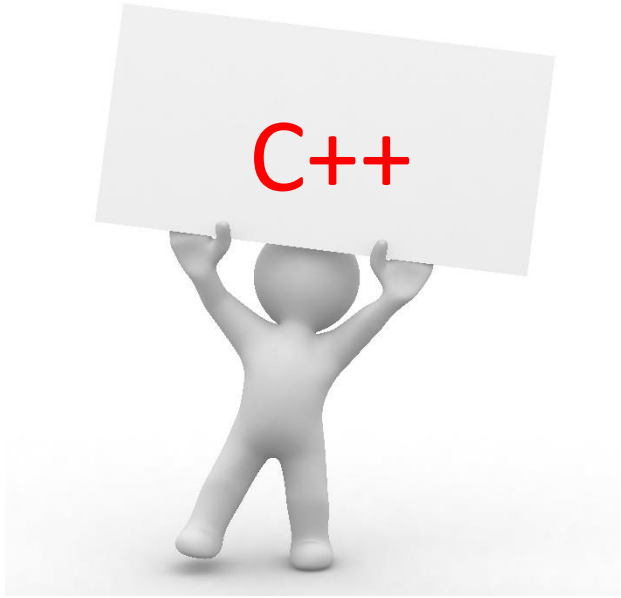
Язык программирования C++

ООП

История языка Си++

Язык Си++ был разработан в начале 1980-х гг. Бьерном Страуструпом из компании AT&T Bell Laboratories. Си++ основан на языке Си. Таким образом, Си++ был задуман как язык Си с расширенными возможностями. Большая часть языка Си вошла в Си++ как подмножество, поэтому многие программы на Си можно скомпилировать с помощью компилятора Си++.



Алфавит языка

Алфавит C++ включает:

1. прописные и строчные латинские буквы и знак подчеркивания;
2. арабские цифры от 0 до 9;
3. специальные знаки, например, {, %, # и т.д.
4. пробельные символы: пробел, символы табуляции, символы перехода на новую строку.

Алфавит языка

Из символов алфавита формируются *лексемы* языка:

- идентификаторы;
- ключевые (зарезервированные) слова;
- знаки операций;
- константы;
- разделители (скобки, точка, запятая, пробельные символы).

Алфавит языка

В тесте программы можно использовать **комментарии**.

Если текст начинается с // или заключен между символами /* и */, то компилятор его игнорирует.

Общая структура программы:

```
#include <название библиотеки>
using namespace std;
int main() {
  int i;
  cout << "Введите целое число\n";
  cin >> i;
  cout << "Вы ввели число" << i << ", спасибо!";
return 0;
}
```

Общая структура программы:

#include <название библиотеки>

В первой строке программы с помощью директивы **#include** происходит подключение библиотек.

Библиотеки содержат описание функций и других готовых элементов, которые можно использовать в своих программах после того, как заголовочный файл подключён.

Библиотека iostream

Заголовочный файл **iostream** содержит набор готовых функций для потокового ввода и вывода.

```
#include <iostream>
```

Общая структура программы:

```
using namespace std;
```

Пространство имен (**namespace**) – это способ объединения логически связанных объявлений под общим именем.

```
namespace first
    {
      int a;
      float b;
    }
    namespace second
    {
      int a;
      float b;
    }
```

Пространство имен

Если отсутствует необходимость в использовании разных пространств имён в рамках одной программы, то можно однажды задать пространство и далее обращаться ко всем именам без его указания.

using namespace std;

Общая структура программы:

```
int main(int argc, char* argv[]){
  …
  }
```

Каждая программа, написанная на C++, имеет в себе функцию **main()** , с которой начинается запуск программы. Функция **main(),** как правило, возвращает результат своего выполнения, о чем сигнализирует тип данных **int**, который написан перед функцией **main()** .

При правильном, успешном завершении функция **main()** возвращает в качестве результата **0** . Значение результата, отличное от нуля сигнализирует о нештатном завершении программы.

Общая структура программы:

int i, a=5; - Описание переменных.

В языке C++ определено шесть стандартных простых типов данных для представления целых, вещественных, символьных и логических величин.

На основе этих типов, а также массивов и указателей (указатель не является самостоятельным типом, он всегда связан с каким-либо другим конкретным типом), программист может вводить описание собственных простых или структурированных типов.

К структурированным типам относятся перечисления, функции, структуры, объединения и классы.

Простые типы данных

Простые типы делятся на целочисленные типы и типы с плавающей точкой.

Для описания *стандартных типов* определены следующие ключевые слова:

1. **int** (целый);
2. **char** (символьный);
3. **wchar_t** (расширенный символьный);
4. **bool** (логический);
5. **float** (вещественный);
6. **double** (вещественный с двойной точностью).

