Ввод информации. Конвертация

Присваивание переменной введенного числа

В переменную можно записывать информацию, считанную с клавиатуры. Чтобы введенные данные с консоли сохранить в переменную нужно:

string s = Console.ReadLine();

Все данные, которые считываются с консоли имеют строковый тип данных (**string**). Напишем программу, которая будет спрашивать имя у пользователя и после здороваться с ним:

Console.WriteLine("Привет! Как тебя зовут?");

string name = Console.ReadLine();

Console.WriteLine("Привет, " + name);

Однако, если мы не хотим работать со строковым типом, а необходимо работать с целым числом, то нужно будет перевести строку в число.  Чтобы перевести переменную строкового типа в целый тип данный (**int**), нужно:

string s = Console.ReadLine();  
int a = Convert.ToInt32(s);

Мы воспользовались готовой функцией перевода строки с целое число. Или можно сказать "сконвертировали". Представьте, что вы взяли строку и сконвертировали в конвертере, как это происходит с валютами. В нашем случае, строку поменяли на целое число. Предыдущую программу можно еще сократить, подставив вместо переменной **s** его значение, то есть верхняя программа идентично следующей

int a = Convert.ToInt32(Console.ReadLine());

Чтобы перевести перевести переменную строкового типа в другой тип данных, существуют другие конвертеры типов:

string s = Console.ReadLine();  
int number = Convert.ToInt32(s);   
double double = Convert.ToDouble(s);   
char character = Convert.ToChar(s);   
bool boolean = Convert.ToBoolean(s);

Для лучшего понимания напишем программу, которая принимает два числа и выводит их сумму:

int a = Convert.ToInt32(Console.ReadLine());  
int b = Convert.ToInt32(Console.ReadLine());  
int c = a + b;   
Console.WriteLine(c);

Операции с числами

Все операции делятся на два типа:

* **Унарные**;
* **Бинарные**.

К **унарным** относятся операции, в которых участвует **один** операнд. В бинарных операциях – **два** операнда.

**Операнд** – это данные, которые принимают участие в операции.

Бинарные операторы

Например, оператор сложения **«+»** – бинарный: 2+3 – операндами являются числа 2 и 3.

Список **бинарных** арифметических операций:

* сложение - *a*+*b;*
* вычитание - *a*−*b;*
* умножение - *a*∗*b;*
* деление:
  + нацело - *a*/*b;*
  + с остатком - *a* % *b.*

Операции **+, -, \*** такие же как в математике. Деление в программирование делится на два разных деления: **нацело** и **с остатком**. При делении двух **целых** чисел результатом также будет **целое** число.

**Оператор %** возвращает остаток от деления.

static void Main(string[] args)  
{  
    int a = 5;  
    int b = 2;

int result;  
    Console.WriteLine("a = " + a + ", b = " + b);  
   
    result = a + b;  
    Console.WriteLine("Сложение, a + b = " + result);  
   
    result = a \* b;  
    Console.WriteLine("Умножение, a \* b = " + result);  
   
    result = a / b;  
    Console.WriteLine("Деление, a / b = " + result);  
   
    result = a % b;  
    Console.WriteLine("Остаток от деления, a % b = " + result);  
}

При вычислений выражение содержащих несколько операций, приоритет операций такой же как в математике. Чтобы повысить приоритет операции, используются **скобки**.

Унарные операторы

Унарных арифметических операторов есть всего два:

* **инкрементация** «++»;
* **декрементация** «--».

Инкрементация **увеличивает** операнд на **единицу**, а декрементация - **уменьшает** на **единицу**.

static void Main()

{

int a = 0, b = 5;

a++;

b--;

}

Вещественные числа

Данные вещественного (**double**) типа используются значительно чаще целого типа. Они необходимы в тех случаях, когда числовые значения могут содержать дробные части. Дробная часть отделяется от целой части при при помощи **точки**.

double d = 0.5;

Считываются вещественные переменные точно также как целые, только конвертация происходит в нужный тип данных, то есть:

double d = Convert.ToDouble(Console.ReadLine());

Точно такие же операции поддерживаются и с данным типом данных как и с целым.

Как мы говорили, при делении **двух целых** чисел, результатом также будет **целое** число. Чтобы получить **точный** результат с десятичной точкой, нужно, чтобы делимое и/или делитель были типа **double**. Тогда деление будет происходить как в математике:

int a = 5;

int b = 2;

Console.WriteLine(a / b); // 2

double c = 2;

Console.WriteLine(a / c); // 2.5

Целые и действительные числа в одном арифметическом выражении

Если мы хотим сохранить в вещественной переменной целое число, то достаточноприсвоить переменной типа **double** значение целочисленного арифметического выражения. Это возможно, так как множество вещественных чисел содержит в себе множество целых чисел:

int a = 8;  
double c = a;

**Внимание!** Если оба операнда (те, что стоят слева и справа от знака операции) **целые** и результат записывается в переменную вещественного типа, то все равно операция будет выполняться в **целых** числах, так сначала выполняется правая часть от оператора присвоения (=):

int a = 5, b = 3;  
double c = a / b; // с = 1

Деление будет выполнено в **целых** числах, а уже после этого результат будет преобразован в **действительное** число. Чтобы избавиться от этой проблемы, необходимо **явно** привести один из операндов к типу **double**. Чтобы привести переменную целого типа к вещественному типу, надо в круглых скобках перед именем переменной записать **double**. Таким образом, чтобы в переменной "с" оказался правильный результат, нужно написать:

int a = 5, b = 3;  
double c = (double)a / b;  
Console.WriteLine(c); // 1.6666666667

В приведенном примере, при операции деления, сначала целое число **а преобразуется в вещественное**, затемданное вещественное число делитсяна целое число **b**, следовательно деление происходит как в математике, то есть появляется **дробная часть**.

**Примечание**: работа с вещественными числа очень чревато ошибками в точности. Особенно это проявляется при делении и умножении.

Явное приведение типов

Мы не всегда можем присвоить значение переменной другой переменной, если они разных типов. Для этого нам необходимо **явно привести** значение одной переменной в значенгие второй переменной:

int a = 5;

double b = (double)a;

int a = 5;

string b = (string)a;

double a = 5;

string b = (string)a;

Класс Math

В классе **Math** собраны все основные математические функции: модуль числа, функция возведение числа в степень, нахождение квадратного корня и другие.

Для **нахождения модуля числа**, используется функция **Abs**([число]):

static void Main(string[] args)

{

int b = -9;

double a = Math.Abs(b);

Console.WriteLine(a);

}

Для **возведения числа в степень**, используется функция **Pow**([число], [степень]):

static void Main(string[] args)

{

   double b = 9;

   double a = Math.Pow(b, 2);

   Console.WriteLine(a);

}

  Для **нахождения квадратного корня** служит функция **Sqrt**([число]) возвращаемый тип данных – **double**:

static void Main(string[] args)  
{  
   double b = 9;

   double a = Math.Sqrt(b);  
   Console.WriteLine(a);   
}