# Блиц-курс языка C++

## I. Простейшая программа

**Программа, которая не делает ничего**

```
int main() {
; ; ;
return 0; }
```

; - пустой оператор, каждый оператор в программе заканчивается символом ;

**Программа, которая выводит что-нибудь на консоль**

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

int main() {
   cout << "Hello!" << endl;
   return 0;
}
```

**Программа, которая выводит что-нибудь в файл**

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

int main() {
   freopen ("output.txt", "w", stdout);
   cout << "Hello!" << endl;
   return 0;
}
```

## II. Типы данных, константы, описание переменных

**Базовые типы данных:**

**int** – целые числа от $-2^{31}$ до $2^{31}-1$
**long long** - целые числа от $-2^{63}$ до $2^{63}-1$
**double** – вещественные (действительные) числа
**char** – целые числа от -128 до 127, используются для хранения символов

**Примеры целых констант типа int:**

123
-7777

**Примеры целых констант типа long long (справа дописываются буквы ll):**

123ll
-7777ll

**Примеры вещественных констант:**

2.7
33.
.008
1e-2
2.33E10

**Примеры констант типа char:**

'a'
'\'' – символ апостроф
'\n' – символ конца строки

**Переменные описываются так:**

```
int a, b=1, c, B=-99;
long long x=12345678912345678911, y=0, z;
double e=2.73;
char c1, c2, c3='*';
```

Можно задавать или не задавать начальное значение переменной. Описание переменных можно размещать в любом месте программы, в том чисел и вне функций.

## III. Операции, выражения

**Арифметические операции**
Унарный минус: - (применяется к одному числу или переменной для изменения знака)
Сложение: +
Вычитание: -
Умножение: *
Деление: /
Остаток от деления: %

Если целое число делится на целое, то результат деления – целое число, т.е. дробная часть отбрасывается. Если делимое или делитель вещественные, то результат вещественный. Например, 2/3 это 1, а 2./3 это 1.6666…

**Операции сравнения**
Больше: >
Меньше: <
Больше или равно: >=
Меньше или равно: <=
Равно: ==
Не равно: !=

Результат, который возвращают операции сравнения – целое число 1 или 0. Если сравнение выполнилось, то результат 1, иначе 0.

**Логические операции**
Логическое отрицание: !
Логическое «и»: &&
Логическое «или»: ||

**Правила выполнения логических операций**

| A | B | !A | A && B | A \|\| B |
|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |

**Операции присваивания**
Обычное присваивание: =
Сложные присваивания (только часть, есть и другие): += -= *= /= %=

В левой части операции присваивания должна стоять переменная, в правой части – любое выражение.

**Примеры присваиваний**

```
int a=1;
```

```
a+=2; // переменная получит значение 3
a*=5; // переменная получит значение 15
a-=6; // переменная получит значение 9
a/=2; // переменная получит значение 4
a%=3; // переменная получит значение 1
```

**Операция явного преобразования типа**

Можно явно изменить тип выражения:

```
int a,b;
long long c = (long long) a * b;
```

Если в одном выражении используются переменные (константы) разных типов, то выбирается наибольший тип. Если все типы в выражении одинаковы, то результат выражения имеет этот тип.

**Операции ++ и --**

Инкрементная операция ++ и декрементная -- применяются только для переменных. Операция ++ увеличивает значение переменной на 1, операция -- уменьшает на 1. Если операция записана слева от переменной, то сначала изменяется переменная, а потом подставляется в выражение. Если операция справа, то сначала старое значение переменной подставляется в выражение, а затем изменяется переменная. Например здесь

```
int a=3, b=7, c=a++*b--;
```

значения переменных станут равными 4, 6 и 21

а здесь

```
int a=3, b=7, c=++a*--b;
```

значения переменных станут равными 4, 6 и 24

## IV. Ввод и вывод

**Программа, которая вводит из файла целое число, не превосходящее по модулю 1000000, и выводит в файл его квадрат**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int x;
   cin >> x;
   cout << (long long) x * x << endl;
   return 0;
}
```

**Программа, которая вводит из файла три целых числа, не превосходящих по модулю 1000000000, и выводит их в файл в обратном порядке**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int a,b,c;
   cin >> a >> b >> c;
   cout << a << " " << b << " " << c << endl;
   return 0;
}
```

