Лекция 5. Операторы языка С++

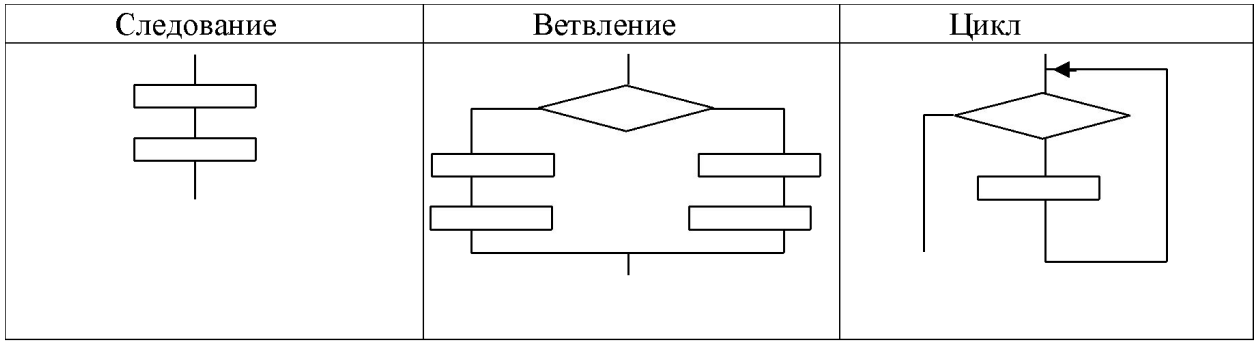
Операторы управляют процессом выполнения программы. Набор операторов языка С++ содержит все управляющие конструкции структурного программирования.

В теории программирования доказано, что программу для решения задачи любой сложности можно составить из трех структур: **линейной, разветвляющейся и циклической.**

Линейной называется конструкция, представляющая собой последовательное соединение двух или более операторов.

Ветвление – задает выполнение одного из двух операторов, в зависимости от выполнения какого либо условия.

Цикл – задает многократное выполнение оператора.



Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости вносить в нее изменения.

Операторы управления работой программы называют **управляющими конструкциями программы.**

К ним относят:

- **оператор «выражение» (пустой оператор);**
- **составные операторы;**
- **операторы выбора;**
- **операторы циклов;**
- **операторы перехода.**

1. Оператор «выражение»

Любое выражение, заканчивающееся точкой с запятой, рассматривается как оператор, выполнение которого заключается в вычислении этого выражения. Частным случаем выражения является пустой оператор **;**.

Примеры: •

```
i++;
a+=2;
x=a+b;
```

Пустой оператор – ; – это оператор, состоящий только из точки с запятой. Он может появиться в любом месте программы, где по синтаксису требуется оператор. Выполнение пустого оператора не меняет состояния программы.

2. Составные операторы

К составным операторам относят собственно составные операторы и блоки. В обоих случаях это последовательность операторов, заключенная в фигурные скобки. Блок отличается от составного оператора наличием определений в теле блока.

Например:

```
{
n++;                    это составной оператор
summa+=n;
}

{
int n=0;
n++;                    это блок
summa+=n;
}
```

3. Операторы выбора

Операторы выбора - это условный оператор **(if)** и переключатель **(switch).**

1. Условный оператор if

Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным.

Условная конструкция в C++ всегда записывается в круглых скобках после оператора if. Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.

Оператор if ("если") позволяет организовать ветвление в программе.

Он имеет две формы: **оператор "если" и оператор "если...иначе".**

Оператор "если" имеет вид	**оператор "если...иначе" имеет вид**

if (условие)
действие;
	if (условие)
	действие1;
	else
	действие2;

Пример конструкции ветвления

```
if (num < 10) {  // Если введенное число меньше 10.
    cout << "Это число меньше 10." << endl;
} else {  // иначе
    cout << "Это число больше либо равно 10." << endl;
}
```

Здесь говорится: «Если переменная num меньше 10 — вывести соответствующее сообщение. Иначе, вывести другое сообщение».

