Всероссийская олимпиада школьников по информатике
Школьный Этап
Республика Бурятия
2020-2021

ст. тренер по информатике ЦПОС РБ
преп. каф. ИТ ИМИ БГУ
Мальцев С.П.

Справочные материалы

Основы языка программирования C++

Инструкции

Программа на C++ состоит из набора инструкций. Каждая инструкция (statement) выполняет определенное действие. В конце инструкции в языке C++ ставится точка с запятой (;). Данный знак указывает компилятору на завершение инструкции. Например:

```
std::cout << "Hello World!";
```

Данная строка выводит на консоль строку "Hello world!", является инструкцией и поэтому завершается точкой с запятой.

Набор инструкций может представлять блок кода. Блок кода заключается в фигурные скобки, а инструкции помещаются между открывающей и закрывающей фигурными скобками:

```
{
    std::cout << "Hello World!";
    std::cout << "Bye World!";
}
```

В этом блоке кода две инструкции, которые выводят на консоль определенную строку.

Функция main

Каждая программа на языке C++ должна иметь как минимум одну функцию - функцию main(). Именно с этой функции начинается выполнение приложения. Ее имя main фиксировано и для всех программ на Си всегда одинаково.

Функция также является блоком кода, поэтому ее тело обрамляется фигурными скобками, между которыми определяется набор инструкций.

В частности, при создании первой программы использовалась следующая функция main:

```
#include <iostream>                 // подключаем заголовочный файл iostream


int main()                          // определяем функцию main
{                                   // начало функции
    std::cout << "Hello World!";    // выводим строку на консоль
    return 0;                       // выходим из функции
}                                   // конец функции
```

Определение функции main начинается с возвращаемого типа. Функция main в любом случае должна возвращать число. Поэтому ее определение начинается с ключевого слова int.

Далее идет название функции, то есть main. После названия в скобках идет список параметров. В данном случае функция main не принимает никаких параметров, поэтому после названия указаны пустые скобки. Однако есть другие варианты определения функции main, которые подразумевают использование параметров. В частности, нередко может встречаться следующее определение функции main, использующей параметры:

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


}
```

И после списка параметров идет блок кода, который и содержит в виде инструкций собственно те действия, выполняемые функцией main.

Как и во многих языках программирования, в C++ для хранения данных используются переменные. Переменная имеет тип, имя и значение. Тип определяет, какую информацию может хранить переменная.

Перед использованием любую переменную надо определить. Синтаксис определения переменной выглядит следующим образом:

```
тип_переменной имя_переменной;
```

Простейшее определение переменной:

```
int age;
```

Здесь определена переменная age, которая имеет тип int. Поскольку определение переменной представляет собой инструкцию, то после него ставится точка с запятой.

Также стоит учитывать, что C++ - регистрозависимый язык, а это значит, что регистр символов имеет большое значение. То есть в следующем коде будут определяться две разные переменные:

```
int age;
int Age;
```

Поэтому переменная Age не будет представлять то же самое, что и переменная age.

Кроме того, в качестве имени переменной нельзя использовать ключевые слова языке C++, например, for или if. Но таких слов не так много: alignas, alignof, asm, auto, bool, break, case, catch, char, char16_t, char32_t, class, const, constexpr, const_cast, continue, decltype, default, delete, do, double, dynamic_cast, else, enum, explicit, export, extern, false, float, for, friend, goto, if, inline, int, long, mutubale, namespace, new, noexcept, nullptr, operator, private, protected, public, register, reinterpret_cast, return, short, signed, sizeof, static, static_assert, static_cast, struct, switch, template, this, thread_local, throw, true, try, typedef, typeid, typename, union, unsigned, using, virtual, void, volatile, wchar_t, while.

Также нельзя объявить больше одной переменной с одним и тем же именем, например:

```
int age;
int age;
```

Подобное определение вызовет ошибку на этапе компиляции.

И в довершеие следует сказать, что переменным стоит давать осмысленные имена, которые будут говорить об их предназначении.

Инициализация

После определения переменной можно присвоить некоторое значение:

```
int age;
age = 20;
```

Каждая переменная имеет определенный тип. И этот тип определяет, какие значения может иметь переменная, какие операции с ней можно производить и сколько байт в памяти она будет занимать. В языке C++ определены следующие базовые типы данных:

bool: логический тип. Может принимать одну из двух значений true (истина) и false (ложь). Размер занимаемой памяти для этого типа точно не определен.

char: представляет один символ в кодировке ASCII. Занимает в памяти 1 байт (8 бит). Может хранить любое значение из диапазона от -128 до 127, либо от 0 до 255

signed char: представляет один символ. Занимает в памяти 1 байт (8 бит). Может хранить любой значение из диапазона от -128 до 127

unsigned char: представляет один символ. Занимает в памяти 1 байт (8 бит). Может хранить любой значение из диапазона от 0 до 255

wchar_t: представляет расширенный символ. На Windows занимает в памяти 2 байта (16 бит), на Linux - 4 байта (32 бита). Может хранить любой значение из диапазона от 0 до 65 535 (при 2 байтах), либо от 0 до 4 294 967 295 (для 4 байт)

char16_t: представляет один символ в кодировке Unicode. Занимает в памяти 2 байта (16 бит). Может хранить любой значение из диапазона от 0 до 65 535

char32_t: представляет один символ в кодировке Unicode. Занимает в памяти 4 байта (32 бита). Может хранить любой значение из диапазона от 0 до 4 294 967 295

short: представляет целое число в диапазоне от –32768 до 32767. Занимает в памяти 2 байта (16 бит).
Данный тип также имеет синонимы short int, signed short int, signed short.

unsigned short: представляет целое число в диапазоне от 0 до 65535. Занимает в памяти 2 байта (16 бит).
Данный тип также имеет синоним unsigned short int.

int: представляет целое число. В зависимости от архитектуры процессора может занимать 2 байта (16 бит) или 4 байта (32 бита). Диапазон предельных значений соответственно также может варьироваться от –32768 до 32767 (при 2 байтах) или от −2 147 483 648 до 2 147 483 647 (при 4 байтах). Но в любом случае размер должен быть больше или равен размеру типа short и меньше или равен размеру типа long
Данный тип имеет синонимы signed int и signed.

unsigned int: представляет положительное целое число. В зависимости от архитектуры процессора может занимать 2 байта (16 бит) или 4 байта (32 бита), и из-за этого диапазон предельных значений может меняться: от 0 до 65535 (для 2 байт), либо от 0 до 4 294 967 295 (для 4 байт).
В качестве синонима этого типа может использоваться unsigned

long: представляет целое число в диапазоне от −2 147 483 648 до 2 147 483 647. Занимает в памяти 4 байта (32 бита).
У данного типа также есть синонимы long int, signed long int и signed long

unsigned long: представляет целое число в диапазоне от 0 до 4 294 967 295. Занимает в памяти 4 байта (32 бита).
Имеет синоним unsigned long int.

long long: представляет целое число в диапазоне от −9 223 372 036 854 775 808 до +9 223 372 036 854 775 807. Занимает в памяти, как правило, 8 байт (64 бита).
Имеет синонимы long long int, signed long long int и signed long long.

unsigned long long: представляет целое число в диапазоне от 0 до 18 446 744 073 709 551 615. Занимает в памяти, как правило, 8 байт (64 бита).
Имеет синоним unsigned long long int.

float: представляет вещественное число ординарной точности с плавающей точкой в диапазоне +/- 3.4E-38 до 3.4E+38. В памяти занимает 4 байта (32 бита)

double: представляет вещественное число двойной точности с плавающей точкой в диапазоне +/- 1.7E-308 до 1.7E+308. В памяти занимает 8 байт (64 бита)

long double: представляет вещественное число двойной точности с плавающей точкой не менее 8 байт (64 бит). В зависимости от размера занимаемой памяти может отличаться диапазон допустимых значений.

void: тип без значения

Таким образом, все типы данных за исключением void могут быть разделены на три группы: символьные (char, wchar_t, char16_t, char32_t), целочисленные (short, int, long, long long) и типы чисел с плавающей точкой (float, double, long double).

Условные конструкции направляют ход программы по одному из возможных путей в зависимости от условия.

Конструкция if

Конструкция **if** проверяет истинность условия, и если оно истинно, выполняет блок инструкций. Этот оператор имеет следующую сокращенную форму:

```
if (условие)
{
    инструкции;
}
```

В качестве *условия* использоваться условное выражение, которое возвращает **true** или **false**. Если условие возвращает true, то выполняются последующие инструкции, которые входят в блок if. Если условие возвращает false, то последующие инструкции не выполняются. Блок инструкций заключается в фигурные скобки.

Например:

```
#include <iostream>

int main()
{
    int x = 60;

    if(x > 50)
    {
        std::cout << "x is greater than 50 \n";
    }

    if(x < 30)
    {
        std::cout << "x is less than 30 \n";
    }

    std::cout << "End of Program" << "\n";
    return 0;
}
```

Здесь определены две условных конструкции if. Они проверят больше или меньше значение переменной x, чем определенное значение. В качестве инструкции в обоих случаях выполняется вывод некоторой строки на консоль.

В первом случае x > 50 условие истинно, так как значение переменной x действительно больше 50, поэтому это условие возвратит true, и, следовательно, будут выполнятся инструкции, которые входят в блок if.

Во втором случае операция отношения x < 30 возвратит false, так как условие ложно, поэтому последующий блок инструкций выполняться не

будет. В итоге при запуске программы вывод консоли будет выглядеть следующим образом:

```
x greater than 50
End of Program
```

Также мы можем использовать полную форму конструкции if, которая включает оператор else:

```
if(выражение_условия)
    инструкция_1
else
    инструкция_2
```

После оператора else мы можем определить набор инструкций, которые выполняются, если условие в операторе if возвращает false. То есть если *условие* истинно, выполняются инструкции после оператора if, а если это выражение ложно, то выполняются инструкции после оператора else.

```
int x = 50;
if(x > 60)
    std::cout << "x is greater than 60 \n";
else
    std::cout << "x is less or equal 60 \n";
```

В данном случае условие x > 60 ложно, то есть возвращает false, поэтому будет выполняться блок else. И в итоге на консоль будет выведена строка "x is less or equal 60 \n".

Однако нередко надо обработать не два возможных альтернативных варианта, а гораздо больше. Например, в случае выше можно насчитать три условия: переменная x может быть больше 60, меньше 60 и равна 60. Для проверки альтернативных условий мы можем вводить выражения **else if**:

```
int x = 60;


if(x > 60)
{
    std::cout << "x is greater than 60 \n";
}
else if (x < 60)
{
    std::cout << "x is less than 60 \n";
}
else
{
    std::cout << "x is equal 60 \n";
}
```

То есть в данном случае мы получаем три ветки развития событий в программе.

Если в блоке if или else или else-if необходимо выполнить только одну инструкцию, то фигурные скобки можно опустить:

```
int x = 60;


if(x > 60)
    std::cout << "x is greater than 60 \n";
else if (x < 60)
    std::cout << "x is less than 60 \n";
else
    std::cout << "x is equal 60 \n";
```

Для выполнения некоторых действий множество раз в зависимости от определенного условия используются циклы. В языке C++ имеются следующие виды циклов:

for

while

do...while

Цикл while

Цикл while выполняет некоторый код, пока его условие истинно, то есть возвращает true. Он имеет следующее формальное определение:

```
while(условие)
{
    // выполняемые действия
}
```

После ключевого слова **while** в скобках идет условное выражение, которое возвращает true или false. Затем в фигурных скобках идет набор инструкций, которые составляют тело цикла. И пока условие возвращает true, будут выполняться инструкции в теле цикла.

Например, выведем квадраты чисел от 1 до 9:

```
#include <iostream>


int main()
{
    int i = 1;
    while(i < 10)
    {
        std::cout << i << " * " << i << " = " << i * i << std::endl;
        i++;
    }
```

```
    return 0;
}
```

Здесь пока условие i < 10 истинно, будет выполняться цикл while, в котором выводится на консоль квадрат числа и инкрементируется переменная i. В какой-то момент переменная i увеличится до 10, условие i < 10 возвратит false, и цикл завершится.

Консольный вывод программы:

```
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
```

Каждый отдельный проход цикла называется итерацией. То есть в примере выше было 9 итераций.

Если цикл содержит одну инструкцию, то фигурные скобки можно опустить:

```
int i = 0;
while(++i < 10)
    std::cout << i << " * " << i << " = " << i * i << std::endl;
```

Цикл for

Цикл for имеет следующее формальное определение:

```
for (выражение_1; выражение_2; выражение_3)
{
    // тело цикла
}
```

выражение_1 выполняется один раз при начале выполнения цикла и представляет установку начальных условий, как правило, это инициализация счетчиков - специальных переменных, которые используются для контроля за циклом.

выражение_2 представляет условие, при соблюдении которого выполняется цикл. Как правило, в качестве условия используется операция сравнения, и если она возвращает ненулевое значение (то есть условие истинно), то выполняется тело цикла, а затем вычисляется *выражение_3*.

выражение_3 задает изменение параметров цикла, нередко здесь происходит увеличение счетчиков цикла на единицу.

Например, перепишем программу по выводу квадратов чисел с помощью цикла for:

```
#include <iostream>


int main()
{
    for(int i =1; i < 10; i++)
    {
        std::cout << i << " * " << i << " = " << i * i << std::endl;
    }


    return 0;
}
```

Первая часть объявления цикла - int i = 1 - создает и инициализирует счетчик i. Фактически это то же самое, что и объявление и инициализация переменной. Счетчик необязательно должен представлять тип int. Это может быть и другой числовой тип, например, float. И перед выполнением цикла его значение будет равно 1.

Вторая часть - условие, при котором будет выполняться цикл. В данном случае цикл будет выполняться, пока переменная i не станет равна 10.

И третья часть - приращение счетчика на единицу. Опять же нам необязательно увеличивать на единицу. Можно уменьшать: i--. Можно изменять на другое значение: i+=2.

В итоге блок цикла сработает 9 раз, пока переменная i не станет равна 10. И каждый раз это значение будет увеличиваться на 1. И по сути мы получим тот же самый результат, что и в случае с циклом while:

```
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
```

Необязательно указывать все три выражения в определении цикла, мы можем одно или даже все из них опустить:

```
int i = 1;
for(; i < 10;)
{
    std::cout << i << " * " << i << " = " << i * i << std::endl;
    i++;
}
```

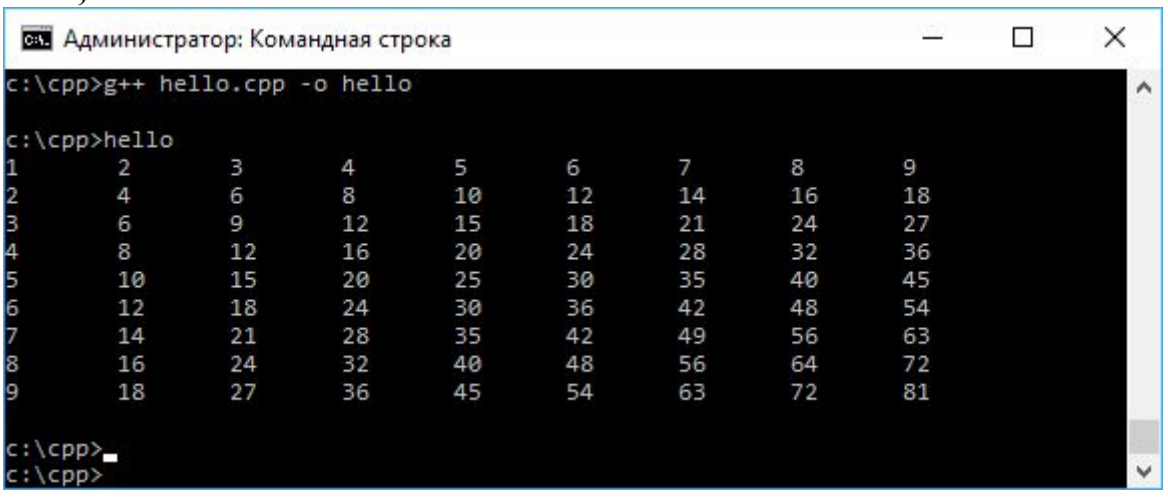
Формально определение цикла осталось тем же, только теперь первое и третье выражения в определении цикла отсутствуют: for (; i < 10;). Переменная-счетчик определена и инициализирована вне цикла, а ее приращение происходит в самом цикле.

Можно определять вложенные циклы. Например, выведем таблицу умножения:

```
#include <iostream>

int main()
{
    for (int i=1; i < 10; i++)
    {
        for(int j = 1; j < 10; j++)
        {
            std::cout << i * j << "\t";
        }
        std::cout << std::endl;
    }

    return 0;
}
```



Цикл do

В цикле do сначала выполняется код цикла, а потом происходит проверка условия в инструкции while. И пока это условие истинно, то есть не равно 0, то цикл повторяется. Формальное определение цикла:

```
do
{
   инструкции
}
while(условие);
```

Например:

```
#include <iostream>


int main()
{
   int i = 6;
   do
   {
      std::cout << i << std::endl;
      i--;
   }
   while(i>0);


   return 0;
}
```

Здесь код цикла сработает 6 раз, пока i не станет равным нулю.

Но важно отметить, что цикл do гарантирует хотя бы однократное выполнение действий, даже если условие в инструкции while не будет истинно. То есть мы можем написать:

```
int i = -1;
do
{
   std::cout << i << std::endl;
   i--;
}
while(i>0);
}
```

Хотя у нас переменная i меньше 0, цикл все равно один раз выполнится.

Операторы continue и break

Иногда возникает необходимость выйти из цикла до его завершения. В этом случае можно воспользоваться оператором **break**. Например:

```
#include <iostream>

int main()
{
   int i = 1;
   for ( ; ; )
   {
      std::cout << i << " * " << i << " = " << i * i << std::endl;
      i++;
      if (i > 9) break;
   }
   return 0;
}
```

Здесь когда значение переменной i достигнет 10, осуществляется выход из цикла с помощью оператора break.

В отличие от оператора break, оператор **continue** производит переход к следующей итерации. Например, нам надо посчитать сумму только нечетных чисел из некоторого диапазона:

```
#include <iostream>
int main()
{
    int result = 0;
    for (int i=0; i<10; i++)
    {
        if (i % 2 == 0) continue;
        result +=i;
    }
    std::cout << "result = " << result << std::endl; // 25
    return 0;
}
```

Чтобы узнать, четное ли число, мы получаем остаток от целочисленного деления на 2, и если он равен 0, то с помощью оператора continue переходим к следующей итерации цикла. А если число нечетное, то складываем его с остальными нечетными числами.

Примеры решения олимпиадных задач

1) Требуется сложить два целых числа A и B.

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

2) Одна из основных операций с числами – их сравнение. Мы подозреваем, что вы в совершенстве владеете этой операцией и можете сравнивать любые числа, в том числе и целые. В данной задаче необходимо сравнить два целых числа.

```
# include <iostream>
using namespace std;
int main()
{
   int a, b;
   cin>>a>>b;
   if(a==b)
      cout<<"=";
   else
      if(a<b)
         cout<<"<";
      else
         cout<<">";
}
```

3) На столе лежат n монеток. Некоторые из них лежат вверх решкой, а некоторые – гербом. Определите минимальное число монеток, которые нужно перевернуть, чтобы все монетки были повернуты вверх одной и той же стороной. В первой строке записано натуральное число N ($1 \le N \le 100$) – число монеток. В каждой из последующих N строк содержится одно целое число – 1 если монетка лежит решкой вверх и 0 если вверх гербом.

```
# include <iostream>
# include <vector>
using namespace std;
int main(){
   int n;
```

```
    int k=0;
    cin>>n;
    vector<int> a(n);
    for(int i=0; i<n; i++)
        cin>>a[i];
    for(int i=0; i<n; i++)
        if(a[i]==1)
            k++;
    if(k>n-k)
        cout<<n-k;
    else
        cout<<k;
}
```

Полезные ссылки для подготовки к олимпиаде

http://www.rosolymp.ru Портал Всероссийской олимпиады школьников

Интернет-ресурсы с коллекциями олимпиадных задач:

- **http://www.olympiads.ru/moscow/index.shtml (сайт московских олимпиад по информатике);**
- **http://neerc.ifmo.ru/school/russia-team/archive.html (сайт с архивом задач Всероссийских командных олимпиад школьников по программированию);**
- **http://www.olympiads.ru/ (сайт по олимпиадной информатике);**

Интернет-ресурсы с коллекциями олимпиадных задач и возможностью их тестирования в реальном масштабе времени:

- **http://acm.timus.ru (сайт Уральского государственного университета, содержащий большой архив задач с различных соревнований по спортивному программированию)**
- **http://acmp.ru ("Школа программиста" Сайт разработан в Красноярском краевом Дворце пионеров)**
- **http://informatics.mccme.ru (Дистанционная подготовка по информатике)**
- **http://acmu.ru/ (Олимпиады по информатике ХМАО - Югра)**

Интернет – турниры

Сайты интернет-олимпиад для школьников:

- **http://olymp.ifmo.ru/ (сайт городских интернет – олимпиад школьников Санкт-Петербурга);**
- **http://neerc.ifmo.ru/school/io/index.html (сайт интернет-олимпиад по информатике, проводимых жюри Всероссийской командной олимпиады школьников по программированию);**
- **http://www.olympiads.ru/online/index.shtml (сайт московских онлайн-олимпиад)**

Олимпиадные сайты зарубежных стран:

- **http://www.topcoder.com/tc (сайт интернет-соревнований компании TopCoder).**