# Лабораторная работа № 1.

## ВВЕДЕНИЕ В C#.
## СОЗДАНИЕ КОНСОЛЬНЫХ ПРИЛОЖЕНИЙ НА C#

**Вопросы:**

1. Язык C#
2. Первая программа на C#
3. Основы C#

**1. Язык C#** (произносится Си-Шарп) - это язык программирования от компании Microsoft. Он входит в версию Visual Studio - Visual Studio.NET. Кроме C# в Visual Studio.NET входят Visual Basic.NET и Visual C++.

Одна из причин разработки нового языка компанией Microsoft - это создание компонентно-ориентированного языка для новой платформы .NET. Другие языки были созданы до появления платформы .NET, язык же C# создавался специально под эту платформу и не несет с собой груза совместимости с предыдущими версиями языков. Хотя это не означает, что для новой платформы это единственный язык.

Еще одна из причин разработки компанией Microsoft нового языка программирования - это создание альтернативы языку Java.

Язык C# является наиболее известной новинкой в области создания языков программирования. В отличие от 60-х годов XX века - периода бурного языкотворчества - в нынешнее время языки создаются крайне редко.

Создателем языка является сотрудник Microsoft Андреас Хейлсберг. Он стал известным в мире программистов задолго до того, как пришел в Microsoft. Хейлсберг входил в число ведущих разработчиков одной из самых популярных сред разработки - Delphi. В Microsoft он участвовал в создании версии Java - J. C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, направленное на возможность повторного использования созданных компонентов.

### 2. Первая программа на C#

Запускаем Visual Studio.NET. Для создания нового пустого проекта C# проекта в Visual Studio необходимо воспользоваться кнопкой **Create Project** на стартовой странице Visual Studio 2008 (рис.1).

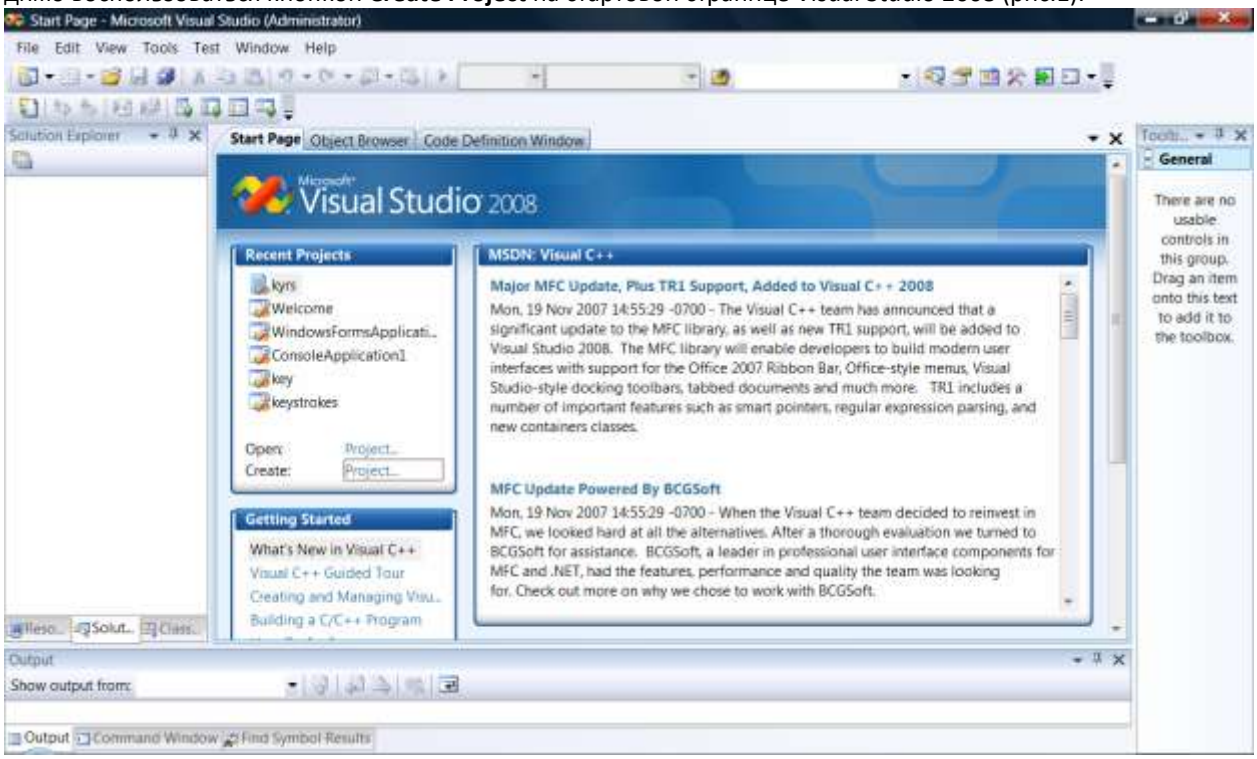


**Рис.1.** Стартовая страница Visual Studio 2008

В появившемся окне New Project слева выбираем, **Visual C#,** а справа тип приложения - **Console Application** (рис.2):

