Лабораторная работа №3

Операторы цикла и операторы передачи управления

Цель работы: Изучить синтаксис и работу операторов цикла и операторов передачи управления

Теоретические сведения

Оператор while

Обобщенная форма оператора **while**:

```
while(выражение)
        оператор
```

Вначале вычисляется выражение. Если результат отличен от нуля, тогда выполняется оператор и управление переходит обратно к началу цикла while. Это приводит к выполнению тела цикла while, а именно оператора, который будет выполняться до тех пор, пока выражение не станет равным нулю. Пример:

```
int i=1, sum=0;
while(i<=10)
{
        sum+=i;
        ++i;
}
```

Оператор for

Оператор for – итерационный оператор, обычно используемый с переменной, которая увеличивается или уменьшается. Конструкция оператора for следующая:

```
for(выражение1; выражение2; выражение3)
        оператор
```

Сначала вычисляется выражение1. Обычно выражение1 инициализирует переменную, используемую в цикле. Это выражение вычисляется только один раз. Затем вычисляется выражение2. Если оно отлично от нуля, то выполняется оператор, обрабатывается выражение3, проверяется выражение2 и т.д., пока выражение2 не станет равным нулю. В операторе for могут отсутствовать любое, либо все выражения, но должны оставаться точки с запятой. Пример:

```
int i, sum=0;
for(i=0; i<=10; i++)                // вычисление суммы 10 чисел
        sum+=i;

for(i=1, sum=0;  ; sum+=i++);   // бесконечный цикл
for(;;);
```

Оператор do_while

Конструкция оператора do_while следующая:

```
do
        оператор
while(выражение);
```

Сначала выполняется оператор, затем вычисляется выражение. Если его результат отличен от нуля, то управление переходит обратно к началу оператора do. Например: суммировать положительные числа.

```
int i=0, sum=0;
do
{
        sum+=i;
        scanf("%d",&i);
} while(i>0);
```

Операторы передачи управления

Оператор switch

В программе часто необходимо в зависимости от того или иного результата реализовать одну либо другую группу инструкций. Оператор switch позволяет выбрать одну из нескольких альтернатив. Он записывается в следующем формальном виде:

```
switch(целое выражение)
{
        case   метка1: вариант 1; break;
        case   метка2: вариант 2; break;
                . . .
        case   метка n: вариант n; break;
        default: вариант n+1; break;
}
```

Порядок работы оператора switch следующий:

1. Вычисляется выражение в круглых скобках, стоящих за switch.
2. Выполняется метка case, совпадающая с тем значением, которое было найдено на этапе 1; если ни одна из case не соответствуют этому значению, выполняется метка default; если метки default нет, switch прерывается.
3. Выполнение switch прерывается, когда встречается инструкция break или когда достигается конец switch.

Оператор break

Оператор break используется в операторах цикла for, while, do_while и в операторе switch. Оператор break вызывает выход из самого глубоко вложенного цикла или оператора switch. Например:

```
// выход из цикла по отрицательному значению
int i;
float x;
```

```
for(i=0; i < 10; i++)
{
        printf("\nВведите число\n");
        scanf("%f",&x);
        if( x< 0.0 )
        {
                printf("\nЧисло отрицательное\n");
                break;      //выход из цикла по отрицательному значению
        }
}
```

Оператор continue

Оператор continue заставляет прекратить текущую итерацию цикла и начать следующую.

```
// вычисление суммы положительных чисел
int i;
float x, sum=0;
for(i=0; i < 10; i++)
{
        printf("\nВведите число\n");
        scanf("%f",&x);
        if( x< 0.0 ) continue;
        sum+=x;
}
```

Порядок выполнения работы

1. Изучить краткие теоретические сведения.
2. Составить блок-схему алгоритма.
3. По разработанной блок-схеме алгоритма написать программу.
4. Отладить и выполнить программу.

Варианты заданий

1. Даны натуральное число n и целые числа $a_1,\ldots a_n$. Вычислить количество и сумму тех членов данной последовательности, которые делятся на 5 и не делятся на 7.

2. Используя оператор цикла, написать программу, в которой вычисляется наибольшее положительное целое число n, удовлетворяющее условию: $7n^3 + 81n^2 - 10^6 < 0$. Значение переменной n вывести на печать.

3. Даны натуральное число n и целые числа $a_1,\ldots a_n$. Вычислить количество и сумму положительных, отрицательных и равных нулю членов данной последовательности.

4. Получить все шестизначные счастливые номера. Про целое число n, удовлетворяющее условию 0 <= n <= 999999, говорят, что оно представляет

собой счастливый номер, если сумма трех его первых цифр равна сумме трех его последних цифр; если в числе меньше шести цифр, то недостающие начальные цифры считаются нулями.

5. Дано натуральное число n (n <= 99). Получить все способы выплаты суммы с помощью монет достоинством 1, 5, 10 и 20 коп.

6. Дано натуральное число n. Как наименьшим количеством монет можно выплатить n копеек? Предполагается, что в достаточно большом количестве имеются монеты в 1, 2, 3, 5, 10, 15, 20 и 50 коп.

7. День Учителя ежегодно отмечается в первое воскресенье октября. Дано натуральное число n, означающее номер года. Определить число, на которое в октябре указанного года приходится День Учителя.

8. Рассмотрим некоторое натуральное n (n > 1). Если оно четно, то разделим его на 2, иначе умножим на 3 и прибавим 1. Если полученное число не равно 1, то повторяется тоже действие и т.д., пока не получится 1. До настоящего времени неизвестно, завершается ли этот процесс для любого n > 1. Даны натуральные числа k, n, m (1 < k <n). Поверить, верно ли, что для любого натурального n из диапазона от k до n процесс завершается не позднее, чем после m таких действий.

9. Написать программу, которая выдает все способы представления числа n в виде суммы $n=b_1 + \ldots + b_k$, где k, $b_1,\ldots,b_k > 0$.

10. Даны натуральные числа a, b, c и a1, b1, c1, где a, a1 – означают день, b, b1 – месяц, c, c1 – год. Вычислить количество дней прошедших между двумя датами и количество полных лет.