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
double num;
cout << "Введите произвольное число: ";
cin >> num;
if (num < 10) {  // Если введенное число меньше 10.
cout << "Это число меньше 10." << endl;
}
else {  // иначе
cout << "Это число больше либо равно 10." << endl;
}
system("pause");
return 0;
}
```

Усовершенствуем программу так, чтобы она выводила сообщение, о том, что переменная num равна десяти.

Усовершенствуем программу так, чтобы она выводила сообщение, о том, что переменная num равна десяти:

```
if (num < 10) {  // Если введенное число меньше 10.
    cout << "Это число меньше 10." << endl;
} else if (num == 10) {
    cout << "Это число равно 10." << endl;
} else {  // иначе
    cout << "Это число больше 10." << endl;
}
```

Здесь мы проверяем три условия:
Первое — когда введенное число меньше 10-ти
Второе — когда число равно 10-ти
И третье — когда число больше десяти
Заметьте, что во втором условии, при проверке равенства, мы используем **оператор равенства** — ==, а не оператор присваивания, потому что мы не изменяем значение переменной при проверке, а сравниваем ее текущее значение с числом 10.

Если поставить оператор присваивания в условии, то при проверке условия, значение переменной изменится, после чего это условие выполнится.

Каждому оператору **if** соответствует только один оператор **else**. Совокупность этих операторов — **else if** означает, что если не выполнилось предыдущее условие, то проверить данное. Если ни одно из условий не верно, то выполняется тело оператора else.

Если после оператора if, else или их связки else if должна выполняться только одна команда, то фигурные скобки можно не ставить. Предыдущую программу можно записать следующим образом:

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
double num;
cout << "Введите произвольное число: ";
cin >> num;
if (num < 10) // Если введенное число меньше 10.
 cout << "Это число меньше 10." << endl;
else if (num == 10)
cout << "Это число равно 10." << endl;
else // иначе cout << "Это число больше 10." << endl;
 return 0;
}
```

Такой метод записи выглядит более компактно. Если при выполнении условия нам требуется выполнить более одной команды, то фигурные скобки необходимы. Например:

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
double num;
int k;
cout << "Введите произвольное число: ";
cin >> num;
 if (num < 10) { // Если введенное число меньше 10.
cout << "Это число меньше 10." << endl;
k = 1;
 } else if (num == 10) {
cout << "Это число равно 10." << endl;
k = 2;
} else { // иначе cout << "Это число больше 10." << endl;
 k = 3;
} cout << "k = " << k << endl;
return 0;
}
```

11. Записать программу и получить результат работы данной программы

Данная программа проверяет значение переменной **num**. Если она меньше 10, то присваивает переменной k значение единицы. Если переменная num равна десяти, то присваивает переменной k значение двойки. В противном случае — значение тройки. После выполнения ветвления, значение переменной k выводится на экран.

2. Оператор switch (C++)

Использование множества if-else для проверки одной переменной – практика распространенная, но C++ предоставляет альтернативный и более эффективный условный оператор ветвления, называемый **switch**.

Программа с использованием оператора switch имеет следующий вид:

```
switch (выражение)
{
case значение_1: операторы_1;
break;
case значение_2: операторы_2;
break;
case значение_3: операторы_3;
break;
…
case значение_n: операторы_n;
break;
default: операторы;
break;
}
```

Оператор работает следующем образом. Вычисляется значение *выражения*. Затем выполняются *операторы*, помеченные *значением,* совпадающим со значением *выражения.* То есть если, выражение принимает *значение_1*, то выполняются *операторы_1* и т.д.. Если выражение не принимает ни одного из значений, то выполняются операторы, расположенные после слова **default**.

Ветвь **default** может отсутствовать, тогда оператор имеет вид:

```
switch (выражение)
{
case значение_1: операторы_1;
break;
case значение_2: операторы_2;
break;
case значение_3: операторы_3;
break;
…
case значение_n: операторы_n;
break;
}
```

Оператор **break** необходим для того, чтобы осуществить выход из операторы switch. Если он не указан, то будут выполняться следующие операторы из списка, несмотря на то, что значение, которым они помечены, не совпадает со значением выражения.

Следует помнить о важных моментах применения оператора switch:

- switch отличается от if тем, что он может выполнять только операции проверки строгого равенства, в то время как if может вычислять логические выражения и отношения.
- Не может быть двух констант в одном операторе switch, имеющих одинаковые значения. Конечно, оператор switch, включающий в себя другой оператор switch, может содержать аналогичные константы.

- Если в операторе **switch** используются символьные константы, они автоматически преобразуются к целочисленным значениям.
- Оператор **default** выполняется, если не найдено соответствий, default необязателен и, если его нет, то в случае отсутствия совпадений ничего не происходит. Когда обнаруживается совпадение, операторы, ассоциированные с соответствующим case, выполняются до тех пор, пока не встретится оператор break. В случае default (или последнего case, если отсутствует default), оператор switch заканчивает работу при обнаружении конца.
- С технической точки зрения операторы **break** являются необязательными в операторе switch. Они используются для окончания работы последовательности операторов, ассоциированных с данной константой. Если оператор break отсутствует, продолжают выполняться операторы следующего раздела, пока не будет достигнут оператор break или конец оператора switch.

Пример:

```
#include <iostream.h>
void main()
{
int i;
cout<<"\nEnter the number";
cin>>i;
switch(i)
{
case 1:cout<<"\nthe number is one";
case 2:cout<<"\n2*2="<<i*i;
case 3: cout<<"\n3*3="<<i*i;break;
case 4: cout<<"\n"<<i<<" is very beautiful!";
default: cout<<"\nThe end of work";
}
}
```

```
#include <iostream>
using namespace std;
int main()
{
int i;
cout << "\nEnter the number";
cin >> i;
switch (i)
{
case 1:cout << "\nthe number is one";
case 2:cout << "\n2*2=" << i * i;
case 3: cout << "\n3*3=" << i * i; break;
case 4: cout << "\n" << i << " is very beautiful!";
default: cout << "\nThe end of work";
}
system("pause");
return 0;
}
```

Результаты работы программы:

1. При вводе 1 будет выведено:
The number is one
2*2=1
3*3=1
2. При вводе 2 будет выведено:
2*2=4
3*3=4
3.При вводе 3 будет выведено:
3*3=9
4.При вводе 4 будет выведено:
4 is very beautiful!
5.При вводе всех остальных чисел будет выведено:
The end of work

12. Записать программу и получить результат работы данной программы

Задание № 1

Дано целое число n = 1..3, которое есть номером функции. По значению переменной n вычислить значение соответствующей функции с помощью оператора switch:

1) $-2x^2-4$; 2) $5x+2$; 3) $15-3x$.

13. Записать программу и получить результат работы данной программы

Фрагмент кода, который решает данную задачу с помощью сокращенной формы оператора switch.

