Lecture 6. C Loop Control Statements

Loop statements allow you to execute a statement, or a group of statements, multiple times. Types of loops in C programming:

for loop

while loop

do...while loop

Jump statements alter the normal execution sequence of a program:

continue – used to skip some statements inside the loop

break – used to terminate the execution of loop and switch-case statements

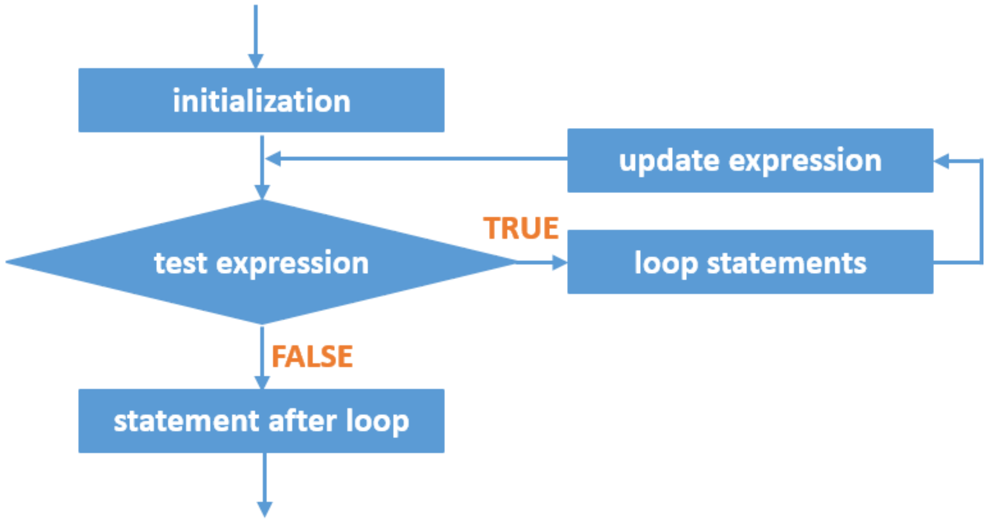
goto – used to jump from one statement to another within a function

for Loop

for loop is usually used when the number of iterations is known.

Syntax

```
for (initialization; test expressions; update expression)
{
    statements;
}
```



for Statement Flowchart

How it works?

1. The **initialization** is executed first and only once to initialize the loop control variable/counter
2. The **test expression** is evaluated
    - If test expression is found to be false (0), the loop statements will not be executed. Program control jumps to the next statement after the “for” loop.
    - If test expression is found to be true, the statements within the loop will be executed AND
      The **update expression** is executed. It will update the loop control variable/counter.
3. Step 2 is repeated until the test expression becomes false or loop is terminated using break statement.

Example:

```
/* program finds sum of all integer numbers from 1 to n,
where n is entered by the user */
#include<stdio.h>

int main(){
    int n, counter, sum=0;
    printf("Enter integer number: ");
    scanf("%d", &n);
    for(counter=1; counter <= n; counter++)
    {
        sum+= counter;
    }

    printf("Sum of integer numbers from 1 to %d = %d", n, sum);
    return(0);
}
```

Important to remember

for is a keyword and must be used only in lower case letters

for statement can be an empty statement Example:

```
for(i=1; i<10; i++)
{
}
```

Result: Variable **i** is incremented

Semicolon at the end of **for** loop is legal C statement and it will produce the same result as **for** loop with no body.

Example:

```
for(i=1; i<10; i++);
{
    statements;
}
```

Result: Variable **i** is incremented

Every **for** statement must include initialization, test expression and update expression. They can be empty but must be separated with semicolon (;).

Example	Explanation

`int i;`
`for(i=1; i<10; i++)`
`{`
`    statements;`
`}`
	Typical **for** loop

`int i=1;`
`for(; i<10; i++)`
`{`
`    statements;`
`}`
	Initialization is not included □
	Loop control variable/counter is initiated before the loop

`int  i;`
`for(i=1; i<10;)`
`{`
`    statements;`
`    i++;`
`}`
	Update expression is not included □
	The update expression is in the loop

`int i=1;`
`for(; i<10;)`
`{`
`    statements;`
`    i++;`
`}`
	Initialization and update expressions are not included □
	Loop control variable/counter is initiated before the loop &
	The update expression is in the loop

`for(i=1, j=1; i<5 && j<=10; i++, j++)`	There are: ● 2 initializations, separated by comma (,) ● Test expression consists of 2 conditions joined together by using logical operator AND (&&) ● 2 update expressions, separated by comma (,)

Nested for Loops

C programming allows using nesting **for** loops – one loop is inside another loop.

Example:

```
int i,j;
for(i=1; i<5; i++)
{
      for(j=1; j<=10; j++)
      {
           printf("i=%, j=%d, i+j=%d", i,j,i+j);
      }
      printf("\n");
}
```

References

Tan, H.H., and T.B. D'Orazio. *C Programming for Engineering & Computer Science*. USA: WCB McGRaw-Hill. 1999. Print.
C4learn. "C for loop - types - C Programming." *Learn Programming Language Step By Step*. N.p., 24 Nov. 2013. Web. 05 Mar. 2017. <http://www.c4learn.com/c-programming/c-for-loop-types/>.
Tutorialspoint.com. "For loop in C." *Www.tutorialspoint.com*. N.p., n.d. Web. 05 Mar. 2017. <https://www.tutorialspoint.com/cprogramming/c_for_loop.htm>.