LOOPS

REPETITION OR LOOP CONTROL INSTRUCTION

Introduction

If you have to perform an action over and over, often with variations in the details each time, the mechanism which meets this need is the 'loop'.

The versatility of the computer lies in its ability to perform a set of instructions repeatedly. This involves repeating some portion of the program either specified number of times or until a particular condition is being satisfied. This repetitive operation is done through a loop control instruction.

To repeat a part of program there are three methods,

a. Using **WHILE** loop.
b. Using **FOR** loop.
c. Using **DO-WHILE** loop.

While Loop

The while loop can be used if you don't know how many times a loop must run. Here is an example:

```
#include <stdio.h>
int main()
{
    int counter, howmuch;
    scanf ("%d",&howmuch);
    counter = 0;
```

```
	while(counter<howmuch)
	{
		counter++;
		printf("%d\n",counter);
	}
	return 0 ;
}
```

Let's take a look at the example: First you must always initialize the counter before the while loop starts (counter = 1). Then the while loop will run if the variable counter is smaller then the variable "howmuch". If the input is ten, then 1 through 10 will be printed on the screen. A last thing you have to remember is to increment the counter inside the loop (counter++). If you forget this the loop becomes infinitive.

As said before (after the for loop example) it makes a difference if prefix incrementing (++i) or postfix incrementing (i++) is used with while loop. Take a look at the following postfix and prefix increment while loop example:

```
int main()
{
	Int I;
	I = 0;
	while(i++<5)
	{
        printf(“%d\n”,i);
    }
    printf(“\n”);
    I = 0;
    while(++i<5)
    {
        printf(“%d\n”,i);
    }

    return 0;

}
```

Output :-

```
1
2
3
4
5
1
2
3
4
```

- i++ will increment the value of i, but is using the pre-incremented value to test against < 5. That’s why we get 5 numbers.

++i will increment the value of i, but is using the incremented value to test against < 5. That's why we get 4 numbers.

Example 2

Calculation of simple interest:-

```
main ()
{
        int p,n,count;
        float        r,si;
        count = 1;
        while (count <=3)
        {
                printf ("\n enter values of p,n,r");
                scanf("%d%d%f",&p,&n,&r);
                si = p*n*r;
                prinf ("simple interest is =%f",si);
                count = count+1;
        }
}
```

Output:-

```
Enter values of p,n,r 1000,5,13.5
Simple interest is = 675.000000.
```

