П.П.","Белорусь","Ц","Минск","Ленина",2
,101);

//Вывести оба экземпляра Класса
Console.WriteLine("Адресс Иванова И.И.:
\n{0}, {1} регион, город {2}, улица
{3}, д. {4}, кв. {5}", adr1.Strana,
adr1.Region, adr1.Gorod, adr1.ulitsa,
adr1.dom, adr1.kv);
Console.WriteLine("Адресс Петрова
П.П..: \n{0}, {1} регион, город {2},
улица {3}, д. {4}, кв. {5}",
adr2.Strana, adr2.Region, adr2.Gorod,
adr2.ulitsa, adr2.dom, adr2.kv);

Console.ReadLine();
```

Задание 9. Создание класса для арифметических вычислений

Рассмотрим Программу решения квадратного уравнения с использование класса KvUr

Создать новый проект и набрать код программы, проверить программу для разных вариантов: когда тел. номер начинается на 77 или заканчивается на 5.

Основная программа

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

```
namespace ConsoleApplication1
{
/// <summary>
/// Тестирование решения квадратного уравнения
/// </summary>
   class Program
   {
/// <summary>
/// Стартовая процедура
/// </summary>
/// <param name="args"></param>
    static void Main(string[] args)
    {
    double a, b, c;
//Вводим с консоли коэффициенты квадратного уравнения
    Console.Write("a = ");
// Parse  - преобразует строку в число типа Double аналогично Convert.ToDouble
    a = double.Parse(Console.ReadLine());
    Console.Write("b = ");
    b = double.Parse(Console.ReadLine());
    Console.Write("c = ");
    c = double.Parse(Console.ReadLine());

//Определяем переменную uravn и вызываем конструктор KvUr  с тремя параметрами
    KvUr uravn = new KvUr(a,b,c);

//Выводим количество корней, вызывая метод Count класса KvUr
    Console.WriteLine("Количество  решений уравнения {0}",uravn.CountK());
//Выводим корни, вызывая метод Out класса KvUr
    Console.WriteLine("Решения  уравнения {0}", uravn.Out());
    Console.ReadLine();
    }
```

```
    }
}
```

Класс KvUr

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

namespace ConsoleApplication1
{
/// <summary>
/// Класс, решающий квадратное уравнение
/// </summary>
    class KvUr
    {
// поля класса
private double a, b, c; // коэффициенты уравнения
// конструктор
    public KvUr(double a, double b, dou-
    ble c) {
    this.a = a;
    this.b = b;
    this.c = c; }

// метод: вычисление дискриминанта
    private double Discr()
    { return Math.Pow(b, 2) - 4 * a * c;
    } //Внутренний метод Класса, возвращает(return) значение дискриминанта

    // метод: вычисление количества корней
    public int CountK()
    {
    double d = Discr(); //вызывает метод вычисления дискриминанта
```

```
    if (d > 0) return 2;          //Возвращает
    соответственно 2, 1, 0 решений
    else if (d == 0) return 1;
    else return 0;
    }

    // метод: вывод корней уравнения
    public string Out()
    {
    int n = CountK(); //Вызов метода количества
    решений
    if (n == 0) return "Нет корней";
    else
    if (a == 0) return "Вырожденное квад-
    ратное уравнение";
    else
    if (n == 1) return "x = " + (-b / 2 /
    a).ToString();
    else
    {
    double d = Math.Sqrt(Discr());
    double x1 = (-b + d) / 2 / a;
    double x2 = (-b - d) / 2 / a;
    return "x1 = " + x1.ToString() + "
    x2 = " + x2.ToString();
    }
    } //Возвращает значения корней уравнения
    }
}
```

Обработка ошибок.

Принимая во внимание, что .NET Framework включает большое количество предопределенных классов исключений, возникает вопрос: как их использовать в коде для перехвата ошибочных условий? Для того чтобы справиться с возможными ошибочными ситуациями в коде C#, программа обычно делится на блоки трех разных типов:

- Блоки **try** инкапсулируют код, формирующий часть нормальных действий программы, которые потенциально могут столкнуться с серьезными ошибочными ситуациями.
- Блоки **catch** инкапсулируют код, который обрабатывает ошибочные ситуации, происходящие в коде блока try. Это также удобное место для протоколирования ошибок.
- Блоки **finally** инкапсулируют код, очищающий любые ресурсы или выполняющий другие действия, которые обычно нужно выполнить в конце блоков try или catch. Важно понимать, что этот блок выполняется независимо от того, сгенерировано исключение или нет.

Try и catch

Основу обработки исключительных ситуаций в C# составляет пара ключевых слов try и catch. Эти ключевые слова действуют совместно и не могут быть использованы порознь. Ниже приведена общая форма определения блоков try/catch для обработки исключительных ситуаций:

```
try
{
// Блок кода, проверяемый на наличие ошибок.
}
catch (Тип ошибки1)
{
// Обработчик исключения для ошибки 1
}
catch (Тип ошибки 2) {
// Обработчик исключения для ошибки 2.
}
...
```

Задание 10. Обработка ошибок.
Задача 1.
Пример программы деления двух чисел с обработкой 2 исключений: деления на ноль и если вместо числа введена буква.
Создать новый проект и набрать код программы, проверить программу для разных вариантов.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    link1:
    try
    {
    Console.Write("Введите x: ");
    int                x                =
    int.Parse(Console.ReadLine());
    Console.Write("Введите y: ");
    int                y                =
    int.Parse(Console.ReadLine());

    int result = x / y;
    Console.WriteLine("Результат:   "   +
    result);
    }
```

// Обрабатываем исключение возникающее при делении на ноль

```
    catch (DivideByZeroException)
    {
```

```
        Console.WriteLine("Деление на 0 за-
        прещено!!!\n");
        goto link1;
        }
```

// Обрабатываем исключение при некорректном вводе числа в консоль

```
        catch (FormatException)
        {
        Console.WriteLine("Это НЕ число!!!\n");
        goto link1;
        }
        Console.ReadLine();
        }
    }
}
```

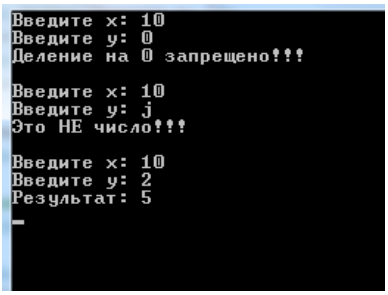


Рисунок 6. Пример работы программы

Все определяемые на уровне пользователя и системы исключения в конечном итоге всегда наследуются от базового класса System.Exception, который, в свою очередь, наследуется от класса System.Object.

Обработка ошибок. Задача 2.

Пример программы обработки ошибок форматов введенных данных:

Создать новый проект и набрать код программы, проверить программу для разных вариантов

Основная программа

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
      Person p = new Person(); // вызов конструктора
      p.Vyvod(); // вывод на экран введенных данных
      Console.ReadLine();
    }
  }
}
```

Класс Person

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

namespace ConsoleApplication1
{
/// <summary>
/// Класс "Человек".
/// Конструктор - иллюстрация методов класса Convert и обработки ошибок
/// </summary>
  class Person
  {
// поля класса (закрытые)
  private string fam; // Фамилия
  private string im; // Имя
```

```
private System.DateTime dr; // Дата рождения Системная операция преобраования данных
private double rost; // рост, м
private int ves; // Вес, кг

// Конструктор по умолчанию. Ввод полей с контролем правильности

public Person()
{
Console.Write("Фамилия ");
fam = Console.ReadLine();

Console.Write("Имя ");
im = Console.ReadLine();

Console.Write("Дата рождения "); //Если Дата введена не в формате даты дд.мм.гг, то обработчик исключений выводит дату 01/01/2001
try
{
string s = Console.ReadLine();
dr = Convert.ToDateTime(s);
}
catch (FormatException)
{
dr = Convert.ToDateTime("01/01/2001");
}

Console.Write("Рост, м ");
try
{
string s = Console.ReadLine(); // надо вводить десятичную запятую!
rost = Convert.ToDouble(s);
}
catch (FormatException)
{
```

```
rost = 1.6;   // если введена буква или символ,
выводится средний рост 1 м 60 см.
}

Console.Write("Вес, кг ");
try
{
string s = Console.ReadLine();
ves = Convert.ToInt32(s);
}
catch (FormatException)
{
ves = 50; // если введена буква или символ, вы-
водится средний вес 50 кг
}
}

// Метод - вывод данных на экран
public void Vyvod()
{
Console.Clear(); // очистить экран
// для нетекстовых переменных автоматически вы-
зывается метод ToString
Console.WriteLine("Фамилия     "     +
this.fam);
Console.WriteLine("Имя " + this.im);
Console.WriteLine("Дата рождения " +
this.dr);
Console.WriteLine("Рост      "      +
this.rost);
Console.WriteLine("Вес " + this.ves);
}
}
}
```

Самостоятельная работа

Простые классы.

1. Опишите, используя Класс, каталог книг в библиотеке. Составьте программу, выдающую список книг А. Дюма, хранящихся в библиотеке.
2. Опишите, используя Класс, таблицу дат и событий русской истории. Составьте программу, выдающую список событий XIX века.
3. Опишите, используя Класс, школьный класс (Фамилия и инициалы, дата рождения, месяц рождения, год рождения). Составьте программу, выдающую список учеников, рожденных в мае месяце.
4. Опишите, используя Класс, записную книжку. Составьте программу, выдающую список друзей, кому в этом году исполняется 20 лет. (Фамилия и инициалы, год рождения, день рождения, месяц рождения).
5. Опишите, используя Класс, школьный класс (Фамилия и инициалы, день рождения, месяц рождения, год рождения). Составьте программу, выдающую день рождения класса (среднее арифметическое дней и месяцев).
6. Опишите, используя Класс, школьную нагрузку (фамилия преподавателя, класс, часы). Определить у какого преподавателя самая большая нагрузка и кого самая низкая.
7. При сдаче норм ГТО, были получены результаты забега на 100 метров и прыжков в длину, задайте нормы ГТО по этим видам, определите списки учеников, не выполнивших нормативы, количество учеников сдавших нормативы, а также списки 3 лучших.
8. Опишите, используя Класс, вступительные экзамены. Абитуриенты сдавали три экзамена, для поступления необходимо набрать 12 баллов. Определите списки абитуриентов, зачис-

ленных в институт, количество не сдавших экзамены, списки абитуриентов сдавших три экзамена на 5.

9. Опишите, используя Класс, данные на учеников (фамилия, улица, дом, квартира). Составьте программу, определяющую сколько учеников живет на улице Свердлова, списки учеников, живущих в доме номер 45.
10. Опишите, используя Класс, почтовую сортировку (город, улица, дом, квартира, кому, ценность). Составьте программу, определяющую: 1) сколько посылок отправлено в г. Москву; 2) сколько и куда (список городов) отправленно посылок ценностью выше 10 рублей; 3) есть ли адреса, куда отправлено более 1 посылки, если есть то сколько и кому.
11. В экзаменационной ведомости можно выделить сведения о ведомости (предмет, номер группы, дата экзамена), сведения о человеке (фамилия, номер зачетной книжки, оценка за экзамен). Определите, сколько человек не сдали информатику, выдать их списки: фамилия, номер группы.
12. Опишите, используя Класс, товар (наименование товара, старая цена, новая цена). Составьте программу, определяющую на какие товары повысятся цены и на сколько процентов.
13. Опишите, используя Класс, время (часы, минуты, секунды). Определите какое время t1 или t2 меньше, увеличьте время T на 1 секунду, вычислите интервал времени, прошедший от времени t1 до времени t2 (t1>t2).
14. В анкетных данных обозначены фамилия, пол, рост. Определите средний рост женщин, фамилию самого высокого мужчины.
15. Дана анкета фамилия, пол, число, месяц, год рождения. Выберите самого старшего мужчину, напечатайте все фамилии, начинающиеся с буквы "Б", и даты рождения этих людей

Лабораторная работа № 5

Краткое описание:

- Класс Random – случайное число
- одномерные массивы
- двумерные массивы

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Класс Random имеет конструктор класса: для того, чтобы вызывать методы класса, нужно вначале создавать экземпляр класса. Этим Random отличается от класса Math, у которого все поля и методы - статические, что позволяет обойтись без создания экземпляров класса Math.

Перегруженный метод **public int Next()** при каждом вызове возвращает положительное целое, равномерно распределенное в некотором диапазоне. Диапазон задается параметрами метода. Три реализации метода отличаются набором параметров:

- **public int Next** () - метод без параметров выдает целые положительные числа во всем положительном диапазоне типа int;
- **public int Next (int max)** - выдает целые положительные числа в диапазоне [0,max];
- **public int Next (int min, int max)** - выдает целые положительные числа в диапазоне [min,max].

Задание 11. Программа, содержащая 4 метода: создание и вывод случайного числа до 10, нахождение максимума, минимума и суммы 3 чисел.
Введите код программы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы.

Модифицируйте программу, добавив метод нахождения суммы квадратов трех целых чисел.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    ThreeInt chisla = new ThreeInt();
    chisla.Vyvod();
    Console.WriteLine("Максимальное
    число {0}",chisla.MaxDig());
    Console.WriteLine("Минимальное чис-
    ло {0}", chisla.MinDig());
    Console.WriteLine("Сумма чисел " +
    chisla.Summa());
    Console.ReadLine();
    }
  }
}
```

Класс Random

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
  /// <summary>
  /// Класс: 3 целых числа и разные действия с ними
  /// </summary>
  class ThreeInt
    { // поля класса
    private int d1, d2, d3;
```

```
/// <summary>
/// Конструктор без параметров: создаются 3 случайных числа до 10
/// </summary>
public ThreeInt()
{  // демонстрация класса Random
const int maxd = 10;

// первоначально создается объект случ величины
Random rnd = new Random();

d1 = rnd.Next(maxd);
d2 = rnd.Next(maxd);
d3 = rnd.Next(maxd);
}

/// <summary>
/// Вывод всех чисел на экран
/// </summary>
public void Vyvod()
/*При использовании в качестве типа возвращаемого значения для метода, void указывает, что метод не возвращает значение*/
{
Console.WriteLine("Первое      число
{0}",d1);
Console.WriteLine("Второе      число
{0}",d2);
Console.WriteLine("Третье      число
{0}",d3);
}

/// <summary>
/// Поиск максимального из чисел
/// </summary>
/// <returns>Возвращает максимум</returns>
public int MaxDig()
{
```

```
        int dmax;

        if (d1 > d2) dmax = d1; else dmax =
        d2;
        if (d3 > dmax) dmax = d3;
        return dmax;
        }

        /// <summary>
        /// Поиск минимального из чисел
        /// </summary>
        /// <returns>Возвращает минимум</returns>
        public int MinDig()
        {
        int dmin;

        dmin = (d1 > d2) ? d2 : d1;
        dmin = (d3 > dmin) ? dmin : d3;
        return dmin;
        }

        /// <summary>
        /// Расчет суммы чисел
        /// </summary>
        /// <returns>Возвращает сумму</returns>
        public int Summa()
        {
        return d1 + d2 + d3;
        }
    }
}
```

Задание 12. Введите код программы «Решения треугольников по 3 сторонам», комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, добавив код, проверяющий, является ли треугольник прямоугольным.
Обратите внимание на генерацию исключений.

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

namespace ConsoleApplication1
{
   /// <summary>
   /// Тема: операторы языка
   /// </summary>
   class Program
   {
      static void Main(string[] args)
      {
      try
      {
      Treug t = new Treug(5, 3, 4);
      t.Vyvod();
      Console.WriteLine("Периметр     тре-
      угольника {0}", t.Perimetr());
      Console.WriteLine("Площадь      тре-
      угольника {0}", t.Ploshad());
      }
      catch(ArgumentOutOfRangeException)
      {
      Console.WriteLine("Ошибка  определе-
      ния сторон треугольника");
      }
      }
   }
}
```

Класс Треугольник

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

```
namespace ConsoleApplication1
{
   /// <summary>
   /// Треугольник по трем сторонам
   /// </summary>
   class Treug
   {
      // поля - стороны треугольника
      private double a, b, c;

      /// <summary>
      /// Конструктор
      /// </summary>
      /// <param name="a"></param>
      /// <param name="b"></param>
      /// <param name="c"></param>
      public Treug(double a, double b,
      double c)
      {
      if (a <= 0 || b <= 0 || c <= 0)
      throw                          (new
      ArgumentOutOfRangeException());    //
      генерация исключения

      // отсортировать, чтобы a < b < c
      if (a > b) { double t = a; a = b; b
      = t; }
      if (a > c) { double t = a; a = c; c
      = t; }
      if (b > c) { double t = b; b = c; c
      = t; }

      // проверить соблюдения основного соотношения
      сторон
      if (a + b <= c) throw (new
      ArgumentOutOfRangeException());
      // генерация исключения

      // поля класса отсортированы по возрастанию
```

```
        this.a = a; this.b = b; this.c = c;
        }


        /// <summary>
        /// Вывод данных
        /// </summary>
        public void Vyvod()
        {
        Console.WriteLine("a=          {0}",
        this.a);
        Console.WriteLine("b=          {0}",
        this.b);
        Console.WriteLine("c=          {0}",
        this.c);
        }

        /// <summary>
        /// Расчет периметра
        /// </summary>
        /// <returns></returns>
        public double Perimetr()
        {
        return this.a + this.b + this.c;
        }

        /// <summary>
        /// Расчет площади по формуле Герона
        /// </summary>
        /// <returns></returns>
        public double Ploshad()
        {
        double p = Perimetr() / 2;

        return Math.Sqrt(p * (p - this.a) *
        (p - this.b)*(p - this.c));
        }
    }
}
```

Массивы.
Массивом называют упорядоченную совокупность элементов одного типа. Каждый элемент массива имеет индексы, определяющие порядок элементов. Число индексов характеризует размерность массива.

