ТИПЫ ДАННЫХ. ВВОД-ВЫВОД C#

ТИПЫ ДАННЫХ

Как и во многих языках программирования, в C# есть своя система типов данных, которая используется для создания переменных.

Тип данных определяет их внутреннее представление, множество значений, которые может принимать объект, а также допустимые действия, которые можно осуществлять над объектом.

В ЯЗЫКЕ C# ЕСТЬ СЛЕДУЮЩИЕ ПРИМИТИВНЫЕ ТИПЫ ДАННЫХ:

bool — хранит значение true или false (логические литералы).

Представлен системным типом **System.Boolean**

```
bool alive = true;
bool isDead = false;
```

byte — хранит целое число от 0 до 255 и занимает 1 байт. Представлен системным типом **System.Byte**

```
byte bit1 = 1;
byte bit2 = 102;
```

short — хранит целое число от -32768 до 32767 и занимает 2 байта. Представлен системным типом **System.Int16**

```
short n1 = 1;
short n2 = 102;
```

int — хранит целое число от -2147483648 до 2147483647 и занимает 4 байта.

Представлен системным типом **System.Int32**.

Все целочисленные литералы по умолчанию являются значениями типа **int**:

```
int a = 10;
int b = 0b101; // бинарная форма b =5
int c = 0xFF;  // шестнадцатиричная форма c = 255
```

long — хранит целое число от -9223372036854775808 до 9223372036854775807 и занимает 8 байт.

Представлен системным типом **System.Int64**

```
long a = -10;
long b = 0b101;
long c = 0xFF;
```

float — хранит число с плавающей точкой от -3.4*1038 до 3.4*1038 и занимает 4 байта.

Представлен системным типом **System.Single**;

double — хранит число с плавающей точкой от ±5.0*10-324 до ±1.7*10308 и занимает 8 байтов.

Представлен системным типом **System.Double**;

decimal — хранит десятичное дробное число. Если употребляется без десятичной запятой, имеет значение от ±1.0*10-28 до ±7.9228*1028, может хранить 28 знаков после запятой и занимает 16 байтов.

Представлен системным типом **System.Decimal**;

char — хранит одиночный символ в кодировке Unicode и занимает 2 байта.

Представлен системным типом **System.Char**, которому соответствуют символьные литералы

```
char a = 'A';
char b = '\x5A';
char c = '\u0420';
```

string — хранит набор символов **Unicode**.

Представлен системным типом **System.String**, которому соответствуют символьные литералы.

```
string hello = "Hello";
string word = "world";
```

object — может хранить значение любого типа данных и занимает 4 байта на 32-разрядной платформе и 8 байтов на 64-разрядной платформе.

Представлен системным типом **System.Object**, который является базовым для всех других типов и классов .NET:

```
object a = 22;
object b = 3.14;
object c = "hello code";
```

НАПРИМЕР, ОПРЕДЕЛИМ НЕСКОЛЬКО ПЕРЕМЕННЫХ РАЗНЫХ ТИПОВ
И ВЫВЕДЕМ ИХ ЗНАЧЕНИЯ НА КОНСОЛЬ:

```
static void Main(string[] args)
    {
        string name = "Tom";
        int age = 33;
        bool isEmployed = false;
        double weight = 78.65;
         Console.WriteLine($"Имя: {name}");
        Console.WriteLine($"Возраст: {age}");
        Console.WriteLine($"Вес: {weight}");
        Console.WriteLine($"Работает: {isEmployed}");
    }
```

Для вывода данных на консоль здесь применяется **интерполяция**: перед строкой ставится знак **$**, и после этого мы можем вводить в строку значения переменных в фигурных скобках.

Консольный вывод программы:

Имя: Tom

Возраст: 33

Вес: 78,65

Работает: False

ИСПОЛЬЗОВАНИЕ СУФФИКСОВ.

При присвоении значений надо иметь в виду следующую тонкость: все вещественные литералы рассматриваются как значения типа **double**.

А чтобы указать, что дробное число относится к типу **float** или **decimal**, необходимо к литералу добавлять суффикс: **F/f** — для **float** и **M/m** — для **decimal**.

```
float a = 3.14F;
float b = 30.6f;

decimal c = 1005.8M;
decimal d = 334.8m;
```

Подобным образом все целочисленные литералы рассматриваются как значения типа **int**.

Чтобы явно указать, что целочисленный литерал является значением типа **long** — используйте суффикс **L/l**.

```
long b = 20L;
```

НЕЯВНАЯ ТИПИЗАЦИЯ.

Ранее мы явно указывали тип переменных, например int x;. и компилятор при запуске уже знал, что x хранит целочисленное значение.

Однако можно использовать модель **неявной типизации**:

```
var hello = "Hello to World";
var c = 20;

Console.WriteLine(c.GetType().ToString());
Console.WriteLine(hello.GetType().ToString());
```

При неявной типизации вместо названия типа данных используется ключевое слово var.

Затем при компиляции компилятор сам выводит тип данных, исходя из присвоенного значения.

В примере выше использовалось выражение **Console.WriteLine(c.GetType().ToString());,** которое позволяет узнать выведенный тип переменной **c**.