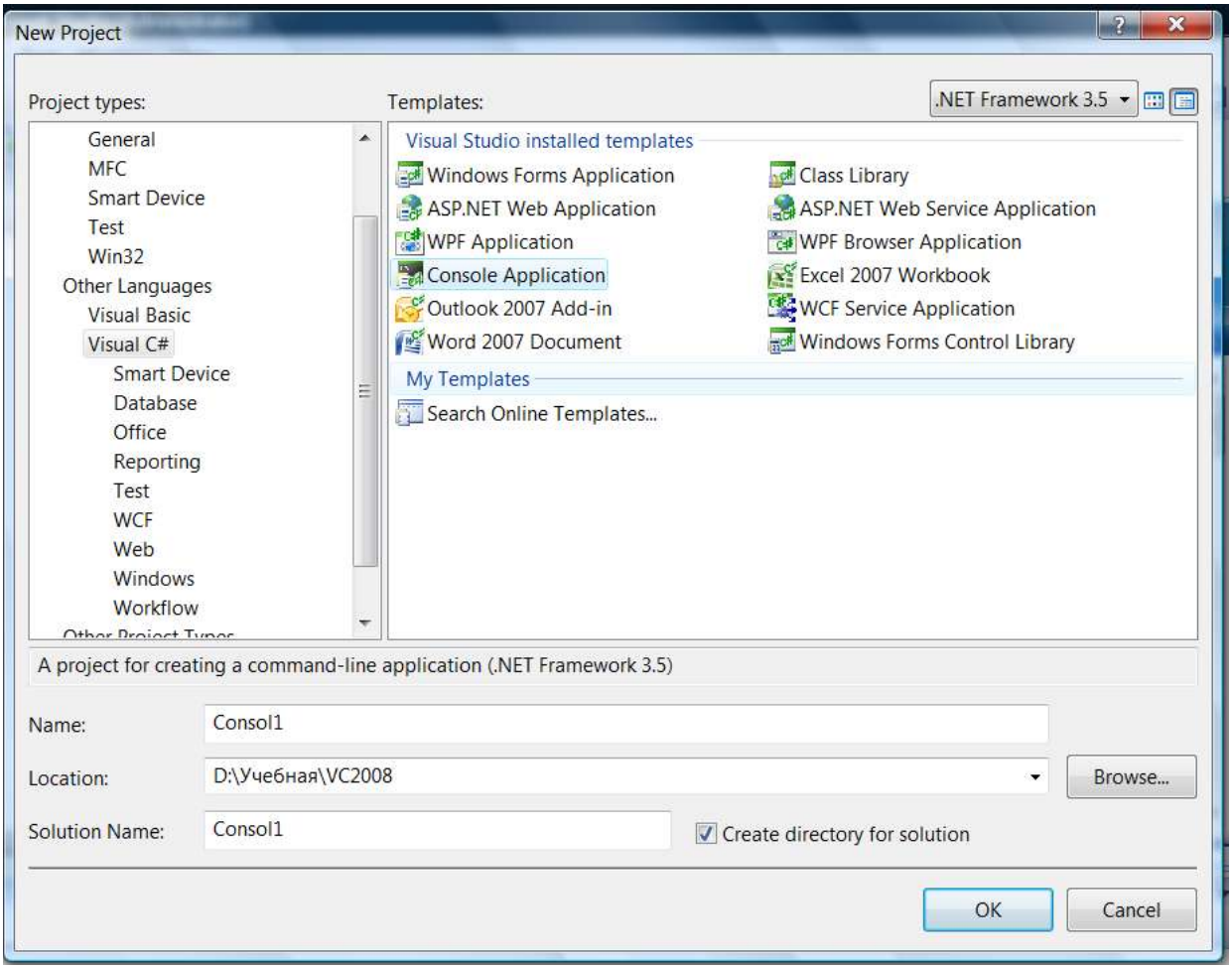


**Рис.2.** Окно создания нового проекта

В качестве имени проекта (Name) наберите **Consol1**, или что-то в этом роде. Нажмите на кнопку ОК для закрытия данного диалогового окна. В качестве места (**Location**), где будет храниться папка проекта с файлами, укажите сетевой диск **Z**. По умолчанию, предусмотрено, что имя папки рабочей области и имя проекта совпадают, что не обязательно.

В результате чего, на диске будут созданы папка рабочей области Consol1, а в этой папке еще одна папка проекта Consol1, в которой будут храниться файлы проекта, и создан файл кода программы *Program.cs*. Кроме этого, в папке рабочей области создаются еще файл с именем рабочей области и расширением **sln**, который содержит информацию, о настройках рабочей области, файлах, папках и с помощью которого происходит загрузка рабочего пространства в среду VisualStudio. В папке с названием проекта, создаются папки **bin**, **obj**, **Properties**. В папке bin находится вложенная папка **Debug**, где впоследствии будет находиться исполняемый файл проекта Consol1.exe. Папка **obj** используется для хранения информации об объектных модулях, папка **Properties** – информации о свойствах проекта (рис.3).

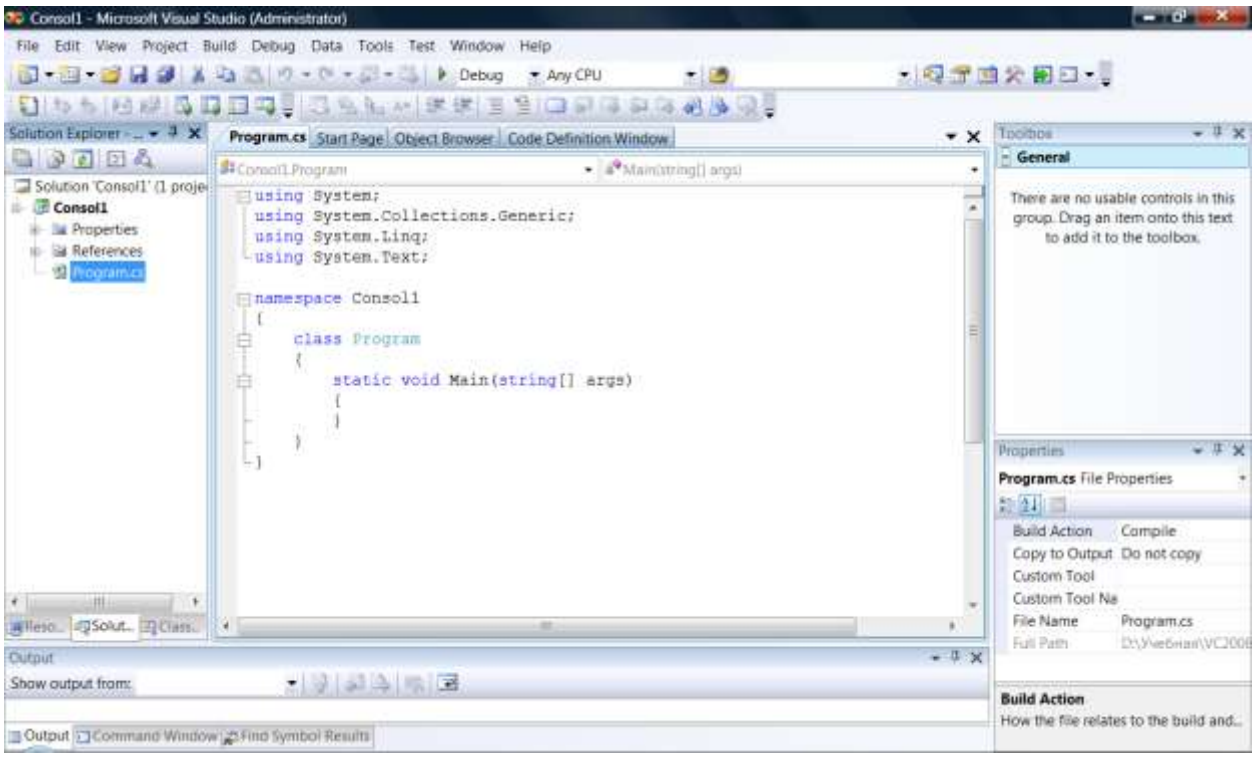


**Рис.3.** Окно рабочей области проекта

Добавим в код программы строку, которая выведет некоторое сообщение в консольное окно.

```
Console.WriteLine("Привет из C#");
```

Так как в программе автоматически создана строка `using System`, то вместо длинных можно использовать короткие имена методов, в частности, вместо System.Console можно писать просто Console, как записано в предыдущей строке.