1. Одномерные массивы

Объявление одномерных массивов

<тип элементов> [] <имя массива>;

Например:
`int[] k;` // k- одномерный массив, состоящих из целых чисел
`int[] a,b,c;` //одновременное объявление 3 массивов целых чисел

После объявления, так как массив представляет ссылочный объект, то для создания массива необходима инициализация

`k=new int [3];` //создание массива из 3 чисел

Очень часто инициализация совпадает с объявлением.
В первом случае инициализация является явной и задается константным массивом.

`double[] x= {5.5, 6.6, 7.7};` // константный массив

Следуя синтаксису, элементы константного массива следует заключать в фигурные скобки.

Во втором случае создание и инициализация массива выполняется в объектном стиле с вызовом конструктора массива. И это наиболее распространенная практика объявления массивов.

int[] d= new int[5]

В C# нумерация элементов массива идет с нуля d[0], d[1], d[2], d[3], d[4]. Элемента d[5] в данном массиве не существует.

Задание 13. Введите код программы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, добавив массивы v3=v1+v2; v4=a+c+u. Вывести напечать все массивы.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    //объявляются три одномерных массива A,B,C
    int[] a, b, c;
    a = new int[5];
    b = new int[6];
    c = new int[4];
    Random rnd = new Random();
    //Заполняем массив a случайными числами
    for (int i = 0; i < a.GetLength(0)/*обозначает "до конца массива a"*/; i++)
    a[i] = rnd.Next(1, 100);
    //Заполняем массив b случайными числами
```

```
    for (int i = 0; i < b.GetLength(0)/*обозначает "до конца массива b"*/; i++)
    b[i] = rnd.Next(1, 100);


    //Рассчитываем массив с
    for (int i = 0; i < c.GetLength(0); i++)
    c[i] = a[i] + b[i];
    //объявление массива с явной инициализацией
    int[] x = {5, 5, 6, 6, 7, 7};
    //объявление массивов с отложенной инициализацией
    int[] u, v1,v2;
    u = new int[3];
    for (int i = 0; i < 3; i++)
    u[i] = i + 1;

    //v1={1,2,3};недопустимое присваивание
    v1 = new int[4];
    v1 = u; //допустимое присваивание

    v2 = new int[4];
    for (int i = 0; i < 3; i++)
    v2[i] = u[i]*2;

    // Создадим Класс Arr с методом PrintArr для вывода массивов на печать
    Arr.PrintArr("a", a);
    Arr.PrintArr("b", b);
    Arr.PrintArr("c", c);
    Arr.PrintArr("x", x);
    Arr.PrintArr("u", u);
    Arr.PrintArr("v1", v1);
    Arr.PrintArr("v2", v2);
    Console.ReadLine();
    }
}
```

```
}
```

Класс Arr для вывода на печать

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

namespace ConsoleApplication1
{
  class Arr
  {
    public static void PrintArr(string
    name, int[] T)
    {
    Console.WriteLine(name);
    for (int i = 0; i < T.GetLength(0);
    i++)
    Console.Write("\t"    +    name    +
    "[{0}]={1}", i, T[i]);
    Console.WriteLine();
    }
  }
}
```

2. Двумерные массивы

Одномерные массивы позволяют задавать такие математические структуры как векторы, двумерные - матрицы, трехмерные - кубы данных, массивы большей размерности - многомерные кубы данных.

Так задается двумерный массив:

```
int[,] v = new int [2,3];
```

Обратите внимание, что пара квадратных скобок только одна. Естественно, что в нашем примере у массива 6 (=2*3) элементов (v[0,0] - первый, v[1,2] - последний). Аналогично мы можем задавать многомерные массивы.

Пример трехмерного массива:

```
int[,,] a = new int [2,10,7];
```

А вот так можно сразу инициализировать двумерные массивы:

```
int[,]   b   =   {{2,-2},{3,-22},{0,4}};
```

//константный массив

Задание 14. Пример программы умножения элементов матрицы на 7.

Введите код программы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, добавив умножение элементов главной диагонали на 3. Вывести новую матрицу.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    //объявление динамического массива v
    Console.WriteLine("Введите     число
    строк матрицы ");
    int size1 =
    int.Parse(Console.ReadLine());
    Console.WriteLine("Введите     число
    столбцов матрицы ");
    int size2 =
```

```
int.Parse(Console.ReadLine());
int[,] v = new int[size1, size2];

//Заполнение матрицы случайными числами
Random rnd = new Random();
for (int i = 0; i < v.GetLength(0);
i++)
for (int j = 0; j < v.GetLength(1);
j++)
{
v[i,j] = rnd.Next(1, 10); ;
}
//Вывод исходной матрицы
Console.WriteLine("Исходная матри-
ца");
for (int i = 0; i < v.GetLength(0);
i++)
{
for (int j = 0; j < v.GetLength(1);
j++)
{
Console.Write(v[i, j]+"\t" );
}
Console.WriteLine("\n");
}
Console.WriteLine();
//Умножение всех элементов матрицы на 7
for (int i = 0; i < v.GetLength(0);
i++)
for (int j = 0; j < v.GetLength(1);
j++)
{
v[i, j] = v[i, j] * 7;
}

//Вывод новой матрицы
Console.WriteLine("Матрица*7");
for (int i = 0; i < v.GetLength(0);
i++)
```

```
        {
        for (int j = 0; j < v.GetLength(1);
        j++)
        {
        Console.Write(v[i, j]+"\t");
        }
        Console.WriteLine("\n");
        }
        Console.ReadLine();
        }
    }
}
```

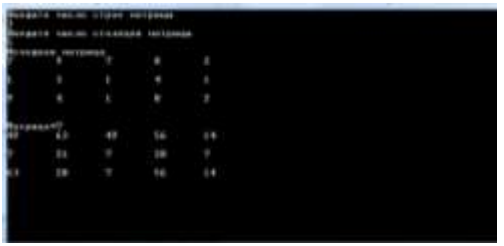


Рисунок 7. Пример работы программы

Задание 15. Класс квадратное уравнение. Одномерные массивы.

Откройте программу вычисления корней квадратного уравнения из Лабораторной работы №4, внесите изменения для выведения корней в массив.

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

namespace ConsoleApplication1
{
/// <summary>
/// Тема: одномерные массивы
```

```
/// </summary>
class Program
   {
/// <summary>
/// Стартовая процедура
/// </summary>
/// <param name="args"></param>
     static void Main(string[] args)
     {
     double a, b, c;
     Console.Write("a = ");
     a =
     double.Parse(Console.ReadLine());
     Console.Write("b = ");
     b =
     double.Parse(Console.ReadLine());
     Console.Write("c = ");
     c =
     double.Parse(Console.ReadLine());

     KvUr uravn = new KvUr(a,b,c);

     Console.WriteLine("Количество  кор-
     ней уравнения {0}",uravn.k);

     if (uravn.k > 0)
       {
       Console.WriteLine("Корни  уравне-
       ния:");
       for (int i = 0; i < uravn.k; i++)
       Console.WriteLine("Корень {0} ра-
       вен {1}", i + 1, uravn.x[i]);
       }
     }
   }
}
```

Класс Корни уравнения

```
using System;
```

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

namespace ConsoleApplication1
{
   /// <summary>
   /// Класс, решающий квадратное уравнение для любого случая
   /// </summary>
   class KvUr
   {
      // поля класса
      private double a, b, c; // коэффициенты уравнения
      public int k; // количество корней
      public double[] x; // корни уравнения
      // конструктор
      public KvUr(double a, double b,
      double c)
      {
      if (a == 0 && b==0 && c==0)
      throw(new ArgumentNullException());
      // вырожденный случай
      this.a = a; this.b = b; this.c = c;
      Reshenie();
      }

      // метод: вычисление дискриминанта
      private double Discr()
      { return Math.Pow(b, 2) - 4 * a *
      c; }

      // метод: вычисление количества корней
      private void CountK()
      {
        if (this.a == 0) this.k = 0;
        else
        {
```

```
        double d = Discr();
        if (d > 0) this.k = 2;
        else if (d == 0) this.k = 1;
        else this.k = 0;
        }
    }

    // метод: вычисление корней уравнения
    private void Reshenie()
    {
    CountK();
    if (this.k == 1)
        {
        this.x = new double[1];
        this.x[0] = -b / 2 / a;
        }
    else
        {
        this.x = new double[2];
        double d = Math.Sqrt(Discr());
        this.x[0] = (-b + d) / 2 / a;
        this.x[1] = (-b - d) / 2 / a;
        }
    }
  }
}
```

Задание 16. Введите код программы «Ввод, транспонирование и вывод матрицы», комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы.

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

namespace ConsoleApplication1
```

```
{
  /// <summary>
  /// Тема: двумерные массивы
  /// </summary>
  class Program
  {
    static void Main(string[] args)
    {
    //Размероность Матрицы задается константами 3x4
    const int m=3;
    const int n=4;

    //определение Класса Matrica с передачей размеров матрицы
    Matrica x = new Matrica(m, n);

    //r - переменная для создания случайного числа
    Random r = new Random();

    // заполнить случайными числами от 0 до 1
    for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++)
    x.a[i,j] = r.NextDouble();

    Console.WriteLine("\nИсходная матрица ");
    x.Vyvod(); //Метод вывода для матрицы X

    Console.WriteLine("\nТранспонированная матрица ");
    Matrica y = Matrica.Transpon(x);
    //метод транспонирования
    y.Vyvod(); //Метод выовда для матрицы Y
    }
  }
}
```

Класс Матрица

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

namespace ConsoleApplication1
{
   /// <summary>
   /// Класс: двумерная матрица
   /// </summary>
   class Matrica
   {
   // поля
      public double[,] a; // двумерный массив
      действительных чисел

      /// <summary>
      /// Конструктор
      /// </summary>
      /// <param name="m"></param>
      /// <param name="n"></param>

      public Matrica(int m, int n)
      {
      if (m <= 0 || n <= 0) throw (new
      ArgumentOutOfRangeException());
      this.a = new double[m, n]; // Выделить
      память для массива
      }

      /// <summary>
      /// Вывод матрицы по строкам
      /// </summary>
      public void Vyvod()
      {
```

```
        for (int i = 0; i < this.a.GetLength(0); i++) // ф-ция GetLength() возвращает кол-во элементов в данной размерности (аргумент)
        {
        for (int j = 0; j < this.a.GetLength(1); j++)
        Console.Write("{0,8:f4}",this.a[i,j]);
        // 8 - ширина поля вывода, f-fixed point, 4 - число знаков после точки
        Console.WriteLine();
        }
        }

        /// <summary>
        /// Транспонирование матрицы
        /// </summary>
        /// <param name="mass">mass - транспонируемая матрица</param>
        /// <returns></returns>

        public static Matrica Transpon(Matrica mass)
        {
        int m = mass.a.GetLength(0);
        int n = mass.a.GetLength(1);

        Matrica b = new Matrica(n, m);
        // число строк = числу столбцов исх массива. И наоборот

        for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
        b.a[i, j] = mass.a[j, i];

        return b;
        }
    }
}
```

Самостоятельная работа.

Одномерные массивы.

1. Элементы массива A записать в виде массивов B и C, причем в массив B записать элементы, стоящие на нечетных местах в массиве A, а в массив C записать элементы, стоящие на четных местах в массиве A.
2. В данном массиве целых чисел найти минимальное значение и все номера элементов массива, равных ему.
3. Элементы линейного вещественного массива X[n] вычисляются следующим образом: X[1] = 1, X[2] = 2, ... , X[k] = X[k-1] * X[k-2], k= 3,4,...,n. Написать программу вычисления элементов массива X.
4. Написать программу замены положительных элементов линейного вещественного массива Y[n] на 0.
5. Вычислить произведение сумм положительных и отрицательных элементов массива A[n].
6. Требуется ввести последовательность целых чисел и проверить, есть ли среди них отрицательные. Если они есть, вывести новую последовательность, состоящую из отрицательных членов исходной последовательности, записанных в том же порядке, в каком они встречались в исходной. Если их нет, сообщить об этом.
7. В одномерном массиве переставьте максимальный и минимальный элементы местами.
8. В одномерном массиве требуется найти наибольший элемент и номер второго такого элемента, если их несколько.
9. Написать программу замены элементов линейного вещественного массива Y[n], имею-

щих нечетные порядковые номера, на число 100.

10. Написать программу, с помощью которого первые k элементов массива A[n], kєn, заполняются последовательностью 1,3,1,3,...
11. В данной последовательности чисел найти минимальное значение среди положительных элементов и все номера членов последовательности, равных ему.
12. "Сожмите массив", "выбросив" каждый второй его элемент (дополнительные массивы использовать не разрешается).
13. Задан одномерный массив A[n], состоящий только из нулей и единиц. Проверьте, строго ли они чередуются.
14. Составить программу, увеличивающую ненулевые элементы линейного вещественного массива Y[n] на 1.
15. Для линейного массива A[n] найти сумму всех элементов: а) равных заданному числу D; б) не равных заданному числу D; в) больше заданного числа D; г) не больше заданного числа D; д) меньше заданного числа D; е) не меньше заданного числа D.
16. Написать программу, увеличивающий на 2 все элементы вещественного массива Y[n], которые больше 10.
17. Заменить максимальный элемент массива A[n] его индексом.
18. Осуществите циклическую перестановку элементов массива: первый элемент должен стать вторым, второй - третьим и т.д., последний - первым. Нового массива не заводить.
19. Для линейного целого массива A[n] найти сумму всех: а) четных элементов; б) нечетных элементов; в) элементов, кратных 3.
20. Определить, имеются ли в целочисленном массиве C [10], два подряд идущих нулевых элемента.

21. Дан массив A[n] и число X. Написать программу, который печатает "ДА", если X совпадает с одним из элементов данного массива, и "НЕТ" - в противном случае.
22. Дан массив A с N элементами. Написать программу, которая изменяет значения элементов массива по правилу: A[i] равно сумме элементов A[k] для к=1,...,i. Дополнительный массив не использовать.
23. Написать программу, при выполнении которой элементы массива заменяются на сумму предыдущего и последующего элементов, если такие существуют.
24. Написать программу нахождения номеров наибольшего неположительного и наименьшего неотрицательного элемента массива A[n].

Двумерные массивы.

1. Дан двумерный массив A[m, n]. Написать программу построения одномерного массива B[m], элементы которого соответственно равны:
   а) суммам элементов строк,
   б) произведениям элементов строк,
   в) наименьшим элементам строк.
2. Массив A[m,n] содержит два (и только два) одинаковых числа. Требуется напечатать их индексы. Обратите внимание на то, чтобы никакой элемент массива не сравнивался сам с собой!
3. Массив A[m,n] содержит вещественные числа. Требуется ввести целое число K и вычислить сумму элементов A[i,j], для которых i+j=K. Прежде, однако следует убедиться, что значение K позволяет найти решение, в противном случае нужно использовать обработчик ошибок.

4. Элемент двумерного массива называется локальным минимумом, если он строго меньше всех имеющихся у него соседей. Подсчитать количество локальных минимумов заданной матрицы размером NxN.
5. Дан массив A[n,n]. Составить программу, которая прибавила бы каждому элементу данной строки элемент, принадлежащий этой строке и главной диагонали.
6. Сформируйте двумерный массив NxN по следующему правилу: элементы главной диагонали равны 1, ниже главной диагонали - 0, а выше - сумме индексов.
7. Транспонируйте произвольный двухмерный массив. Дополнительные массивы не используйте.
8. Вычислить след заданной квадратной матрицы A[N,N]. След квадратной матрицы - это число, равное сумме элементов главной диагонали.
9. Из массива A[n,n] сформируйте массив B[n*n], "развернув" его по столбцам (вариант: по строкам).
10. Из предложенного одномерного массива длины N сформируйте двумерный массив так, чтобы первая строка нового массива содержала четные по номеру элементы исходного массива, а вторая - нечетные (варианты: четное или нечетное N).
11. Напишите программу, находящую в двумерном массиве номера строк с наибольшей суммой элементов и записывающую эти номера в одномерный массив
12. Определите, имеются ли в массиве A[4,4] строки, равные первой строке. Если имеются, выведите ДА, если отсутствуют - НЕТ.
13. Найти сумму тех элементов двумерного массива, которые делятся на 2.
14. Написать программу для перестановки максимальных элементов строк двумерного массива

B[L, L] на главную диагональ. Элементы главной диагонали B[i,i] ставить на место максимального элемента в этой строке.

15. Назовем максимальный элемент двумерного массива S[n,n] - главным. Написать программу, которая находит произведение чисел той строки двумерного массива, в которой расположен главный элемент.
16. Написать программу заполнения массива N*N нулями и единицами в шахматном порядке.
17. Дан массив A[n,m]. Найти номер столбца, для которого среднеарифметическое значение его элементов: a) максимально; b)минимально.
18. Написать программу, которая меняет местами значения элементов двумерного массива чисел A[n,n] симметричных относительно главной диагонали.
19. Написать программу, которая позволяет напечатать таблицу Пифагора. Таблица Пифагора - это квадратная матрица из 10 строк и 10 столбцов, каждый элемент которой определяется формулой: A[i,j] = i*j.
20. Заполнить массив A следующим образом

    1 2 . . .10
    11 12 . . .20
    21 22 . . .30
    . . . . . .
    91 92 . . .100
21. Заполнить массив A следующим образом

    1 2 3 . . .10
    0 1 2 . . . 9
    0 0 1 . . . 8
    . . . . .
    0 0 0 . . . 1
22. Получить массив B из массива A удалением n-й строки и k-го столбца.

Написать отчет по лабораторной работе.

Лабораторная работа № 6

Краткое описание:

- Методы Класса **System.String**
- Виртуальный Класс Array
- Посимвольный анализ Массива
- Разбор строки
- Регулярные выражения

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение
Строковые типы данных

Строковые константы

Без констант не обойтись. В C# существуют два вида строковых констант:

• обычные константы, которые представляют строку символов, заключенную в кавычки;

• @-константы, заданные обычной константой с предшествующим знаком @.

В обычных константах некоторые символы интерпретируются особым образом. Связано это прежде всего с тем, что необходимо уметь задавать в строке непечатаемые символы, такие, как, например, символ табуляции. Возникает необходимость задавать символы их кодом - в виде escape-последовательностей. Для всех этих целей используется комбинация символов, начинающаяся символом "\" - обратная косая черта. Так, пары символов: "\n", "\t", "\\", "\"" задают соответственно символ перехода на новую строку, символ табуляции, сам символ обратной косой черты, символ кавычки, вставляемый в строку, но не сигнализирующий о ее окончании. Комбинация "\xNNNN" задает символ, определяемый шестнадцатеричным кодом NNNN. Хотя такое решение возникающих проблем совершенно естественно, иногда возникают неудобства: например, при задании констант, определяющих путь к файлу, приходится каждый раз удваивать символ обратной

косой черты. Это одна из причин, по которой появились @-константы.
В @-константах все символы трактуются в полном соответствии с их изображением. Поэтому путь к файлу лучше задавать @-константой. Единственная проблема в таких случаях: как задать символ кавычки, чтобы он не воспринимался как конец самой константы. Решением является удвоение символа.
Строки в C# - это экземпляры класса **System.String.** В C# есть тип string, но класс System.String является более продвинутым, так что его использование часто оказывается более оправданным и простым.
Этот класс имеет множество методов и свойств, некоторые из которых перечислены ниже:
Свойство **Length**. Возвращает длину строки. Пример использования:

```
String s="Здравствуй, Мир!";
int k=s.Length; //в k запишется 16
```

Метод **Compare**. Статический метод, сравнивающий две строки. Возвращает 0, есть строки равны, отрицательное значение, если первая строка меньше второй и положительное значение, если первая строки больше второй (больше и меньше в алфавитном смысле, разумеется).