**Программа, которая вводит из файла два натуральных числа, не превосходящих 1000000000, и выводит в файл их частное с пятью дробными цифрами**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int a,b;
   cin >> a >> b;
   cout << fixed << setprecision(5) << (double)a/b << endl;
   return 0;
}
```

## V. Условный оператор, операция выбора

**Условный оператор:**

```
if (<логическое выражение>) <оператор или блок>
```

или

```
if (<логическое выражение>) <оператор или блок> else <оператор или блок>
```

*<логическое выражение>* - любое выражение, если его значение не ноль, то оно считается истинным, и операторы после if выполняются. Если значение выражения равно нулю, то ничего не происходит (в первом случае) или выполняются операторы после else.

Если нужно выполнить один оператор, то его просто записывают, если больше одного, то пишут последовательность операторов, заключая их в круглые скобки.

**Программа, которая вводит из файла целое число (здесь и далее будет достаточно тиап int), и выводит в файл слово "yes", если число чётное**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int x;
   cin >> x;
   if(x%2==0) cout << "yes" << endl;
   return 0;
}
```

**Программа, которая вводит из файла целое число и выводит в файл слово "yes" и частное от деления большего числа на меньшее, если одно число кратно другому, или слово "no" в противном случае**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int a, b;
   cin >> a >> b;
   if(a%b==0 || b%a==0) {
        cout << "yes  ";
        if (a>=b) cout << a/b;
```

```
        else cout << b/a;
      }
   else cout << "no";
   cout << endl;
   return 0;
}
```

**Операция выбора:**

```
<логическое выражение>?<выражение 1>:<выражение 2>
```

Если логическое выражение истинно, то есть не равно нулю, то операция выбора возвращает первое выражение, в противном случае второе. Например выбрать большее их двух чисел можно так

```
cin >> a >> b;
cout << a>b?a:b << endl;
```

## VI. Операторы цикла

**Цикл while:**

```
while (<логическое выражение>) <оператор или блок>
```

Вычисляется логическое выражение. Если оно не ноль, то выполняется тело цикла (оператор или блок после выражения в скобках), иначе цикл заканчивается.

**Программа, которая вводит из файла целое число N и выводит в файл слово сумму всех чисел от 1 до N**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N;
   cin >> N;
   int sum=0, i=1;
   while (i<=N) {
        sum+=i;
        i++;
   }
   cout << sum;
   return 0;
}
```

Вычисление суммы можно записать и короче

```
while (i<=N) sum+=i++;
```

**Цикл do while:**

```
do  <оператор или блок> while (<логическое выражение>)
```

Сначала выполняется тело цикла, а потом вычисляется логическое выражение. Если оно равно нулю, то цикл заканчивается. Цикл while называется циклом с предусловием, а цикл do while – с постусловием.

**Программа, которая вводит из файла целое число N и выводит в файл слово сумму всех чисел от 1 до N**

```
#include <iostream>
```

```
using namespace std;

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N;
   cin >> N;
   int sum=0, i=1;
   do sum+=i++; while (i<=N);
   cout << sum;
   return 0;
}
```

**Цикл for:**

```
for (<выражение 1>; <логическое выражение>; <выражение 2>) <оператор или блок>
```

Перед началом цикла (только один раз) вычисляется выражение 1, язык разрешает непосредственно в этом выражении описывать переменные. Цикл завершается, если оно оказывается равным нулю. После каждого шага цикла вычисляется выражение 2.

**Вычисление суммы всех чисел от 1 до N циклом for**

```
int i=1, sum=0;
for (; i<=N; i++) sum+=i;
```

или

```
int i, sum;
for (i=1, sum=0; i<=N; i++) sum+=i;
```

Здесь в первом выражении использована особая операция «запятая», она может соединять любые выражения в одно, таким способом в цикл for можно уместить любое количество необходимых действий.

Ещё один вариант

```
int i, sum;
for (i=1, sum=0; i<=N; sum+=i, i++);
```

или

```
int i, sum;
for (i=1, sum=0; i<=N; sum+=i++);
```

или

```
int sum=0;
for (int i=1; i<=N; sum+=i++);
```

## VII. Циклы ввода

**Программа, которая вводит из файла натуральное число N, затем N целых чисел, не превосходящих по модулю 1000000000**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N, x;
```

```
   cin >> N;
   for (int i=0; i<N; i++) cin >> x;
   cout << "введено " << N << " чисел" << endl;
   return 0;
}
```

**Программа, которая вводит из файла целые числа, не превосходящие по модулю 1000000000, количество чисел в файле заранее не известно**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int x, k=0;
   while (cin >> x) k++;
   cout << "введено " << k << " чисел" << endl;
   return 0;
}
```

## VIII. Одномерные массивы

**Описание одномерного массива**

```
<тип элемента> <имя массива> [<длина>];
```

Например

```
int a[10000];
long long b[n*m];
char c[LMAX];
```

Массивы можно описывать внутри функции или вне функций. Длину внешних массивов можно задавать только константами. Все элементы внешних массивов зануляются в начале программы, для внутренних массивов этого не происходит.

