# Первая программа. Основы синтаксиса C++

Традиционно изучение нового языка программирования начинается с программы, выводящей в консоль приветствие `hello␣world`. На C++ эта программа может выглядеть так.

```
#include <iostream>
using namespace std;
int main() {
  cout << "hello world" << endl;
  return 0;
}
```

Рассмотрим синтаксис этой программы. Слова «`using`», «`namespace`», «`int`», «`return`» называются ключевыми и служат для описания объектов и инструкций. Ключевые слова распознаются транслятором и имеют одинаковое значение во всех программах, поэтому написание и значение ключевых слов необходимо запоминать. Также программа содержит строковый литерал «`"hello␣world"`» и целочисленный «`0`». Литералами называют константы в тексте программы, запись которых представляет их значение. Как видим, признаком строкового литерала являются двойные кавычки. Числовые литералы записываются без кавычек и состоят из цифр. Кроме того в программе присутствуют идентификаторы «`std`», «`main`», «`cout`», «`endl`», которые задают различные объекты программы:

- «`cout`» — это стандартная переменная, связанная с потоком вывода;
- «`std`» — пространство имен для библиотечных функций C++;
- «`main`» — функция, с которой начинается исполнение программы;
- «`endl`» — функция для вывода в консоль конца строки.

Круглые скобки являются признаком описания (описателем) функции, фигурные скобки разбивают программу на части, называемые блоками. Программа состоит из инструкций и описаний, которые завершаются символом «`;`». Символ «`#`» является признаком директивы препроцессора, которая заканчивается в конце строки. Пробелы в C++ используются только для улучшения читаемости программы и не несут смысловой нагрузки.

Теперь разберем программу построчно. В первой строке записана директива препроцессора «`#include`». Она служит для подключения библиотек, позволяющих использовать те или иные функции языка. Имя библиотеки указывается в угловых скобках. В первых программах мы будем использовать три библиотеки:

- «`iostream`» — для потокового ввода и вывода;
- «`string`» — для работы со строковой информацией;
- «`cmath`» — для математических функций.

Вторая строка служит для раскрытия пространства имен «`std`». Идентификаторы в C++ можно объединять в группы, называемые пространствами имен. Этот механизм позволяет избежать путаницы в именах при написании сложных программ. Но в простых программах им часто пренебрегают для сокращения длины кода. Пространство имен «`std`» содержит идентификаторы всех функций стандартной библиотеки C++, и эта директива позволяет далее их использовать без указания самого имени «`std`».

В третьей строке записан заголовок функции «`main`». Вся программа в C++ разбивается на части, называемые функциями, которые работают поочередно, вызывая друг друга. В теле функции, ограниченном фигурными скобками, записываются инструкции программы. Идентификатор «`main`» является стандартным, исполнение всех программ на C++ начинается с функции с этим именем.

Функция «main» содержит две инструкции. Первая — это выражение, осуществляющее вывод в стандартный поток строкового литерала «"hello␣world"» и конца строки. Оператор «<<», примененный к переменной «cout», задает операцию вывода своего правого аргумента. Как видно из примера, он может быть применен многократно.

Вторая инструкция «return␣0» останавливает выполнение функции и указывает, что ее результатом является число 0. Это число воспринимается операционной системой и является признаком корректного завершения работы программы.

## Вторая программа. Переменные

Другой традиционной программой является «a+b» — программа для нахождения суммы двух чисел.

```
#include <iostream>
using namespace std;
int main() {
  int a,b,c;
  cin >> a >> b;
  c=a+b;
  cout << c << endl;
  return 0;
}
```

Эта программа содержит новые элементы. Строка «int␣a,b,c» задает локальные переменные функции. Переменной в программировании называют участок памяти, обладающий уникальным именем, предназначенный для хранения информации определенного типа. В данном случае создаются три переменных с соответствующими именами, которые не могут повторяться в пределах блока, в котором они были объявлены.

Типом в программировании называется множество значений с определенными на нем операциями. Формат хранения информации и ее размер в памяти также определяется по типу данных. Ключевое слово «int» говорит, что переменные будут принадлежать целому знаковому типу.

Глобальные переменные «cin» и «cout» являются стандартными и описаны в библиотеке iostream. Они связаны соответственно с потоками ввода и вывода. Знаки «>>», «<<», «+», «=» называются операторами и задают операции, смысл которых зависит от типов переменных, к которым они были применены. Оператор >>, примененный к потоку ввода, задает операцию чтения данных из входного потока с их записью в соответствующие переменные. Операторы «+», «-», «*», «/», примененные к числам, задают соответствующие арифметические операции. Оператор «=» в C++ обозначает операцию присваивания или инициализацию переменной. Присваивание в программировании — это операция записи значения в участок памяти, отведенный под некоторую переменную. При этом то значение, которое было записано в переменной ранее, безвозвратно стирается.

Таким образом, тело функции содержит одну декларацию и четыре инструкции. В декларации объявляются три переменные целого типа. Первые три инструкции — это выражения, задающие ввод данных из потока в переменные a и b, запись в переменную c суммы a+b и вывода в поток значения переменной c. Последняя инструкция завершает функцию и программу.

## Основы работы

### Целые числа и остаток от деления

При решении задач с целыми типами данных, помимо основных арифметических операций, используется операция взятия остатка от деления, которая записывается при по-

мощи оператора «%». Например, значением `x%3` для неотрицательных значений `x` могут быть 0, 1 или 2. Когда два числа делятся нацело, то остаток от деления считается равным нулю. Операция деления, применяемая к целым числам, выполняется как деление с отбрасыванием остатка, то есть ее результатом является целое число.