```
namespace ConsoleApplication1
{
  class Program
  {
  static void Main(string[] args)
    {
    String s1 = "Молоко", s2 = "Масло",
    s3 = "Сливки";
    Console.WriteLine(String.Compare(s1,
    s1)); //Выдаст 0, т. к. "Молоко" равно
    "Молоко".
```

```
        Console.WriteLine(String.Compare(s1,
        s2));
        //Выдаст 1, т. к. "Молоко" больше "Масло". "о"
        позже "а"
        Console.WriteLine(String.Compare(s1,
        s3));  //Выдаст -1, т. к. "Молоко" меньше
        "Сливки". "м" раньше "с"
        Console.ReadLine();
        }
    }
}
```

Метод **Equals.** Метод, возвращает true, если строки равны, false - если не равны. Может быть как статическим, так и не статическим.

```
String t1="Дом", t2="Дым";
//Статический метод
Console.WriteLine(String.Equals(t1,
t2).ToString());
//Не статический метод
Console.WriteLine(t1.Equals(t2).ToString
());
```

Метод **Substring**. Поволяет извлечь из строки подстроку. Пример использования:

```
String a1 = "Смирно!", a2;
a2 = a1.Substring(1, 3); //записывает в строку
```
a2 3 символа строки a1 начиная со второго,
нумерация символов в строке начинается с 0,
```
Console.WriteLine(a2); //Напечатается "мир"
```

Метод **Insert**. Вставляет в строку другую строку.

```
String y1 = "Графа", y2;
y2 = y1.Insert(4, "ик");
Console.WriteLine(y2);
```

Первый параметр тут - это куда вставляем (нумерация, как всегда, с нуля), второй - что за строчку вставляем.
Метод **IndexOf**. Позволяет найти в строке подстроку.

```
String x1 = "Природа", x2 = "род", x3 =
"дар";
Console.WriteLine(x1.IndexOf(x2));
//Напечатается 3 - начальная позиция подстроки
Console.WriteLine(x1.IndexOf(x3));
//Напечатается -1
```

Этот метод возвращает номер позиции, на котором в строке находится передаваемая в качестве параметра подстрока. Если такой подстроки нет, то возвращается -1.

Метод **Replace.** Производит замену в строке.

```
String r1 = "Малоко", r2 = "а", r3;
r3 = r1.Replace(r2, "о"); // В строке 1 меняем
подстроку 2 на "о"
Console.WriteLine(r3);
```

Методы **EndsWith** и **StartsWith**. Проверяют, не завершается ли строка на заданную строку, или не начинается ли строка с заданной строки соответственно.

```
String b1 = "Палиндром";
if (b1.StartsWith("Пал"))
Console.WriteLine("Строка начинается на
\"Пал\"");
else
Console.WriteLine("Строка не начинается
на \"Пал\"");
if (b1.EndsWith("дом"))
Console.WriteLine("Строка заканчивается
на \"дом\"");
else
Console.WriteLine("Строка не заканчива-
етсяна \"дом\"");
Console.ReadLine();
```

Методы **ToUpper** и **ToLower** переводят строку в верхний или нижний регистр соответственно. Пример использования:

```
String u1 = "фмф";
u1 = u1.ToUpper();
Console.WriteLine("Наш    факультет    -
{0}!",u1);
Console.ReadLine();
```

Методы **Trim, TrimEnds и TrimStart**. Удаляют пробельные символы из начала и конца строки (**Trim**), только с конца строки (**TrimEnds**) и только с начала строки (**TrimStart**).

```
String l1 = "     пропуск слов     ";
Console.WriteLine("Начальный     вариант
{0} строки",l1);
l1=l1.Trim();
Console.WriteLine("Усовершенствованный
{0} строки", l1);
```

Виртуальный класс Array

Каждый создаваемый массив получает большую часть функциональности от класса **System.Array**. Общие члены этого класса позволяют работать с массивом с использованием полноценной объектной модели. Таким образом, методы и свойства, определенные в классе Array, можно использовать с любым массивом C#.

Создание массивов

Класс Array является абстрактным, поэтому создать массив с использованием какого-либо конструктора нельзя. Однако вместо применения синтаксиса C# для создания экземпляров массивов также возможно создавать их с помощью статического метода CreateInstance(). Это исключительно удобно, когда

заранее неизвестен тип элементов массива, поскольку тип можно передать методу CreateInstance() в параметре как объект Type:

```
// Создаем массив типа string, длиной 5
Array myArr = Array.CreateInstance(typeof(string), 5);

// Инициализируем первые два поля массива
myArr.SetValue("Имя", 0);
myArr.SetValue("Возраст", 1);

// Считываем данные из массива
string s1 = (string)myArr.GetValue(0);
string s2 = (string)myArr.GetValue(1);

Console.WriteLine(s1+", "+s2);
Console.ReadLine();
```

Сортировка и поиск. Статические методы класса Array

Статические методы класса Array позволяют решать самые разнообразные задачи:
1. ***Copy*** - позволяет копировать весь массив или его часть в другой массив.
2. ***IndexOf, LastIndexOf*** - определяют индексы первого и последнего вхождения образца в массив, возвращая -1, если такового вхождения не обнаружено.
3. ***Reverse*** - выполняет обращение массива, переставляя элементы в обратном порядке.
4. ***Sort*** - осуществляет *сортировку* массива.
5. ***BinarySearch*** - определяет индекс первого вхождения образца в отсортированный массив, используя алгоритм двоичного *поиска*.

Задание 17. Введите код программы, в которой используются все перечисленные выше методы, комментарии вводить не обязательно, запустите программу на

выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, изменяя переменные и строки вывода.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    int[] myArr = {4, 5, -183, 12, 34,
    0, 2 ,-13}; //явная инициализация массива

    Console.WriteLine("Исходный массив
    чисел: ");
    foreach (int x in myArr) //Цикл со
    счетчиком для массивов
    Console.Write("\t{0}",x);

    // Реализуем сортировку массива
    Console.WriteLine("\n\nОтсортирован
    ный массив:");
    Array.Sort(myArr); //метод сортировки вир-
    туального класса Array
    foreach (int x in myArr)
    Console.Write("\t{0}",x);

    // Организуем поиск числа 12
    Console.WriteLine("\n\nПоиск
    числа:");
    int search = Ar-
    ray.BinarySearch(myArr, 12); //метод би-
    нарного поиска виртуального класса Array
    Console.WriteLine("Число 12 нахо-
    дится на {0} позиции",search+1);
```

```
    }
  }
}
```

Задание 18. Введите код программы, в которой используются все перечисленные выше методы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, добавив еще одну переменную класса «Случайный массив» и отсортируйте новый массив по убыванию.

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

namespace ConsoleApplication1
{
  /// <summary>
  /// Тема: класс Array
  /// </summary>
  class Program
  {
    static void Main(string[] args)
    {
    RandArr ra = new RandArr(1, 10,
    10);

    // Вывод
    ra.Vyvod();

    // Поиск
    Console.Write("Введите число для
    поиска в массиве ");
    int n =
    int.Parse(Console.ReadLine());

    int[] p = ra.Poisk(n);
```

```
        if (p == null)
        Console.WriteLine("В массиве такого
        элемента нет");
        else
        {
        Console.WriteLine("Найденные индек-
        сы массива:");
        for (int i = 0; i < p.Length; i++)
        Console.Write("{0}  ", p[i]);
        Console.WriteLine();
        }

        // Сортировка
        int[] arr = (int[])ra.GetArr();
        Array.Sort(arr);
        Console.WriteLine("Отсортированный
        массив:");
        for (int i = 0; i < arr.Length;
        i++)
        Console.Write("{0}  ", arr[i]);
        Console.WriteLine();

        }
    }
}
```

Класс Случайный массив

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

namespace ConsoleApplication1
{
    /// <summary>
    /// Класс: одномерный массив случайных целых или
    действительных чисел
```

```
/// </summary>
class RandArr
{
   // поле класса
   private Array a;

/// <summary>
/// Конструктор
/// </summary>
/// <param name="t">Тип элементов массива: 1 - целый, 2 - действительный</param>
/// <param name="size">Количество элементов в массиве</param>
/// <param name="max">Максимальный элемент массива (минимальный = 0)</param>

   public RandArr(int t, int size,
   double max)
   {
   // предупреждение возможных ошибок
   if ((t != 1 && t != 2) || size <= 0
   ||      max      <=      0)      new
   ArgumentOutOfRangeException();

   // инициализация датчика случайных чисел
   Random r = new Random();

   // создание массива целых или действительных чисел
   if (t == 1)
   {
   int[] mass = new int[size];
   this.a = mass;
   }
   else
   {
   double[] mass = new double[size];
   this.a = mass;
   }
```

```
// заполнение массива случайными числами
for (int i = 0; i < this.a.Length;
i++)
if        (this.a.GetType()        ==
typeof(Int32[]))
this.a.SetValue(r.Next(0,
(int)max), i); //a[i] = ... нельзя
else
this.a.SetValue(max*r.NextDouble(),
i); //a[i] = ... нельзя

}

/// <summary>
/// Для доступа к массиву извне
/// </summary>
/// <returns></returns>
public Array GetArr()
{
return this.a;
}

/// <summary>
/// Вывод массива на экран
/// </summary>

public void Vyvod()
{
foreach (object elem in this.a)
Console.Write("{0}        ",    el-
em.ToString());
Console.WriteLine();
}


/// <summary>
/// Ищет номера (индексы) вхождения элемента s
в массив a
```

```
        /// </summary>
        /// <param name="s">Искомый элемент (типа int или double)</param>
        /// <returns></returns>

        public int[] Poisk(object s)
        {
        int[] ind = new int[this.a.Length]; // массив для записи номеров элементов. Размер - с запасом
        int[] indexes = null; // выводимый массив. Если элемент не будет найден, ф-ция возвратит null

        int i=0,n = 0;

        while (n != -1)
        {
        n = Array.IndexOf(this.a, s, n); // статический метод
        if (n != -1) ind[i++] = n++;
        }

        if (i > 0)
        {
        indexes = new int[i]; // выделение памяти для выводимого массива
        Array.Copy(ind, indexes, i); // копирование индексов в новый массив
        }

        return indexes; // по окончании функции сборщик мусора автомат. удаляет массив ind из кучи
        }
    }
}
```

Задание 19. Введите код программы «Разбор строки (массив символов). Стандартные классы Array, Char», в которой используются все перечисленные выше методы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, изменяя переменные и строки вывода.

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

namespace ConsoleApplication1
{
  /// <summary>
  /// Тема: символы, массивы символов
  /// </summary>
  class Program
  {
    static void Main(string[] args)
    {
    Console.Write("Введите строку символов ");
    string s = Console.ReadLine();

    RazborStroki str = new RazborStroki(s);

    // объявляем массив символьного типа и вызываем метод Letters класса str
    char[] b = str.Letters();

    char[] c = str.Digits(); //Метод - Цифры
    char[] d = str.Punct(); //Метод - знаки препинания
    char[] e = str.Others(); //Метод - другие символы
```

```
     RazborStroki.PrintArr("Буквы ", b);
     RazborStroki.PrintArr("Цифры", c);
     RazborStroki.PrintArr("Знаки препи-
     нания ",d);
     RazborStroki.PrintArr("Прочие  сим-
     волы",e);
     }
   }
}

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

namespace ConsoleApplication1
{
    /// <summary>
    /// Класс: разбор строки на массивы символов,
    цифр, знаков препинания, прочих символов
    /// </summary>
    class RazborStroki
    {
    // поле: разбираемая строка
       private string s;

    /// <summary>
    /// Конструктор
    /// </summary>
    /// <param name="s"></param>
       public RazborStroki(string s)
       {
       this.s = s;
       }
    /// <summary>
    /// Добавляет символ в конец массива, если его еще
    нет
    /// </summary>
```

```
/// <param name="c">Добавляемый символ</param>
/// <param name="mass">Массив символов</param>
/// <param name="n">Индекс последнего элемента массива</param>
///
   private static void AddChar(char c,char[] mass,ref int n) //ref - изменяемый параметр
   {
   if (Array.IndexOf(mass,c) == -1) //выполняет поиск и указывает индекс объекта, если объекта нет - индекс=-1
   mass[n++] = c;
   }

   /// <summary>
   /// Создание массива символов
   /// </summary>
   /// <param name="vid">вид символа; 1-буква, 2-цифра, 3-препинание, 4 прочие</param>
   /// <returns></returns>
   private char[] MakeArr(int vid)
   {
   char[] temp = new char[this.s.Length]; // вспомогательный массив символов
   char[] symb = null; // выводимый массив символов
   int n=0; // количество найденных символов

   for (int i = 0; i < this.s.Length; i++)
   {
   char c = s[i];
   switch (vid)
   {
```

```
case 1: if (Char.IsLetter(c))
AddChar(c,temp, ref n); break;
case 2: if (Char.IsDigit(c))
AddChar(c,temp, ref n); break;
case 3: if (Char.IsPunctuation(c))
AddChar(c,temp, ref n); break;
case 4: if (!Char.IsLetter(c) &&
!Char.IsDigit(c) &&
!Char.IsPunctuation(c))
AddChar(c,temp, ref n); break;
}
}
```
// скопировать temp в новый массив с реальными размерами
```
if (n > 0)
{
symb = new char[n];
Array.Copy(temp, symb, n);
Array.Sort(symb);
}
return symb;
}

/// <summary>
/// Возвращает массив букв
/// </summary>
/// <returns></returns>
public char[] Letters()
{
return MakeArr(1);
}

/// <summary>
/// Возвращает массив букв
/// </summary>
/// <returns></returns>
public char[] Digits()
{
return MakeArr(2);
```

```
        }

        /// <summary>
        /// Возвращает массив букв
        /// </summary>
        /// <returns></returns>
        public char[] Punct()
        {
        return MakeArr(3);
        }

        /// <summary>
        /// Возвращает массив букв
        /// </summary>
        /// <returns></returns>
        public char[] Others()
        {
        return MakeArr(4);
        }

        //Метод выводы массивов
        public static void PrintArr(string
        msg,char[] a)
        {
        if (a == null)
        Console.WriteLine(msg + " не най-
        ден");
        else
        {
        Console.WriteLine(msg);
        for (int i = 0; i < a.Length; i++)
        Console.Write("{0} ", a[i]);
        Console.WriteLine();
        }
        }
    }
}
```

Задание 20. Введите код программы «Класс слова. Стандартный класс String», комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы. Модифицируйте программу, изменяя переменные и строки вывода Класс слова. Стандартный класс String.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    Slova sl = new Slova("Гляжу, под-
    нимается медленно в гору");
    Console.WriteLine(sl.Out());
    string[] asp = sl.SpisokSlov();
    foreach (string s in asp)
    {
    Console.Write("{0} ", s);
    }
    Console.WriteLine();
    Console.ReadLine();
    }
  }
}
```

Класс Слова

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

namespace ConsoleApplication1
{
```

```
/// <summary>
/// Операции со строками
/// </summary>
class Slova
{
   private string s; // поля класса

   /// <summary>
   /// Конструктор.
   /// </summary>
   /// <param name="s">не должна быть пустая</param>
   public Slova(string s)
   {
   if (s == null || s.Length == 0)
   throw new ArgumentNullException();
   // приводим строку к набору слов, убирая пунктуацию и лишние пробелы
   this.s = DelPunct(s);
   }

   /// <summary>
   /// Вывод строки
   /// </summary>
   /// <returns></returns>
   public string Out()
   {
   return this.s;
   }