Далее в программе объявлен класс Programm. Что такое классы подробно будет рассмотрено позже, сейчас же достаточно знать, что в C# необходимо создать класс и в нем функцию Main *(функция Main обязательно должна быть в каждой программе на C#, и именно с этой функции и начинается выполнение программы)*. Обратите также внимание, что эта функция пишется с прописной (большой) буквы. C# различает строчные и прописные буквы. В функции Main выводим на экран некоторую строчку методом WriteLine (рис. 4).

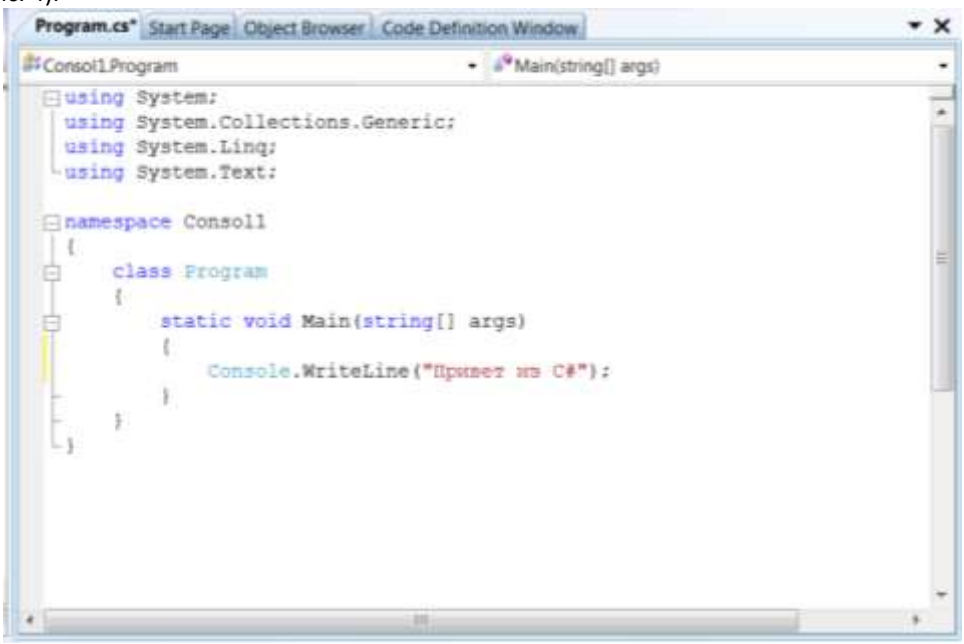


**Рис. 4.** Окно программы

Выбрав в верхнем меню рабочей области **Debug -> StartWithoutDebugging** или **Ctrl+F5** запускаем программу. Результат представлен на рис. 5.

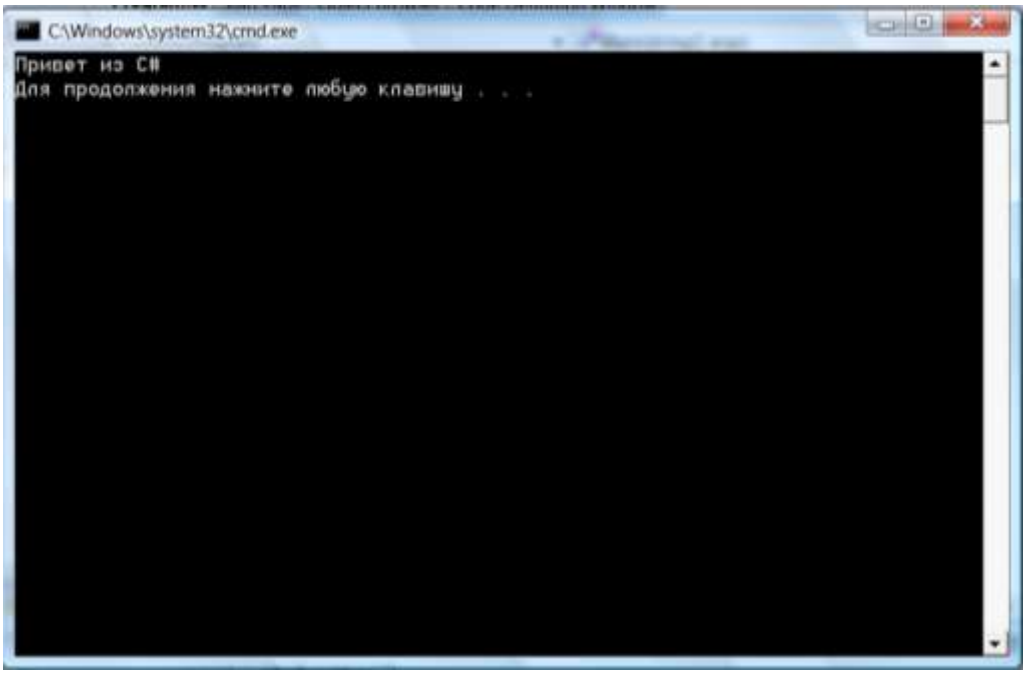


Рис.5. Результат работы программы

Расширим возможности программы, а именно добавим возможность считывания данных с клавиатуры и выполнения простейшим арифметических операций, нахождение суммы, разности, произведения и частного.

Для считывания строки символов, введенной с клавиатуры в консольном окне, используется метод `Console.ReadLine()`, пространства имен System. Для преобразования строки символов в число необходимо использовать метод `Parse()`.

Используем соответствующие методы в программе:

```
float m; //Описание переменной m типа float
Console.WriteLine("Привет из C#");
Console.WriteLine("Введите целое число: ");
//Считаем строку символов методом
//Console.ReadLine() и с помощью метода
//Parse() преобразуем ее к целому типу int
//и присвоим переменной k целого типа.
int k = Int32.Parse(Console.ReadLine());
Console.WriteLine("Было введено число - " + k);
k = k + k;
Console.WriteLine("Сумма k+k =" + k);
k = k * k;
Console.WriteLine("Произведение k*k =" + k);
//Переменная l описана как float - вещественное число
float l = (float)k / ((float)k + (float)k);
Console.WriteLine("Выражение k /(k+k) = " + l);
Console.WriteLine("Введите дробное число ");
Console.WriteLine("(в качестве разделительного знака");
Console.WriteLine(" используйте запятую): ");
//Считывание строки символов и преобразование ее к типу float
m = float.Parse(Console.ReadLine());
m = (m + m)/(m*m);
Console.WriteLine("Выражение(m + m)/(m*m)= " + m);
```

Обратите внимание на строку:

```
float l = (float)k / ((float)k + (float)k);
```

Здесь, в явном виде используется приведение типа переменной k к вещественному типу float, иначе при проведении вычислений, результат от деления будет приведен к целому числу.

