# Типы данных

МАЛЬГИНА Н.Г., 2018

# Тип данных

**Тип данных** (*тип*) — множество значений и операций на этих значениях (IEEE Std 1320.2-1998)

Тип данных характеризует одновременно:

- множество допустимых значений, которые могут принимать данные, принадлежащие к этому типу;
- набор операций, которые можно осуществлять над данными, принадлежащими к этому типу.

# Типы данных делят на:

| *Скалярные (простые):* | *Нескалярные* (агрегатные, составные) |
| --- | --- |
| • *Целые*<br>• *Вещественные*<br>• *Логические*<br>• *Символьные и т.д.* | • Массивы,<br>• Списки<br>• Файлы и т.д. |

# Простой тип

**Простой тип** - тип данных, о объектах которого, переменных или постоянных, можно сказать следующее:

- работа с объектами осуществляется с помощью конструкций языка;
- внутреннее представление значений объектов может зависеть от реализации транслятора (компилятора или интерпретатора) и от платформы;
- объекты не включают в себя другие объекты и служат основой для построения других объектов (составной (сложный) тип).

# Как правило, к простым относятся числовые типы:

- типы для хранения целых чисел с разной точностью;
- типы для хранения символов строк;
- тип для хранения значений true и false;
- типы для хранения вещественных чисел с разной точностью;
- и другие

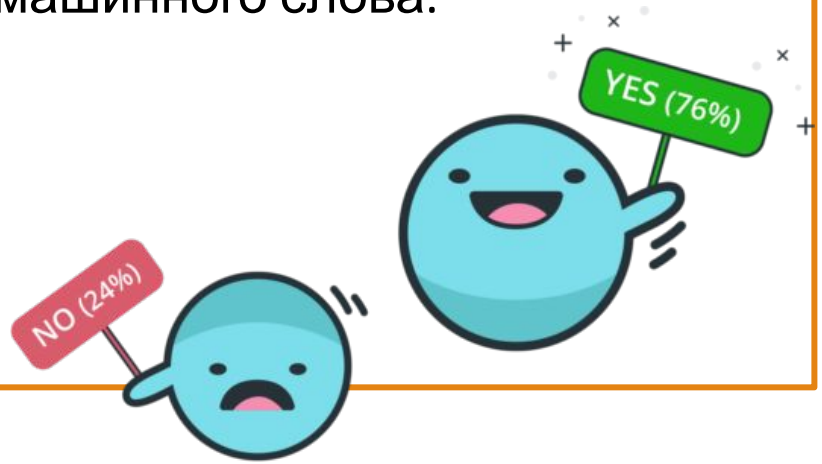
bool alive = true

# Логический тип

Логические, или булевы значения (по фамилии их изобретателя — Буля), могут иметь лишь одно из двух состояний — «истина» или «ложь». В разных языках обозначаются bool или Boolean. «Истина» может обозначаться как true. «Ложь», соответственно, false. В языках C и C++ любое ненулевое число трактуется как «истина», а ноль — как «ложь». В принципе, для реализации типа достаточно одного бита, однако из-за особенностей микропроцессоров, на практике размер булевых величин обычно равен размеру машинного слова.

Логический тип в языке C# - bool

ПРИМЕР: bool A=true;



# Доступные операции с логическим типом данных

***И*** (логическое умножение) (AND),

***ИЛИ*** (логическое сложение) (OR ),

***исключающее ИЛИ*** (сложение с переносом) (XOR ),

***эквивалентность*** (равенство) (EQV ,=,==)

***инверсия*** (NOT , ~, !=)

***сравнение*** (>, <, <=, >=)

# Целочисленные типы

Целочисленные типы содержат в себе значения, интерпретируемые как числа (знаковые и беззнаковые). Служит для представления целых чисел.