   /// <summary>
   /// Отсортированный список слов в строке
   /// </summary>
   /// <returns></returns>

   public string[] SpisokSlov()
   {
   string[] spis = this.s.Split(new
   char[] { ' ' });
```

```
Array.Sort(spis, Sravn);

return spis;
}

/// <summary>
/// Сравнение двух строк по длине.
/// Используется при сортировке массива слов
/// </summary>
/// <param name="s1"></param>
/// <param name="s2"></param>
/// <returns></returns>
private int Sravn(string s1, string
s2)
{
// Используется метод CompareTo прародителя
Object
int n =
s1.Length.CompareTo(s2.Length);
if (n != 0) return n;
else
return s1.CompareTo(s2); // если длины
строк равны, сравниваются сами строки
}

/// <summary>
/// получить в строке s слова, разделенные
пробелом
/// </summary>
/// <param name="s"></param>
/// <returns></returns>

private string DelPunct(string s)
{
string se = new string(' ',
s.Length); // получить строку из пробелов

StringBuilder st = new
StringBuilder(se);
```

```
    for (int i = 0; i < s.Length; i++)
    if (Char.IsLetter(s[i]))
    st[i] = s[i];
    else st[i] = ' ';
    // в string s[i] только для чтения
    // в StringBuilder st[i] для чтения и записи

    se = st.ToString();
    char[] c = { ' ' };
    string[]     w     =     se.Split(c,
    StringSplitOptions.RemoveEmptyEntri
    es);
    se = String.Join(" ", w);
    return se;
    }
  }
}
```

Регулярные выражения.

Регулярные выражения — это часть небольшой технологической области, невероятно широко используемой в огромном диапазоне программ. Регулярные выражения можно представить себе как мини-язык программирования, имеющий одно специфическое назначение: находить подстроки в больших строковых выражениях.

(англ. regular expressions, сокр. RegExp, RegEx) — система синтаксического разбора текстовых фрагментов по формализованному шаблону, основанная на системе записи образцов для поиска

Методика применения
1. Подключить пространство имен System.Text.RegularExpressions
2. Создать экземпляр класса RegEx, задающий поисковую строку **RegEx p = new Regex(“^(a|b)*”);**
3. Вызвать метод Match, возвращающий результат поиска **Match m = p.Match(str); // str – строка, в кот. ищем**
4. Проанализировать, был ли успешным поиск if (m.Success)
5. Вывести первую найденную строку, удовлетворяющую условию m.Value или индекс найденной подстроки m.Index

Условные обозначения – классы символов	
()	Круглые скобки – группа символов
[]	Квадратные скобки – один из множества символов
[A-Z]	Один из диапазона символов
(a|b|c)	Один из символов. Аналогично [abc]
/w	Множество символов: знаки латиницы и цифры
/s	Пробелы
/d	Цифры

Условные обозначения – модификаторы	
.	Точка – любой символ
^	Начало строки
$	Конец строки
*	0 или более соответствий
+	1 или более соответствий
?	0 или 1 соответствие
{n}	Ровно n соответствий (вместо n подставить число)

Более подробно метасимволы можно узнать по ссылке: http://professorweb.ru/my/csharp/charp_theory/level4/4_10.php

Задание 21. Введите код программы, использующей методику поиска подстроки в строке., в которой используются перечисленные выше методы, комментарии вводить не обязательно, запустите программу на выполнение, проверьте различные варианты выполнения программы.
Задание: придумать несколько примеров для поиска одного элемента и несколько примеров для поиска подстроки из нескольких элементов. Модифицируйте программу.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    // Примеры поиска по образцу с использованием
    регулярных выражений
    // s - исходная строка, obrazec - образец подстро-
    ки для поиска
    string s="", obrazec="";
    string[] naideno={};

    PoiskOdnogo(s, obrazec, naideno);
    PoiskMnogih(s, obrazec, naideno);
    }

    static    void    PoiskOdnogo(string
    s,string obrazec,string[] naideno)
    {

    s = "start";
```

```
obrazec = @"a\w+";
// \w - англ буквы и цифры
// + - одно или более соответствий
// поисковая строка: буква a и за ней одна или более букв или цифр

naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

s = "fab77cd ef";
naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

s = "fabaddbddfabbb";
obrazec = "a(b|d)*f";
// | - или
// * - 0 или несколько соответствий
// поисковая строка:буква a, за ней 0 или несколько символов b или d, в конце строки f

naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

obrazec = "b.d";
// . - любой символ, кроме конца строки
naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

// Название диска из пути к файлу
```

```
s = @"c:\Program Files\Microsoft Visual Studio 10.0\Samples\text.txt";
obrazec = @"^[a-z|A-Z]\:"; // ^ - в начале строки
naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

// Расширение из названия файла
s = "Письмо от 25.05.docx";
obrazec = @"\.(\w)+$"; // $ - в конце строки
naideno = RegExpress.FindMatch(s, obrazec);
RegExpress.Print(s, obrazec, naideno);
}
static void PoiskMnogih(string s, string obrazec, string[] naideno)
{
s = "01101111111011";
obrazec = "(00|11)*((01|10)(00|11)*(01|10)(00|11)*)*"; // четное число единиц
naideno = RegExpress.FindMatches(s, obrazec);
RegExpress.Print(s, obrazec, naideno);

s = "кок тот кук тут как кот";
obrazec = "(к|т).(к|т)";
naideno = RegExpress.FindMatches(s, obrazec);
RegExpress.Print(s, obrazec, naideno);
```

```
    s = @"c:\Program Files\Microsoft
    Visual                    Studio
    10.0\Samples\text.txt";
    obrazec      =      @"[а-я|А-Я|\w|
    |\.]+(\\)";  // список папок в пути
    naideno = RegExpress.FindMatches(s,
    obrazec);
    RegExpress.Print(s,        obrazec,
    naideno);

    // обратный поиск
    s = "asaaasdderr";
    obrazec = @"([a-z])\1"; // выражения в
    скобках нумеруются
    naideno = RegExpress.FindMatches(s,
    obrazec);
    RegExpress.Print(s,        obrazec,
    naideno);

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

Класс RegExpress

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
// для работы с регулярными выражениями

namespace ConsoleApplication1
{
  class RegExpress // только статические мето-
  ды
  {
```

```
/// <summary>
/// Найти первое соответствие
/// </summary>
/// <param name="stroka"></param>
/// <param name="poisk"></param>
/// <returns></returns>
static public string[] FindMatch(string stroka, string poisk)
{
Regex pat = new Regex(poisk); // регулярное выражение для поиска подстроки
Match match = pat.Match(stroka); // создан объект, в кот. результаты поиска
string found = "";
if (match.Success) // если что-нибудь найдено
found = match.Value; // в свойстве Value - первая найденная строка

string[] s = {found};

return s; // если не найдено, возвращается пустая строка
}

/// <summary>
/// Найти все соответствия
/// </summary>
/// <param name="stroka"></param>
/// <param name="poisk"></param>
/// <returns></returns>

static public string[] FindMatches(string stroka, string poisk)
{
```

```
  Regex pat = new Regex(poisk); // регулярное выражение для поиска подстроки
  MatchCollection matches = pat.Matches(stroka); // создан объект, в кот. результаты поиска
  string[] temp = new string[matches.Count];

  int i=0;
  foreach (Match match in matches)
  {
  if (match.Length>0) // берем только непустые строки
  temp[i++] = match.Value;
  }

  string[] found = new string[i];

  Array.Copy(temp, 0, found, 0, i);

  return found;
  }

  static public void Print(string stroka, string poisk, string[] found)
  {
  Console.WriteLine("\nИсходная строка {0}", stroka);

  Console.WriteLine("Строка для поиска {0}", poisk);

  foreach(string s in found)

  Console.WriteLine("Найденная строка {0}", s);
  }
}
```

```
}
```

Самостоятельная работа

Посимвольный анализ и преобразование строк. Строки и числа.

1. Дана строка. Подсчитать количество содержащихся в ней цифр.
2. Дана строка. Подсчитать количество содержащихся в ней прописных латинских букв.
3. Дана строка. Подсчитать общее количество содержащихся в ней строчных латинских и русских букв.
4. Дана строка. Преобразовать в ней все прописные латинские буквы в строчные.
5. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные.
6. Дана строка. Преобразовать в ней все строчные буквы (как латинские, так и русские) в прописные, а прописные — в строчные.
7. Дана строка. Если она представляет собой запись целого числа, то вывести 1, если вещественного (с дробной частью) — вывести 2; если строку нельзя преобразовать в число, то вывести 0. Считать, что дробная часть вещественного числа отделяется от его целой части десятичной точкой «.».
8. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке слева направо).
9. Дано целое положительное число. Вывести символы, изображающие цифры этого числа (в порядке справа налево).

10. Дана строка, изображающая целое положительное число. Вывести сумму цифр этого числа.
11. Дана строка, изображающая арифметическое выражение вида «<цифра>±<цифра>±…±<цифра>», где на месте знака операции «±» находится символ «+» или «–» (например, «4+7–2–8»). Вывести значение данного выражения (целое число).
12. Дана строка, изображающая двоичную запись целого положительного числа. Вывести строку, изображающую десятичную запись этого же числа.
13. Дана строка, изображающая десятичную запись целого положительного числа. Вывести строку, изображающую двоичную запись этого же числа.

Обработка строк с помощью стандартных функций. Поиск и замена.

1. Дано целое число N (> 0) и строка S. Преобразовать строку S в строку длины N следующим образом: если длина строки S больше N, то отбросить первые символы, если длина строки S меньше N, то в ее начало добавить символы «.» (точка).
2. Даны целые положительные числа N1 и N2 и строки S1 и S2. Получить из этих строк новую строку, содержащую первые N1 символов строки S1 и последние N2 символов строки S2 (в указанном порядке).
3. Дан символ C и строка S. Удвоить каждое вхождение символа C в строку S.
4. Дан символ C и строки S, S0. Перед каждым вхождением символа C в строку S вставить строку S0.

5. Дан символ C и строки S, S0. После каждого вхождения символа C в строку S вставить строку S0.
6. Даны строки S и S0. Проверить, содержится ли строка S0 в строке S. Если содержится, то вывести True, если не содержится, то вывести False.
7. Даны строки S и S0. Найти количество вхождений строки S0 в строку S.
8. Даны строки S и S0. Удалить из строки S первую подстроку, совпадающую с S0. Если совпадающих подстрок нет, то вывести строку S без изменений.
9. Даны строки S и S0. Удалить из строки S последнюю подстроку, совпадающую с S0. Если совпадающих подстрок нет, то вывести строку S без изменений.
10. Даны строки S и S0. Удалить из строки S все подстроки, совпадающие с S0. Если совпадающих подстрок нет, то вывести строку S без изменений.
11. Даны строки S, S1 и S2. Заменить в строке S первое вхождение строки S1 на строку S2.
12. Даны строки S, S1 и S2. Заменить в строке S последнее вхождение строки S1 на строку S2.
13. Даны строки S, S1 и S2. Заменить в строке S все вхождения строки S1 на строку S2.
14. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и вторым пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.
15. Дана строка, содержащая по крайней мере один символ пробела. Вывести подстроку, расположенную между первым и последним пробелом исходной строки. Если строка содержит только один пробел, то вывести пустую строку.

Разные задачи по теме строки.

1. Замените каждую встреченную в строке X букву "к" сочетанием букв "ken".
2. Вычеркните из строки Y все буквы, стоящие на нечетных местах.
3. Выясните, есть ли в слове X хотя бы одна из букв "к" или "м".
4. Подсчитать, сколько раз цифра 4 встречается в десятичной записи трехзначного натурального числа M.
5. Подсчитайте, сколько раз первая буква строки X встречается в этой строке.
6. Выясните, какая из букв (первая или последняя) встречается в строке X чаще.
7. Определите наибольшую из цифр, используемых в десятичной записи двузначного натурального числа N .
8. Составьте алгоритм подсчета числа одинаковых букв в строках X и Y равной длины, стоящих на одних и тех же местах.
9. Выясните, является ли данная строка "перевертышем".
10. Вычеркните i-ю букву строки.
11. Выясните, какая из букв "а" или "б" встречается в строке X чаще.
12. Из слов "микромир", "мировоззрение", "миротворец" получить слово "мир".
13. Подсчитать, сколько раз встречается сочетание "ал" в строке: 'КАШАЛОТ КАЛОШУ КУШАЛ'.
14. Утройте каждую букву заданной строки.

Написать отчет по лабораторной работе.

Лабораторная работа № 7

Краткое описание:

- Способы преобразования типов
- Коллекции
- Множества

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Преобразование типов.
Существует явное и неявное преобразование типов.
Автоматическое преобразование типов
Когда данные одного типа присваиваются переменной другого типа, неявное преобразование типов происходит автоматически при следующих условиях:

- оба типа совместимы
- диапазон представления чисел целевого типа шире, чем у исходного типа

Если оба эти условия удовлетворяются, то происходит расширяющее преобразование. Например, тип **int** достаточно крупный, чтобы вмещать в себя все действительные значения типа **byte**, а кроме того, оба типа, int и byte, являются совместимыми целочисленными типами, и поэтому для них вполне возможно неявное преобразование.
Приведение несовместимых типов
Несмотря на всю полезность неявных преобразований типов, они неспособны удовлетворить все потребности в программировании, поскольку допускают лишь расширяющие преобразования совместимых типов. А во всех остальных случаях приходится обращаться к приведению типов.
Приведение — это команда компилятору преобразовать результат вычисления выражения в указанный тип. А для этого требуется явное преобразование типов. Ниже приведена общая форма приведения типов:
(целевой_тип) выражение
Здесь ***целевой_тип*** обозначает тот тип, в который желательно преобразовать указанное выражение.

Конвертация типов.

Роль класса *System.Convert*

В пространстве имен *System* имеется класс *Convert*, который тоже может применяться для расширения и сужения данных:

```
byte sum = Convert.ToByte(var1 + var2);
```

Одно из преимуществ подхода с применением класса System.Convert связано с тем, что он позволяет выполнять преобразования между типами данных нейтральным к языку образом. Однако, поскольку в C# есть операция явного преобразования, использование класса Convert для преобразования типов данных обычно является делом вкуса.

```
char ch1='A'; // символом, заключенным в одинарные кавычки
char ch2 ='\x5A'; // escape-последовательностью, задающей код символа;
char ch3='\u0058'; //Unicode-последовательностью, задающей Unicode-код символа.
char ch = new Char(); //Массив символов

int kod;
string s;
ch = ch1;
ch = ch + ch2;

//неявное преобразование символьного типа в тип int
kod = ch1;

//явное преобразование числа в символ
ch1=(char)(kod + 1);

//преобразование символьного типа в строку
//s = ch неявное преобразование невозможно;

s = ch1.ToString() + ch2.ToString()+
ch3.ToString();
```

//Использование возможностей класса System.Convert

```
Console.WriteLine("s=  {0},  ch=  {1},
ch1= {2},ch2= {3},ch3= {4}, kod = {5}",
s, ch, ch1, ch2,ch3, kod);
Console.ReadLine();
```

Коллекции в C#. Множества – коллекции.

В C# коллекция представляет собой совокупность объектов. В среде .NET Framework имеется немало интерфейсов и классов, в которых определяются и реализуются различные типы коллекций. Коллекции упрощают решение многих задач программирования благодаря тому, что предлагают готовые решения для создания целого ряда типичных, но порой трудоемких для разработки структур данных. Например, в среду .NET Framework встроены коллекции, предназначенные для поддержки динамических массивов, связных списков, стеков, очередей и хеш-таблиц.

Краткий обзор коллекций

Главное преимущество коллекций заключается в том, что они стандартизируют обработку групп объектов в программе. Все коллекции разработаны на основе набора четко определенных интерфейсов. Некоторые встроенные реализации таких интерфейсов, в том числе ArrayList, Hashtable, Stack и Queue, могут применяться в исходном виде и без каких-либо изменений. Имеется также возможность реализовать собственную коллекцию, хотя потребность в этом возникает крайне редко.

В среде .NET Framework поддерживаются пять типов коллекций: необобщенные, специальные, с поразрядной организацией, обобщенные и параллельные.

Необобщенные коллекции

Реализуют ряд основных структур данных, включая динамический массив, стек, очередь, а также словари, в которых можно хранить пары "ключ-значение". В отношении необобщенных коллекций важно иметь в виду следующее: они оперируют данными типа object. Таким образом, необобщенные коллекции могут служить

для хранения данных любого типа, причем в одной коллекции допускается наличие разнотипных данных. Очевидно, что такие коллекции не типизированы, поскольку в них хранятся ссылки на данные типа object. Классы и интерфейсы необобщенных коллекций находятся в пространстве имен System.Collections.

Специальные коллекции

Оперируют данными конкретного типа или же делают это каким-то особым образом. Например, имеются специальные коллекции для символьных строк, а также специальные коллекции, в которых используется однонаправленный список. Специальные коллекции объявляются в пространстве имен System.Collections.Specialized.

Поразрядная коллекция

В прикладном интерфейсе Collections API определена одна коллекция с поразрядной организацией — это BitArray. Коллекция типа BitArray поддерживает поразрядные операции, т.е. операции над отдельными двоичными разрядами, например И, ИЛИ, исключающее ИЛИ, а следовательно, она существенно отличается своими возможностями от остальных типов коллекций. Коллекция типа BitArray объявляется в пространстве имен System.Collections.

Обобщенные коллекции

Обеспечивают обобщенную реализацию нескольких стандартных структур данных, включая связные списки, стеки, очереди и словари. Такие коллекции являются типизированными в силу их обобщенного характера. Это означает, что в обобщенной коллекции могут храниться только такие элементы данных, которые совместимы по типу с данной коллекцией. Благодаря этому исключается случайное несовпадение типов. Обобщенные коллекции объявляются в пространстве имен System.Collections.Generic.