Если аргументы операции деления целые положительные числа, то ее результат можно считать делением с округлением вниз $\lfloor \frac{a}{b} \rfloor$. В некоторых случаях нам может потребоваться деление с округлением вверх $\lceil \frac{a}{b} \rceil$. Для выполнения такой операции можно воспользоваться равенством $\lceil \frac{a}{b} \rceil = \lfloor \frac{a+b-1}{b} \rfloor$ или, в записи на C++, `(a+b-1)/b`.

### Числа с плавающей точкой

Для работы с вещественными числами используется специальный тип данных, называемый числами с плавающей точкой («`double`», или «`long␣double`»). Такие числа хранятся в памяти в виде $m2^p$, где $m$ и $p$ можно считать целыми числами. Числа с плавающей точкой уже не обязательно являются целыми, в частности, при отрицательных степенях $p$.

В программах эти числа представляются в десятичной записи, поскольку мы привыкли работать именно с такими числами. Однако при преобразовании двоичных дробей в десятичные как правило возникает погрешность, поэтому по возможности следует выбирать способ решения задач, не использующий чисел с плавающей точкой.

Операция деления для чисел с плавающей точкой работает также, как с десятичными дробями. Если одно из чисел является целым, а другое нет, то целое число будет преобразовано к типу с плавающей точкой.

Литералы с плавающей точкой могут быть записаны в экспоненциальном формате с использованием символа «`E`». Например, число $2.7 \cdot 10^3$ может быть записано в виде `2.7E3`. Признаком того, что литерал является числом с плавающей точкой, является наличие в его записи буквы «`E`» или символа «`.`». Таким образом, следующий фрагмент программы выведет сначала число 3, потом 3.75.

```
int x=15;
cout << x/4 << endl;
cout << x/4.0 << endl;
```

При потоковом выводе можно указать способ печати вещественного числа при помощи функций-манипуляторов «`fixed`» (формат с фиксированной точкой) и «`scintific`» (формат с плавающей точкой).

```
double x=0.0025;
cout << fixed << x << endl;
cout << scintific << x << endl;
```

На печать будут выведены числа `0.0025` и `2.5e-3`. Для указания количества дробных знаков, которые требуется выводить в формате с фиксированной точкой можно использовать функцию «`cout.precision(n)`». В следующем примере будет выведено `10.66666667`.

```
double x=32.0/3;
cout.precision(8)
cout << fixed << x << endl;
```

Большая часть функций библиотеки `cmath` работает с числами с плавающей точкой. Чаще всего используются следующие:

- «`sqrt(x)`» — квадратный корень;
- «`pow(a,p)`» — вычисление $a^p$;
- «`log(x)`» — натуральный логарифм $x$.;
- «`sin(x)`», «`cos(x)`», «`tan(x)`» — тригонометрические функции;

- «`asin(x)`», «`acos(x)`», «`atan(x)`» — обратные тригонометрические функции;
- «`atan2(y,x)`», — вычисление угла луча из точки $(0,0)$ проходящего через точку $(x,y)$.

Приведем пример работы. Константа П определяется как угол к точке $(-1,0)$, далее для точки с координатами $(x,y)$ находим расстояние и угол от начала координат, уменьшаем угол на 30 градусов и вновь находим декартовы координаты точки. Значения выводятся в формате с фиксированной точкой с 6 знаками.

```
double pi=atan2(0.0,-1.0);
double x=2.0,y=1.0;
double r=sqrt(x*x+y*y);
double alph=atan2(y,x);
alph-=pi/6;
x=r*cos(alph);
y=r*sin(alph);
cout.precision(6);
cout<<fixed<<"("<<x<<","<<y<<")"<<endl;
```

### Символы и строки

Для хранения информации символьного типа предназначен тип «`char`». В памяти хранится код символа, поэтому с переменными символьного типа можно работать также как и с целыми числами. Символьные литералы, в отличие от строк, записываются в одиночных кавычках. Помимо стандартных символов, которые есть на клавиатуре, существуют также специальные управляющие символы. Их запись начинается с косой черты. Важнейшим из них является символ переноса строки, записываемый «`'\n'`». В следующем примере значения переменных `a` и `b` выводятся через пробел на одной строке, а их сумма на другой.

```
int a=10,b=20;
cout << a <<' '<< b <<'\n' << a+b << endl;
```

Между выводом `'\n'` и `endl` есть некоторые различия. Все команды вывода не приводят к непосредственной записи информации на внешнее устройство, так как изначально она записывается в буфер. Вывод манипулятора `endl` приводит также к сбросу буфера, то есть операция записи происходит только в этот момент. Поэтому вывод `endl` работает медленнее чем `'\n'`, однако он гарантирует сохранение информации при сбоях в программе.

Строка — это последовательность символов. Для работы со строками можно использовать тип «`string`», хранящийся в одноименной библиотеке. Символы в строке пронумерованы, начиная с нуля. Для доступа к одному из символов строки можно использовать операцию индексации «`[]`». Длину строки можно узнать при помощи метода «`s.size()`». Поскольку символы пронумерованы с нуля, номером последнего символа будет «`s.size()-1`». В C++ строки можно изменять. Например, в следующей программе вводится строка и в ней переставляется первый и последний символ.

```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string str;
  cin >> str;
  char t=str[0];
  str[0]=str[str.size()-1];
  str[str.size()-1]=t;
  cout << str << endl;
```

```
  return 0;
}
```

Для строк также определены некоторые операции. Соединение двух строк можно записать при помощи оператора «`+`». При этом следует иметь в виду, что выражение «`s=s+q`» может работать существенно медленнее чем «`s+=q`>, поэтому следует использовать второй вариант. Кроме того, определены операции сравнения, которые реализованы посимвольно в порядке алфавита (как в словаре). Такие сравнения называют лексикографическими.