Практическое занятие 12

Тема: «Операторы цикла »

Теоретическая часть

Одним из мощных механизмов управления ходом последовательности выполнения программы является использование циклов. Цикл задает многократное выполнение одного и того же программного кода (итерации). Он имеет точку вхождения, проверочное условие и, как правило, точку выхода. Цикл, не имеющий точки выхода, называется бесконечным. Для бесконечного цикла проверочное условие всегда принимает истинное значение. Проверка условия может осуществляться перед выполнением (циклы **for**, **while**) или после окончания (**do-while**) тела цикла. Циклы могут быть вложенными друг в друга.

Циклы for

Синтаксис цикла for имеет вид:

for(выражение1; выражение2; выражение3)
оператор_или_блок_операторов;

Оператор **for** работает следующим образом:

1) Выполняется выражение1.
2) Вычисляется величина выражения2. Если полученный результат принял истинное значение, выполняется тело цикла (оператор_или_блок__опера«горов). В противном случае выполнение цикла прекращается и осуществляется переход к оператору, который следует непосредственно за телом цикла.
3) После выполнения тела цикла вычисляется выражение3 и осуществляется переход к пункту 2 (вычисление величины выражения2 и т. д.).

Выражение1 чаще всего служит в качестве инициализации какой-нибудь переменной, выполняющей роль счетчика итераций. **Выражение2** используется как проверочное условие и на практике часто содержит выражения с операторами сравнения. По-умолчанию величина **выражения2** принимает истинное значение. **Выражение3** служит чаще всего для приращения значения счетчика циклов либо содержит выражение, влияющее, каким бы то ни было образом, на проверочное условие.

Все три выражения могут отсутствовать в конструкции, однако синтаксис не допускает пропуска символа точка с запятой (;). Поэтому простейший пример бесконечного цикла for (выполняется постоянно до принудительного завершения программы) выглядит следующим образом:

```
for( ; ; )
cout << "Бесконечный цикл... " ;
```

Если в цикле должны синхронно изменяться несколько переменных, которые зависят от переменной цикла, вычисление их значений можно поместить в оператор **for**, воспользовавшись оператором "запятая". Изменение значения счетчика цикла for выполняется в его конструкции (выражение3), а в теле цикла изменений счетчика цикла не допускают.

В качестве примера выполним суммирование набора из десяти целых чисел, начиная с 10.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int Sum = 0;
    for(int i=10; i<20; i++)
       Sum = Sum + i;
    cout << "Сумма чисел от 10 до 19 равна: " << Sum;
    cin.get();
    return EXIT_SUCCESS;
}
```

В цикле **for** счетчиком является переменная *i* (в данном случае целочисленная), которая при инициализации получает начальное значение 10. В качестве **выражение3** используется *i++* , что означает увеличение *i* на единицу. Поэтому после каждой итерации значение *i* возрастает на 1 пока не достигнет 20 (**выражение2:** *i*<20). Именно при этом значение *i* происходит выход из цикла. На каждой итерации выполняется суммирование предыдущего значения переменной Sum с текущим значением *i* и полученный результат присваивается переменной Sum. При достижении переменной *i* значения 20 **выражение2** (*i*<20) становится ложным, поэтому тело цикла (в данном случае суммирование) не выполняется, а просто происходит выход из цикла **for**. Так как переменная *i* объявлялась в конструкции цикла **for** , то при выходе из цикла она становится неопределенной.

Приведем пример вычисления факториала числа, в котором демонстрируется использование в качестве тела цикла блочного оператора.

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
    int a=0;
    unsigned long fact = 1;
    cout << "Введите число: ";
    cin >> a;
    if((a >= 0) && (a < 33))
    {
        for(int i=1; i<=a; i++)
        {
            if(a!=0)
              fact=fact*i;
            else
              fact=1;
        }
    cout << fact << '\n';
    }
    cin.get();
```

```
    return EXIT_SUCCESS;
}
```

Практическая часть

1. Изучите примеры использования оператора **for**, приведенные в теоретической части. Реализуйте приведенные примеры. Изучите работу примеров для нескольких значений задаваемых параметров.

2. Используя оператор цикла **for** выполните задания 1 и 3 **Практического занятия 10** так, чтобы полученная программа позволяла вычислять 10 значений функции в цикле и поясните получаемый результат работы программ.

3. Подготовьте отчет.