Типы данных C#. Объявление и инициализация

Система типов

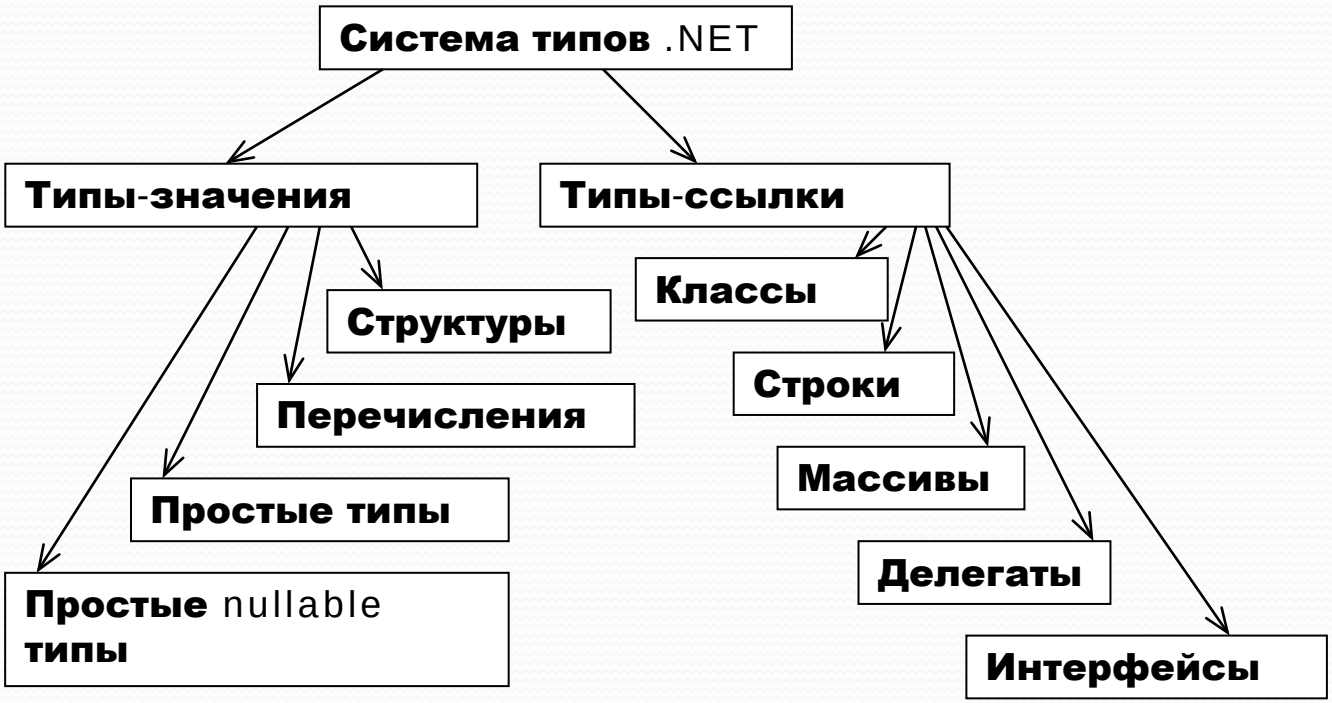
C# является языком со строгой типизацией данных.

.NET Framework предоставляет общую систему типов CTS (Common Type System), использование которой позволяет разрабатывать приложение на любом из языков, поддерживающих эту среду.

Система типов поддерживает две категории типов, каждая из которых разделена на подкатегории:

- **типы значений (типы–значения),**
- **ссылочные типы (типы–ссылки).**

Схема типов представлена на рисунке :



Для переменной типа значение в отведенной для неё ячейке памяти содержится значение этой переменной.

Для переменной ссылочного типа в отведенной для неё ячейке памяти содержится адрес области памяти, в которой записано значение этой переменной.

Отличия типов-значения от типов-ссылок

	Типы-значения value-types	Типы-ссылки reference-types
Объект представлен	непосредственно значением	ссылкой в куче
Объект располагается	в стеке	в куче
Значение по умолчанию	0, false, '\0', null	ссылка имеет значение null
При выполнении операции присваивания копируется	значение	ссылка

Для типов-ссылок необходимо явно выделять место в памяти, используя метод ***New().***

Все типы (типы-значения и типы-ссылки), за исключением простых типов-значений и пары предопределённых ссылочных типов (string и object), *должны определяться* (если уже не были ранее специально определены) программистами в рамках объявлений.