**Обращение к элементам массива**

Индексы элементов массива могут быть от 0 до значения *<длина>*-1. Чтобы обратиться к элементу, нужно записать имя массива и значение индекса в квадратных скобках. Например

```
a[9999]
a[i+j]
b[0]
c[++k]
```

В сях не принято контролировать значение индекса, разрешается задавать любой индекс, в том чисел даже отрицательный, но такие действия приводят чаще всего к аварийному завершению программы. Программист должен сам следить, чтобы индекс не выходил за пределы допустимого диапазона.

С элементами массивов можно делать то же самое, что с обычными (скалярными или простыми) переменными.

**Программа, которая вводит из файла натуральное число N, затем N целых чисел, не превосходящих по модулю 1000000000, помещая их в массив**

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

```
int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N;
   cin >> N;
   int X[N];
   for (int i=0; i<N; i++) cin >> X[i];
   cout << "введено " << N << " чисел" << endl;
   for (int i=0; i<N; i++) cout << X[i] << endl;
   return 0;
}
```

**Программа, которая вводит из файла натуральное число N, не превышающее 100, затем N целых чисел, не превосходящих по модулю 1000000000, помещая их в массив, и выводит числа в файл в порядке не убывания**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N;
   cin >> N;
   int X[N];
   for (int i=0; i<N; i++) cin >> X[i];
   for (int i=0; i<N-1; i++)
       for (int j=i+1; j<N; j++)
           if (X[i]>X[j]) swap(X[i],X[j]);
   for (int i=0; i<N; i++) cout << X[i] << " ";
   return 0;
}
```

В этой программе использован простейший алгоритм сортировки – сортировка обменами.

## IX. Двумерные массивы

**Описание двумерного массива**

```
<тип элемента> <имя массива> [<длина первого измерения>] [<длина второго измерения>];
```

Например

```
int A[100][100], B[n][m+2];
long long D[2][100000];
```

Правила задания длин измерений такие же, как у одномерных массивов. Если массив внешний, то длины задаются константами, у внутреннего массива можно задавать длины измерений константами и переменными.

Двумерный массив используется в программе для хранения прямоугольной таблицы – матрицы.

**Обращение к элементам двумерных массивов**

```
A[0][i+j]
B[15][25]
D[1][99999]
```

**Программа, которая вводит из файла натуральные числа N и M, не превышающие 100, затем матрицу размером N на M из целых чисел, не превосходящих по модулю 1000000000, и выводит матрицу в табличном виде**

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

int main() {
   freopen ("input.txt", "r", stdin);
   freopen ("output.txt", "w", stdout);
   int N, M;
   cin >> N >> M;
   int X[N][M];
   for (int i=0; i<N; i++)
       for (int j=0; j<M; j++) cin >> X[i][j];
   for (int i=0; i<N-1; i++){
       for (int j=0; j<M; j++) cout << X[i][j] << " ";
       cout << endl;
      }
   return 0;
}
```

**Многомерные массивы**

Массивы могут иметь любое количество измерений (любую размерность). Длина каждого измерения записывается при описании массива в отдельных квадратных скобках. Чтобы обратиться к элементу многомерного массива, нужно задать столько индексов, какова размерность массива. Например

```
int Z[2][3][10][100], i=1, j=2, k=3;
Z[i][j][(i+1)*j][99]=-1;
```

## X. Функции

Программа на сях всегда состоит из некоторого количества функций. Помимо функции main, которая обязательно должна быть, могут использоваться и другие функции, как написанные программистом, так и стандартные функции языка. Функции описываются в виде:

```
<тип возвращаемого значения> <имя функции> (<тип параметра> <имя параметра>, …)
```

Если у функции нет параметров (как у функции main), то в скобках ничего не пишется. Если функция ничего не должна возвращать, то в качестве типа возвращаемого значения пишут void.