В строке:

```
m = float.Parse(Console.ReadLine());
```

используется метод `Parse()` для преобразования строки символов число вещественного типа float. Результат преобразования присваиваться переменной m. Тип переменной описан в строке:

```
float m; //Описание переменной m типа float
```

Результат работы программы представлен на рис. 6.

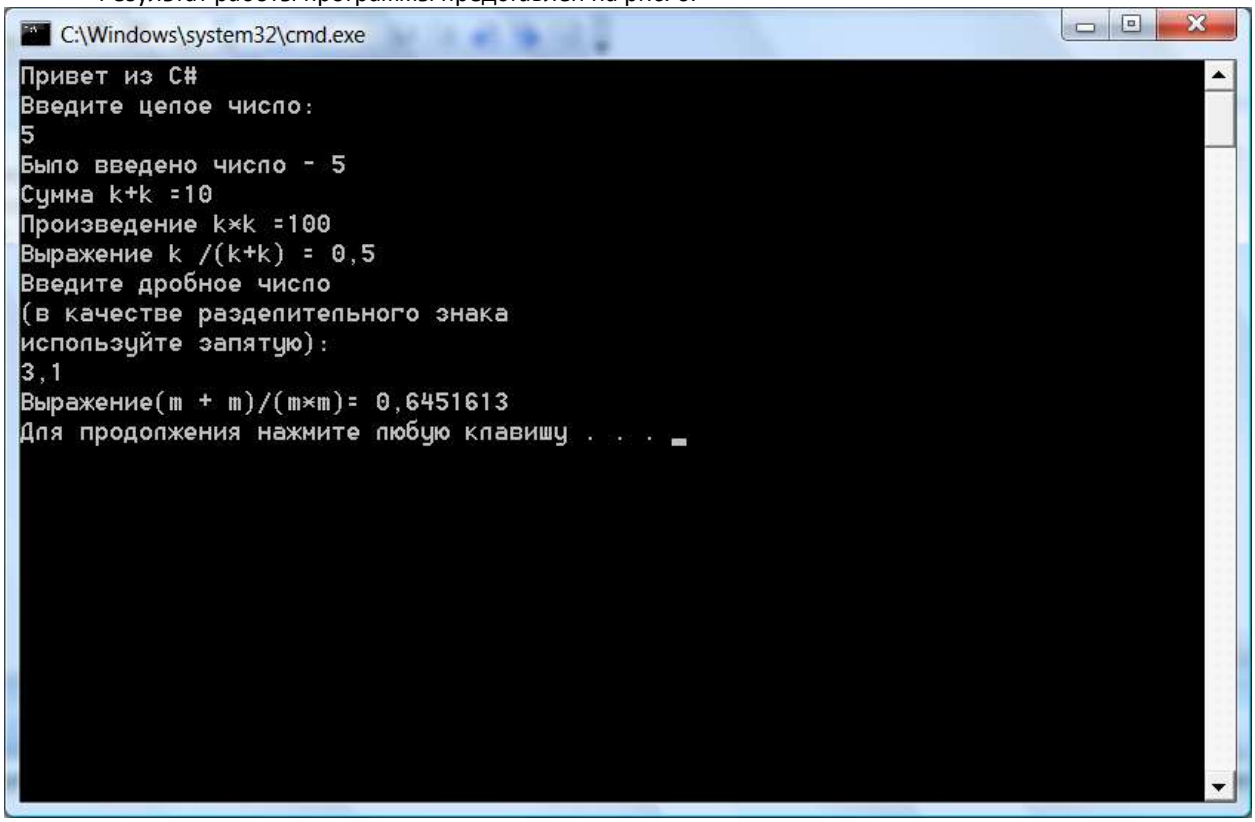


**Рис.6.** Результат работы программы

## 3. Основы C#

### 3.1. Переменные языка C#

Для каждого типа данных C# существует соответствующий тип данных в среде CRL (Common Language Runtime), что означает, что каждый тип имеет два названия - полный (из CLR, его можно использовать в любом языке .NET) и сокращенный, который используется в C#. При разработке программ на C# можно использовать как полный так и сокращенный тип записи.

Таким образом, следующие три объявления переменной k равносильны:

```
int k;
using System;
...
Int32 k;
```

и

```
System.Int32 k;
```

Аналогично и с другими типами языка C#.

Основные типы данных и их характеристики перечислены в следующей таблице 1:

Таблица 1

| **Логический тип** | | | |
|---|---|---|---|
| **Имя типа (C#)** | **Системный тип (CLR)** | **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 | - 2,147,483,648… 2,147,483,647 | Знаковое, 32 Бит |
| Uint | System.UInt32 | 0… 4,294,967,295) | Беззнаковое, 32 Бит |
| Long | System.Int64 | - 9,223,372,036,854,775,808 … 9,223,372,036,854,775,807 | Знаковое, 64 Бит |
| Ulong | System.UInt64 | 0… 18,446,744,073,709,551,615 | Беззнаковое, 64 Бит |
| **Арифметический тип с плавающей точкой** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Точность** |
| Float | System.Single | - 3.402823e38 … + 3.402823e38 | 7 цифр |
| Double | System.Double | - 1.79769313486232e308… 1.79769313486232e308 | 15-16 цифр |
| **Арифметический тип с фиксированной точкой** | | | |
| **Имя типа** | **Системный тип** | **Диапазон** | **Точность** |
| Decimal | System.Decimal | -79,228,162,514,264,337,593,543,950,335 79,228,162,514,264,337,593,543,950,335 | 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 | |

Объявление переменной можно совместить с инициализацией (заданием начального значения):

```
int z=88;
```

Набор операторов для C# достаточно стандартен **+ , -, *, /** - действуют как и в любом другом языке. Отметим только, что / (деление) применительно к целым числам дает целую часть от деления. Так, фрагмент

```
int k=100999, n=1000, s;
s=k/n;
Console.WriteLine(s.ToString());
```

выведет на экран 100, а не 101, т. е. никакого округления не происходит.