Подлежащие объявлению типы называются производными типами.

Простые (элементарные) типы

Это типы, имя и основные свойства которых известны компилятору. Относительно элементарных типов компилятору не требуется никакой дополнительной информации. Свойства и функциональность этих типов известны.

Используемые в .NET языки программирования основываются на общей системе типов. Между именами простых типов в C# и именами типов, объявленных в Framework Class Library, существует взаимно однозначное соответствие (см. табл.).

Соответствие простых типов C# FCL-типу

Тип в C#	Соответствует FCL–типу	Описание
sbyte	System.SByte	**Целый. 8–разрядное со знаком. Диапазон значений: 128 … 127**
byte	System.Byte	**Целый. 8–разрядное без знака. Диапазон значений: 0 … 255**
short	System.Int16	**Целый. 16–разрядное со знаком. Диапазон значений: –32768 … 32767**
ushort	System.UInt16	**Целый. 16–разрядное без знака. Диапазон значений: 0 … 65535**
int	System.Int32	**Целый. 32–разрядное со знаком. Диапазон значений: –2147483648 … 2147483647**
uint	System.UInt32	**Целый. 32–разрядное без знака. Диапазон значений: –0 … 4294967295**

Соответствие простых типов C# FCL-типу

Тип в C#	Соответствует FCL–типу	Описание
long	System.Int64	Целый. 64–разрядное со знаком. Диапазон значений: –9223372036854775808 … 9223372036854775807
ulong	System.UInt64	Целый. 64–разрядное без знака. Диапазон значений: 0 … 18446744073709551615
char	System.Char	16 (!) разрядный символ UNICODE.
float	System.Single	Плавающий. 32 разряда. Стандарт IEEE.
double	System.Double	Плавающий. 64 разряда. Стандарт IEEE.
decimal	System.Decimal	128–разрядное значение повышенной точности с плавающей точкой.
bool	System.Boolean	Значение true или false.

Литералы. Представление значений

В программах на языках высокого уровня (C# в том числе) ***литералами*** называют последовательности входящих в алфавит языка программирования символов, обеспечивающих явное представление значений, которые используются для обозначения начальных значений в объявлении членов классов, переменных и констант в методах класса.

Арифметические литералы кодируют значения различных (арифметических) типов. Тип арифметического литерала определяется следующими интуитивно понятными внешними признаками

Символьные литералы

Представляют собой **заключённые в одинарные кавычки** вводимые с клавиатуры одиночные символы: 'X', 'p', 'Q', '7',

Строковые литералы

Это последовательность символов и символьных управляющих последовательностей, **заключённых в двойные кавычки.**

```
…"c:\\My Documents\\sample.txt"…
```

Объявление и инициализация

Выполнение оператора объявления переменной типа-значения в методе класса приводит к созданию в памяти объекта соответствующего типа, возможно проинициализированного определённым значением. Это значение может быть задано в виде литерала соответствующего типа

Например,

```
//определения переменных типов-значений:
int a;
System.Int32 a;


//определение и инициализация переменных типа значения:
int a = 0;
int a = new int();
System.Int32 a = 0;
System.Int32 a = new System.Int32();
```

Область видимости

При определении классов и методов используются фигурные скобки. Содержимое между этими скобками называется блоком. Блоки могут быть вложены друг в друга. Любая переменная, объявленная внутри конкретного блока, называется локальной переменной для этого блока, и она не существует вне, то есть ее нельзя использовать в других блоках, если они не являются вложенными.

```
{
        {
                double a=3;
                . . .
                {
                        Console.Writeln(a);      //правильно
                }
        }
        Console.Writeln(a);              //ошибка компиляции
}
```

Не допускается объявление переменной во вложенном блоке с тем же именем, что и переменная, объявленная в основном блоке:

```
{
        double a=3;
        . . .
        {
            double a;    //ошибка компиляции
        }
}
```

Константы

Объявляются с дополнительным спецификатором **const.** Требуют непосредственной инициализации:

```
//константа инициализируется литералом 3.14.
        const float Pi = 3.14;
```

Пример

```
Int32 x; Double y; string s;

Console.Write("Введите X=");
    s = Console.ReadLine();
    x = Convert.ToInt32(s);
 Console.Write("Введите Y=");
    s = Console.ReadLine();
    y = Convert.ToDouble(s);
Console.WriteLine("Произведение X*Y= {0,5:g}", x * y);
            Console.ReadKey();
```