Параллельные коллекции

Поддерживают многопоточный доступ к коллекции. Это обобщенные коллекции, определенные в пространстве имен System.Collections.Concurrent.

Коллекция, содержащаяся только отличающиеся элементы, называется множеством (set). В составе .NET 4 имеются два множества — **HashSet<T> и SortedSet<T>.** Оба они реализуют интерфейс **ISet<T>.** Класс **HashSet<T>** содержит неупорядоченный список различающихся элементов, а в **SortedSet<T>** элементы упорядочены.

Имя	Описание
Add	Добавляет указанный элемент в набор.
Clear	Удаляет все элементы из объекта HashSet<T>.
Contains	Определяет, содержит ли объект HashSet<T> указанный элемент.
CopyTo(T[])	Копирует элементы объекта HashSet<T> в массив.
Equals(Object)	Определяет, равен ли заданный объект текущему объекту. (Унаследовано от Object.)
Overlaps	Определяет, имеются ли общие элементы в текущем объекте HashSet<T> и в заданной коллекции.
Remove	Удаляет указанный элемент из объекта HashSet<T>.

Задание 22. Программа, реализующая основные действия над множествами с помощью SortedSet<T>

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

namespace ConsoleApplication1
```

```
{
  class Program
  {
    static void Main(string[] args)
    {
    // Создадим два множества символьных элемен-
    тов
    SortedSet<char>     ss     =     new
    SortedSet<char>();
    SortedSet<char>     ss1     =     new
    SortedSet<char>();

    //добавлеям элементы 1 множества
    ss.Add('A');
    ss.Add('B');
    ss.Add('C');
    ss.Add('Z');
    ShowColl(ss, "Первая коллекция: ");

    //добавлеям элементы 2 множества
    ss1.Add('A');
    ss1.Add('B');
    ss1.Add('C');
    ShowColl(ss1, "Вторая коллекция");

    if      (ss.SetEquals(ss1))      Con-
    sole.WriteLine("Равны");
    else Console.WriteLine("не равны");
    Console.WriteLine();

    ss.SymmetricExceptWith(ss1);
    ShowColl(ss,"Исключили    разноимен-
    ность двух множеств: ");

    ss.UnionWith(ss1);
    ShowColl(ss,"Объединение множеств:");

    ss.ExceptWith(ss1);
    ShowColl(ss, "Вычитание множеств");
```

```
      Console.ReadLine();
      }

      static void
      ShowColl(SortedSet<char> ss, string
      s) // Метод посимвольного вывода на консоль
      {
      Console.WriteLine(s);
      //вывод на консоль, используя цикл со счетчиком
      "До конца массива"
      foreach (char ch in ss)
      Console.Write(ch + " ");
      Console.WriteLine("\n");
      }
   }
}
```

Задание: в предыдущей программе заменить SortedSet<T> на HashSet<T>. Проверить особенности HashSet<T> на примере неупорядоченных множеств чисел.

Для получения дополнительной информации перейдите по ссылке
http://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx _
руководство по программированию на C#

Самостоятельная работа.

Множества.

Для решения некоторых задач можно использовать регулярные выражения.

1. Имеются три множества символьного типа, которые заданы своими конструкторами:

Y1=['A','B','D','R','H']

Y2=['R','A','H','D']
Y3=['A','R']

Сформировать новое множество

X=(Y1|Y2) U (Y1|Y2)

Вывести на печать полученное множество.

Далее проверить, включено ли множество Y3 во множество X.

2. Из множества целых чисел 1..20 выделить:

а) множество чисел, делящихся на 6 без остатка;

b) множество чисел, делящихся без остатка или на 2 или на 3.

3. Напишите программу, переводящую школьные отметки в школьные оценки.
4. Определить количество гласных букв латинского алфавита в некотором тексте.
5. В заданной строке, состоящей из букв латинского алфавита и оканчивающейся точкой, определить общее число вхождений в нее букв 'a','e','c','h'.
6. Из первой сотни натуральных чисел найти все простые с помощью решета Эратосфена. Для представления решета используйте множество!
7. Переменной S присвоить:

а) пустое множество;

б) множество из строчных гласных латинских букв (a, e, i, o, u);

г) множество из всех цифр.

8. Вычислить значения отношений:

а) [2]<>[2,2,2]; б) ['a','b']=['b','a'];

в) [4,5,6]=[4..6]; г) ['c','b']=['c'..'b'];

д) [2,3,5,7]<=[1..9]; е) [3,6..8]<=[2..7,9];

ж) []<=['0'..'9']; з) 'q' in ['a'..'z'];

и) [2]<[1..3]; к) 66=[66]

9. Дано множество ень_недели = (пн,вт,ср,чт,пт,сб,вс);

Описать множественный тип, включающий в себя множества из:

а) названий любых дней недели;

б) названий рабочих дней недели.

10. Месяц=1..12;
    Написать программу, определяющую количество дней в месяце m невисокосного года.
11. letters= множество от 'a'..'z';
    Написать программу, печатающую в алфавитном порядке все элементы множества A, имеющего тип letters.
12. Вычислить значения выражений:

а) [1,3,5]+[2,4]; б) [1,3,5]*[2,4]; в) [1,3,5]-[2,4];
г) [1..6]+[3..8]; д) [1..6]*[3..8]; и) [2,4]-[1..5];
к) []+[4]; л) []*[4]; м) []-[4];

13. M = 0..99;
    Написать программу, подсчитывающую количество элементов в множестве A типа M. Например, для множества [5,8,23] должно быть выведено 3

Написать отчет по лабораторной работе.

Лабораторная работа № 8

Краткое описание:

- Описание Классов
- Варианты доступа к полям и методам Класса
- Процедуры и функции

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение
Классы. Описание и структура.

Синтаксис класса
Синтаксис описания класса:
[атрибуты][модификаторы]class
имя_класса[:список_родителей]
{тело_класса}

Возможными модификаторами в объявлении класса могут быть модификаторы new, abstract, sealed, о кото-

рых подробно будет говориться при рассмотрении наследования, и четыре модификатора доступа, два из которых - private и protected - могут быть заданы только для вложенных классов. Обычно класс имеет атрибут доступа public, являющийся значением по умолчанию. Так что в простых случаях объявление класса выглядит так:

public class Rational {тело_класса}

В теле класса могут быть объявлены:

- константы;
- поля;
- конструкторы и деструкторы;
- методы;
- события;
- делегаты;
- классы (структуры, интерфейсы, перечисления).

Поля класса

Поля класса синтаксически являются обычными переменными (объектами) языка. Их описание удовлетворяет обычным правилам объявления переменных.

Содержательно поля задают представление той самой абстракции данных, которую реализует класс. Поля характеризуют свойства объектов класса. Когда создается новый объект класса (в динамической памяти или в стеке), то этот объект представляет собой набор полей класса. Два объекта одного класса имеют один и тот же набор полей, но разнятся значениями, хранимыми в этих полях.

Доступ к полям

Каждое поле имеет модификатор доступа, принимающий одно из четырех значений: public, private, protected, internal. Атрибутом доступа по умолчанию является атрибут private.

Методы класса

Методы класса синтаксически являются обычными процедурами и функциями языка. Их описание удовле-

творяет обычным правилам объявления процедур и функций.

Методы-свойства

Методы, называемые свойствами (Properties), представляют специальную синтаксическую конструкцию, предназначенную для обеспечения эффективной работы со свойствами. При работе со свойствами объекта (полями) часто нужно решить, какой модификатор доступа использовать, чтобы реализовать нужную стратегию доступа к полю класса. Перечислю пять наиболее употребительных стратегий:

• чтение, запись (Read, Write);
• чтение, запись при первом обращении (Read, Write-once);
• только чтение (Read-only);
• только запись (Write-only);
• ни чтения, ни записи (Not Read, Not Write).

Общий синтаксис методов-свойств. Пусть name - это закрытое свойство. Тогда для него можно определить открытый метод-свойство (функцию), возвращающую тот же тип, что и поле name. Имя метода обычно близко к имени поля (например, Name). Тело свойства содержит два метода - get и set, один из которых может быть опущен. Метод get возвращает значение закрытого поля, метод set - устанавливает значение, используя передаваемое ему значение в момент вызова хранящееся в служебной переменной со стандартным именем value. Поскольку get и set - это обычные процедуры языка, то программно можно реализовать сколь угодно сложные стратегии доступа.

Индексаторы

Свойства являются частным случаем метода класса с особым синтаксисом. Еще одним частным случаем является индексатор. Метод-индексатор является обобщением метода-свойства. Он обеспечивает доступ к закрытому полю, представляющему массив. Объекты класса индексируются по этому полю.

Синтаксически объявление индексатора - такое же, как и в случае свойств, но методы get и set приобретают аргументы по числу размерности массива, задающего индексы элемента, значение которого читается или обновляется. Важным ограничением является то, что у класса может быть только один индексатор и у этого индексатора стандартное имя this. Так что если среди полей класса есть несколько массивов, то индексация объектов может быть выполнена только по одному из них.

Задание 23.
Методы-свойства. Индексаторы. (Класс Person)
Пример программы, использующей различные стратегии доступа к данным.
Задание: добавить вторую персону по фамилии Иванов, статус – пенсионер, имеющий 3 детей: служащего, студента и школьника.

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

//демонстрация методов-свойств и индексатора

namespace ConsoleApplication1
{
  class Program
  {
   static void Main(string[] args)
   {
   Person pers1 = new Person(), pers2 =
   new Person(), pers3 = new Person();
   // поля получают значения по умолчанию

   // Задание значений методов-свойств
   pers1.Fam = "Петров";
```

```
    pers1.Vozr = 42;
    pers1.Zarpl = 10000;

    // Вывод значений методов-свойств
    Console.WriteLine("Фамилия {0} Воз-
    раст {1} Статус {2}", pers1.Fam,
    pers1.Vozr, pers1.Status);

    // Дети
    pers2.Fam = "Петров";
    pers2.Vozr = 21;
    pers3.Fam = "Петрова";
    pers3.Vozr = 5;
    pers1[0] = pers2;
    pers1[1] = pers3;

    // вывод данных детей
    Console.WriteLine("Сведения о де-
    тях");
    for (int i=0;i<pers1.Kol_det;i++)
    Console.WriteLine("Фамилия {0} Воз-
    раст {1} Статус {2}", pers1[i].Fam,
    pers1[i].Vozr, pers1[i].Status);
    Console.ReadLine();
    }
   }
}
```

Класс Person

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

namespace ConsoleApplication1
{
/// <summary>
/// Класс Персона
/// </summary>
  class Person
```

```
{
// поля (закрытые). Начальные значения полей при задании конструктора без параметров
   private string fam = "", status = "";
   private int vozr = 0, zarpl = 0, kol_det = 0;
   private Person[] deti = new Person[10]; // массив Дети, выделена память под макс. число детей

/// <summary>
/// Метод-свойство Read, Write-once
/// </summary>
   public string Fam
   {
   set { if (fam == "") fam = value; }
   get { return fam; }
   }

/// <summary>
/// Метод-свойство: Только Read
/// </summary>
   public string Status
   {
   get { return status; }
   }

/// <summary>
/// Метод-свойство: Чтение/Запись. Можно изменять и другие поля
/// </summary>

   public int Vozr
   {
   set
   {
   vozr = value;
   if (vozr < 7) status = "ребенок";
```

```
   else if (vozr < 17) status =
   "школьник";
   else if (vozr < 22) status = "сту-
   дент";
   else if (vozr < 60) status = "слу-
   жащий";
   else status = "пенсионер";
   }
   get { return vozr; }
   }

/// <summary>
/// Метод-свойство: Только запись
/// </summary>


   public int Zarpl
   {
   set { zarpl = value; }
   }

/// <summary>
/// Метод-свойство: только чтение
/// </summary>
   public int Kol_det
   {
   get { return kol_det; }
   }
/// <summary>
/// Индексатор. Массив Дети
/// </summary>
/// <param name="i"></param>
/// <returns></returns>
   public Person this[int i]
   // индексатор всегда имеет имя this
   {
   get
   {
```

```
    if (i > 0 && i < kol_det) return
    deti[i];
    else return deti[0];
    }
    set
    {
    if (i >= 0 && i < 10)
    {
    if (i <= kol_det) deti[i] = value;
    // либо заменяем данные i-го ребенка
    if (i == kol_det) kol_det++;
    // либо вводим нового
    }
    }
    }
  }
}
```

Процедуры и функции

Процедуры и функции. Отличия.

Функция отличается от процедуры двумя особенностями:

• всегда вычисляет некоторое значение, возвращаемое в качестве результата функции;

• вызывается в выражениях.

Процедура C# имеет свои особенности:

• возвращает формальный результат void, указывающий на отсутствие результата;

• вызов процедуры является оператором языка;

• имеет входные и выходные аргументы, причем выходных аргументов - ее результатов - может быть достаточно много.

Функции с переменным числом параметров

В стек помещается только массив объектов, а сами объекты размещаются теперь в куче, а не на стеке (в отличии от C++).

Для компилятора указывается ключевое слово params:

Задание 24: Набрать исходный код программы, проанализировать ее работу для разных исходных данных. Изменить код программы так, чтобы в качестве аргументов передавался массив чисел, набранных с клавиатуры.

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

namespace ConsoleApplication1
{
  class Program
  {
   static void F(params int[] args)
   //функция,выводящая аргументы
   {
   Console.WriteLine("Количество аргу-
   ментов: {0}", args.Length);
   //Количество аргументов по длине массива пара-
   метров
   for (int i = 0; i < args.Length;
   i++)
   {
   Console.WriteLine("\targs[{0}]      =
   {1}", i, args[i]); //аргументы функции
   }
   }
   static void Main()
   {
   F();//аргументы отсутствуют
   F(1);
   F(1, 2);
   F(1, 2, 3);
   F(new int[] { 1, 2, 3, 4 });
   //аргументы в виде массива целых чисел
```

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

Примерный результат выполнения программы:

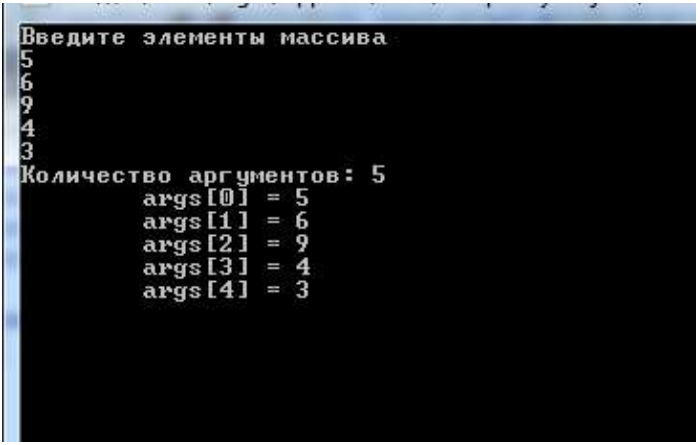


Рисунок 8. Пример работы программы

Задание 25: **Перегрузка операций. Программа вычисления простых дробей.**

Набрать исходный код программы, проанализировать ее работу для разных исходных данных. Изменить код программы – добавить умножение дробей.

