СИНТАКСИС ЯЗЫКА С++

СИНТАКСИС ЯЗЫКА C++

Операторы

Условный оператор выбора *if*

Оператор **if** предназначен для выполнения тех или иных действий в зависимости от истинности или ложности некоторого условия. Условие задается выражением, возвращающим результат **булева** типа.

```
if (условие) оператор;
```

```
if (B > A) C = B;
C = A;
```

```
if (условие) оператор1;
else оператор2;
```

Тернарный оператор if

```
1  операнд1? операнд2 : операнд3
```

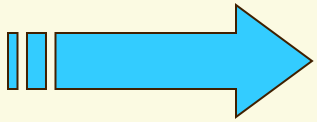
```
int a {5};
int b {8};
a > b ? std::cout << a-b : std::cout << a+b;
```

СИНТАКСИС ЯЗЫКА С++

Операторы

Условный оператор выбора *if*

```
if (условие1)
  if (условие2)
    оператор1;
  else оператор2;
```



```
if (условие1)
  { if (условие2)
      оператор1;
    else оператор2;}
```

```
if (условие1)
  { if (условие2)
      оператор1;}
    else оператор2;
```

СИНТАКСИС ЯЗЫКА С++

Операторы

Условный оператор множественного выбора *switch*

```
switch (выражение_выбора) {
        case  значение_1 : оператор_1;
                                break;          //не обязательно
        ……..
        case значение_n : оператор_n;
                                break;          //не обязательно
        default : оператор;                     //не обязательно
}
```

В этой конструкции выражение выбора должно иметь **порядковый** тип - целый, перечислимый и т.п.

СИНТАКСИС ЯЗЫКА С++

Операторы

Оператор передачи управления *goto*

Оператор **goto** позволяет прервать обычный поток управления и передать управление в произвольную точку кода, помеченную специальной меткой.

```
goto  L1;
…
second:  …
…
L1:  …
…
if (…) goto L1;
   else goto second;
```

СИНТАКСИС ЯЗЫКА С++

Оператор цикла *for*

Оператор **for** обеспечивает циклическое повторение некоторого оператора (в т.ч. составного) *заданное число раз*. Повторяемый оператор называется *телом* цикла. Повторение цикла обычно определяется некоторой управляющей переменной (*счетчиком*), которая изменяется при каждом выполнении тела цикла. Повторение заканчивается когда управляющая переменная достигает заданного значения.

for (выражение1; выражение2; выражение3) оператор;

СИНТАКСИС ЯЗЫКА C++

Оператор цикла *for*

Необязательные параметры for

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

Несколько счетчиков в списке инициализации for

```
#include <iostream>

int main()
{
    int numbers[]{1, 2, 3, 4};
    int sum {};
    for (int i {1}, j {5}; i < 6 && j < 10; i++, j++)
    {
        std::cout  << i << "*" << j << "="<< i * j << std::endl;   // Sum: 15
    }

}
```

СИНТАКСИС ЯЗЫКА C++

Оператор цикла *for*

Перебор значений в стиле for-each

```
for(тип переменная : последовательность)
{
    инструкции;
}
```

```
#include <iostream>

int main()
{
    for (int n : {2, 3, 4, 5})
    {
        std::cout << n << std::endl;
    }
}
```

```
#include <iostream>

int main()
{
    for (char c : "Hello")
    {
        std::cout << c << std::endl;
    }
}
```

СИНТАКСИС ЯЗЫКА С++

Оператор цикла *do … while*

Структура **do…while** используется для организации циклического выполнения оператора или операторов, называемых *телом цикла*, до тех пор, пока *не окажется* ***нарушенным*** *некоторое условие*

```
do
    оператор
 while  (условие);
```

СИНТАКСИС ЯЗЫКА С++

Оператор цикла *while*

Оператор **while** используется для организации циклического выполнения *тела цикла*, *пока* ***выполняется*** *некоторое условие*.

while (условие) оператор ;

СИНТАКСИС ЯЗЫКА С++

Прерывание цикла: операторы break, Continue, return, функция Abort

📖 Оператор **break** прерывает выполнение тела любого цикла **for**, **do**, **while** и передает управление следующему за циклом выполняемому оператору.

📖 Для прерывания циклов, расположенных в функциях, можно воспользоваться оператором **return**. В отличие от оператора **break**, оператор **return** прервет не только выполнение цикла, но и выполнение той функции, в которой расположен цикл.

📖 Функция **Abort** прерывает выполнение блока, в котором расположен цикл, генерируя исключение, не связанное с сообщением об ошибке.

📖 Оператор **Continue** прерывает выполнение текущей итерации цикла и передает управление на следующую итерацию.

СИНТАКСИС ЯЗЫКА С++

Массивы

Одномерные массивы

Массив представляет собой структуру данных, позволяющую хранить под одним именем совокупность данных любого, но только одного какого-то типа.

Массив характеризуется своим именем, типом хранимых элементов, размеров, нумерацией элементов и размерностью.

тип переменная [константное_выражение]

```
int A[10];
```

СИНТАКСИС ЯЗЫКА C++

Массивы

Одномерные массивы

Код

```
A[0] = 1;
A[1] = 1;
for  (int  i = 2;  i < =9;  i++)
        A[i] = A[i-2] + A[i-1];
```

заполняет массив числами Фибоначчи, первые два из которых равны 2, а каждое последующее равно сумме двух предыдущих.