Количество чисел в машинном изображении множества целых чисел зависит от длины машинного слова, обычно выражаемой в битах. Например, при длине машинного слова 1 байт (8 бит) диапазон представимых целых чисел со знаком от -128 до 127. В беззнаковом формате байтовое представление числа будет от 0 до 255 ($2^8$ - 1). Если используется 32-разрядное машинное слово, то целое со знаком будет представлять значения от −2 147 483 648 ($-2^{31}$) до 2 147 483 647 ($2^{31}-1$); всего $1\ 0000\ 0000_{16}$ ($4\ 294\ 967\ 296_{10}$) возможных значений.

Ограничение длины машинного слова обусловлено конкретной аппаратной реализацией того или иного компьютера.

# Представление целых чисел

Многие языки программирования предлагают выбор между **короткими** (*short*), **длинными** (*long*) и целыми стандартной длины. Длина *стандартного целого типа*, как правило, совпадает с размером машинного слова на целевой платформе. Для 16-разрядных операционных систем — этот тип (int) составляет 2 байта и совпадает с типом short int (можно использовать как short, опуская слово int), для 32-разрядных операционных систем он будет равен 4 байтам и совпадает с длинным целым long int (можно использовать как long, опуская слово int), и в этом случае будет составлять 4 байта. Короткое целое short int, для 16-разрядных операционных систем, 32-разрядных операционных систем, и для большинства 64-разрядных операционных систем составляет — 2 байта. Также в некоторых языках может использоваться тип данных двойное длинное long long, который составляет 8 байт.

# Целочисленные типы в C#

| Тип | Описание | Пример |
|---|---|---|
| **byte** | хранит целое число от 0 до 255 и занимает 1 байт. Представлен системным типом System.Byte | `byte bit1 = 1;`<br>`byte bit2 = 102;` |
| **sbyte** | хранит целое число от -128 до 127 и занимает 1 байт | `sbyte bit1 = -101;`<br>`sbyte bit2 = 102;` |
| **short** | хранит целое число от -32768 до 32767 и занимает 2 байта | `short n1 = 1;`<br>`short n2 = 102;` |
| **ushort** | хранит целое число от 0 до 65535 и занимает 2 байта | `ushort n1 = 1;`<br>`ushort n2 = 102;` |
| **int** | хранит целое число от -2147483648 до 2147483647 и занимает 4 байта. | `int a = 10;` |
| **uint** | хранит целое число от 0 до 4294967295 и занимает 4 байта. | `uint a = 10;` |
| **long** | хранит целое число от –9 223 372 036 854 775 808 до 9 223 372 036 854 775 807 и занимает 8 байт | `long a = -10;` |
| **ulong** | хранит целое число от 0 до 18 446 744 073 709 551 615 и занимает 8 байт. | `ulong a = 10;` |

# Системные целочисленные типы в C#

| Тип | Описание |
|---|---|
| **byte** | Представлен системным типом System.Byte |
| **sbyte** | Представлен системным типом System.SByte |
| **short** | Представлен системным типом System.Int16 |
| **ushort** | Представлен системным типом System.UInt16 |
| **int** | Представлен системным типом System.Int32. |
| **uint** | Представлен системным типом System.UInt32 |
| **long** | Представлен системным типом System.Int64 |
| **ulong** | Представлен системным типомSystem.UInt64 |

# Использование системных типов

Выше при перечислении всех базовых типов данных для каждого упоминался системный тип. Потому что название встроенного типа по сути представляет собой сокращенное обозначение системного типа. Например, следующие переменные будут эквивалентны по типу:

```
int a = 4;
System.Int32 b = 4;
```

# Операции над целыми

**Сравнение**.

- Здесь применимы соотношения «равно» («=»; «==»; «eq»), «не равно» («!=»; «<>»; «ne»), «больше» («>»; «gt»), «больше или равно» («>=»; «ge»), «меньше» («<»; «lt») и «меньше или равно» («<=»; «le»).

**Инкремент** («++») и **декремент** («--»)

- Арифметическое увеличение или уменьшение числа на единицу. Выделено в отдельные операции из-за частого использования с переменными-счётчиками в программировании.