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

   // тема: перегрузка операций
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
```

```
        Rational a = new Rational(2, 6);
        Rational b = new Rational(-12, -16);
        Rational c = new Rational(2, 12);

        a.print("a ");
        b.print("b ");
        c.print("c ");


        Rational d = a + b;
        Rational e = d - c;
        Rational f = d * e;
        Rational k = (d + e - f)/(a - b);

        d.print("d = a + b ");
        e.print("e = d - c ");
        k.print("k = (d + e - f)/(a - b) ");

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

Класс Rational

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

namespace ConsoleApplication1
{
/// <summary>
/// Класс рациональных чисел.
/// Рациональное число представляет собой несократимую дробь вида m/n
/// где m и n - целые числа
/// </summary>
   class Rational
   {
```

```
// поля класса (закрытые)
int m, n;

/// <summary>
/// Конструктор. Создает несократимую дробь.
/// Если b=0, то создается число 0 (числитель 0,
///знаменатель 1)
/// иначе сокращается
/// </summary>
///<param name="a"> числитель </param>
///<param name="b"> знаменатель </param>
public Rational(int a, int b)
{
if (b == 0) { this.m = 0; this.n = 1; }
else
{
// приведение знака
if (b < 0) { b = -b; a = -a; }
// приведение к несократимой дроби
int d = nod(a, b);
// наибольший общий делитель
this.m = a / d; this.n = b / d;
}
}

/// <summary>
/// Вычисление наибольшего общего делителя
///целых чисел a и b
/// по алгоритму Евклида
/// </summary>
/// <param name="a"></param>
/// <param name="b"></param>
/// <returns></returns>
static private int nod(int a, int b)
{
int c = 0; // Вспомогательная переменная.
Инициализация обязательна по нормам языка

a = Math.Abs(a); b = Math.Abs(b);
```

```
// приводим к числам без знака

if (b > a) { c = a; a = b; b = c; }
// приводим к виду a>b

do  // цикл поиска НОД
{
c = a % b; a = b; b = c;
}
while (b != 0);

return a;
}

/// <summary>
/// Вывод рационального числа
/// </summary>
/// <param name="soob">описание числа</param>

public void print(string soob)
{
Console.WriteLine("{0}     {1}/{2}",
soob,this.m, this.n);
}

/// <summary>
/// перегруженная операция сложения
/// </summary>
/// <param name="r1">первое слагаемое</param>
/// <param name="r2">второе слагаемое</param>
/// <returns></returns>

public  static  Rational  operator
+(Rational r1, Rational r2)
{
int m1 = r1.m * r2.n + r2.m * r1.n;
int n1 = r1.n * r2.n;

return new Rational(m1, n1);
```

```
        }


        /// <summary>
        /// перегруженная операция вычитания
        /// </summary>
        /// <param name="r1"></param>
        /// <param name="r2"></param>
        /// <returns></returns>

        public static Rational operator -
        (Rational r1, Rational r2)
        {
        int m1 = r1.m * r2.n - r2.m * r1.n;
        int n1 = r1.n * r2.n;

        return new Rational(m1, n1);
        }


        /// <summary>
        /// перегруженная операция деления
        /// </summary>
        /// <param name="r1"></param>
        /// <param name="r2"></param>
        /// <returns></returns>
        public  static  Rational  operator
        /(Rational r1, Rational r2)
        {
        int m1 = r1.m * r2.n;
        int n1 = r1.n * r2.m;

        return new Rational(m1, n1);
        }
    }
}
```

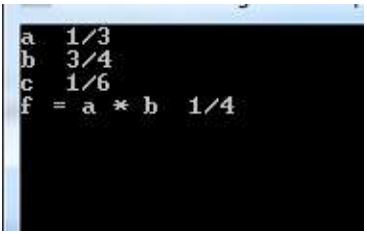


Рисунок 9. Пример работы программы

Самостоятельная работа

1. Описать функцию Sign(X) целого типа, возвращающую для вещественного числа X следующие значения:
   –1, если X < 0; 0, если X = 0; 1, если X > 0.
2. С помощью этой функции найти значение выражения Sign(A) + Sign(B) для данных вещественных чисел A и B.

3. Описать функцию RootsCount(A, B, C) целого типа, определяющую количество корней квадратного уравнения A·x2 + B·x + C = 0 (A, B, C — вещественные параметры, A □ 0). С ее помощью найти количество корней для каждого из трех квадратных уравнений с данными коэффициентами. Количество корней определять по значению дискриминанта: D = B2 – 4·A·C.

4. Описать функцию CircleS(R) вещественного типа, находящую площадь круга радиуса R (R — вещественное). С помощью этой функции найти площади трех кругов с данными радиусами. Площадь круга радиуса R вычисляется по формуле $S = \pi \cdot R^2$. В качестве значения π использовать 3.14.

5. Описать функцию RingS(R1, R2) вещественного типа, находящую площадь кольца, заклю-

ченного между двумя окружностями с общим центром и радиусами R1 и R2 (R1 и R2 — вещественные, R1 > R2). С ее помощью найти площади трех колец, для которых даны внешние и внутренние радиусы. Воспользоваться формулой площади круга радиуса R: $S = \pi \cdot R^2$. В качестве значения π использовать 3.14.

6. Описать функцию TriangleP(a, h), находящую периметр равнобедренного треугольника по его основанию a и высоте h, проведенной к основанию (a и h — вещественные). С помощью этой функции найти периметры трех треугольников, для которых даны основания и высоты. Для нахождения боковой стороны b треугольника использовать теорему Пифагора:$b2 = (a/2)^2 + h^2$.

7. Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых чисел от A до B включительно (A и B — целые). Если A > B, то функция возвращает 0. С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.

8. Описать функцию Calc(A, B, Op) вещественного типа, выполняющую над ненулевыми вещественными числами A и B одну из арифметических операций и возвращающую ее результат. Вид операции определяется целым параметром Op: 1 — вычитание, 2 — умножение, 3 — деление, остальные значения — сложение. С помощью Calc выполнить для данных A и B операции, определяемые данными целыми N1, N2, N3.

9. Описать функцию Quarter(x, y) целого типа, определяющую номер координатной четверти, в которой находится точка с ненулевыми веще-

ственными координатами (x, y). С помощью этой функции найти номера координатных четвертей для трех точек с данными ненулевыми координатами.

10. Описать функцию Even(K) логического типа, возвращающую True, если целый параметр K является четным, и False в противном случае. С ее помощью найти количество четных чисел в наборе из 10 целых чисел.

11. Описать функцию IsSquare(K) логического типа, возвращающую True, если целый параметр K (> 0) является квадратом некоторого целого числа, и False в противном случае. С ее помощью найти количество квадратов в наборе из 10 целых положительных чисел.

12. Описать функцию IsPower5(K) логического типа, возвращающую True, если целый параметр K (> 0) является степенью числа 5, и False в противном случае. С ее помощью найти количество степеней числа 5 в наборе из 10 целых положительных чисел.

13. Описать функцию IsPowerN(K, N) логического типа, возвращающую True, если целый параметр K (> 0) является степенью числа N (> 1), и False в противном случае. Дано число N (> 1) и набор из 10 целых положительных чисел. С помощью функции IsPowerN найти количество степеней числа N в данном наборе.

14. Описать функцию IsPrime(N) логического типа, возвращающую True, если целый параметр N (> 1) является простым числом, и False в противном случае (число, большее 1, называется простым, если оно не имеет положительных делителей, кроме 1 и самого себя). Дан набор из 10 целых чисел, больших 1.

С помощью функции IsPrime найти количество простых чисел в данном наборе.

15. Описать функцию DigitCount(K) целого типа, находящую количество цифр целого положительного числа K. Используя эту функцию, найти количество цифр для каждого из пяти данных целых положительных чисел.

16. Описать функцию DigitN(K, N) целого типа, возвращающую N-ю цифру целого положительного числа K (цифры в числе нумеруются справа налево). Если количество цифр в числе K меньше N, то функция возвращает –1. Для каждого из пяти данных целых положительных чисел K1, K2, …, K5 вызвать функцию DigitN с параметром N, изменяющимся от 1 до 5.

17. Описать функцию IsPalindrom(K), возвращающую True, если целый параметр K (> 0) является палиндромом (то есть его запись читается одинаково слева направо и справа налево), и False в противном случае. С ее помощью найти количество палиндромов в наборе из 10 целых положительных чисел. При описании функции можно использовать функции DigitCount и DigitN из заданий 14 и 15.

18. Описать функцию DegToRad(D) вещественного типа, находящую величину угла в радианах, если дана его величина D в градусах (D — вещественное число, 0 < D < 360). Воспользоваться следующим соотношением: 180° = π радианов. В качестве значения π использовать 3.14. С помощью функции DegToRad перевести из градусов в радианы пять данных углов.

19. Описать функцию RadToDeg(R) вещественного типа, находящую величину угла в градусах, если дана его величина R в радианах (R — вещественное число, $0 < R < 2 \cdot \pi$). Воспользоваться следующим соотношением: $180° = \pi$ радианов. В качестве значения π использовать 3.14. С помощью функции RadToDeg перевести из радианов в градусы пять данных углов.

20. Описать функцию Fact(N) вещественного типа, вычисляющую значение факториала $N! = 1 \cdot 2 \cdot \ldots \cdot N$ (N > 0 — параметр целого типа; вещественное возвращаемое значение используется для того, чтобы избежать целочисленного переполнения при больших значениях N). С помощью этой функции найти факториалы пяти данных целых чисел.

21. Описать функцию Fact2(N) вещественного типа, вычисляющую двойной факториал:
    $N!! = 1 \cdot 3 \cdot 5 \cdot \ldots \cdot N$, если N — нечетное;
    $N!! = 2 \cdot 4 \cdot 6 \cdot \ldots \cdot N$, если N — четное
22. (N > 0 — параметр целого типа; вещественное возвращаемое значение используется для того, чтобы избежать целочисленного переполнения при больших значениях N). С помощью этой функции найти двойные факториалы пяти данных целых чисел.

23. Описать функцию Fib(N) целого типа, вычисляющую N-й элемент последовательности чисел Фибоначчи FK, которая описывается следующими формулами:
    $F1 = 1$, $F2 = 1$, $FK = FK{-}2 + FK{-}1$,
    $K = 3, 4, \ldots$.
    Используя функцию Fib, найти пять чисел Фибоначчи с данными номерами N1, N2, …, N5.

Написать отчет по лабораторной работе.

Лабораторная работа № 9

Краткое описание:

- Отношения между классами
- Наследование
- Сериализация объектов

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Наследование. Абстрактные классы. Абстрактные, полиморфные, заменяемые, специфические методы.

Отношения между классами

Каждый класс, как не раз отмечалось, играет две роли: он является модулем - архитектурной единицей, и он имеет содержательный смысл, определяя некоторый тип данных. Но классы программной системы - это ансамбль, в котором классы, играя свои роли, не являются независимыми - все они находятся в определенных отношениях друг с другом. Два основных типа отношений между классами определены в ОО-системах. Первое отношение "клиенты и поставщики", называется часто клиентским отношением или отношением вложенности (встраивания). Второе отношение "родители и наследники" называется отношением наследования.

Определение 1. Классы A и B находятся в отношении "клиент-поставщик", если одним из полей класса B является объект класса A. Класс A называется поставщиком класса B, класс B называется клиентом класса A.

Определение 2. Классы A и B находятся в отношении "родитель - наследник", если при объявлении класса B

класс А указан в качестве родительского класса. Класс А называется родителем класса В, класс В называется наследником класса А.
Оба отношения - наследования и вложенности - являются транзитивными. Если В - клиент А и С - клиент В, то отсюда следует, что С - клиент А. Если В - наследник А и С - наследник В, то отсюда следует, что С - наследник А.

Наследование
Мощь ООП основана на наследовании. Когда построен полезный класс, то он может многократно использоваться. Повторное использование - это одна из главных целей ООП. Но и для хороших классов неизбежно наступает момент, когда необходимо расширить возможности класса, придать ему новую функциональность, изменить интерфейс. Всякая попытка изменять сам работающий класс чревата большими неприятностями - могут перестать работать прекрасно работавшие программы, многим клиентам класса вовсе не нужен новый интерфейс и новые возможности. Здесь-то и приходит на выручку наследование. Существующий класс не меняется, но создается его потомок, продолжающий дело отца, только уже на новом уровне.
Класс-потомок наследует все возможности родительского класса - все поля и все методы, открытую и закрытую часть класса. Правда, не ко всем полям и методам класса возможен прямой доступ потомка. Поля и методы родительского класса, снабженные атрибутом private, хотя и наследуются, но по-прежнему остаются закрытыми, и методы, создаваемые потомком, не могут к ним обращаться напрямую, а только через методы, наследованные от родителя. Единственное, что не наследует потомок - это конструкторы родительского класса.

Задание 26: Набрать код программы, проверить выполнение для различных вариантов входных данных. Самостоятельно добавить Класс Треугольник – наслед-

ник от Класса Фигура и посчитать площадь и периметр равнобедренного треугольника.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    Figure f1 = new Krug(2.5);
    Figure f2 = new Kvadrat(3.2);
    Figure f3 = new Prjamoug(2.5,3.2);


    Console.WriteLine("\nФигура f1");
    f1.Print();  // вызывается метод базового типа
    ((Krug)f1).Print();  // вызывается метод приведенного типа
    Console.WriteLine("Площадь    {0:F3}
    Периметр    {1:F3}",    f1.Ploshad(),
    f1.Perimetr());

    Console.WriteLine("\nФигура f2");
    f2.Print();
    ((Kvadrat)f2).Print();
    Console.WriteLine("Площадь    {0:F3}
    Периметр    {1:F3}",    f2.Ploshad(),
    f2.Perimetr());
    Console.WriteLine("Диагональ
    {0:F3}",  ((Kvadrat)f2).Diagonal());
    // приведение типа здесь обязательно

    Console.WriteLine("\nФигура f3");
```

```
      f3.Print();
      ((Prjamoug)f3).Print();
      Console.WriteLine("Площадь    {0:F3}
      Периметр    {1:F3}",    f3.Ploshad(),
      f3.Perimetr());
      Console.WriteLine("Диагональ
      {0:F3}",
      ((Prjamoug)f3).Diagonal());
      Console.ReadLine();
      }
   }
}
```

Класс Фигура

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

namespace ConsoleApplication1
{
/// <summary>
/// Абстрактный класс: двумерная фигура
/// </summary>
   abstract class Figure
   {
   protected double a;  // характеристика фигуры (радиус, сторона и т.п.)

   /// <summary>
   /// конструктор
   /// </summary>
   /// <param name="a"></param>
   public Figure(double a)
   {
   this.a = a;
   }

   /// <summary>
```

```
    /// Абстрактный метод: вычисление площади
    /// </summary>
    /// <returns></returns>
    public abstract double Ploshad();


    /// <summary>
    /// Абстрактный метод: вычисление периметра
    /// </summary>
    /// <returns></returns>
    public abstract double Perimetr();

    /// <summary>
    /// Вывод
    /// </summary>
    /// <returns></returns>
    public void Print()
    {
    Console.WriteLine("Характеристика:
    {0:F3}",this.a);
    }
    }
}
```

Класс Круг - наследник от Класса Фигура

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

namespace ConsoleApplication1
{
  class Krug:Figure // Наследник класса Figure
  {
    public Krug(double a): base(a) // кон-
    структор (не наследуется)
    {
    }
```

```
    public override double Ploshad() // полиморфный метод
    {
    return Math.PI*this.a*this.a;
    }

    public override double Perimetr() // полиморфный метод
    {
    return 2*Math.PI * this.a ;
    }

    public new void Print() // заменяемый метод
    {
    Console.WriteLine("Радиус: {0:F3}", this.a);
    }
  }
}
```

Класс Квадрат - наследник от Класса Фигура

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

namespace ConsoleApplication1
{
  class Kvadrat : Figure
  // Наследник класса Figure
  {
    public Kvadrat(double a): base(a)
    // конструктор (не наследуется)
    {
    }
    public override double Ploshad()
    // полиморфный метод
    {
```

```
    return this.a * this.a;
    }
    public override double Perimetr()
    // полиморфный метод
    {
    return 4 * this.a;
    }
    public new void Print()
    // заменяемый метод
    {
    Console.WriteLine("Сторона:
    {0:F3}", this.a);
    }

    public double Diagonal()
    // специфический метод наследника
    {
    return Math.Sqrt(2) * this.a;
    }
  }
}
```

Класс Прямоугольник - наследник от Класса Фигура

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

namespace ConsoleApplication1
{
  /// <summary>
  /// Прямоугольник
  /// </summary>
  class Prjamoug : Figure // Наследник
  класса Figure
  {
    protected double b; // вторая сторона
    public Prjamoug(double a, double b)
```

```
    : base(a)  // конструктор (не наследуется)
    {
    this.b = b;
    }
    public override double Ploshad()
    // полиморфный метод
    {
    return this.a*this.b;
    }
    public override double Perimetr()
    // полиморфный метод
    {
    return 2*(this.a+this.b) ;
    }
    public new void Print()
    // заменяемый метод
    {
    Console.WriteLine("Сторона1:  {0:F3}
    Сторона2: {0:F3}", this.a, this.b);
    }
    public double Diagonal()
    // специфический метод наследника
    {
    return   Math.Sqrt(this.a*this.a   +
    this.b*this.b) ;
    }
  }
}
```

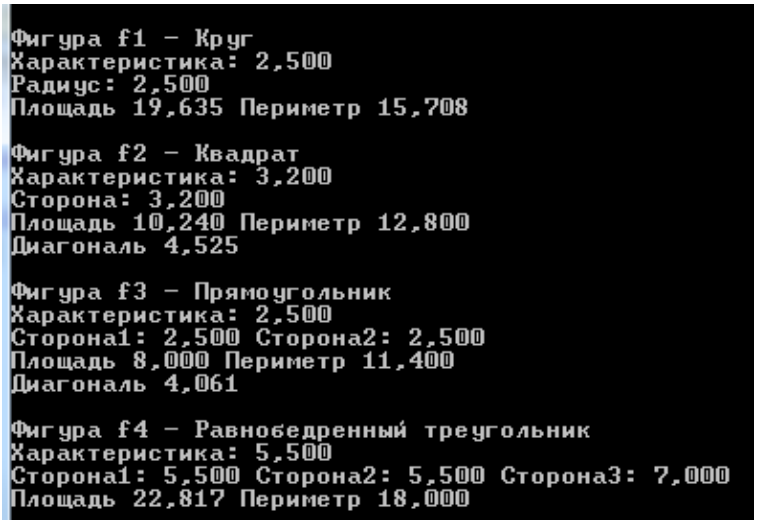


Рисунок 10. Примерный вывод данных программы

Интерфейсы. (Класс Person).

Интерфейс представляет собой полностью абстрактный класс, все методы которого абстрактны. От абстрактного класса интерфейс отличается некоторыми деталями в синтаксисе и поведении. Синтаксическое отличие состоит в том, что методы интерфейса объявляются без указания модификатора доступа. Отличие в поведении заключается в более жестких требованиях к потомкам. Класс, наследующий интерфейс, обязан полностью реализовать все методы интерфейса. В этом - отличие от класса, наследующего абстрактный класс, где потомок может реализовать лишь некоторые методы родительского абстрактного класса, оставаясь абстрактным классом.

Более подробно про Интерфейсы можно почитать по ссылке

http://msdn.microsoft.com/ru-ru/library/ms173156.aspx

Встроенные интерфейсы

Являются частью библиотеки FCL. Они используются многими классами-библиотеками так же, как и классами, создаваемыми пользователем.

Упорядоченность объектов и интерфейс IComparable

Часто, когда создается класс, желательно задать отношение порядка на его объектах. Такой класс следует объявить наследником интерфейса IComparable. Этот интерфейс имеет всего один метод CompareTo (object obj), возвращающий целочисленное значение, положительное, отрицательное или равное нулю, в зависимости от выполнения отношения "больше", "меньше" или "равно".