Есть еще один оператор - **%**. Это - остаток от деления. Следующий фрагмент выведет на экран 999:

```
int k=100999, n=1000, s;
s=k%n;
Console.WriteLine(s.ToString());
```

В C# существуют операторы инкремента и декремента. Так, после следующего фрагмента k увеличится на 1, а n - уменьшится на 1:

```
k++;
n--;
```

## 3.2. Операторы языка C#

### Оператор присваивания

В C# присваивание формально считается операцией. Запись:

```
X = expr;
```

следует считать оператором присваивания, так же, как и одновременное присваивание со списком переменных в левой части:

```
X1 = X2 = ... = expr;
```

#### Блок или составной оператор

С помощью фигурных скобок несколько операторов можно объединить в единую синтаксическую конструкцию, называемую блоком или составным оператором:

```
{ //Аналог begin в Pascal
   оператор_1
   ...
   оператор_N
}//Аналог end в Pascal
```

Синтаксически блок воспринимается как единичный оператор и может использоваться всюду в конструкциях, где синтаксис требует одного оператора. Тело цикла, ветви оператора if, как правило, представляются блоком.

### 3.2.1. Логические операторы

В C# существуют следующие логические операторы:

| Оператор | Описание | Пример |
|---|---|---|
| && | Логическое И. Результат равен true, только если оба операнда равны true | (x==8) && (y==5) |
| \|\| | Логическое ИЛИ. Результат равен false, только если оба операнда равны false | (y>8) \|\| (y<5) |
| ! | Отрицание. Изменяет логическое значение на противоположное | if(!(a==b))... |

Все эти операторы возвращают результат типа bool.

Обратите внимание, что для логического **равно** (т. е. для ответа на вопрос "Верно ли, что что-то равно чему-то") используется знак двойного равенства (==). Знак же одинарного равенства (=) используется для **присваивания**. Для знака == существует парный знак != ("не равно"). Так, приведенный выше пример для оператора ! можно переписать так:

```
if(!(a==b))...
```

В C#, , нельзя вместо false использовать 0, а вместо true - любое ненулевое число. Так, следующий фрагмент содержит ошибку:

```
  int k;
  ...
  if(k) //Ошибка!
  ...
```

### 3.2.2. Операторы выбора

В языке C# для выбора одной из нескольких возможностей используются две конструкции - if и switch. Первую из них обычно называют альтернативным выбором, вторую - разбором случаев.

#### Оператор if

Синтаксис оператора if:

```
if(выражение_1) оператор_1
else if(выражение_2) оператор_2
...
else if(выражение_K) оператор_K
else оператор_N
```

Выражения if должны заключаться в круглые скобки и быть булевого типа ( выражения должны давать значения true или false). Арифметический тип не имеет явных или неявных преобразований к булевому типу. По правилам синтаксиса языка, then-ветвь оператора следует сразу за круглой скобкой без ключевого слова then, типичного для большинства языков программирования. Каждый из операторов может быть блоком - в частности, if-оператором. Поэтому возможна и такая конструкция:

```
if(выражение1) if(выражение2) if(выражение3) ...
```

Ветви else и if, позволяющие организовать выбор из многих возможностей, могут отсутствовать. Может быть опущена и заключительная else-ветвь. В этом случае краткая форма оператора if задает альтернативный выбор - делать или не делать - выполнять или не выполнять then-оператор.

Выражения if проверяются в порядке их написания. Как только получено значение true, проверка прекращается и выполняется оператор (это может быть блок), который следует за выражением, получившим значение true. С завершением этого оператора завершается и оператор if. Ветвь else, если она есть, относится к ближайшему открытому if.

**If** служит для разветвления программы на два направления. Если некоторое условие выполняется, то программа идет в одну сторону, если не выполняется - то в другую. Пример, определяющий, четное или нечетное число ввел пользователь:

```
class Program
    {
        static void Main(string[] args)
        {
            //Читаем символ введенный с клавиатуры ( метод ReadLine())
            //с помощью метода Parse преобразуем его
            //в целое число и присваиваем значение переменной k
            Console.WriteLine("Введите целое число");
            int k = Int32.Parse(Console.ReadLine());
            int b = k / 2;
            if (2*b == k)
            {
                Console.WriteLine("Четное число");
            }
            else
            {
                Console.WriteLine("Нечетное число");
            }
        }
    }
```

Фигурные скобки можно не писать в случае одного оператора. Ветка else тоже не является обязательной - все зависит от конкретной задачи.

### Оператор switch

Важным случаем выбора из нескольких вариантов является ситуация, при которой выбор варианта определяется значениями некоторого выражения. Соответствующий оператор C# называется оператором **switch**. Вот его синтаксис:

```
switch(выражение)
{
   case константное_выражение_1: [операторы_1 оператор_перехода_1]
   ...
   case константное_выражение_K: [операторы_K оператор_перехода_K]
   [default: операторы_N оператор_перехода_N]
}
```

Ветвь default может отсутствовать. Заметьте, по синтаксису допустимо, чтобы после двоеточия следовала пустая последовательность операторов, а не последовательность, заканчивающаяся оператором перехода. Константные выражения в case должны иметь тот же тип, что и switch-выражение.

Вначале вычисляется значение switch-выражения. Затем оно поочередно в порядке следования case сравнивается на совпадение с константными выражениями. Как только достигнуто совпадение, выполняется соответствующая последовательность операторов case-ветви. Поскольку последний оператор этой последовательности является оператором перехода (чаще всего это оператор break), то обычно он завершает выполнение оператора switch. Case-ветвь может быть пустой последовательностью операторов. Тогда в случае совпадения константного выражения этой ветви со значением switch-выражения будет выполняться первая непустая последовательность очередной case-ветви. Если значение switch-выражения не совпадает ни с одним константным выражением, то выполняется последовательность операторов ветви default, если же такой ветви нет, то оператор switch эквивалентен пустому оператору.

Пример:

```
class Program
    {
```

```
        static void Main(string[] args)
        {
            Console.WriteLine("Введите оценку");
            int k = Int32.Parse(Console.ReadLine());
            Console.WriteLine("Введенная оценка: " + k.ToString());
            switch (k)
            {
                case 1:
                case 2:
                    Console.WriteLine("Неудовлетворительно");
                    break;
                case 3:
                    Console.WriteLine("Удовлетворительно");
                    break;
                case 4:
                    Console.WriteLine("Хорошо");
                    break;
                case 5:
                    Console.WriteLine("Отлично");
                    break;
                default:
                    Console.WriteLine("Ошибка");
                    break;
            }
        }
    }
```

В приведенном примере в зависимости от введенного пользователем числа на экран выводится та или иная оценка. Если число k не лежит в промежутке от 1 до 5, то выполняются операторы в ветке default и выводится надпись "Ошибка". Ветка default необязательна. Обратите внимание на оператор break. Если его не написать, то будут выполнятся операторы из следующей ветки case до строки с break'ом . Если в некоторой ветке case или default есть операторы, то написание break обязательно. Так, в следующих двух участках кода программы есть ошибки:

```
...
    case 1:
        Console.WriteLine("Совсем неудовлетворительно");
        //Ошибка! Тут пропущен break
    case 2:
        Console.WriteLine("Неудовлетворительно");
        break;
    ...
...
    default:
        Console.WriteLine("...");
        //Ошибка! Тут пропущен break
    }
```

Применяя оператор switch, необходимо заканчивать каждую case-ветвь оператором break.