Flow chart

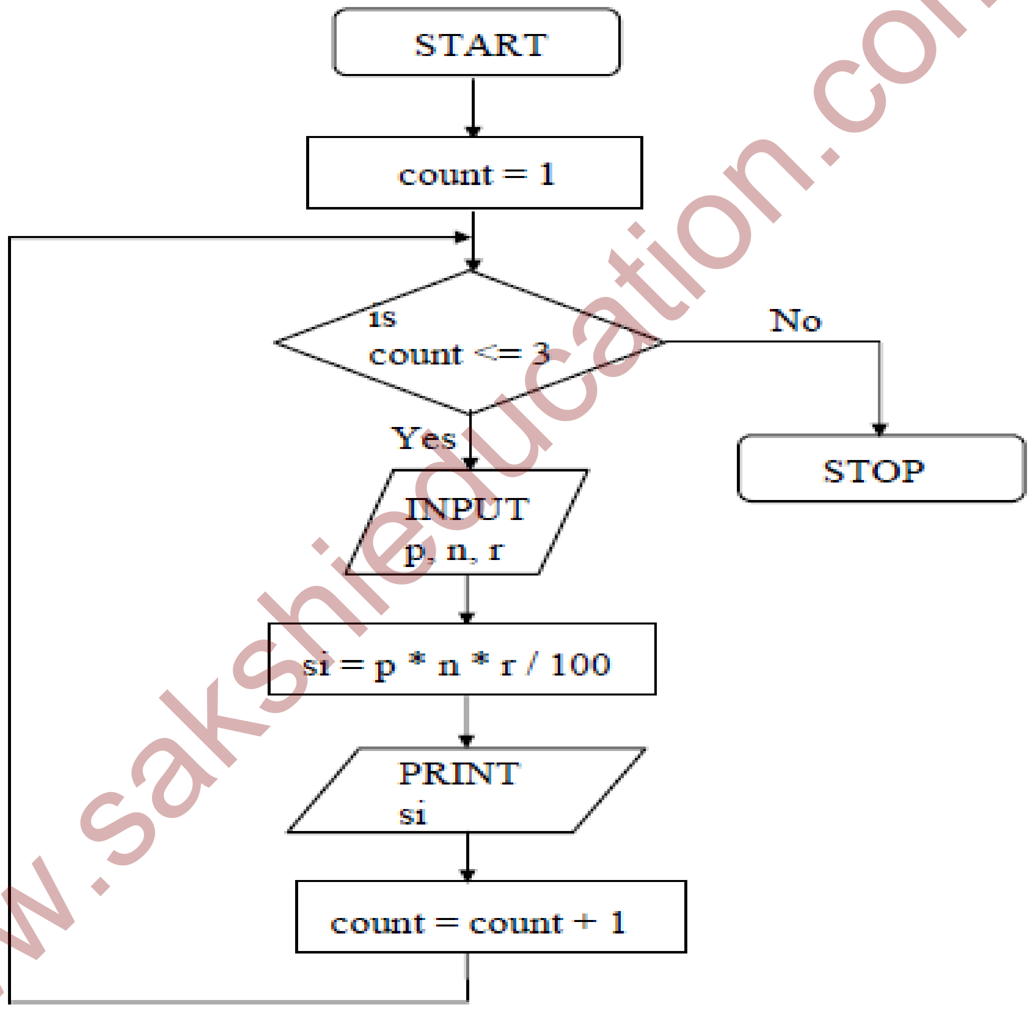

Figure 4.4.2:-Flow chart for the Example 2

Do –While

The “do while loop” is almost the same as the while loop. The “do while loop” has the following form:

Syntax:-

```
do
{
        Do something;
}
while (expression);
```

Do something first and then test if we have to continue. The result is that the loop always runs once.

Example:-

```
#include <stdio.h>
int main()
{
        int counter, howmuch;
        scanf (“%d”,&howmuch);
        counter = 0;
                do
                {
                counter++;
                printf(“%d\n”,counter);
                }
        while(counter<howmuch);
        return 0 ;
}
```

“for” Loop

The “for loop” loops from one number to another number and increases by a specified value each time. The “for loop” uses the following structure:

Syntax:-

```
for(initialize counter ; test counter ; increment counter)
{
      Do this;
     Do this;
     Do this;
}
```

Example:-

The program for calculating simple interest,

```
main ( )
{
        int p, n, count ;
        float r, si ;
        for ( count = 1 ; count <= 3 ; count = count + 1 )
          {
              printf ( "Enter values of p, n, and r " ) ;
              scanf ( "%d %d %f", &p, &n, &r ) ;
              si = p * n * r / 100 ;
              printf ( "Simple Interest = Rs.%f\n", si ) ;
        }
}
```

Flow chart:-

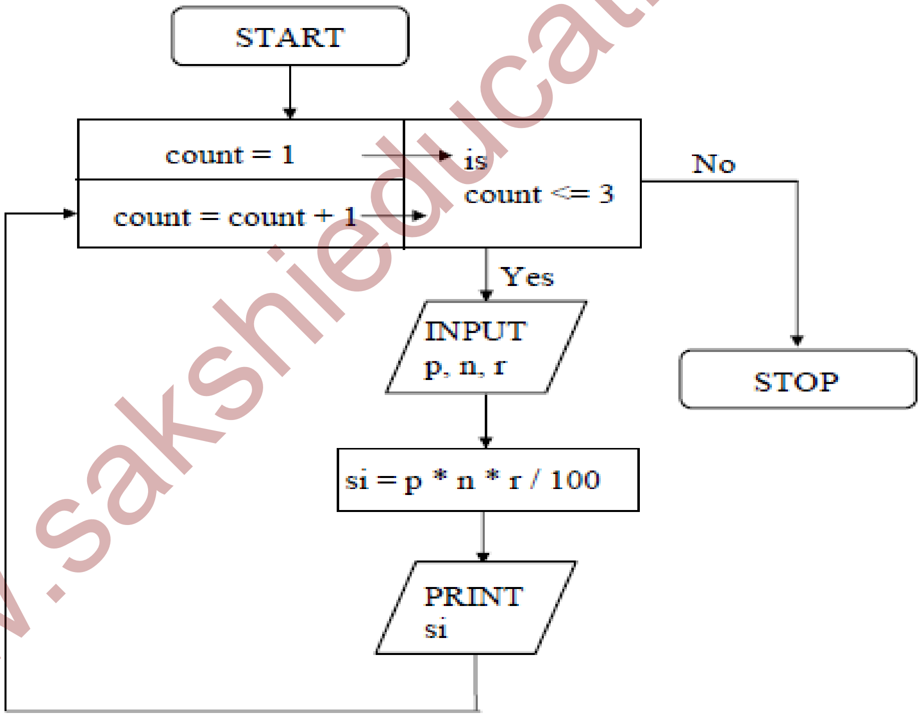

Figure 4.4.4:- Flow chart for the above example

If this program is compared with the one written using **while**, it can be seen that the three steps—initialization, testing and incrementation required for the loop construct have now been incorporated in the **for** statement.

