TUTORIAL

Iteration - for loop

Chapter

The for loop: It is generally available in all programming languages. It allows very flexibility and power to program. The general form of for loop is as below:

```
for(initialization; condition; increment)
{
   // set of statements also known as loop body
}
```

All these initialization, condition, increment and statements are usual C expressions. So the general form can be re-written as:

```
for(expression; expression; expression)
{
   // set of expressions
}
```

The sequence of statements in for loop will be as follows: The initialization is an assignment statement that is used to set the loop control variable. The condition is a relational expression that determines when the loop exits. The increment defines how the loop control variable changes each time the loop is repeated. You must separate these three major sections by semicolons.

First, the initialization will execute, then the condition will be checked for its true value, if condition evaluates to true, the set of statements or loop body will be executed. Then the increment will be executed, and again control moves to the condition statement to decide whether to enter in the loop again or exit the loop and continue with the first statement after the scope of loop.

The following program will print the numbers from 1 to 10.

```c
#include<stdio.h>

int main()
{
  int c;
  printf("Before the for loop. c = %d\n",c);
  for(c=1; c<= 10; c=c+1)
  {
    printf("%d \n",c);
  }
  printf("After the for loop. c = %d",c);
  return 0;
}

```

The variable c generally treated as loop counter will be initialized to 1 at the very beginning. Then the condition will be evaluated which check the value to be less than or equal to 10. As the condition satisfies, the printf statement will execute and after that the increment will happen. After increment, control will move to condition again and continue in this manner till the value of c becomes 11, which makes the condition false, hence control will come out of loop body and execute the printf statement after the loop. Instead of i = i + 1, the statements i++ or i += 1 can also be used.

Each part of for loop has its own significance and used to control the execution of loop body. If we change the above program as the increment of 2 instead of 1 in c, it will behave as:

```c
#include<stdio.h>

int main()
{
  int c=0;
  printf("Before the for loop. c = %d\n",c);
  for(c=1; c<= 10; c=c+2)
  {
    printf("%d \n",c);
  }
  printf("After the for loop. c = %d",c);
  return 0;
}

```

We can also write the loops in reverse direction with decrements in value of loop counter as below: -

```c
#include<stdio.h>

int main()
{
  int c=0;
  printf("Before the for loop. c = %d\n",c);
  for(c = 10; c >= 1; c = c - 1)
  {
    printf("%d \n",c);
  }
  printf("After the for loop. c = %d",c);
  return 0;
}

```

So we can control how to control the flow of execution of statements in our program.

If the condition of for loop will not satisfy in the first check the control will not go in the body of loop for a single time. Instead it will just come out of loop and execute the first statement after the block of for loop. For example, if we change the above program as:

```c
#include<stdio.h>

int main()
{
  int c=0;
  printf("Before the for loop. c = %d\n",c);
  for(c = 10; c < 5; c = c + 1)
  {
    printf("%d \n",c);
  }
  printf("After the for loop. c = %d",c);
  return 0;
}

```

After initializing c to 10, while checking the condition for c to be less than 5, which evaluates to false, hence control directly come out of for loop and execute the printf() statement outside the for loop. As we check the condition to enter the for loop before entering the loop body, for loop is called entry-controlled loop.