### 3.2.3. Операторы цикла

#### Оператор for

Оператор цикла for обобщает известную конструкцию цикла типа арифметической прогрессии. Его синтаксис:

```
for(инициализаторы; условие; список_выражений) оператор
```

Оператор, стоящий после закрывающей скобки, задает тело цикла. В большинстве случаев телом цикла является блок. Сколько раз будет выполняться тело цикла, зависит от трех управляющих элементов, заданных в скобках. **Инициализаторы** задают начальное значение одной или нескольких переменных, часто называемых счетчиками или просто переменными цикла. В большинстве случаев цикл for имеет один счетчик. Условие задает условие окончания цикла, соответствующее выражение при вычислении должно получать значение true или false. **Список выражений**, записанный через запятую, показывает, как меняются счетчики цикла на каждом шаге выполнения. Если условие цикла истинно, то выполняется тело цикла, затем изменяются значения счетчиков и снова проверяется условие. Как только условие становится ложным, цикл завершает свою работу. В цикле for тело цикла может ни разу не выполняться, если условие цикла ложно после инициализации, а может происходить зацикливание, если условие всегда остается истинным. В нормальной ситуации тело цикла выполняется конечное число раз.

Счетчики цикла зачастую объявляются непосредственно в инициализаторе и соответственно являются переменными, локализованными в цикле, так что после завершения цикла они перестают существовать.

В тех случаях, когда предусматривается возможность преждевременного завершения цикла с помощью одного из операторов перехода, счетчики объявляются до цикла, что позволяет анализировать их значения при выходе из цикла.

Пример цикла **for**:

```
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введите число");
            int k = Int32.Parse(Console.ReadLine());
            int sum = 0;
            for (int i = 1; i <= k; i++)
            {
                sum += i; //Эквивалентно sum=sum+i
            }
            Console.WriteLine(sum);
        }
    }
```

Этот пример подсчитывает сумму чисел от 1 до введенного пользователем числа k. Сумма записывается в переменную sum и выводится на экран.

#### Циклы While

Цикл while (выражение) является универсальным видом цикла, включаемым во все языки программирования. Тело цикла выполняется до тех пор, пока остается истинным выражение while. В языке C# у этого вида цикла **две модификации** - с проверкой условия в начале и в конце цикла. Первая модификация имеет следующий синтаксис:

```
while(выражение) оператор
```

Эта модификация соответствует алгоритму: "сначала проверь, а потом делай". В результате проверки может оказаться, что и делать ничего не нужно. Тело такого цикла может ни разу не выполняться. Каждое выполнение тела цикла - это очередной шаг к завершению цикла.

Цикл, проверяющий условие завершения в конце, соответствует алгоритму: "сначала делай, а потом проверь". Тело такого цикла выполняется, по меньшей мере, один раз. Вот синтаксис этой модификации:

```
do
   оператор
while(выражение);
```

Оба эти цикла используются, как правило, тогда, когда точно не известно, сколько раз цикл должен выполнится. Например, при вводе пользователем пароля или при подсчете чего-либо с определенной точ-

ностью. Оба эти цикла будут выполняться до тех пор, пока условие в круглых скобках после слова while будет истинно. Как только условие станет равным false, выполнение цикла прекращается. Самое важное отличие между while и do-while в том, что while может не выполниться ни одного раза, тогда как do-while по крайней мере один раз выполнится. Пример использования:

```
class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введите пароль");
            string password;
            do
            {
                password = Console.ReadLine();
            }
            while (password != "C#");
            //-----------------------------------
        }
    }
```

В примере цикл будет выполняться до тех пор, пока пользователь не введет правильный пароль (C#).

### 3.2.4. Операторы перехода

Операторов перехода, позволяющих прервать естественный порядок выполнения операторов блока, в языке C# имеется несколько.

#### Операторы break и continue

Операторы break и continue позволяют при выполнении некоторого условия выйти из цикла, из оператора выбора, из блока.

Оператор break может стоять в теле цикла или завершать case-ветвь в операторе switch. Пример его использования в операторе switch уже демонстрировался. При выполнении оператора break в теле цикла завершается выполнение самого внутреннего цикла. В теле цикла, чаще всего, оператор break помещается в одну из ветвей оператора if, проверяющего условие преждевременного завершения цикла:

```
public void Jumps()
{
   int i = 1, j=1;
   for(i = 1; i<100; i++)
   {
      for(j = 1; j<10;j++)
      {
         if (j>=3) break;
      }
      Console.WriteLine("Выход из цикла j при j = {0}", j);
      if (i>=3) break;
   }
   Console.WriteLine("Выход из цикла i при i= {0}", i);
}
```

Оператор continue используется только в теле цикла. В отличие от оператора break, завершающего внутренний цикл, continue осуществляет переход к следующей итерации этого цикла.

#### Оператор return

Еще одним оператором, относящимся к группе операторов перехода, является оператор return, позволяющий завершить выполнение процедуры или функции. Его синтаксис:

```
return [выражение];
```

Для функций его присутствие и аргумент обязателен, поскольку выражение в операторе return задает значение, возвращаемое функцией.

## 3.3. Класс Math

Стандартные математические функции в C# собраны в отдельном классе Math и являются методами класса.

Класс Math, содержащий стандартные математические функции, содержит два статических поля, задающих константы **e** и **π**, а также 23 статических метода. Методы задают:

- тригонометрические функции - Sin, Cos, Tan;
- обратные тригонометрические функции - ASin, ACos, ATan, ATan2 (sinx, cosx);
- гиперболические функции - Tanh, Sinh, Cosh;
- экспоненту и логарифмические функции - Exp, Log, Log10;
- модуль, корень, знак - Abs, Sqrt, Sign;
- функции округления - Ceiling, Floor, Round;
- минимум, максимум, степень, остаток - Min, Max, Pow, IEEERemainder.

В особых пояснениях эти функции не нуждаются.