Let us now examine how the **for** statement gets executed:

- When **for** statement is executed for the first time, the value of **count** is set to an initial value 1.
- Now the condition **count <= 3** is tested. Since **count** is 1 the condition is satisfied and the body of the loop is executed for the first time.
- Upon reaching the closing brace of **for**, control is sent back to the **for** statement, where the value of **count** gets incremented by 1.
- Again the test is performed to check whether the new value of **count** exceeds 3.
- If the value of **count** is still within the range 1 to 3, the statements within the braces of **for** are executed again.
- The body of the **for** loop continues to get executed till **count** doesn't exceed the final value 3.
- When **count** reaches the value 4 the control exits from the loop and is transferred to the statement (if any) immediately after the body of **for.**

Note

Difference between for loop and while loop

While loop

Used for looping until a condition is satisfied and when it is unsure how many times the code should be in loop.

For loop

Used for looping until a condition is satisfied but it is used when you know how many times the code needs to be in loop.

TEST

1. **What will be output when you will execute following c code?**

```
#define PRINT printf("Siddhartha");printf(" Singh");
#include<stdio.h>
void main(){
   int x=1;
   if(x--)
       PRINT
   else
       printf("welcome");
}
```

a. siddhartha Singh
b. siddhartha
c. welcome
d. compilation error.

2. **What will be output when you will execute following c code?**

```
#define True 5==5
#include<stdio.h>
void main(){
         if(.001-0.1f)
```

```
            printf("Hello");
             else if(True)
                        printf("IIIT");
            else
                        printf("HYDERABAD");
        }
```

OPTIONS

A. Hello

B. IIIT

C. Hyderabad

D. warning: condition is always true

E. warning :unreachable code

3. What is the output of the following code?

```
#include<stdio.h>
        void main()
        {
                if(sizeof(void))
                 printf ("Hello");
                else
                printf("IIIT");
        }
```

OPTIONS:-

a) Hello

b) IIIT

c) Warning, condition is always false.

d) Compilation error

3. Find the output of the given program below ?

#include <stdio.h>

```
void main()
 {
         int ch;
        printf("enter a value btw 1 to 2:");
         scanf("%d", &ch);
         switch (ch, ch + 1,ch+3,ch+5)
         {
         case 1:
                       printf("1\n");
                break;
         case 2:
                 printf("2");
                 break;
        case 6:
                 printf("6");
                 break;
         default:
                printf("choice not valid");
     }
   }
```

Options :-

(a) **1**
(b) **2**
(c) **1 2**
(d) **6**
(e) **Compilation error**

4.

4 What will be output when you will execute following c code?

```
#include<stdio.h>
void main(){
   int check=2;
   switch(check){
     case 1: printf("A");
     case 2: printf(" B");
     case 3: printf(" C");
     default: printf(" c");
   }
}
```

Options:

a. A

b. B

c. ABC

d. none of these

5. **What will be output when you will execute following c code?**

```
#include<stdio.h>
void main(){
    int const X=0;
    switch(5/4/3){
      case X:  printf("Clinton");
              break;
      case X+1:printf("Gandhi");
              break;
      case X+2:printf("Gates");
              break;
      default: printf("Brown");
    }
}
```

a. Clinton
b. Gandhi
c. Gates
d. compilation error

6 **What will be output of following c code?**

```
#include<stdio.h>
int    main(){
   int x=011,i;
   for(i=0;i<x;i+=3){
       printf("Start ");
       continue;
       printf("End");
```

```
    }
    return 0;
}
```

OPTIONS:-

a. start

b. start end

c. startstartend

d. startstartstart

7. What will be output of following c code?

```
#include<stdio.h>
int i=40;
extern int i;
int main(){
    do{
        printf("%d",i++);
    }
    while(5,4,3,2,1,0);
    return 0;
}
```

Answer ?

8.Write a program to print the following below pattern

```
      1
     2   3
   4   5   6
  7   8   9   10
```

9. Write a program to find whether the given number is Armstrong or not?

10. Write a program to find whether a given number is palindrome or not.?