Задание 27: Набрать код программы, проверить выполнение для различных вариантов входных данных. Добавить Семью Сидоровых: Николай, Ирина, Михаил и Светлана. Николай и Ирина – родители, Михаил – супруг Петровой Юлии (Персона №4)

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    Person[] p = new Person[6]; // массив
    персон

    p[0] = new Person("Иванов", "Иван",
    "23.06.1960");
    p[1]   =   new   Person("Иванова",
    "Мария", "14.03.1964");
```

```
        p[2]    =    new    Person("Иванов",
        "Михаил", "01.02.1990");
        p[3]    =    new    Person("Иванова",
        "Анна", "15.09.1992");
        p[4]    =    new    Person("Петрова",
        "Юлия", "18.10.1966");
        p[5] = new Person("Петров", "Влади-
        мир", "16.12.1991");

        p[0].Suprug = p[1];
        p[1].Suprug = p[0];
        p[2].Father = p[0];
        p[2].Mother = p[1];
        p[3].Father = p[0];
        p[3].Mother = p[1];
        p[5].Mother = p[4];
        p[3].Suprug = p[5];
        p[5].Suprug = p[3];


        Array.Sort(p);  // сортируем массив, исходя
        из метода интерфейса ICompatible

        for (int i = 0; i < 6; i++)
        p[i].Print();

        }
    }
}
```

Класс Персона

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

namespace ConsoleApplication1
{
/// <summary>
```

```
/// Класс: Персона с полями того же класса и с интерфейсом сравнения
/// </summary>
   class Person:IComparable
   {
      // поля класса (закрытые)
      string fam, im;
      DateTime dr;
      Person father, mother, suprug;


      // конструктор
      public Person(string fam, string
      im, string dr)
      {
      this.fam = fam;
      this.im = im;
      this.dr = Convert.ToDateTime(dr);
      }

      // методы-свойства
      public Person Father
      {
      set { father = value; }
      }

      public Person Mother
      {
      set { mother = value; }
      }

      public Person Suprug
      {
      set { suprug = value; }
      }

      // переопределенный метод родительского класса
      object
      public override string ToString()
```

```
{
return this.fam.ToString() + " " +
this.im.ToString()    +    "    "    +
this.dr.ToString("d");   //   "d"   -
краткий формат даты
}

// метод интерфейса IComparable
public int CompareTo(object pers)
{
Person p = pers as Person;  // as при-
водит к типу. Если невозможно, возвращает null

if (p == null)
throw (new ArgumentException()); //
выбрасывает исключение, если сравнение не с
переменной класса Person
else
if (this.dr.CompareTo(p.dr) == 0)
return this.fam.CompareTo(p.fam); //
в порядке фамилий, если даты рождения одина-
ковы
else
return this.dr.CompareTo(p.dr); // в
порядке старшинства
}

// вывод всех данных о персоне
public void Print()
{
Console.WriteLine("\nПерсона   {0}",
this);

if (this.father != null)
Console.WriteLine("Отец        {0}",
this.father);

if (this.mother!=null)
```

```
    Console.WriteLine("Мать        {0}",
    this.mother);

    if (this.suprug!=null)
    Console.WriteLine("Супруг(а)   {0}",
    this.suprug);
    }
  }
}
```

Сериализация объектов (сказка о рыбаке и рыбке)
Под сериализацией понимают процесс сохранения объектов в долговременной памяти (файлах) в период выполнения системы. Под десериализацией понимают обратный процесс - восстановление состояния объектов, хранимых в долговременной памяти. Механизмы сериализации C# и Framework .Net поддерживают два формата сохранения данных - в бинарном файле и XML-файле. В первом случае данные при сериализации преобразуются в бинарный поток символов, который при десериализации автоматически преобразуется в нужное состояние объектов. Другой возможный преобразователь (SOAP formatter) запоминает состояние объекта в формате xml.
Сериализация позволяет запомнить рубежные состояния системы объектов с возможностью последующего возвращения к этим состояниям. Она необходима, когда завершение сеанса работы не означает завершение вычислений. В этом случае очередной сеанс работы начинается с восстановления состояния, сохраненного в конце предыдущего сеанса работы.

Задание 28: Набрать код программы, проверить выполнение для различных вариантов входных данных.

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;
using System.Runtime.
Serialization.Formatters.Binary;
// для сериализации
using System.IO; // для объектов FileStream

// Тема: сериализация в форме XML
namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    const String filename =
    "state.bin"; // имя файла для сохранения
    состояния класса
    Personage starik = new
    Personage("Старик", 70);
    Personage staruha = new
    Personage("Старуха", 70);

    // соединение двух персонажей
    starik.Svadba(staruha);

    // первоначальное состояние персонажей
    staruha.IzmStatus("крестьянка",
    "корыто");
    starik.IzmStatus("рыбак", "сеть");

    // сохранение состояния персонажей
    BinaryFormatter bf = new
    BinaryFormatter();
    FileStream fs = new
    FileStream(filename,
    FileMode.Create, FileAccess.Write);
    bf.Serialize(fs, starik);
    fs.Close();
```

```
        // вывод начального состояния
        starik.Print("В начале сказки");

        // первое желание
        staruha.IzmStatus("дворянка", "име-
        ние");
        starik.IzmStatus("муж дворянки",
        "сеть");
        starik.Print("Первое желание");

        // второе желание
        staruha.IzmStatus("боярыня", "поме-
        стья");
        starik.IzmStatus("муж боярыни",
        "сеть");
        starik.Print("Второе желание");

        // третье желание
        staruha.IzmStatus("государыня",
        "страна");
        starik.IzmStatus("муж государыни",
        "сеть");
        starik.Print("Третье желание");

        // последнее желание

        // восстановление состояния персонажей
        fs = new FileStream(filename,
        FileMode.Open, FileAccess.Read);
        starik = (Person-
        age)bf.Deserialize(fs);
        fs.Close();

        starik.Print("Последнее желание");

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

Класс Персонаж

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


namespace ConsoleApplication1
{
// Персонаж сказки о рыбаке и рыбке
[Serializable] // можно сохранять
  class Personage
  {
    // поля класса (закрытые)
    private String im; // имя персонажа
    private int vozr; // возраст
    private String status;
    private String bogatstvo;
    private Personage suprug;


    // конструктор
    public Personage(String im, int
    vozr)
    { this.im = im; this.vozr = vozr; }

    // соединение двух персонажей
    public void Svadba(Personage s)
    {
    this.suprug = s;
    s.suprug = this;
    }

    // задание статуса и богатства персонажа
    public void IzmStatus(String sta-
    tus, String bogatstvo)
    {
    this.status = status;
```

```
        this.bogatstvo = bogatstvo;
        }

        // вывод данных персонажа
        public void Print(String sost)
        {
        if (sost.Length > 0) Con-
        sole.WriteLine("\n {0}",sost);
        Console.WriteLine("\n {0}", sost);
        Console.WriteLine("Имя: {0}, Воз-
        раст {1}, Статус {2}, Богатство
        {3}", this.im, this.vozr,
        this.status, this.bogatstvo);
        Console.WriteLine("Имя: {0}, Воз-
        раст {1}, Статус {2}, Богатство
        {3}",this.suprug.im,
        this.suprug.vozr,
        this.suprug.status,
        this.suprug.bogatstvo);
        }
    }
}
```

Самостоятельная работа

1. Описать процедуру PowerA3 (A, B), вычисляющую третью степень числа A и возвращающую ее в переменной B (A — входной, B — выходной параметр; оба параметра являются вещественными). С помощью этой процедуры найти третьи степени пяти данных чисел.
2. Описать процедуру PowerA234(A, B, C, D), вычисляющую вторую, третью и четвертую степень числа A и возвращающую эти степени соответственно в переменных B, C и D (A — входной, B, C, D — выходные параметры; все параметры являются вещественными).

С помощью этой процедуры найти вторую, третью и четвертую степень пяти данных чисел.

3. Описать процедуру Mean(X, Y, AMean, GMean), вычисляющую среднее арифметическое AMean = (X+Y)/2 и среднее геометрическое GMean = (X·Y)1/2 двух положительных чисел X и Y (X и Y — входные, AMean и GMean — выходные параметры вещественного типа). С помощью этой процедуры найти среднее арифметическое и среднее геометрическое для пар (A, B), (A, C), (A, D), если даны A, B, C, D.
4. Описать процедуру TrianglePS(a, P, S), вычисляющую по стороне a равностороннего треугольника его периметр P = 3·a и площадь S = a2·(3)1/2/4 (a — входной, P и S — выходные параметры; все параметры являются вещественными). С помощью этой процедуры найти периметры и площади трех равносторонних треугольников с данными сторонами.
5. Описать процедуру RectPS(x1, y1, x2, y2, P, S), вычисляющую периметр P и площадь S прямоугольника со сторонами, параллельными осям координат, по координатам (x1, y1), (x2, y2) его противоположных вершин (x1, y1, x2, y2 — входные, P и S — выходные параметры вещественного типа). С помощью этой процедуры найти периметры и площади трех прямоугольников с данными противоположными вершинами.
6. Описать процедуру DigitCountSum(K, C, S), находящую количество C цифр целого положительного числа K, а также их сумму S (K — входной, C и S — выходные параметры целого типа). С помощью этой процедуры найти количество и сумму цифр для каждого из пяти данных целых чисел.

7. Описать процедуру InvertDigits(K), меняющую порядок следования цифр целого положительного числа K на обратный (K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры поменять порядок следования цифр на обратный для каждого из пяти данных целых чисел.
8. Описать процедуру AddRightDigit(D, K), добавляющую к целому положительному числу K справа цифру D (D — входной параметр целого типа, лежащий в диапазоне 0–9, K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры последовательно добавить к данному числу K справа данные цифры D1 и D2, выводя результат каждого добавления.
9. Описать процедуру AddLeftDigit(D, K), добавляющую к целому положительному числу K слева цифру D (D — входной параметр целого типа, лежащий в диапазоне 1–9, K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры последовательно добавить к данному числу K слева данные цифры D1 и D2, выводя результат каждого добавления.
10. Описать процедуру Swap(X, Y), меняющую содержимое переменных X и Y (X и Y — вещественные параметры, являющиеся одновременно входными и выходными). С ее помощью для данных переменных A, B, C, D последовательно поменять содержимое следующих пар: A и B, C и D, B и C и вывести новые значения A, B, C, D.
11. Описать процедуру Minmax(X, Y), записывающую в переменную X минимальное из значений X и Y, а в переменную Y — максимальное из этих значений (X и Y — вещественные параметры, являющиеся одновременно входными и выходными). Используя четыре

вызова этой процедуры, найти минимальное и максимальное из данных чисел A, B, C, D.

12. Описать процедуру SortInc3(A, B, C), меняющую содержимое переменных A, B, C таким образом, чтобы их значения оказались упорядоченными по возрастанию (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры упорядочить по возрастанию два данных набора из трех чисел: (A1, B1, C1) и (A2, B2, C2).
13. Описать процедуру SortDec3(A, B, C), меняющую содержимое переменных A, B, C таким образом, чтобы их значения оказались упорядоченными по убыванию (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры упорядочить по убыванию два данных набора из трех чисел: (A1, B1, C1) и (A2, B2, C2).
14. Описать процедуру ShiftRight3(A, B, C), выполняющую правый циклический сдвиг: значение A переходит в B, значение B — в C, значение C — в A (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить правый циклический сдвиг для двух данных наборов из трех чисел: (A1, B1, C1) и (A2, B2, C2).
15. Описать процедуру ShiftLeft3(A, B, C), выполняющую левый циклический сдвиг: значение A переходит в C, значение C — в B, значение B — в A (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить левый циклический сдвиг для двух данных наборов из трех чисел: (A1, B1, C1) и (A2, B2, C2).

Написать отчет по лабораторной работе.

Лабораторная работа № 10

Краткое описание:

- делегаты как указатели функций
- основные операторы
- математические операторы
- операторы условия

Запустить Microsoft Visual Studio, выбрав приложение Microsoft Visual C#
Начать новый проект – Консольное приложение

Делегаты как указатели на функции .

Слово делегат (delegate) используется в C# для обозначения хорошо известного понятия. Делегат задает определение функционального типа (класса) данных. Экземплярами класса являются функции. Описание делегата в языке C# представляет собой описание еще одного частного случая класса. Каждый делегат описывает множество функций с заданной сигнатурой. Каждая функция (метод), сигнатура которого совпадает с сигнатурой делегата, может рассматриваться как экземпляр класса, заданного делегатом. Синтаксис объявления делегата имеет следующий вид:

[<спецификатор доступа>] delegate <тип результата > <имя класса> (<список аргументов>);

Этим объявлением класса задается функциональный тип - множество функций с заданной сигнатурой, у которых аргументы определяются списком, заданным в объявлении делегата, и тип возвращаемого значения определяется типом результата делегата.

Делегат представляет собой объект, который может ссылаться на метод. Следовательно, когда создается

делегат, то в итоге получается объект, содержащий ссылку на метод. Более того, метод можно вызывать по этой ссылке. Иными словами, делегат позволяет вызывать метод, на который он ссылается.

По сути, делегат — это безопасный в отношении типов объект, указывающий на другой метод (или, возможно, список методов) приложения, который может быть вызван позднее.

В частности, объект делегата поддерживает три важных фрагмента информации:

- адрес метода, на котором он вызывается;
- аргументы (если есть) этого метода;
- возвращаемое значение (если есть) этого метода.

Как только делегат создан и снабжен необходимой информацией, он может динамически вызывать методы, на которые указывает, во время выполнения. Каждый делегат в .NET Framework (включая специальные делегаты) автоматически снабжается способностью вызывать свои методы синхронно или асинхронно. Этот факт значительно упрощает задачи программирования, поскольку позволяет вызывать метод во вторичном потоке выполнения без ручного создания и управления объектом Thread.

Задание 29: Набрать код программы **«Простой целочисленный калькулятор»**, проверить выполнение для различных вариантов входных данных. Модифицировать программу: добавить два метода – вычитание и целочисленное деление.

```
using System;
namespace ConsoleApplication1
{
   // Создадим делегат до определения основного класса
   delegate int IntCalc(int i, int j);
```

```
  class Program
  {
  // Организуем метод суммы 2 чисел
     static int Sum(int x, int y)
     {
     return x + y;
     }

     // Организуем метод произведения 2 чисел
     static int Prz(int x, int y)
     {
     return x * y;
     }

     //Основной метод класса
     static void Main()
     {
     // Сконструируем делегат
     IntCalc op = new IntCalc(Sum);

     int    k=Convert.    ToInt32    (Con-
     sole.ReadLine());
     int m =
     Convert.ToInt32(Console.ReadLine());

     int result = op(k, m);
     Console.WriteLine("Сумма:  "  +  re-
     sult);

     // Изменим ссылку на метод
     op = new IntCalc(Prz);
     result = op(k, m);
     Console.WriteLine("Произведение:   "
     + result);

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

Задание 30: Набрать код программы вычисления интеграла методом трапеций на интервале [a,b] с точностью ε=0.000001, проверить выполнение для различных вариантов входных данных. Делегатом выступает подынтегральная функция.

Задание: подсчитать интеграл:

$4\cos(x+2)$

Класс Функция

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

namespace ConsoleApplication1
{
   // Класс - функции от одного аргумента
   class Functions
   {
      static public double func1(double
      x)
      {
      return 2.0 * x + 3.0;
      }
      static public double func2(double
      x)
      {
      return 3.0 * x * x + 4.0 * x + 2.0;
      }
   }
}
```

Класс Интеграл

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
// Вычисление интеграла по методу трапеций
   class Integral
   {
      public delegate double
      PodIntegrFunc(double x); // делегат: по-
      динтегральная функция от одного переменного: y
      = f(x)

      /// <summary>
      /// Вычисление интеграла
      /// </summary>
      /// <param name="a">Нижняя граница</param>
      /// <param name="b">Верхняя граница</param>
      /// <param name="eps">Точность</param>
      /// <param name="f">Подинтегральная функ-
      ция</param>
      /// <returns>Возвращает значение интеграла с
      требуемой точностью</returns>
      static public double
      CalcIntegral(double a, double b,
      double eps, PodIntegrFunc f)
      {
      int n = 4; // минимальное количество раз-
      биений интервала интегрирования
      double i0 = 0;
      double i1 = Trap(a, b, n, f); // вы-
      числить интеграл по методу трапеций с разбие-
      нием интервала на n частей

      while (Math.Abs(i1 - i0) > eps) //
      предполагаем, что метод обязательно сходится
      {
      i0 = i1;
```

```
      n *= 2; // увеличиваем вдвое количество разбиений
      i1 = Trap(a, b, n, f);
      }
      return i1;
      }

      // Вычисление интеграла по методу трапеций
      static private double Trap(double
      a, double b, int n, PodIntegrFunc
      f)
      {
      double x = a;
      double dx = (b - a) / n;
      double sum = f(a) / 2; // в начальной
      точке берется половинное значение функции

      for (int i = 2; i <= n; i++)
      {
      x += dx; sum += f(x);
      }
      sum += f(b) / 2; // в конечной точке берется половинное значение функции
      return sum * dx;
      }
   }
}
```

Основная программа

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

namespace ConsoleApplication1
{
  class Program
  {
```

```
        // Тестирование класса вычисления интеграла
        static void Main(string[] args)
        {
        double eps = 1.0E-6; // точность вычисления
        double a = 0.0, b = 1.0;
        Integral.PodIntegrFunc f1 = new Integral.PodIntegrFunc(Functions.func1); // задание делегата
        double z1 = Integral.CalcIntegral(a, b, eps, f1);
        Console.WriteLine("\nФункция y=2x+3, пределы от {0} до {1}, интеграл равен {2,8:f4}", a, b, z1);

        a = 1.0; b = 3.0;
        Integral.PodIntegrFunc f2 = new Integral.PodIntegrFunc(Functions.func2); // задание делегата
        double z2 = Integral.CalcIntegral(a, b, eps, f2);
        Console.WriteLine("\nФункция y=3x^2+4x+2, пределы от {0} до {1}, интеграл равен {2,8:f4}", a, b, z2);
        Console.ReadLine();
        }
    }
}
```

