Лабораторная работа 7. Управляющие операторы в языке C++. Часть 2

1. ЦЕЛЬ РАБОТЫ

Изучить возможности использования простейших управляющих конструкций в программах на языке C++.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Цикл for. Цикл `for`, наиболее универсальный из всех циклов языка C++, выглядит так:

```
for ([инициализация]; [условие]; [модификация]) оператор
```

Прежде всего выполняется *инициализация* цикла; секция инициализации может содержать любое выражение. Инициализация производится только один раз перед началом работы цикла.

Оценивается выражение *условия.* Если оно истинно (не равно 0), выполняется *оператор* тела цикла; если условие ложно, происходит выход из цикла и управление передается следующему оператору.

После исполнения тела цикла производится *модификация,* после чего управление возвращается заголовку цикла и все повторяется снова. Секция модификации может содержать любое выражение; обычно в ней изменяют значения *управляющих переменных* цикла.

Пример 1. Программа запрашивает с клавиатуры числа a и b, а затем выводит на экран все целые числа из диапазона от a до b.

```
include <iostream>
#include <conio.h>

using namespace std;

int main()
{int i, a, b;
 cout << "Enter a: "; cin >> a;
 cout << "Enter b: "; cin >> b;
 for (i=a; i<=b; i++) cout << i << "  ";
 getch(); return 0;
}
```

Пример 2 использования инструкции `for`. В следующей программе находится сумма всех натуральных чисел от 1 до N.

```
#include <iostream>
#include <conio.h>
using namespace std;

int main()
{ int i,N;
  long int S;
  cout << "N = "; cin >> N;
```

```
    for (S=0,i=1; i<=N; i++) S+=i;
    cout << "Summa = " << S;
    getch();  return 0;
}
```

Цикл while. Это оператор цикла с предусловием. Общий формат оператора цикла `while` следующий:

while (<условие>) <оператор>

Оператор тела цикла будет циклически повторяться до тех пор, пока вычисление «условия» не даст значения **ложь** или **ноль**. Условное выражение вычисляется и проверяется перед выполнением тела цикла. Например, в следующем примере значение переменной `p` увеличивается на единицу в цикле до тех пор, пока не достигнет 10:

```
while (p < 10 ) p++;
```

Оператор `while(1);` реализует бесконечный пустой цикл. Использование в качестве условия константы 1 приводит к тому, что условие повторения цикла остается все время истинным и работа цикла никогда не заканчивается.

Как обычно, одиночный оператор тела цикла можно заменить блоком, заключенным в фигурные скобки:

while *(условие_продолжения){операторы_тела цикла}*

Обратите внимание, что в цикле `while` проверка условия делается *перед* выполнением тела цикла. Если условие изначально ложно, то тело цикла не исполняется вообще, ни одного раза.

Пример 3. Дано натуральное число. Найти первую (старшую) цифру этого числа.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{unsigned int N;
 cout << "N = "; cin >> N;
 while (N>10) N = N/10; // или можно так: N /= 10
 cout << "\nFirst number = " << N;
 getch(); return 0;
 }
```

Цикл do–while. Это оператор цикла с постусловием. Общий формат этого оператора следующий:

do <оператор> **while** (условие);

<оператор> циклически повторяется до тех пор, пока условие истинно (вычисление <условия> не даст 0). Главное отличие этого оператора от оператора `while` состоит в том, что <условие> здесь проверяется не до, а после первого выполнения тела цикла. Гарантировано как минимум одно его выполнение.

Пример 4. Запрашиваются натуральные числа *n* (*n*<1000000) и *m* (*m*<10). Проверить, есть ли в записи числа *n* цифра *m*.

