

# Основы программирования

## Основы языка программирования C/C++

# Полезные сайты для изучения C/C++

[cppstudio.com](http://cppstudio.com) — для начинающих

[cplusplus.com](http://cplusplus.com) — руководства по C++

[ru.stackoverflow.com](http://ru.stackoverflow.com) — вопросы и ответы

[cpp.com.ru](http://cpp.com.ru) — книги по C++ на русском

Книги Герберта Шилдта

# Структура программы на C/C++

- Директивы препроцессора  
(**#define**, **#include**,...)
- Описания глобальных типов данных  
(**typedef**, **struct**, **class**,...)
- Описания глобальных данных  
(переменных, констант)
- Описания заголовков функций
- Главная функция **main** или **\_tmain**
- Описания функций

```
#include <iostream>
using namespace std;
double min(double, double);
int main(int argc, char *argv[])
{
    double x, y;
    for (int i = 0; i < 10; i++)
    {
        cin >> x >> y;
        cout << min(x, y) << endl;
    }
    return 0;
}

double min(double a, double b)
{
    if (a <= b) return a; else return b;
}
```

# Арифметические типы

Типы переменных:

- **int** – целые со знаком
- **float** – короткие вещественные (с плавающей точкой)
- **double** – вещественные с двойной точностью

Модификаторы **unsigned**, **short**, **long**

**Имя переменной (идентификатор)** – последовательность букв и/или цифр, начинается всегда с буквы

# Примеры описаний переменных

```
int i, count;  
unsigned k; (unsigned int k;)  
short mask;  
float leng2;  
double x1, x2, y1, y2;
```

Константы описываются значением:

77    -256    -1    10000 – целые (**int**)  
-100.0    3.1415    1e-3    (= 0.001) –  
вещественные (**double**)

# Внутренний формат

- int** – 4 байта (32 бита),  
диапазон: -2147483648...2147483647,  
числа  $\geq 0$  – прямой двоичный код,  
числа  $< 0$  – дополнительный код
- float** – 4 байта (8 бит – порядок, 23 бита –  
мантисса, 6-7 точных десятичных цифр  
числа)
- double** – 8 байт (11 бит – порядок, 52 бита –  
мантисса, 15 точных десятичных цифр числа)

# Операции и их приоритеты

Приоритет	Операции	Порядок исполнения
1	() [] -> .	→
2	! ~ + - ++ -- & * (<тип>) sizeof	←
3	* / %	→
4	+ -	→
5	<< >>	→
6	< <= >= >	→
7	== !=	→
8	&	→
9	^	→
10		→
11	&&	→
12		→
13	? :	←
14	= *= /= %= += -= &= ^=  = <<= >>=	←
15	,	→



# Приведение типов

```
double x, y; int k, m;  
x = 1.7; y = 8; k = 10; m = 3;
```

Результаты операций:

```
x + 10,    k - 3.0,    k * y - double
```

```
k + 10,    k / 3,    k * 5 - int
```

```
x > 0,    x <= k - true
```

```
k == k/3*m,    k != 10 - false
```

```
m == k/3,    m == 3 - true
```

```
y/3*3 == y - false (неправильно!)
```

```
abs(y/3*3 - y) < 1e-6 - true (правильно!)
```

# Арифметическое выражение (формула)

**Формула** записывается линейно, в строку

**Операции:**  $+$ ,  $-$ ,  $*$  (умножение),  $/$  (деление, при делении целых чисел частное будет целым),  $\%$  (остаток от деления для целых).

**Стандартные функции:** `sqrt` (квадратный корень), `abs` (абсолютное значение), `sin` (синус), `cos` (косинус), `exp` (экспонента) и др.

**Порядок вычислений:** слева-направо, но вначале операции  $*$ ,  $/$ ,  $\%$ , а затем  $+$  и  $-$ .

**Круглые скобки** изменяют **порядок вычислений**

# Список основных операторов

- присваивание
- ввод/вывод
- составной
- условный
- цикл **for**
- цикл **while**
- **return**

Все операторы, кроме составного, завершаются символом **‘;’**

# Оператор присваивания

Формат: переменная = формула

Примеры:

```
double x, y, a, b, c; int k;
```

```
y = sin(a*x) - b/2;
```

```
c = sqrt(x*x + y*y);
```

```
k = k + 2;
```

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

```
a = k * 10 + 2;
```

```
k = (int) (a / 5 - k); - целая часть
```

# Сокращенная запись

`double x, y; int k, m;`

**k++** – значение **k** сначала используется, а затем увеличивается на 1

**++k** – значение **k** сначала увеличивается на 1, а затем используется

**+= -= ...** - расчет нового значения на основе старого

Эквивалентные	выражения
<code>k = m++;</code>	<code>k = m; m = m + 1;</code>
<code>k = ++m;</code>	<code>m = m + 1; k = m;</code>
<code>y *= x - 2;</code>	<code>y = y * (x - 2);</code>
<code>k += 2;</code>	<code>k = k + 2;</code>
<code>m -= 1; <b>или</b> m--;</code>	<code>m = m - 1;</code>

# Ввод/вывод чисел

**cin** – стандартный поток ввода – это некий источник, который:

- передает в программу последовательность байт, соответствующую введенным с клавиатуры символам
- преобразует эту последовательность в значения входных переменных

**cout** – стандартный поток вывода:

- формирует последовательность символов, соответствующую выводимым значениям
- передает эту последовательность для отображения на экране

# Ввод/вывод чисел

Операторы, необходимые для подключения потоков:

```
#include <iostream>  
using namespace std;
```

