

# Язык программирования Си

Элементы языка, типы данных, переменные,  
программа

*Югов Иван Олегович  
МОУ Гимназия №10, г. Тверь*

# Элементы языка

Язык Си включает следующие элементы:

- **Ключевые слова** — оформляют различные конструкции языка: `else`, `int`, `return`;
- **Знаки** — играют разделительную роль и обозначают некоторые операции: `;`, `+`, `&`, `(`;
- **Литералы (константы)** — фиксированные значения: `false`, `0`, `'Z'`, `25.4E2`, `"Привет"`;
- **Идентификаторы (имена)** — определяют конкретные объекты программы: `main`, `scanf`, `Temp`, `atan2`;
- **Директивы препроцессору** — определяют, как должен быть обработан код программы перед компиляцией: `#include`, `#define`;
- **Комментарии** — любой текст, заключённый между символами `/*` и `*/` и не содержащий символов `*/`:  
`/* Вычисление длины окружности */.`

# Основные типы

## Целочисленные:

- `char`
- `short int`
- `int`
- `long int`
- `long long int`

## Вещественные:

- `float`
- `double`
- `long double`

## «Пустой»:

- `void`

## Логический

# Числовые константы

*Литерал* — любое фиксированное значение, явно записанное в коде программы.

У компилятора — правило:

*«Если что-то начинается с цифры, то это числовая константа»*

Поэтому:

*Числовая константа всегда начинается с цифры*  
(не считая знаков +, – и десятичной точки)

# Целочисленные типы

Перед указанием типа можно ставить `signed` (знаковый) или `unsigned` (беззнаковый).

Тип	Бит	Диапазон <code>signed</code>	Диапазон <code>unsigned</code>
<code>char</code>	8	−128 ... 127	0 ... 255
<code>short int</code>	16	−32 768 ... 32 767	0 ... 65 535
<code>int</code>		как <code>short</code> или <code>long</code> , в зависимости от системы	
<code>long int</code>	32	−2 147 483 648 ... 2 147 483 647	0 ... 4 294 967 295
<code>long long int</code>	64	−9 223 372 036 854 775 808 ... 9 223 372 036 854 775 807	0 ... 18 446 744 073 709 551 615

По умолчанию всё — `signed`.

«`int`» можно не писать: `unsigned int ~ unsigned`; `long int ~ long`.

Определение разрядности типа — функция `sizeof()`.

Возвращает размер в байтах: `sizeof(int)`.

# Целочисленные типы

Основные типы	Синонимы Microsoft Visual C++
<code>char</code>	<code>__int8</code>
<code>short int</code>	<code>__int16</code>
<code>long int</code>	<code>__int32</code>
<code>long long int</code>	<code>__int64</code>

По умолчанию всё — signed.



# Целочисленные типы

Используемые системы счисления:

- **десятичная**: 5, +77, -190, 1000000000;
- **восьмеричная** — начинается с 0:  
015, 0100000, -0777;
- **шестнадцатеричная** — начинается с 0x  
или 0X: 0x9A, 0X294, -0x100000f;
- **двоичная** (только для GCC) — начинается  
с 0b или 0B: 0b111, 0B100001, -0b1111110.

# Целочисленные типы

Константы по умолчанию имеют тип `(signed) int`.

Тип можно переопределить:

- **беззнаковый** — заканчивается символом `u` или `U`:  
`5u`, `012U`, `0x5CAu`;
- **длинное (*long*)** — заканчивается символом `l` или `L`:  
`99l`, `0xABCDL`, `-0315L`;
- **«очень длинное» (*long long*)** — заканчивается символами `ll` или `LL`: `5LL`; `0XBaLL`; `-0105ll`.

Непротиворечивые указания типа можно комбинировать: `5LU`; `2ull`.



# Вещественные типы

Тип	Бит	Диапазон	Точность
<code>float</code>	32	$\pm 1,40129846 \cdot 10^{-45} \dots \pm 3,40282347 \cdot 10^{38}$	7—8 знаков
<code>double</code>	64	$\pm 4,9406564584124654 \cdot 10^{-324} \dots \pm 1,7976931348623157 \cdot 10^{308}$	15—16 знаков
<code>long double</code>	80	$\pm 1,9 \cdot 10^{-4932} \dots \pm 1,1 \cdot 10^{4932}$	19—20 знаков

Тип `long double`:

- в Microsoft Visual C++ соответствует типу `double`;
- может занимать 96 или 128 бит; работают всегда 80 бит.

Вещественные числа — это потери точности,  $-0$ , NaN,  $\infty \dots$

***Не используйте вещественные числа без необходимости.***

# Вещественные типы

Обязательна точка, разделяющая целую и дробную части:

$-2.0$ ,  $-0.5$ ,  $3.1415927$ .

Одну из частей можно не указывать:  $143.$ ,  $.005$ ,  $-.0$ .

Возможна экспоненциальная форма записи:

- 1) целая или вещественная мантисса (не обязательно нормализованная);
- 2) символ  $e$  или  $E$ ;
- 3) целый порядок в десятичной записи (допускаются ведущие нули):

$6.02e23$ ;  $-1.6e-19$ ;  $0.042E+09$ ;  $-.52E+6$ ;  $4e4$ .

