Lecture 6

Loop Statements. Pre-condition (Pre-test) and Post-condition (Post-test) Loops.

One of the most common programming tasks is to perform the same set of statements (operations) more times. Rather than repeat a set of statements again and again, a **loop** can be used to perform the same set of statements repeatedly.

A *Loop Statement* is a series of instructions that is repeated if a certain condition is true.

Each pass through the loop is called an ***iteration,*** and corresponding algorithm is called **iterative algorithm,** that very often used in computer programming. A loop can be viewed as a cycle repeating a specific number of operations.

Depending on the position of the loop condition there are two types of loops: **pre-condition (pre-test) loop** and **post-condition (post-test) loop.**

Depending on the kind of the loop condition there are also two types of loops: **event-controlled loop** and **count-controlled loop.**

The difference between Pre-test and Post-test Loops

A Pre-test Loop is one in which the block of code, named the body of the loop, is to be repeated while the specified condition is true, and the condition is situated at the beginning of the loop and is tested before the loop is executed.

A Post-test Loop is one in which the block of code is to be repeated while the specified condition is true, and the condition is situated at the end of the loop and is tested after the loop is executed.

Loop Statements in C

C language gives us a choice of three types of loop statements: **while, for** and **do-while**.

The ***while*** and the ***for*** loop statements refer to ***Pre-test loop statements.*** The ***do – while*** loop statement corresponds to ***Post-test loop statement.***

Loop Statement *while*

The ***while*** loop statement structure consists of a block of code and a condition. The loop will repeat as long as the condition is true. The while tests its condition at the top of the loop. Therefore, if the condition is false to begin with, the loop will not execute at all. Because the condition of the loop is tested before the block is executed, the while loop is called a ***pre-test loop statement.***

The syntax of the ***while*** loop statement:

```
initialization;
while(condition)
{ block of statements (body of the loop); }
```

Example of program using ***while*** loop statement:

```
#include<stdio.h>
 int main( )
{ int i = 0, n= 5;
  while (i<n)
  {printf("The value     of i is %d\n", i);
    i  = i + 1; }  // update operation
   return 0; }
```

Loop Statement *for*

The ***for*** loop is usually used when the number of iterations is predetermined (count- controlled loop) . Similar to the while loop, the for loop is a ***pre-test loop.***

Unlike other kinds of loops, such as the while loop, the for loop is often distinguished by an explicit loop counter or loop variable. This allows the body of the for loop (the code that is being repeatedly executed) to know about the sequencing of each iteration.

The syntax for a 'for loop' is the following:

for(expression1 used for **initialization;**expression2 used for **condition**;expression3 for **last operation**)
{block of statements (body of the loop);}

General features of a for loop statement:

- **initialization** sets the initial value of the loop control variable, it is executed only once before looping
- **condition** is a logical expression, it tests the value of the loop control variable. If not given, it is assumed to be true. If condition is false, the loop is terminated
- **last operation** updates the loop control variable

Example of program using ***for*** loop statement:

```
#include <stdio.h>
 int main( )
 {int i,n = 5;
    for(i=0; i<n; i++)
           {  printf("The value of i is %d \n",i);}
    return 0;}
```

Advantage of for loop:

Any for loop is equivalent to some while loop, so the language doesn't get any additional power by having the for statement.

For certain type of problem, a for loop can be easier to construct and easier to read than the corresponding while loop.

The for statement makes a common type of while loop easier to write. It is a very good (perhaps the best) choice for counting loops, very often used for arrays processing.

Do-while loop statement

The ***do - while*** construct consists of a block of code and a condition. First, the code within the block is executed, and then the condition is evaluated at the end. If the condition is true the code within the block is executed again. This repeats until the condition becomes false.

Because do-while loops check the condition after the block is executed, the do-while loop is also known as a **post-test loop**. The syntax is: initialization**;** **do** { body of the loop**;**} **while** (condition);

The do-while loop statement is used when you want to execute the loop body at least once.

Example of program using ***do-while*** loop statement:

```
#include <stdio.h>
 int main( )
{int i= 0, n=5;
 do{
     printf("The value of i is %d \n",i);
     i = i +1;
     }while( i<n) ;
  return 0;}
```

Conclusion

- Instead of performing the same set of statements multiple times use a loop statement to perform the same set of statements repeatedly;
- C language gives you a choice of three loop statements, while, do while and for;
- The ***while loop*** keeps repeating an action while an associated test returns true. This is useful where the programmer does not know in advance how many times the loop will be traversed;
- The ***do while loop*** is similar, but the test occurs after the loop body is executed. This ensures that the loop body is run at least once;
- The ***for loop*** is frequently used, usually where the loop will be traversed a fixed number of times;

Lecturer: Mihail Kulev