**Сложение** («+») и **вычитание** («-»).

**Умножение** («*»).

**Деление** («/»; «\») и получение **остатка от деления** ( «%»).

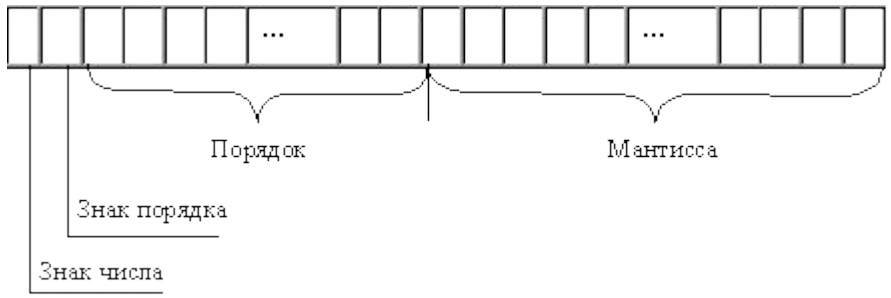
**Инверсия знака** и **получение абсолютного значения** .

**Возведение в степень** («^»).

# Числа с плавающей запятой

Используются для представления вещественных (не обязательно целых) чисел.

В этом случае число записывается в виде x=a*10^b. Где 0<=a<1, а b — некоторое целое число из определённого диапазона. a называют мантиссой, b — порядком. У мантиссы хранятся несколько цифр после запятой, а b — хранится полностью.



# Типы для хранения вещественных чисел

| Тип | Описание | |
|---|---|---|
| **float** | хранит число с плавающей точкой от $-3.4*10^{38}$ до $3.4*10^{38}$ и занимает 4 байта. | Представлен системным типом System.Single |
| **double** | хранит число с плавающей точкой от $\pm5.0*10^{-324}$ до $\pm1.7*10^{308}$ и занимает 8 байта. | Представлен системным типом System.Double |
| **decimal** | хранит десятичное дробное число, занимает 16 байт | Представлен системным типом System.Decimal |

# Строковые типы

Последовательность символов, которая рассматривается как единое целое в контексте переменной.

В программировании, **строковый тип** (*string*) — тип данных, значениями которого является произвольная последовательность (строка) символов алфавита. Каждая переменная такого типа (**строковая переменная**) может быть представлена фиксированным количеством байтов либо иметь произвольную длину.

# Представление в памяти строковых данных

Некоторые языки программирования накладывают ограничения на максимальную длину строки, но в большинстве языков подобные ограничения отсутствуют. При использовании Unicode каждый символ строкового типа может требовать двух или даже четырёх байтов для своего представления.

# В представлении строк в памяти компьютера существует два принципиально разных подхода:

**Представление массивом символов.** От названия языка Pascal, где этот метод был впервые реализован, данный метод получил название *Pascal strings*.

**Метод «завершающего байта»:**

Одно из возможных значений символов алфавита (как правило, это символ с кодом 0) выбирается в качестве признака конца строки, и строка хранится как последовательность байтов от начала до конца. Есть системы, в которых в качестве признака конца строки используется не символ 0, а байт 0xFF (255) или код символа «$». Наибольшее распространение метод получил в языке Си.

# Символьные и строковые данные в C#

| Тип | Описание | Пример |
|---|---|---|
| **char** | хранит одиночный символ в кодировке Unicode и занимает 2 байта. Представлен системным типом System.Char | `char a = 'A';` |
| **string** | хранит набор символов Unicode. Представлен системным типом System.String | `string hello = "Hello";` |

# Операции со строками

**Простейшие операции со строками**

**Производные операции**

**Операции при трактовке строк как списков**

**Более сложные операции**

**Возможные задачи для строк на естественном языке**

# Простейшие операции со строками

1. получение символа по номеру позиции (индексу);
2. конкатенация (соединение) строк

# Производные операции

- получение подстроки по индексам начала и конца;
- проверка вхождения одной строки в другую (поиск подстроки в строке);
- проверка на совпадение строк (с учётом или без учёта регистра символов);
- получение длины строки;
- замена подстроки в строке.