Пример использования:

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

namespace temp11
{
    class Program
    {
        static void Main(string[] args)
        {

   double a, b,t,t0,dt,y;
   string NameFunction;
   Console.WriteLine("Введите имя F(t)исследуемой функции a*F(b*t)" + " (sin,
cos, tan, cotan)");
   NameFunction = Console.ReadLine();
   Console.WriteLine("Введите параметр a (double)");
   a= double.Parse(Console.ReadLine());
   Console.WriteLine("Введите параметр b (double)");
   b= double.Parse(Console.ReadLine());
   Console.WriteLine("Введите начальное время t0(double)");
   t0= double.Parse(Console.ReadLine());
   const int points = 10;
   dt = 0.2;
   for(int i = 1; i<=points; i++)
   {
      t = t0 + (i-1)* dt;
      switch (NameFunction)
      {
         case ("sin"):
            y = a*Math.Sin(b*t);
            break;
         case ("cos"):
            y = a*Math.Cos(b*t);
            break;
         case ("tan"):
            y = a*Math.Tan(b*t);
            break;
         case ("cotan"):
            y = a/Math.Tan(b*t);
            break;
         case ("ln"):
            y = a*Math.Log(b*t);
            break;
         case ("tanh"):
            y = a*Math.Tanh(b*t);
            break;
         default:
```