Так как по умолчанию все целочисленные значения рассматриваются как значения типа **int**, переменная с будет иметь тип **int** или **System.Int32**.

ОГРАНИЧЕНИЯ

Мы не можем сначала объявить неявно типизируемую переменную, а затем инициализировать:

```
// этот код работает
int a;
a = 20;
 // этот код не работает
var c;
c= 20;
```

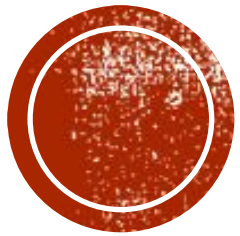
Мы не можем указать в качестве значения неявно типизируемой переменной **null**:

```
// этот код не работает
var c=null;
```

Так как значение **null**, компилятор не сможет вывести тип данных.

DOUBLE ИЛИ DECIMAL

Decimal чаще находит применение в финансовых вычислениях, тогда как **double** — в математических операциях.



КОНСОЛЬНЫЙ ВВОД-ВЫВОД

КОНСОЛЬНЫЙ ВЫВОД.

Для вывода информации на консоль выше использовался встроенный метод **Console.WriteLine**.

То есть если мы хотим вывести некоторую информацию на консоль, то нам надо передать ее в метод **Console.WriteLine**:

```
string hello = "Привет мир";
        Console.WriteLine(hello);
        Console.WriteLine("Добро пожаловать в C#!");
        Console.WriteLine("Пока мир...");
        Console.WriteLine(24.5);


        Console.ReadKey();
```

Консольный вывод:

```
Привет мир!
Добро пожаловать в C#!
Пока мир...
24,5
```

Нередко возникает необходимость в выводе на консоль в одной строке значений сразу нескольких переменных. В этом случае можно использовать прием, который называется **интерполяцией**:

```
string name = "Tom";
int age = 34;
double height = 1.7;
Console.WriteLine("Имя: {0} Возраст: {2} Рост: {1}м", name, height, age);
Console.ReadKey();
```

Этот способ подразумевает, что первый параметр при использовании метода **Console.WriteLine** является выводимой строкой

("Имя: {0} Возраст: {2} Рост: {1}м").

Все последующие параметры — значения, которые могут быть встроены в эту строку

(name, height, age).

При этом важен порядок подобных параметров.

Например, в данном случае сначала идет **name**, потом **height** и затем **age**.

Поэтому **name** будет иметь параметр с номером **0** (нумерация начинается с нуля), **height** — с номером **1**, **age** — с номером **2**.

Поэтому в строке "**Имя: {0} Возраст: {2} Рост: {1}м**" на место плейсхолдеров **{0}**, **{2}**, **{1}** будут вставляться значения соответствующих параметров.

Кроме **Console.WriteLine**() можно использовать метод **Console.Write**(); он работает так же, за исключением того, что не осуществляет переход на следующую строку.

КОНСОЛЬНЫЙ ВВОД.

Кроме вывода информации на консоль можно получать информацию с консоли. Для этого предназначен метод **Console.ReadLine**().

Он позволяет получить введенную строку:

```
Console.Write("Введите свое имя: ");
        string name = Console.ReadLine();
        Console.WriteLine($"Привет {name}");
         Console.ReadKey();
```

В данном случае все, что вводит пользователь, с помощью метода **Console.ReadLine** передается в переменную **name**.

Однако минусом этого метода является то, что **Console.ReadLine** считывает информацию именно в виде **строки**.

Поэтому по умолчанию можно присвоить ее только переменной типа **string**.

Как быть, если, допустим, мы хотим ввести возраст в переменную типа **int** или другую информацию в переменные типа **double** или **decimal**?

По умолчанию платформа **.NET** предоставляет ряд методов, которые позволяют преобразовать различные значения к типам **int**, **double** и т.д. Некоторые из этих методов:

- **Convert.ToInt32() (преобразует к типу int);**
- **Convert.ToDouble() (преобразует к типу double);**
- **Convert.ToDecimal() (преобразует к типу decimal).**

```
Console.Write("Введите имя: ");
        string name = Console.ReadLine();
---------------------------------------------------------
         Console.Write("Введите возраст: ");
        int age = Convert.ToInt32(Console.ReadLine());
-----------------------------------------------------------
 Console.Write("Введите рост: ")
   double height = Convert.ToDouble(Console.ReadLine());
-----------------------------------------------------------
 Console.Write("Введите размер зарплаты: ");
      decimal salary = Convert.ToDecimal(Console.ReadLine());
-----------------------------------------------------------
Console.WriteLine($"Имя: {name} Возраст: {age} Рост: {height}м
Зарплата: {salary}$");
```

При вводе важно учитывать **текущую операционную систему.**

В одних культурах разделителем между целой и дробной частью является **точка** (США, Великобритания и др.), в других — **запятая** (Россия, Германия и др.).

Например, если текущая операционная система (ОС) русскоязычная, значит, надо вводить дробные числа с **запятой** в качестве разделителя.

Если локализация англоязычная, значит, разделителем целой и дробной частей при вводе будет **точка**.