Для потоков определены операции:

>> – извлечение значения из потока **cin**

<< – вставка значения в поток **cout**

# Примеры ввода значений

```
int k, m; double x, y;
```

Эквивалентные последовательности операторов:

1. `cin >> x; cin >> y; cin >> k; cin >> m;`
2. `cin >> x >> y; cin >> k >> m;`
3. `cin >> x >> y >> k >> m;`

Возможные варианты ввода для всех 3 случаев  
(Enter после каждой строки запускает ввод):

- 3.14 2.71 13 256
- 3.14 2.71  
13 256
- 3.14  
2.71 13 256



# Примеры вывода значений

```
int k, m; k = 25; m = 100;
```

1. `cout << k << m << k + m;`  
`cout << k - m; cout << endl;`

**ВЫХОДНАЯ строка:** 25100125-75

2. `cout << k << " " << m << " ";`  
`cout << k + m << " " << k - m << endl;`

**ВЫХОДНАЯ строка:** 25 100 125 -75

3. `cout << "k=" << k << ", m=" << m << endl;`  
`cout << "k+m=" << k + m;`  
`cout << ", k-m=" << k - m << endl;`

**2 ВЫХОДНЫЕ строки:**

k=25, m=100

k+m=125, k-m=-75

# Составной оператор

**Формат:**

```
{  
    оператор_1;  
    оператор_2;  
    ...  
    оператор_n;  
}
```

# Условный оператор

**2 варианта – полный и сокращенный:**

```
if (условие) оператор_1; else оператор_2  
if (условие) оператор_1
```

**условие** – любое выражение (например, сравнение), значением которого может быть либо **true** (истина), либо **false** (ложь)

**оператор\_1** и **оператор\_2** – любые операторы C++ (в том числе, составные)

**Порядок работы:** вычисляется значение условия; если оно истинно, то выполняется оператор\_1; если условие ложно и оператор включает **else**, то выполняется оператор\_2

# Примеры условных операторов

```
int k, m, i;
```

1. `if (k < 0) k = -k;`

2. `if (k >= m) cout << k; else cout << m;`

3. `if (k < m) { i = k; k = m; m = i; }`

4. `if (k == 1) k *= 2;`

`else if (k == 2) k -= 2;`

`else if (k == 3) k++;`

`else cout << "error";`

# Целочисленные значения в условиях

```
if (k - m) k += (m + 1) * 2;
```

ЭКВИВАЛЕНТНО

```
if (k - m != 0) k += (m + 1) * 2;
```

```
if (!k) k++;
```

ЭКВИВАЛЕНТНО

```
if (k == 0) k++;
```

# Цикл for

## Формат:

**for** (действия\_1 ; условие ; действия\_2)  
оператор

**действия\_1** и **действия\_2** –  
последовательности операторов,  
разделенных символом ``,'` (обычно это 1 или  
несколько операторов присваивания, но  
может быть и пустой оператор `;`)

**условие** – любое выражение, значением  
которого может быть истина, ложь или целое  
число (как в условном операторе)

**оператор** – любой оператор C++ (в том числе,  
составной или пустой) – это тело цикла

# Порядок работы цикла for

**for** (действия\_1 ; условие ; действия\_2)  
оператор

1. Выполняются начальные **действия\_1** (если они заданы)
2. Проверяется **условие**. Если оно истинно, то сначала выполняется **оператор**, а потом **действия\_2**. Если условие ложно, то работа всего оператора цикла завершается
3. Производится возврат к пункту 2.

# Примеры циклов for

```
int k, n;
```

3 эквивалентных цикла (сумма чисел от 1 до 10):

1. `for (n = 0, k = 1; k <= 10; k++) n += k;`
2. `for (n = 0, k = 1; k <= 10; n += k, k++) ;`
3. `n = 0;`  
`for (k = 1; k <= 10;) { n += k; k++; }`

Другие варианты решения той же задачи:

4. `n = 0; k = 10;`  
`for (; k != 0; k--) n += k;`
5. `n = 0; k = 10;`  
`for (; k;) { n += k; k -= 1; }`



# Примеры циклов `for`

```
int k, n;
```

Циклы, в которых выполняются только начальные действия:

```
for (n = 0, k = 1; k < n; k++) n += k;
```

```
for (n = 0, k = 1; n; n += k, k++) ;
```

Бесконечные циклы:

```
for (n = 0, k = 1; k; k++) n += k;
```

```
for (k = 1; 1; k++)
```

```
for (; true;)
```

Цикл, который выполнится 5 раз:

```
for (n = 10, k = 1; k < n; k++, n--);
```

# Цикл while

## Формат:

**while** (условие) оператор

**условие** – любое выражение, значением которого может быть истина, ложь или целое число (как в условном операторе)

**оператор** – любой оператор C++ (в том числе, составной или пустой) – это тело цикла

## Порядок работы:

Проверяется **условие**. Если оно ложно, то работа всего оператора цикла завершается. Если условие истинно, то выполняется **оператор**, затем вновь проверяется **условие** и т.д.

**Оператор** должен включать действия, которые когда-либо приведут к нарушению истинности **условия**.

## Примеры циклов `while`

```
int k, n;
```

Подсчет суммы чисел от 1 до 10:

```
n = 0; k = 1;
```

```
while (k <= 10) { n += k; k++; }
```

```
n = 0; k = 10;
```

```
while (k) { n += k; k -= 1; }
```

Бесконечные циклы:

```
while (1) { ... }
```

```
while (true) { ... }
```