```
#include <iostream>
#include <conio.h>
using namespace std;
int main()
{unsigned int n,m,k;
 bool f;
 cout << "n = "; cin >> n;
 cout << "m = "; cin >> m;
 f=false;
 do {k = n % 10;
     if (k==m) f=true;
     n /= 10;
    } while (n>0) ;
 if (f) cout << "Est'! "; else cout << "Net.";
 getch(); return 0;
 }
```

Любую конкретную структуру повторения, требуемую для решения некоторой задачи, можно реализовать на основе любого из циклов C++, однако всегда какой-то из них подходит к данному случаю наилучшим образом, позволяя написать более ясный и компактный код. Так, если необходимое число итераций цикла известно заранее (как при обработке массива), проще всего применить цикл `for`. Если же число итераций заранее определить нельзя, например, при операциях поиска в списке, применяют цикл `while` или `do...while`.

Операторы прерывания блока. Часто бывает необходимо “досрочно” выйти из некоторого цикла, до того, как будет удовлетворено условие его завершения (говоря точнее, до того, как условие продолжения станет ложным). Например, вы просматриваете массив на предмет поиска заданного значения. Как только нужный элемент массива найден, выполнять цикл далее нет необходимости. Для досрочного завершения циклов в C++ применяются операторы `break` и `continue`.

- Оператор `break` вызывает прерывание ближайшего (самого внутреннего) заключающего его блока `switch`, `while`, `do... while` или `for`. Управление немедленно передается следующему за блоком оператору.
- Оператор `continue` воздействует только на блоки циклов. Он передает управление в конец тела цикла, пропуская, таким образом, все следующие за ним операторы блока. Здесь досрочно завершается не сам цикл, а его текущая итерация.

3. ЗАДАНИЕ НА РАБОТУ

1. Проверить работу программ, приведенных в кратких теоретических положениях.

2. Разработать самостоятельно приложения для решения трех задач по своему варианту.

4. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Задание 1. Ознакомьтесь с теоретическим материалом, приведенным в пункте «Краткие теоретические положения» данных методических указаний.

Задание 2. Создайте новое консольное приложение, как указано выше. Наберите текст программы из примера 1. Проверьте работу приложения. Покажите результат преподавателю.

Задание 3. Аналогично проверьте работу программ из примеров 2 - 4, приведенных в кратких теоретических положениях. Измените программу (**Пример 22**) таким образом, чтобы в ней находилось произведение всех натуральных чисел от 1 до N. Покажите результаты преподавателю.

Задание 4. Проверьте работу примера 3. Измените его так, чтобы программа находила сумму старшей и младшей цифры введенного числа. Покажите результаты преподавателю.

Задание 5. Проверьте работу примера 4. Измените программу таким образом, чтобы определялось, сколько раз встречается цифра m в числе n. Если же цифры m в числе n нет, программа должна сообщить об этом. Покажите результаты преподавателю.

Задание 6. Далее необходимо разработать три приложения по своему варианту (см. таблицу).

Покажите результаты преподавателю. Оформите отчет по работе.

5. ОФОРМЛЕНИЕ ОТЧЕТА

Отчет по работе должен содержать:

- название и цель работы;
- номер варианта;
- для задач по своему варианту – текст задачи, текст кода программы, схема алгоритма программы, результаты выполнения разработанной программы для разных наборов исходных данных; расчеты, выполненные вручную для тех же наборов исходных данных.

6. БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Шилдт Г. C++: базовый курс, 3-е издание. : Пер. с англ. – М.: «Издательский дом «Вильямс», 2005. – 624 с.

7. ПРИМЕРЫ ЗАДАНИЙ

1. Даны целочисленные координаты трех вершин прямоугольника, стороны которого параллельны координатным осям. Найти координаты его четвертой вершины.

2. Заданы координаты левой верхней и правой нижней вершин прямоугольника (x1, y1), (x2, y2). Определить площадь части прямоугольника, расположенной в первой координатной четверти.

3. Дано натуральное число. Найти сумму цифр этого числа.

4. Дано действительное число *a*, натуральное число $n \geq 1$. Найти $S = a(a+1)(a+2)\ldots(a+n-1)$.

5. Дано действительное число *a*, натуральное число $n \geq 1$. Найти $F = \frac{1}{a} + \frac{1}{a+1} + \ldots + \frac{1}{a+n}$