# Вещественные типы

Константы по умолчанию имеют тип `double`, но можно указать:

- **обычной точности (*float*)** — заканчивается символом `f` или `F`:  
`7.F`, `-0.6f`, `+1.99E+08F`;
- **двойной точности (*double*)** — заканчивается символом `d` или `D`:  
`0.1D`, `-72.4d`, `+2.4E-03d`;
- **«длинное» вещественное (*long double*)** — заканчивается символами `l`, `L`, `dl` или `DL`: `5.03L`, `2E0dl`, `-4.935e+45L`;
- **шестнадцатеричное** (только для GCC) — мантисса шестнадцатеричная, порядок целый десятичный. Разделитель — символ `r` или `R`, обозначающий степень двойки. Порядок обязателен: `0xAP1`, `-0x3.Fr+11`, `-0x2.ar-1`.

`0x1P1 = 2.0`, `0x0.BP+10 = 704.0`, `0x1P1 = 0x8P-2 = 2.0`.

Непротиворечивые указания можно совмещать:

`0xEP5dl`, `10xP3LL`.

# Логический тип

В стандартах языка Си полная поддержка логического типа отсутствует.

Тип имеет имя `_Bool`.

Определены два значения данного типа — 1 (истина) и 0 (ложь).

При подключении заголовочного файла `stdbool.h` можно использовать имя типа `bool` и литералы `true` (истина) и `false` (ложь).

# «Пустой» тип

Тип `void` используется, когда формально тип требуется указать, но на самом деле тип не нужен:

- при объявлении «процедур»;
- при объявлении нетипизированных указателей и т. п.

Не существует значений типа `void`.



# Переменные

*Переменная* — ячейка памяти, предназначенная для хранения некоторого значения.

Переменная имеет:

- *значение*;
- *тип* — определяет, значения какого типа она может хранить;
- *адрес* в памяти, по которому можно обратиться к значению;
- *имя* (как правило), по которому можно обратиться к значению.



# Переменные

Чтение значения из переменной — *обращение* к переменной (к значению переменной).

Запись значения в переменную — *присваивание* значения переменной.

*Имя (идентификатор)* — последовательность символов, определяющая переменную (и другие сущности):

- может состоять из заглавных и строчных латинских букв, цифр и знаков подчёркивания: *A ... Z, a ... z, 0 ... 9, \_*;
- не должно начинаться с цифры;
- не должно совпадать с ключевыми словами языка Си.

Желательно не начинать имена с символа *\_*.

Регистр символов в именах различается: *main* ≠ *Main*.

# Переменные

Переменные необходимо **объявлять** до их первого использования.

Сначала указывается тип, затем список имён.

Завершается объявление точкой с запятой:

```
char a, b, c; unsigned char d, e, f;
```

```
unsigned long long int P;
```

```
signed long F;
```

```
float Argument, Result;
```

```
long double LongDouble;
```

# Переменные

Присваивание значений переменным:

```
a = -5; b = 0xA; P = 75000L; Result =  
2.5E6L;
```

Присваивание переменной начального значения — *инициализация* переменной.

Инициализация возможна в объявлениях:

```
int U = 2;
```

```
double X = -5.2d, Y = 2.6d, R;
```

Оператор присваивания может сам возвращать присваиваемое значение, поэтому можно писать так:

```
a = b = c = 5;
```

# Программа

Программа на языке Си состоит из функций.

Описание функции:

```
тип имя (список формальных параметров)
{
    тело функции
}
```

Здесь **тип** — тип возвращаемого функцией значения.

Скобки { и } — начало и конец тела функции.

# Программа

Выполнение программы начинается с функции, имеющей имя `main`.

```
main()
```

```
{
```

```
}
```

или

```
void main(void)
```

```
{
```

```
}
```



# Программа

Бывает и так:

```
int main(int argc, char **argv)
{

    return 0;

}
```



# Переменные

Переменные могут быть объявлены в любом месте функции (*локально*) или вне функций (*глобально*):

```
int P;
```

```
void main(void) { char A; }
```

```
void other(void) { char B; }
```

Глобальные переменные видны всем.

Локальные переменные видны там, где они объявлены (после объявления).

*Область видимости переменной* — совокупность всех областей программы, в которых значение переменной доступно для чтения и записи.

# Переменные

Скобками { и } можно выделить **блок**:

```
int main(int argc, char **argv)
{
    int Z = 9, P = 2;
    {
        int Z = 7;
    }
    return 0;
}
```

Локальные переменные блока видны только в нём.

Переменные функции видны и в ней, и в блоке.

Локальные переменные временно перекрывают видимость «более глобальных».

# Переменные

Переменную можно объявить с *классом памяти*:

- **auto** (по умолчанию для локальных);
- **static** — позволяет сохранять значения между вызовами функции; инициализируется по умолчанию нулевым значением;
- **register** — попытаться ускорить работу с переменной; работает только для типов с размером, как у **int**; видимость переменных — как у **auto**;
- **extern** — переменная уже объявлена в другом файле; делает её видимой в данном файле.

# Переменные

Объявление переменных с классом памяти:

```
static unsigned long long int Loops = 100;  
extern int Size;  
auto R;
```

Глобальные переменные — в *сегменте данных* программы.

Локальные переменные с классом памяти `static` — также в *сегменте данных*.

Прочие локальные переменные — в *сегменте стека*.  
*Контролируйте стек, чтобы он не переполнился.*