# Операции при трактовке строк как списков

1. свёртка;
2. отображение одного списка на другой;
3. фильтрация списка по критерию.

# Более сложные операции

1. нахождение минимальной надстроки, содержащей все указанные строки;
2. поиск в двух массивах строк совпадающих последовательностей *(задача о плагиате)*.

# Возможные задачи для строк на естественном языке

1. сравнение на близость указанных строк по заданному критерию;
2. определение языка и кодировки текста на основании вероятностей символов и слогов.

# Переменные

Для хранения данных в программе применяются **переменные**.

Переменная представляет именованную область памяти, в которой хранится значение определенного типа.

Переменная имеет **тип**, **имя** и **значение**.

Тип определяет, какого рода информацию может хранить переменная.

Перед использованием любую переменную надо определить. Синтаксис определения переменной выглядит следующим образом:

```
тип имя_переменной;
```

# Ограничения на имя переменной:

имя может содержать любые цифры, буквы и символ подчеркивания, при этом первый символ в имени должен быть буквой или символом подчеркивания

в имени не должно быть знаков пунктуации и пробелов

имя не может быть ключевым словом языка C#. Таких слов не так много, и при работе в Visual Studio среда разработки подсвечивает ключевые слова синим цветом.

# Особенности описания переменных на языке C#

C# является регистрозависимым языком, поэтому следующие два определения переменных будут представлять две разные переменные:

```
string name;
string Name;
```

После определения переменной можно присвоить некоторое значение:

```
string name;
name = "Tom";
```

можем сразу при определении присвоить переменной значение.

```
string name = "Tom";
```

# Неявные преобразования

Иногда может потребоваться скопировать значение в переменную другого типа. Для встроенных числовых типов неявное преобразование можно выполнить, если сохраняемое значение может уместиться в переменной без усечения или округления. Например, переменная типа long (64-разрядное целое число) может хранить любое значение, которое может хранить переменная int (32-разрядное целое число).

Существует таблица неявных числовых преобразований.

```
int num = 2147483647;
long bigNum = num;
```

# Явные преобразования

Приведение — это способ явно указать компилятору, что необходимо выполнить преобразование и что вам известно, что может произойти потеря данных. Чтобы выполнить приведение, укажите тип, в который производится приведение, в круглых скобках перед преобразуемым значением или переменной.

Существует также таблица явных числовых преобразований.

```
double x = 1234.7;
int a;
// Cast double to int.
a = (int)x;
```

При преобразовании вещественного типа в целочисленный, число округляется. Либо возникает исключение (ошибка), если оно не помещается в заданный диапазон.

# Использование суффиксов

При присвоении значений надо иметь в виду следующую тонкость: все вещественные литералы рассматриваются как значения типа **double**. И чтобы указать, что дробное число представляет тип **float** или тип **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;
```

# Неявная типизация

Для неявной типизации вместо названия типа данных используется ключевое слово var. Затем уже при компиляции компилятор сам выводит тип данных исходя из присвоенного значения. Т.е. в этом случае мы обязаны присвоить переменной начальное значение.

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

# Выбор типа

Из выше перечисленного списка типов данных очевидно, что если мы хотим использовать в программе числа до 256, то для их хранения мы можем использоват переменные типа byte.

При использовании больших значений мы можем взять тип short, int, long.

То же самое для дробных чисел - для обычных дробных чисел можно взять тип float, для очень больших дробных чисел - тип double.

Тип decimal здесь стоит особняком в том плане, что несмотря на большую разрядность по сравнению с типом double, тип double может хранить большее значение. Однако значение decimal может содержать до 28-29 знаков после запятой, тогда как значение типа double - 15-16 знаков после запятой.