Простые типы данных

Существует четыре *спецификатора типа*, уточняющих внутреннее представление и диапазон значений стандартных типов:

1. ***short*** (короткий);
2. ***long*** (длинный);
3. ***signed*** (со знаком);
4. ***unsigned*** (без знака).

Простые типы данных

Диапазоны значений простых типов данных

Тип	Диапазон значений	Размер (байт)
`bool`	`true` и `false`	**1**
`signed char`	–128 .. 127	**1**
`unsigned char`	0 .. 255	**1**
`signed short int`	–32 768 .. 32 767	**2**
`unsigned short int`	0 .. 65 535	**2**
`signed long int`	–2 147 483 648 .. 2 147 483 647	**4**
`unsigned long int`	0 .. 4 294 967 295	**4**
`float`	$3.4e^{-38}$ .. $3.4e^{+38}$	**4**
`double`	$1.7\ e^{-308}$ .. $1.7\ e^{+308}$	**8**
`long double`	$3.4\ e^{-4932}$ .. $3.4\ e^{+4932}$	**10**

Преобразование типов данных

В C++ различают два вида преобразования типов данных: явное и неявное.

- **Неявное преобразование** происходит автоматически. Это выполняется во время сравнения, присваивания или вычисления выражения различных типов.

Наивысший приоритет получает тот тип, при котором информация теряется менее всего. Не стоит злоупотреблять неявным преобразованием типов, так как могут возникнуть разного рода непредвиденные ситуации.

Преобразование типов данных

Явное приведение осуществляется с помощью указания целевого типа данных (того, к которому нужно привести) в круглых скобках перед выражением:

```
double s = 2.71;
int t = (int) s;
cout << t << endl; // 2
cout << (int) 3.14 << endl; // 3
cout << (int) (2.5 + t) << endl; // 4
```

Приведение к целым числам от вещественных осуществляется путём отбрасывания целой части (не округлением).

Общая структура программы:

```
cout << "Введите целое число\n";
cin >> i;
cout << "Вы ввели число" << i << ", спасибо!";
```

Ввод данных

Ввод данных в C++ осуществляется с помощью команды **cin** (**C**onsole **In**put).

Аргумент этой функции передаётся через оператор **>>**.

Вывод данных

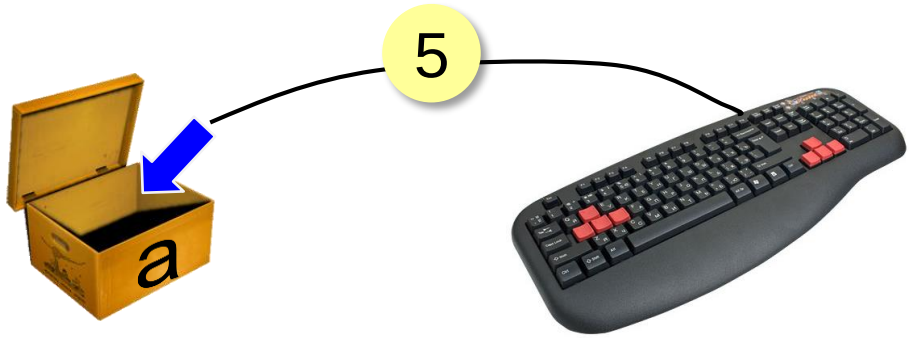
Вывод данных в C++ осуществляется с помощью команды **cout** (**C**onsole **Out**put).

Оператор вывода **<<**.

Ввод значения с клавиатуры

ввести значение **a** из входного потока

```
cin >> a;
```

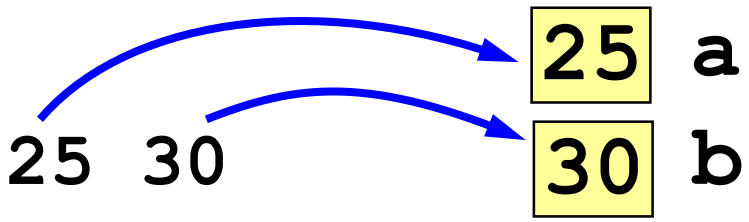


1. Программа ждет, пока пользователь введет значение и нажмет *Enter*.
2. Введенное значение записывается в переменную **a**.