```
int n;
float f, x;
x = 3;
switch (n)
{
   case 1:
      f = -2*x*x-4;
   break;
   case 2:
      f = 5*x+2;
   break;
   case 3:
      f = 15-3*x;
   break;
}
```

13. Записать программу и получить результат работы данной программы

Задание: написать программу, которая складывает, вычитает, умножает, делит два числа введенных с клавиатуры. Разработать пользовательский интерфейс с применением **оператора switch**.

14. Записать программу и получить результат работы данной программы

Лекция 6 -7 Операторы циклов

Операторы цикла используются для организации многократно повторяющихся вычислений.

Различают:

1) **итерационные циклы;**
2) **арифметические циклы.**

Итерационные циклы – это циклы, в которых число повторений циклов заранее не известно и зависит от некоторых условий.

Арифметический цикл - это цикл, число повторений которого известно или может быть вычислено. Окончание определяется сравнением параметра цикла с концом цикла.

Группа действий, повторяющихся в цикле, называется его **телом**. Однократное выполнение цикла называется его **шагом**.

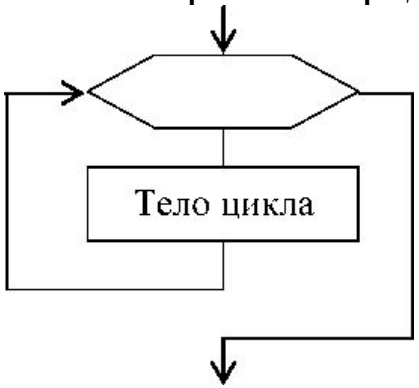
В итерационных циклах известно условие выполнения цикла.

В языке C++ существует три типа циклических конструкций:

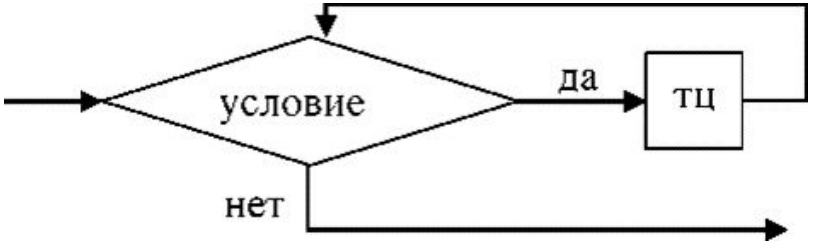
- цикл с предусловием (while);
- цикл с постусловием (do while);
- цикл с параметром (с заданным количеством повторений (for)).

- ***цикл*** - серия команд ***(тело цикла)*** выполняется многократно. Разновидности циклов:

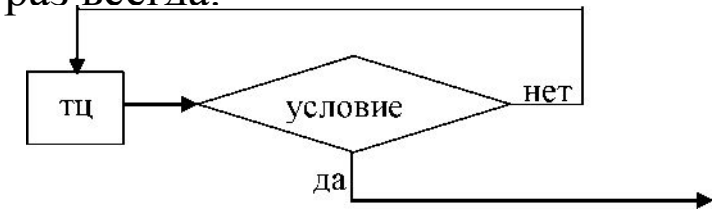
Цикл со счётчиком (for) - выполняется заранее определённое количество раз:



Цикл с предусловием (while) - проверяется условие и, если оно выполняется, то тело цикла (тц) повторяется, если нет - происходит переход к действию, следующему за телом цикла. Если условие не выполняется при первой проверке, то тело цикла не выполняется ни одного раза:



Цикл с постусловием (do while) - тело цикла (тц) выполняется, затем проверяется условие и, если оно не выполняется, то тело цикла повторяется, если выполняется, то происходит переход к действию, следующему за телом цикла. В этом варианте тело цикла выполняется хотя бы один раз всегда.



Когда мы не знаем, сколько итераций должен произвести цикл, нам понадобится цикл **while** или **do...while**.

1. Цикл с предусловием:

while

Выполнение цикла с предусловием начинается с условия, если оно истинно, выполняется оператор цикла. Если при первой проверке условие **ложно**, то **цикл не выполнится ни разу**. В общем виде цикл с предусловием приведен ниже:

```
while (условие)
{
блок инструкций
}
```

Пример

```
while (a!=0)
{
cin>>a;
s+=a;
}
```

Программа:

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
int a=1;
int s=1;
while (a!= 0)
{
cin >> a;
cout << "\n s+=a равняется " << (s+= a) << endl;
}
system("pause");
return 0;
}
```

Здесь пока условие a!= 0 истинно, будет выполняться цикл while, в котором выводится на консоль величина s+= a.

15. Записать программу и получить результат работы данной программы

Задание № 1 Вывести квадраты чисел от 1 до 9.

16. Записать программу и получить результат работы данной программы

Задание 2. Запишите код программы, считающей сумму всех целых чисел от 1 до 1000.

17. Записать программу и получить результат работы данной программы

Задание 3. Для закрепления материала по циклу while разработаем всем известную игру «Отгадай число».
Загадывать число будет компьютер, а мы — отгадывать. Используя генератор случайных чисел, компьютер загадает число от 1 до 10, включительно, пока мы не отгадаем число выход из цикла не произойдет.