```
            y=1;
            break;
        }
        Console.WriteLine ("t = " + t + "; " + a +"*" +
           NameFunction +"(" + b + "*t)= " + y + ";");
    }
    double u = 2.5, v = 1.5, p;
    p= Math.Pow(u,v);
    Console.WriteLine ("u = " + u + "; v= " + v + "; power(u,v)= " + p );
}
        }
     }
```

В данном примере пользователь определяет, какую функцию он хочет вычислить и при каких значениях ее параметров.

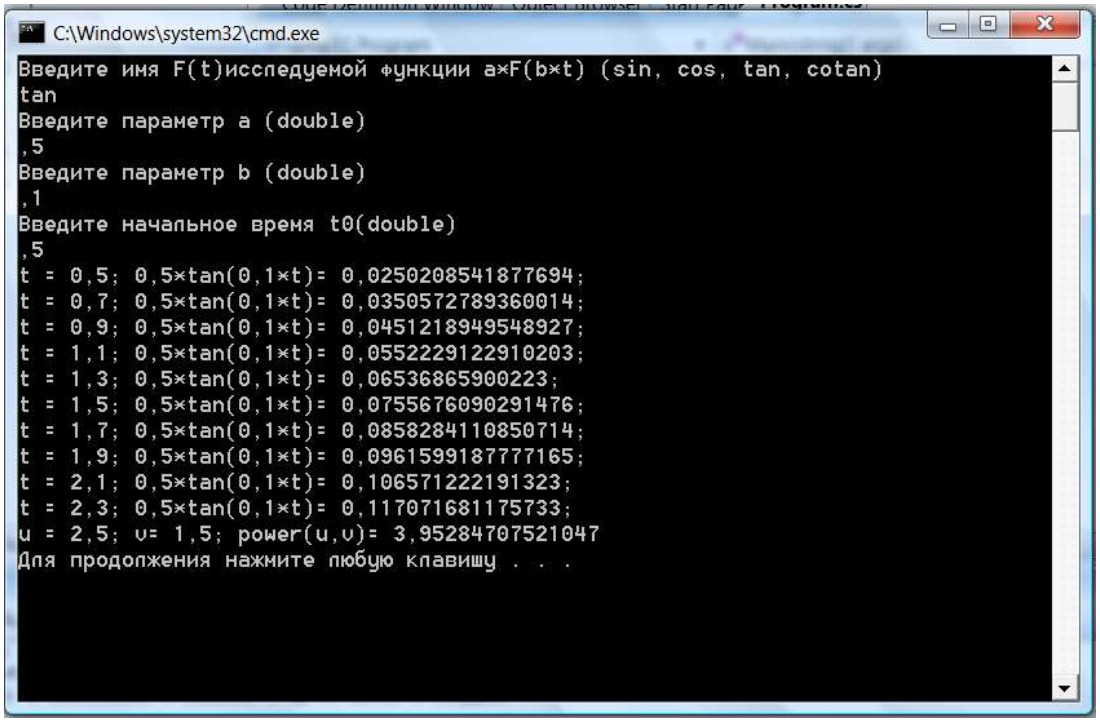


**Рис.7.** Результаты работы программы

## Задание.

Изучить материал, приведенный выше, и в соответствии с заданным вариантом написать программы из Задача1 … Задача 4.

## Варианты индивидуальных заданий.

### Задача 1.

Составить программу, задав исходные данные самостоятельно.

1. Цветочная клумба имеет форму круга. По заданному радиусу вычислить ее периметр и площадь.
2. Вычислить периметр и площадь прямоугольного треугольника по заданному катету и острому углу.
3. По заданному диаметру вычислить длину и площадь окружности.
4. Участок леса имеет форму равнобочной трапеции. Вычислить по заданным сторонам ее периметр и площадь.

5. Ресторан покупает каждый день масло $m_1$ кг по 8.50 грн. за киллограм, сметану $m_2$ кг по 2.40 грн., сливки $m_3$ кг по 4.10 грн. Определить суммы, необходимые для закупки отдельных продуктов и общую сумму.
6. Сколько секунд имеют сутки, неделя, год ?
7. Вычислить кинетическую $E=mv^2/2$ и потенциальную $P=mgh$ энергии тела заданной массы $m$, которое движется на высоте $h$ со скоростью $v$.
8. Цены на два вида товаров выросли на $p$ процентов. Вывести старые и новые цены.
9. Вычислить площадь поверхности $S=4\pi r^2$ и объем $V=4nr^3/3$ шара по радиусу $r$.
10. Скорость света 299792 км/с. Какое расстояние преодолевает свет за час, сутки?
11. Ввести урожайность трех сортов пшеницы (36, 40, 44 т/га) и размеры трех соответственных полей (у га). Сколько собрали пшеницы с каждого поля и со всех вместе?
12. Радиус Луны 1740 км. Вычислить площадь поверхности $S=4\pi r^2$ и объем планеты $V=(4/3)nr^3$.
13. Вычислить длину гипотенузы и площадь прямоугольного треугольника по двум катетам.
14. По известному ребру вычислить объем и площадь боковой поверхности куба.
15. По производительности трех труб и времени их работы по наполнению бассейна, определить сколько воды набрано в бассейн.
16. Определите площадь и периметр квадрата, который описан вокруг круга заданной площади $S$.
17. Тело падает с ускорением $g$. Определить путь тела $h=gt^2/2$ после первой и второй секунд падения.
18. По заданным катетам вычислить периметр и площадь прямоугольного треугольника.
19. Телефонные разговоры с тремя населенными пунктами стоят $c_1$, $c_2$, $c_3$ коп/мин. Разговоры длились $t_1$ $t_2$, $t_3$ мин. Какую сумму насчитает компьютер за каждый и все разговоры вместе.
20. По заданной высоте $h$ и радиусу основания $r$ вычислить площадь боковой поверхности $S=2\pi rh$ и объем $V=nr^2h$ бочки.
21. Цветочная клумба имеет форму квадрата. По заданной стороне вычислить площадь и периметр.
22. По заданному катету и гипотенузе вычислить другой катет и площадь прямоугольного треугольника.
23. Вычислить сторону и площадь квадрата $S=d^2/2$, если задана его диагональ $d$.
24. По заданной высоте $h$, образующей $l$ и радиусу основания $r$ вычислить площадь боковой поверхности $S=\pi rl$ и объем конуса $V=\pi r^2h/3$.
25. Поезд ехал $t_1$час со скоростью $v_1$ км/час, $t_2$ час со скоростью $v_2$ и $t_3$ час со скоростью $v_3$. Определить пройденный путь с разной скоростью и полный путь.

**Задача 2.**

По произвольному значению $x$ вычислить значение функции:

$$y = \begin{cases} f(\varphi), \text{если } |x| < 10, \\ f(\omega), \text{если } |x| \geq 10 \end{cases}, \text{ где}$$

$\varphi = tg(x/i + a) - ln|bi + 7|$, $\omega = c\sqrt[i]{x^2 + di^{1,2}}$ , $i$ - номер варианта. Входные данные вводятся с клавиатуры.

**Задача 3.**

Составить программу для решения поставленной задачи двумя способами: используя: 1) команды case 2) команды if. Данные сформировать таким образом, что бы выбор был из 3-5 альтернатив.

1. По номеру студента в списке группы вывести его фамилию.
2. Имеются данные 4 моделей автомобиля. По номеру модели получить характеристики: год выпуска и цену.
3. По номеру поезда вывести название пункта назначения.
4. По названию первой буквы страны вывести название столицы.
5. По номеру для недели напечатать его название.
6. По номеру троллейбуса вывести название конечной остановки.
7. По первой букве названия страны вывести название континента.
8. По номеру месяца вывести название поры года.
9. По номеру студента в списке группы вывести его имя.
10. По названию первой буквы города вывести справку по количеству населения.
11. По номеру месяца вывести номер квартала.
12. По номеру маршрута автобуса вывести количество остановок его маршрута.
13. По перовой букве названия страны вывести количество городов этой страны.
14. По телефонному коду города вывести название города.
15. По номеру дня недели вывести количество пар в этот день.

16. По цифровому коду шести групп товаров выводить его цену.
17. По номеру месяца выводить количество дней в этом месяце.
18. По коду группы вывести количество студентов в группе.
19. По числу из диапазона 0-5 напечатать его написание на английском языке.
20. По номеру поезда вывести время его отправления.
21. По названию реки вывести ее длину.
22. По первой букве названия реки вывести название страны, где она протекает.
23. По номеру одного из четырех друзей вывести его имя.
24. По номеру квартиры в доме вывести количество жильцов.
25. По цифре из диапазона 5-9 вывести ее значение словом.

### Задача 4.

Вычислить выражение согласно варианта:

1) z=a+b;
2) z=ab;
3) z=tg(b)–a;
4) $z=(a+b)^2$;
5) z=5ab–4;
6) z=sin(a)+b;
7) z=b tga;
8) $z=a^2+3b$;
9) $z=(|ab|)^{1}/^{4}$;
10) z=ab–π;
11) z=a–2b;
12) z=a tgb;
13) z=cos(ab);
14) z=|a–b|;
15) z=ctg(2a)–b;
16) $z=e^{3ab}$;
17) z=4ba–b;
18) z=2a–b;
19) z=|12a–cos(b)|;
20) z=2a–b;
21) z=tg(a+b);
22) z=ln|a+4b|;
23) z=3ab–cos(b);
24) $z=4a+e^{b}$;
25) z=5a–2b,

где

$$a=\sum_{x=i}^{i+8} f(x), \qquad b=\prod_{x=i}^{i+5} f(x)$$

i – номер варианта, x – целое число. Значение функции $f(x)$ выбираются из таблицы. Вывести полученный результат и исходные значения.

| n | Функція f(x) |
|---|---|
| 1 | $9,2\cos x^2-\lvert\sin x/1,1\rvert$ |
| 2 | $12,4\sin\lvert x/2,1\rvert-8,3\cos 1,2x$ |
| 3 | $\lvert\cos x/2,7\rvert+9,1\sin(1,2x+1)$ |
| 4 | $\lvert\sin x/3,12+\cos x^2\rvert-8,3\sin 3x$ |
| 5 | $\cos\lvert 2x\rvert/1,12-\cos(3x-2)+6,15$ |
| 6 | $\sin x\cos x^2\sin(x+1,4)+5,14$ |
| 7 | $\lvert\sin(2x-1,5)+3\sin x^2\rvert+2,38$ |
| 8 | $\cos x^2\sin(2x-1)+4,29$ |
| 9 | $\cos(x^2+1)-\lvert\sin 2x-5,76\rvert$ |
| 10 | $\sin x-\cos x^3\sin(x^2-4,2)+4,27$ |
| 11 | $\lvert\sin 12x\cos\lvert 2x\rvert/3\rvert+4,21$ |
| 12 | $\cos x^3/2,1+\cos x^2/1,1-8.3\sin(3x+1)$ |
| 13 | $\sin x^2\cos x^3-\sin x+5,2$ |
| 14 | $2\sin x\sin(2x-1,5)\cos(2x+1,5)-6$ |
| 15 | $\lvert\cos x^2-0,51\rvert\sin(3x-4)-4,44$ |
| 16 | $\cos 2,1x\sin\lvert x\rvert/0,15-5,8$ |
| 17 | $\lvert\cos 2x^3+2\sin(x/1,2-3,4)\rvert+10,51\cos\lvert 3x\rvert$ |
| 18 | $\lvert\sin(x^2/1,5-2)\rvert+11,73\cos(1,6x-1)$ |
| 19 | $13,4\cos\lvert x\rvert\sin(x^2-2,25)$ |
| 20 | $\lvert\cos(x^2-3,8)\rvert/4,5-9,7\sin(x-3,1)$ |
| 21 | $13,4\sin(-1,26)\cos\lvert x/7,5\rvert$ |
| 22 | $2\sin\lvert 2x\rvert\cos 2x-11,6\sin(x/0,4-1)$ |
| 23 | $\sin\lvert x\rvert/0,1+ 9,4\sin(3x-2,5)$ |

| | |
|---|---|
| 24 | 10,8cos($x^2$/1,13)\|sin(x+1,4) |
| 25 | 11,2cos(2x−1)+\|sinl,5x\|/1,7 |