**Стандартные математические функции**

Стандартных функций в языке много, они сгруппированы по своему назначению и описаны в соответствующих заголовочных файлах. Например, стандартная функция freopen описана в заголовочном файле iostream (и в других заголовочных файлах), который необходимо подключить к программе, использующей эту функцию.
Некоторые из стандартных математических функций:

| | |
|---|---|
| **int abs(int x)** | **\|x\|** |
| **long long llabs(long long x)** | **\|x\|** |
| **double fabs(double x)** | **\|x\|** |
| **double ceil(double x)** | **наименьшее целое число ≥ x** |
| **double floor(double x)** | **наибольшее целое число ≤ x** |
| **double pow(double x, double y)** | $x^y$ |
| **double exp(double x)** | $e^x$ |
| **double log(double x)** | **ln(x)** |
| **double sin(double x)** | **sin(x)** |
| **double cos(double x)** | **cos(x)** |
| **double atan(double x)** | **arctg(x)** |

Обратите внимание, что функция, возвращающая абсолютную величину числа, для аргументов разных типов называется по-разному.

**Пользовательские функции**

Обычно в программе выделяют в отдельные функции фрагменты алгоритма, которые повторяются в программе. Функции также помогают упростить логическую структуру программы.

**Функция, получающая целое число (предполагается, что число неотрицательное), и возвращающая первую цифру в десятичной записи числа**

```
int first(long long x){
    while (x>9) x/=10;
    return x;
}
```

**Программа, читающая из файла целые числа, и находящая среди них наибольшее число, начинающееся цифрой 7**

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

int first(long long x){
    while (x>9) x/=10;
    return x;
}

int main() {
    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);
    long long x, M=0;
    while (cin >> x)
          if (x>M && first(x)==7) M=x;
    if (!M) cout << "нет чисел, начинающихся цифрой 7»;
    else cout << "наибольшее число, начинающееся цифрой 7 " << M;
    return 0;
}
```

Функция first не «испортит» переменную x, так как по умолчанию все параметры передаются в функции по значению – создаются копии параметров, с которыми и работает функция, тем самым аргумент, переданный функции, не может измениться. Чтобы аргумент изменился в функции, его надо передать по адресу, для этого перед именем соответствующего параметра пишут символ &

**Функция, меняющая местами значения двух переменных**

```
void swap(long long &a, long long &b){
     long long tmp=a;
     a=b;
     b=tmp;
}
```

Функции могут получать в качестве параметров и обрабатывать массивы. Для параметра - одномерного массива длину можно не указывать.

**Функция, находящая наибольший элемент массива чисел типа int**

```
int max(int n, int x[]){
    int m=x[0];
    for (int i=1; i<n; i++)
        if (m<x[i]) m=x[i];
    return m;
```

}

Элементы массива внутри функции могут изменяться, и эти изменения передаются наружу.

**Функция, сортирующая массив чисел long long по не убыванию, используя сортировку обменами**

```
void sort(int n, long long x[]){
     for (int i=0; i<n-1; i++)
         for (int j=i+1; j<n; j++)
              if (x[i]>x[j]) swap(x[i],x[j]);
}
```

## XI. Стандартная функция sort

В языке C++ есть стандартная функция sort, выполняющая сортировку данных. Она описана в заголовочном файле algorithm (также и в других заголовочных файлах, но чтобы программа правильно обрабатывалась любым компилятором, лучше добавить в неё строку `#include <algorithm>`).

Функция sort сортирует массивы (или части массивов) любого типа. Пусть, например, необходимо отсортировать по возрастанию (не убыванию) все элементы массива

```
int A[n];
```

оператор

```
sort(A,A+n);
```

решит эту задачу. Если нужно упорядочить по возрастанию только элементы с A[k] по A[m] включтительно, а остальные элементы массива оставить на своих местах, следует записать оператор

```
sort(A+k,A+m+1);
```

Конечно, должны выполняться условия $0 \le k \le m < n$. Если нужна сортировка не по возрастанию, а в каком-то ином порядке, функции sort следует передать ещё один аргумент – функцию-компаратор. Функция-компаратор получает два параметра того же типа, какой имеют элементы сортируемого массива, и возвращает 1, если первый параметр должен стоять в упорядоченном массиве левее, чем второй параметр. То есть эта функция моделирует операцию «меньше». Пусть, например, массив A, описанный выше, нужно упорядочить по убыванию (не возрастанию), это можно сделать так.

`int comp(int a, int b){ return a>b; }` // компаратор в программе находится выше, чем функция, в которой вызывается sort

…

```
int n;
cin >> n;
int A[n];
for (int i=0; i<n; i++) cin >> A[i];
sort(A, A+n, comp);
```

Теперь упорядочим массив по не убыванию первой десятичной цифры числа. Сначала напишем функцию, возвращающую первую цифру числа, учтём, что число может быть отрицательным.