Генерация случайных чисел в C++

C++ имеет свой собственный встроенный генератор случайных чисел.

Он реализован в двух отдельных функциях, которые находятся в **заголовочном файле** cstdlib:

srand() – устанавливает передаваемое значение пользователем, как стартовое число. srand() следует вызывать только один раз — в начале программы (обычно в верхней части функции main()).

rand() — генерирует следующее случайное число в последовательности.

Оно будет из диапазона от 0 до RAND_MAX (**константа** в cstdlib, значение которой — 32767).

Функция **srand** выполняет инициализацию генератора случайных чисел **rand.** Генератор псевдо-случайных чисел инициализируется с помощью аргумента **seed,** который играет роль зерна.
Для любого другого значения, передаваемого через параметр **seed,** и используемого при вызове функции **srand**, алгоритм генерации псевдо-случайных чисел может генерировать различные числа с каждым последующим вызовом функции rand.

Если использовать одно и то же значение seed, с каждым вызовом функции rand, алгоритм генерации псевдо-случайных чисел будет генерировать ту же самую последовательность чисел.

Если **seed** установлен в 1, генератор инициализируется до первоначального значения и производит те же значения, как перед любым вызовом рандов или srand.

Для того, чтобы генерировать случайные числа, функция **srand** обычно инициализируется некоторыми различными значения, например, такие значения генерируются функцией **time**.
srand(time(0));
Значение, возвращенное функцией time (объявлена в заголовке **<ctime>)** отличается каждую секунду, что дает возможность получать совершенно случайные последовательности чисел, при каждом новом вызове функции **rand.**

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

int main(int argc, char* argv[])
{
srand(time(0));
int unknown_number = 1 + rand() % 10; // загадываемое число
int enter_number; // переменная для хранения введённого числа
cout << "Enter unknown number [1:10] : "; // начинаем отгадывать
cin >> enter_number;
while (enter_number != unknown_number)
{
cout << "Enter unknown number [1:10] : ";
cin >> enter_number; // продолжаем отгадывать
}
cout << "You win!!!\n";
system("pause");
return 0;
}
```

18. Записать программу и получить результат работы данной программы

2. Цикл с постусловием:

Цикл **do while** очень похож на цикл while. Единственное их различие в том, что при выполнении цикла do while один проход цикла будет выполнен независимо от условия.

Тело цикла выполняется до тех пор, пока выражение-условие истинно.

Пример:

```
do
{
cin>>a;
s+=a;
}
while(a!=0);
```

Пример программы

```
#include <iostream>
using namespace std;
int main()
{
setlocale(0, "");
int a = 0;
int s = 0;
do
{
cin >> a;
cout << "\n s+=a равняется " << (s += a) << endl;
}
while (a != 0);
system("pause");
return 0;
}
```

19. Записать программу и получить результат работы данной программы

Задание 4

Запишите решение задачи на поиск суммы чисел от 1 до 1000, с применением цикла **do while.**

20. Записать программу и получить результат работы данной программы

Решение

Задание 4:

```
#include <iostream>
 using namespace std;
int main ()
{
setlocale(0, "");
int i = 0; // инициализируем счетчик цикла.
int sum = 0; // инициализируем счетчик суммы.
Do
{// выполняем цикл.
i++;
sum += i;
}
while (i < 1000); // пока выполняется условие.
cout << "Сумма чисел от 1 до 1000 = " << sum << endl;
return 0; }
```

Принципиального отличия нет, но если присвоить переменной i значение, большее, чем 1000, то цикл все равно выполнит хотя бы один проход.

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

int main()
{
srand(time(0));
int Ne = 1 + rand() % 20;
int Y;
cout << "Enter Y number [1:20] : ";
cin >> Y;
while (Ne != Y)
{
cout << "Enter Y number [1:20] : ";
cin >> Y;
}
cout << “Very good!!!\n";
system("pause");
return 0;
}
```