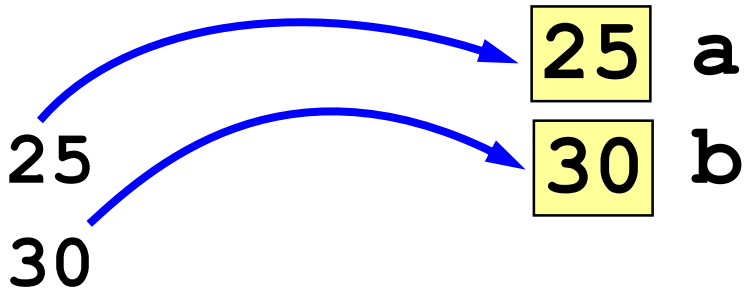
Ввод значений двух переменных

```
cin >> a >> b;
```

через пробел:

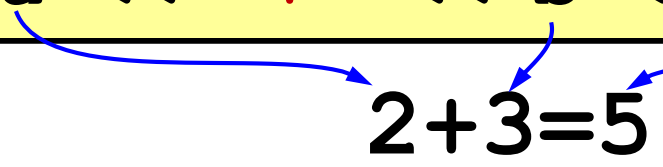


через *Enter*:



Вывод данных

```
cout << a;                    //вывод значения
                              //переменной a

cout << a << endl;            //...и переход
                              //на новую строку

cout << "Привет!";            //вывод текста

cout << "Ответ: " << c;

//вывод текста и значения переменной c

cout << a << "+" << b << "=" << c;
```



Вывод данных (продолжение):

Вычисление выражений:

```
cout << a << "+" << b << "=" << a+b;
```

Форматный вывод:

```
#include <iomanip>
...
a = 123;
cout << setw(5) << a;
```

манипуляторы для управления потоками

␣␣123

5 знаков

set width – установить ширину поля

Библиотека iostream

return 0

Ноль означает отсутствие ошибки.

Для функции main, даже объявляя её как возвращающую целое число, и только для неё, можно ничего не возвращать.

Сложение чисел: полное решение

```
#include <iostream>
int main()
{
  int a, b, c;
  cout << "Введите два целых числа\n";
  cin >> a >> b;
  c = a + b;
  cout << a << "+" << b << "=" << c;
}
```

подсказка

Протокол:

```
Введите два целых числа
25 30
25+30=55
```

компьютер

пользователь

Арифметическое выражения

```
          3   1 2   4       5    6
a = (c + b*5*3 - 1) / 2 * d;
```

Приоритет (*старшинство*):

1) скобки
2) умножение и деление, остаток от деления
3) сложение и вычитание

$$a = \frac{c + b \cdot 5 \cdot 3 - 1}{2} \cdot d$$

Деление

Результат деления целого на целое – **целое** число (остаток отбрасывается):

```
int a = 3, b = 4;
float x;
x = 3 / 4;
x = 3. / 4;
x = 3 / 4.;
x = a / 4;
x = a / 4.;
x = a / b;
x = float(a) / 4;
x = a / float(b);
```

Что запишется в **x**?

Остаток от деления

`%` – остаток от деления

```
int a, b, d;
d = 85;
b = d / 10;
a = d % 10;
d = a % b;
d = b % a;
```

Для отрицательных чисел:

```
int a = -7;
b = a / 2;
d = a % 2;
```

В математике не так!

остаток ≥ 0

$-7 = (-4)*2 + 1$

Сокращенная запись операций

```
int a, b;
...
a ++;      // a = a + 1;
a --;      // a = a - 1;
a += b;    // a = a + b;
a -= b;    // a = a - b;
a *= b;    // a = a * b;
a /= b;    // a = a / b;
a %= b;    // a = a % b;
```

Вещественные числа

Целая и дробная части числа разделяются **точкой**!

Форматы вывода:

```
float x = 123.456;
cout.width(10);
cout.precision(5);
cout << x << endl;
```

5 значащих цифр

␣␣␣␣123.46

всего 10 знаков

```
cout.width(10);
cout.precision(2);
cout << x << endl;
```

␣␣␣1.2e+002

$1{,}2 \cdot 10^2$

Стандартные функции

```
#include <cmath>
```

подключить математическую библиотеку

abs(x) — модуль целого числа
fabs(x) — модуль вещественного числа
sqrt(x) — квадратный корень
sin(x) — синус угла, заданного **в радианах**
cos(x) — косинус угла, заданного **в радианах**
exp(x) — экспонента e^x
ln(x) — натуральный логарифм
pow(x,y) — x^y: возведение числа x в степень y
floor(x) — округление «вниз»
ceil(x) — округление «вверх»

```
float x;
x=floor(1.6);// 1
x=ceil(1.6); // 2
```

```
x=floor(-1.6);//-2
x=ceil(-1.6); //-1
```

Задачи

«A»: Ввести с клавиатуры три целых числа, найти их сумму, произведение и среднее арифметическое.

Пример:

```
Введите три целых числа:
5 7 8
5+7+8=20
5*7*8=280
(5+7+8)/3=6.667
```

«B»: Ввести с клавиатуры координаты двух точек (A и B) на плоскости (вещественные числа). Вычислить длину отрезка AB.

Пример:

```
Введите координаты точки A:
5.5 3.5
Введите координаты точки B:
1.5 2
Длина отрезка AB = 4.272
```