СИНТАКСИС ЯЗЫКА С++

Массивы

Одномерные массивы

Объявление переменной массива можно совмещать с инициализацией элементов массива.

```
int  A[10] = {1,2,3,4,5,6,7,8,9,10};

char  S[10] = {“abcdifghi\0”};
```

Если начальных значений меньше, чем элементов в массиве, оставшиеся элементы автоматически получают нулевые начальные значения.

```
int  A[10] = {0};          //присваивает нулевые значения всем
                           //элементам массива
```

СИНТАКСИС ЯЗЫКА С++

Массивы

Одномерные массивы

Все размеры массивов в программе следует определять именованными константами или макросами. Это делает программу более понятной и существенно облегчает ее отладку и сопровождение.

СИНТАКСИС ЯЗЫКА С++

Массивы

Многомерные массивы

int A2[10][3];

Этот оператор объявляет двумерный массив - таблицу, состоящую из 10 строк и 3 столбцов.

Доступ к элементам массива осуществляется через индексы, каждый из которых заключается в квадратные скобки.

Первый индекс обозначает номер строки, второй - номер столбца. Индексы нумеруются начиная с **нуля**.

СИНТАКСИС ЯЗЫКА С++

Массивы

Многомерные массивы

Если многомерный массив инициализируется при его объявлении, список значений по каждой размерности заключается в *фигурные скобки*.

```
int A3[4][3][2] = {{{0,1},{2,3},{4,5}},
                   {{6,7},{8,9},{10,11}},
                   {{12,13},{14,15},{16,17}},
                   {{18,19},{20,21},{22,23}}};
```

0	**1**
2	**3**
4	**5**

6	**7**
8	**9**
10	**11**

12	**13**
14	**15**
16	**17**

18	**19**
20	**21**
22	**23**

A3[0][1][0] = 2

A3[3][0][1] = 19

ФУНКЦИИ

Объявление и описание функций

тип_возвращаемого_значения имя_функции (список_парам.)

тип_параметра идентификатор_парам.

```
double FSum (double X1, double X2, int A)  //точка с запятой
{                                          //после объявл. функции
        операторы тела функции;            //не ставиться
}
```

```
void SPrint (char[20] S)
{
        операторы тела функции;
}
```

ФУНКЦИИ

Прототипы функций

```
double  FSum (double  X1, double  X2, int  A);
void  SPrint (char[20]  S);
void  F1 (void);
```

или

```
double  FSum (double, double, int);
 void  SPrint (char[20]);
void  F1 ( );
```

Обращение к функции:

```
Fsum(Right, Left, A);
SPrint(“My Name is:”);
F1();
```

ФУНКЦИИ

Возврат из функций

```
double  FSum (double  X1, double  X2, int  A)
{
        return A*(X1 + X2);
}
```

```
void  SPrint (char[20]  S)
{
        if (S != “ ”)
            cout<<S<<endl;
}
```

СИНТАКСИС ЯЗЫКА С++

Передача параметров в функции

Передача параметров по значению

double FSum(**double** X1, **double** X2, **int** A)

Вызов функции: FSum(Y, X2, 5)

Затраты времени и затраты памяти при копировании параметров могут быть значительными в случае использования в качестве параметров массивов.

СИНТАКСИС ЯЗЫКА С++

Передача параметров в функции

Передача параметров по ссылке

```
void  square(int  &);   //Прототип функции вычисления квадрата
void  square(int  &a)   //Заголовок функции
{
    a  = a * a;         //Изменение значения параметра
}
```

Вызов функции:

```
int  x1 = 2;
square(x1);             //a = 4
```

СИНТАКСИС ЯЗЫКА С++

Передача параметров в функции

Передача параметров по ссылке

int &a
int & a
int& a

} идентично

1. Реально в функцию передается не сама переменная, а ее адрес, полученный операцией адресации (&).

2.Оператор вызова дает вызываемой функции возможность прямого доступа к передаваемым данным, а также возможность изменения этих данных.

3. Производительность повышается, но защищенность данных снижается.

СИНТАКСИС ЯЗЫКА С++

Передача параметров в функции

Применение при передаче параметров спецификации *const*

Аргумент при подобном способе передачи не копируется при вызове функции, но внутри функции изменить значение переданного ей параметра невозможно.

double F1(**const** &A)

СИНТАКСИС ЯЗЫКА С++

Передача параметров в функции

Параметры со значениями по умолчанию

```
double  Arh(double  V = 1, double  P = 0.5, double  PH20 =1,
                    double  G = 9.81)
{
        return  G * V * (PH20 - P);
}
```

Вызов функции:

```
F = Arh();
```

Аргументы по умолчанию должны быть самыми правыми (последними) аргументами в списке параметров функции.

СИНТАКСИС ЯЗЫКА C++

Встраиваемые функции

📖 В C++ для снижения накладных расходов на вызовы функций - особенно небольших функций - предусмотрены встраиваемые (*inline*) функции. Спецификация **inline** перед указанием типа результата в объявлении функции «советует» компилятору сгенерировать копию кода функции в соответствующем месте программы, чтобы избежать вызова функции.

```
inline double Circ(double R)
{
        return 6.28318 * R;
}
```

СИНТАКСИС ЯЗЫКА С++

Перегрузка функций

📖 С++ позволяет определить несколько функций с одним и тем же именем, если эти функции имеют разные наборы параметров (по меньшей мере разные типы параметров), - это **называется перегрузкой функций**.

📖 Перегруженные функции, которые решают сходные задачи, делают программы более понятными и легко читаемыми.