```
int first(int x){
      if (x<0) x=-x;
      while (x>9) x/=10;
      return x;
      }
```

Затем напишем компаратор

```
int compare(int a, int b){
      return first(a)<first(b);
}
```

И затем вызовем sort в функции main или другой функции, где это необходимо.

```
int n;
cin >> n;
int A[n];
for (int i=0; i<n; i++) cin >> A[i];
sort(A, A+n, compare);
```

# XII. Символьные строки

Часто в программе требуется обработать символьную (текстовую) информацию. Для хранения символьных строк (последовательностей символов) в C++ есть специальный контейнер **string**. Контейнером в сях называется совокупность данных (например, массива) и функций, обрабатывающих эти данные. Детальное рассмотрение устройства контейнеров оставим на потом, глубокое понимание этого вопроса не обязательно для использования контейнеров. Для использования контейнеров string добавьте в программу строчку

```
#include <string>
```

Переменные string описываются так

```
string a, b, c;
```

Можно описать массив с элементами string

```
string A[1000];
```

**Ввод и вывод строк**

Ввести символьную строку можно обычным потоковым вводом

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

В этом случае пробелы считаются разделителям и не вводятся. Так если во входном файле было записано

```
ABC    DEFG    HIJK
```

то значения введённых переменных будут a: “ABC”, b: “DEFG”, c: “HIJK”. Если нужно ввести целиком строку входного файла, вместе с пробелами, то используют функцию getline

```
getline(cin, a);
```

Выводят символьные строки обычным образом

```
cout << a << endl << b << endl << c << endl;
```

**Присваивание строк**

Для контейнера string определена операция = (присвоить). В правой части присваивания можно писать другую строку string или строковую константу

```
a=”123+456”; b=”*****”; c=a;
```

**Конкатенация строк**

Для строк определена операция +, она в данном случае называется «конкатенация» или «сцепление» строк, результат операции есть строка, полученная дописыванием второй строки справа к первой.

```
string a, b;
a=”abc”;
b=”def”;
cout << a+b;
```

**Текущая длина строки**

Длину строки возвращает метод length или метод size. Методами называются функции, которые лежат в контейнерах, они вызываются специальным образом

```
string s;
s=”abcdefgh”;
cout << s.length() << “ “ << s.size() << endl;
```

Программа выведет 8 8.

**Доступ к отдельным символам строки и подстрокам**

Обратиться к отдельному символу строки можно по индексу, так же, как к элементу массива

```
string s;
s=”abcdef”;
for (int i=0; i<s.length(); i++) cout << s[i];
count << endl;
for (int i=s.length()-1; i>=0; i--) cout << s[i];
count << endl;
```

Программа выведет

```
abcdef
fedcba
```

Метод substr возвращает подстроку данной строки, он получает номер позиции, с которой начинается подстрока, и длину подстроки

```
string s;
s=”ABCDEFGH”;
cout << s.substr(2,3);
```

Программа выведет

```
CDE
```

**Изменение строки**

Метод **push_back** дописывает один символ в конец строки. Метод **insert** вставляет внутрь строки другую строку или отдельный символ. Метод **erase** удаляет из строки подстроку. Метод **clear** удаляет из строки все символы, строка становится пустой.

```
string s;
s=”abcdefghij”;
s.push_back(‘k’); // отдельный символ пишется в апострофах, а не кавычках
cout << s << endl;
s.insert(3,”+++”); // метод получает номер позиции и вставляемую строку
cout << s << endl;
s.insert(7,”**”);
cout << s << endl;
s.erase(5,4); // метод получает номер позиции и длину удаляемой подстроки
cout << s << endl;
s.clear();
cout << s.size();
```

Программа выведет

```
abcdefghijk
abc+++defghijk
abc+++d**efghijk
abc++efghijk
0
```

Позиции в строке отсчитываются, начиная с нуля.