```
Функция y=2x+3, пределы от 0 до 1, интеграл равен   4,0000
Функция y=3x^2+4x+2, пределы от 1 до 3, интеграл равен  46,0000
Функция y=3sin(2x)+4cos(x+2), пределы от -1,57 до 1,57, интеграл равен  -3,3292
```

Рисунок 11. Примерный вывод данных программы

Делегаты как свойства. Контейнеры.

Задание 31: Набрать код программы, проверить выполнение для различных вариантов входных данных. Задание: добавить сортировку по идентификатору.

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

namespace ConsoleApplication1
{
  class Program
  {
    static void Main(string[] args)
    {
    Persons coll = new Persons();
    coll.AddPerson(new Person
    "Соколов", 123, 750.00));
    coll.AddPerson(new Person
    "Синицын", 128, 850.00));
    coll.AddPerson(new Person
    "Воробьев", 223, 750.00));
    coll.AddPerson(new  Person("Орлов",
    129, 800.00));
    coll.AddPerson(new Person
    "Соколов", 133, 1750.00));
    coll.AddPerson(new  Person("Орлов",
    119, 750.00));

    Console.WriteLine("\nСортировка  по
    имени");
    coll.SortPersons(Person.SortName);
    coll.PrintAll();

    Console.WriteLine("\nСортировка  по
    зарплате и имени");
    coll.SortPersons(Person.SortZarplNa
    me);
    coll.PrintAll();
```

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

Класс Persons

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

namespace ConsoleApplication1
{
  // контейнер экземпляров класса Person
  // организован в виде массива
  class Persons
  {
    // делегат
    public delegate int
    CompareItems(Person a, Person b);

    const int maxn = 100; // максимальный
    размер контейнера
    private int svob = 0; // номер первого
    свободного элемента массива
    private Person[] persons = new Per-
    son[maxn]; // массив контейнера -
    резервирование памяти

    // индексатор. Для доступа к закрытому массиву
    persons

    public Person this[int n]
    {
    get { return persons[n]; }
    set { persons[n] = value; }
    }
```

// набор стандартных методов для контейнера

// Добавление нового элемента
```
public void AddPerson(Person elem)
{
if (svob < maxn) persons[svob++] =
elem;
else            throw            (new
ArgumentOutOfRangeException());
}
```

// Вывод на экран всей коллекции
```
public void PrintAll()
{
for (int i = 0; i < svob; i++)
Console.WriteLine("{0,5}      {1,10}
{2,8:f2}",    persons[i].Id,    per-
sons[i].Name, persons[i].Zarpl);
}
```

// Сортировка элементов коллекции (пузырьком) по заданной функции сравнения

```
public                              void
SortPersons(CompareItems compare)
{
Person temp = new Person("", 0, 0);
```
// вспомогательная переменная

```
for (int i = 1; i < svob; i++)
for (int j = svob - 1; j >= i; j--)
if (compare(persons[j], persons[j -
1]) == -1)
{
temp = persons[j - 1];
persons[j - 1] = persons[j];
persons[j] = temp;
}
```

```
    }
  }
}
```

Класс Person – для методов сортировки

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

namespace ConsoleApplication1
{
  // класс Person с методами сравнения по разным полям
  class Person
  {
  // поля класса (закрытые)
    private string name; // имя
    private int id; // идентификационный номер
    private double zarpl; // зарплата

    // конструктор
    public Person(string name, int id,
    double zarpl)
    {
    this.name = name;
    this.id = id;
    this.zarpl = zarpl;
    }

    // свойства для доступа к полям
    public int Id
    {
    get {return this.id;}
    }

    public string Name
```

```
{
get { return this.name; }
}

public double Zarpl
{
get { return this.zarpl; }
}
```
// различные методы сравнения двух экземпляров класса Person

// у всех методов одинаковая сигнатура!
```
private static int
CompareName(Person obj1, Person
obj2)
```

// по имени
```
{ return
obj1.name.CompareTo(obj2.name);
}

private static int
CompareZarplName(Person obj1, Per-
son obj2)
```
// по зарплате и имени

```
{
int n =
 obj1.zarpl.CompareTo(obj2.zarpl);

if (n == 0) return
obj1.name.CompareTo(obj2.name);
else return n;
}
```

// Статические свойства как реализация делегата класса Persons

```
public static Persons.CompareItems
SortName
```

```
    {
    get{return new Persons.CompareItems
    (CompareName); }
    }

    public static Persons.CompareItems
    SortZarplName

    {
    get{return new Persons.CompareItems
    (CompareZarplName); }
    }
  }
}
```

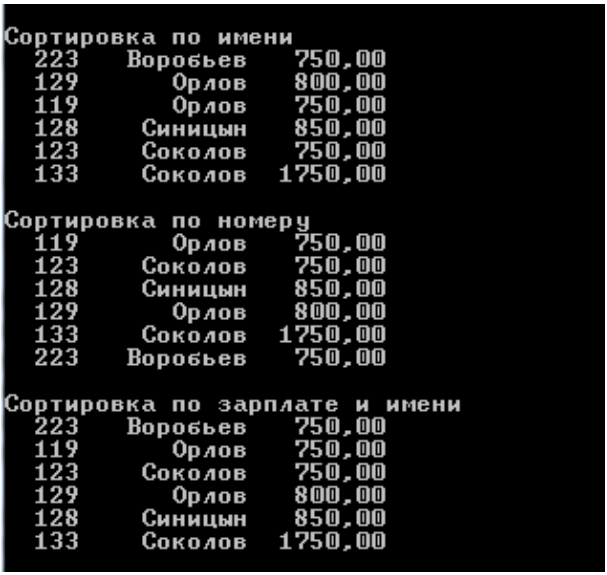


Рисунок 12. Примерный вывод данных программы

Делегаты и события. (Класс Город и службы города)

Задание 32: Набрать код программы, проверить выполнение для различных вариантов входных данных.

Задание: рассмотреть варианты появления недобросовестной службы.

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

namespace ConsoleApplication1
{
  class Program
  {
  static void Main(string[] args)
  {
    Gorod Psk = new Gorod(1000); // инициализация города с 1000 домов
    Psk.Life(365, 0.0001); // наблюдаем в течение года. Вероятность пожара 1%
    Console.ReadLine();
    }
  }
}
```

Класс «Город»

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

namespace ConsoleApplication1
{
  /// <summary>
  /// Моделирование города
  /// </summary>
  class Gorod
  {
    public delegate void
    FireEventHandler(object Sender,
```

```
FireEventArgs args); // делегат типа события пожар с фиксированной сигнатурой

// поля класса
private int koldomov; // количество зданий в городе
// городские службы
private Police policaj; // полицейский
private Med medik; // скорая помощь
private Pozh pozharn; // пожарные
// список обработчиков события в городе (пожара)
public event FireEventHandler Fire;

// конструктор
public Gorod(int kol)
{
this.koldomov = kol;
// инициализация служб (создание экземпляров объектов)
policaj = new Police(this);
medik = new Med(this);
pozharn = new Pozh(this);
// подключение обработчика события к каждому экземпляру объекта
policaj.OnFire();
medik.OnFire();
pozharn.OnFire();
}

/// <summary>
/// моделирование жизни города
/// </summary>
/// <param name="days">количество дней наблюдения за жизнью города</param>
/// <param name="p">вероятность возникновения пожара</param>
public void Life(int days,double p)
```

```
        {
        string msg;
        Random rnd = new Random();

        for(int i=1; i<=days;i++) // наблюдение ведется каждый день заданного кол-ва дней
        for (int j = 1; j <= this.koldomov; j++) // обследуется каждый из домов
        {
        if (rnd.NextDouble() < p) // если пожар возник
        {
        FireEventArgs e = new FireEventArgs(j, i); // передача событию аргументов
        Fire(this, e); // возникновение события, передача его всем получателям события
        // анализ возвращенного получателями события аргумента
        if (e.Likv) msg = "Пожар потушен. Ситуация нормализована";
        else msg = "Пожар продолжается. Требуются дополнительные средства";
        Console.WriteLine(msg);
          }
         }
        }
    }
}
```

Класс «Пожар» - делегат класса

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

namespace ConsoleApplication1
```

```
{
    /// <summary>
    /// Параметры события пожар
    /// </summary>
    class FireEventArgs : EventArgs // родитель - класс EventArgs, стандартный для параметров события (в нем никаких параметров нет)
    {
        // поля класса - параметры события, входные и выходные
        private bool likv; // ликвидирован ли пожар (выходной параметр)
        private int zd; // порядковый номер дома, в котором произошло событие (входной параметр)
        private int den; // порядковый номер дня события (входной параметр)

        // параметры (поля класса) закрыты от изменений, задаются только при создании экземпляра класса

        public FireEventArgs(int zd, int den) // конструктор. Задаются только входные параметры
        {
        this.zd = zd;
        this.den = den;
        this.likv = false;
        }

        // Свойства. Для входных параметров - только чтение, для выходных - чтение и запись

        public int Zd
        {
        get { return zd; }
        }
```

```
    public int Den
    {
    get { return den; }
    }

    public bool Likv
    {
    get { return likv; }
    set { likv = value; }
    }
  }
}
```

Класс «Получатель тушения пожара»

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

namespace ConsoleApplication1
{
  /// <summary>
  /// Абстрактный класс: получатель события. Родитель всех классов-служб
  /// </summary>
  abstract class Poluch
  {
    // поле город - экземпляр класса, вызывающего событие
    private Gorod gor;

    // конструктор: задание конкретного города
    public Poluch(Gorod g)
    {
    this.gor = g;
    }
```

```
    // обработчик события (пустой в абстрактном
    классе)
    public abstract void Pozhar(object
    sender, FireEventArgs args);

    // Присоединение обработчика события к списку
    обработчиков
    public void OnFire()
    {
    gor.Fire           +=           new
    Gorod.FireEventHandler(Pozhar);
    }
  }
}
```

Классы Службы Города:
– Пожарники

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

namespace ConsoleApplication1
{
  // класс Пожарные
  class Pozh:Poluch
  {
    // конструктор (не наследуется)
    public Pozh(Gorod g) : base(g) { }

    // обработчик события (индивидуален для каждой
    службы)
    public override void Pozhar(object
    sender, FireEventArgs args)
    {
    Console.WriteLine("Пожар в доме {0}
    в    {1}    день!\nПожарные    тушат
    пожар",args.Zd,args.Den);
```

```
      // и всегда добиваются успеха
      args.Likv = true;
      }
   }
}
```

-Скорая помощь

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

namespace ConsoleApplication1
{
   // класс Скорая помощь
   class Med:Poluch
   {
      // конструктор (не наследуется)
      public Med(Gorod g) : base(g) { }

      // обработчик события (индивидуален для каждой
      службы)
      public override void Pozhar(object
      sender, FireEventArgs args)
      {
      Console.WriteLine("Пожар в доме {0}
      в {1} день!\nСкорая помощь спасает
      пострадавших",args.Zd,args.Den);
      // аргумент Likv не меняется, т.к. скорая не тушит
      пожары
      }
   }
}
```

- Полиция

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

```
namespace ConsoleApplication1
{
   // класс Полиция
   class Police:Poluch
   {
   // конструктор (не наследуется)
      public Police(Gorod g) : base(g) {
      }

   // обработчик события (индивидуален для каждой
   службы)
      public override void Pozhar(object
      sender, FireEventArgs args)
      {
      Console.WriteLine("Пожар в доме {0}
      в {1} день!\nПолиция ищет поджига-
      телей",args.Zd,args.Den);
   // аргумент Likv не меняется, т.к. полиция не тушит
   пожары
      }
   }
}
```

Классы с универсальными типами (Класс стек)

Самостоятельно проанализировать код программы «Стек», придумать ваианыт модификации программы. Исходный код программы по адресу http://do.psksu.ru/file.php/355/Klass_s_universalinymi_tipami.zip

Написать отчет по лабораторной работе.

Приложение 1.
Пример оформления отчета по лабораторной работе

Лабораторная работа № 3
Цель работы:

- Разобраться с понятием цикла
- Решить задачи с разными видами циклов

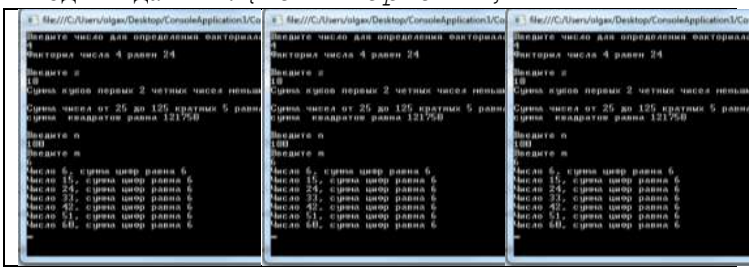
1. Практическая работа

Программа, к которой использованы все виды циклов.

```
{Ваш Код программы из C#
…namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int x, z, f;
double y,k,s, skv;

//Посчитать факториал числа n
Console.WriteLine("Введите число для
определения факториала");
int n = Con-
vert.ToInt32(Console.ReadLine());
f = 1;
x = 1;…
}
```

Варианты работы программы для разных значений входных данных. *{Ваши варианты}*



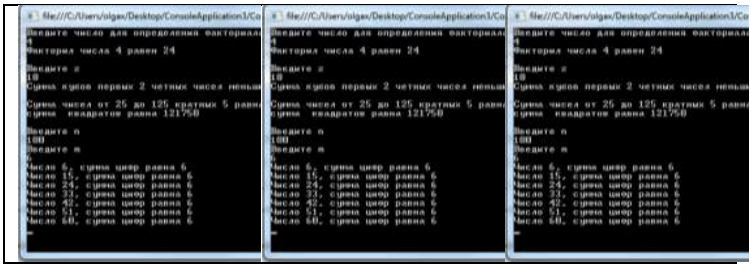


Рисунок 13. Варианты работы программы для разных значений входных данных. Варинтов должно быть не менее шести.

Вывод: 1. В данной программе для нахождения факториала использован цикл с предусловием, который выполняет последовательное умножение до тех пор пока искомое число не достигнет нужного значения.
2. Для нахождения «Суммы кубов четных чисел меньших половины заданного z» использован цикл с постусловием, выполняющийся хотя бы раз до достижения заданного условия.
3. Для нахождения «Суммы чисел от 25 до 125 кратных 5, и суммы квадратов этих чисел» используется цикл со счетчиком, диапазон значений от 25 до 125 с шагом 5.
4. Для решения последней задачи «Даны натуральные числа n, m. Получить все меньшие n натуральные числа, сумма цифр которых равен m.» использованы вложенные циклы, внешний цикл - цикл со счетчиком, выполняется N раз. Внутренний цикл – цикл с предусловие выполняется столько раз, сколько цифр в числе, находит сумму цифр.

2. Самостоятельная работа

2.1 Циклы с предусловием

Вариант № 3

Условие задачи: *{Ваше условие}*

Решение задачи:

```
{Ваш Код программы из C#
…namespace ConsoleApplication1
{
class Program
```

```
{
static void Main(string[] args)
{
int x, z, f;
double y,k,s, skv;

//Посчитать факториал числа n
Console.WriteLine("Введите число для
определения факториала");
int n = Con-
vert.ToInt32(Console.ReadLine());
f = 1;
x = 1;…
}
```

2.2. Циклы с постусловием

Вариант № 7

Условие задачи: *{Ваше условие}*

Решение задачи:

```
{Ваш Код программы из C#
…namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int x, z, f;
double y,k,s, skv;

//Посчитать факториал числа n
Console.WriteLine("Введите число для
определения факториала");
int n = Con-
vert.ToInt32(Console.ReadLine());
f = 1;
x = 1;…
}
```

2.2 Циклы со счетчиком

Вариант № 15

Условие задачи: *{Ваше условие}*

Решение задачи:

```
{Ваш Код программы из C#
…namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int x, z, f;
double y,k,s, skv;

//Посчитать факториал числа n
Console.WriteLine("Введите   число   для
определения факториала");
int          n          =          Con-
vert.ToInt32(Console.ReadLine());
f = 1;
x = 1;…
}
```

2.3 Вложенные циклы

Вариант № 6

Условие задачи: {Ваше условие}

Решение задачи:

```
{Ваш Код программы из C#
…namespace ConsoleApplication1
{
class Program
{
static void Main(string[] args)
{
int x, z, f;
double y,k,s, skv;

//Посчитать факториал числа n
Console.WriteLine("Введите   число   для
определения факториала");
int          n          =          Con-
vert.ToInt32(Console.ReadLine());
f = 1;
```

```
x = 1;…
}
```

Выводы по лабораторной работе:

Мною была проанализирована работа различных видов циклов, преимущества циклов со счетчиком *{…},* преимущества цикла с постусловием: *{…}*, преимущества цикла с постусловием: *{…}*.

Кабаченко Виктор Валентинович
Хмылко Ольга Николаевна

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА C#. КОНСОЛЬНЫЕ ПРИЛОЖЕНИЯ

Учебно-методическое пособие

Технический редактор: Кабаченко В.В.
Компьютерная верстка: Хмылко О.Н.
Корректор:

Подписано в печать 30.06.2015. Формат 60×90/16.
Гарнитура «Times New Roman». Усл. п. л. 9,5.
Тираж 75 экз. Заказ № 2825.
Адрес издательства:
Россия, 180000, Псков, ул. Л. Толстого, 4.
Издательство ПскоГУ