COMPUTER PROGRAMMING - I

Practice time

WAP to find the roots of a quadratic equation by using if else condition

The **term b^2 - 4ac** is known as the determinant of a quadratic equation.
The determinant tells the nature of the roots.
If determinant is greater than 0, the roots are real and different.
If determinant is equal to 0, the roots are real and equal.
If determinant is less than 0, the roots are complex and different.

If determinant > 0,

$$root1 = \frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b - \sqrt{(b^2 - 4ac)}}{2a}$$

If determinant = 0,

$$root1 = root2 = \frac{-b}{2a}$$

If determinant < 0,

$$root1 = \frac{-b}{2a} + i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

$$root2 = \frac{-b}{2a} - i\frac{\sqrt{-(b^2 - 4ac)}}{2a}$$

Practice time

WAP to find the roots of a quadratic equation by using if else condition

```
#include <stdio.h>
#include <math.h>

int main()
{
    double a, b, c, determinant, root1,root2, realPart, imaginaryPart;

    printf("Enter coefficients a, b and c: ");
    scanf("%lf %lf %lf",&a, &b, &c);

    determinant = b*b-4*a*c;
```

Practice time

```
// condition for real and different roots
    if (determinant > 0)
    {
    // sqrt() function returns square root
        root1 = (-b+sqrt(determinant))/(2*a);
        root2 = (-b-sqrt(determinant))/(2*a);

        printf("root1 = %.2lf and root2 = %.2lf",root1 , root2);
    }
```

Practice time

```
//condition for real and equal roots
    else if (determinant == 0)
    {
        root1 = root2 = -b/(2*a);

        printf("root1 = root2 = %.2lf;", root1);
    }
```

Practice time

```
    // if roots are not real
    else
    {
        realPart = -b/(2*a);
        imaginaryPart = sqrt(-determinant)/(2*a);
        printf("root1 = %.2lf+%.2lfi and root2 = %.2f-%.2fi",
realPart, imaginaryPart, realPart, imaginaryPart);
    }

    return 0;
}
```

Errors to be avoided in if-else

- **if( ) ;**
  printf( );
  else
  printf( );
- **if(code= =1 & flag= =0)**
- **if(i= =5) && (j= =10)** // complete condition in brackets
- **if(i = 5)**
- **if( )**
  printf( );
  else ;
  printf( );

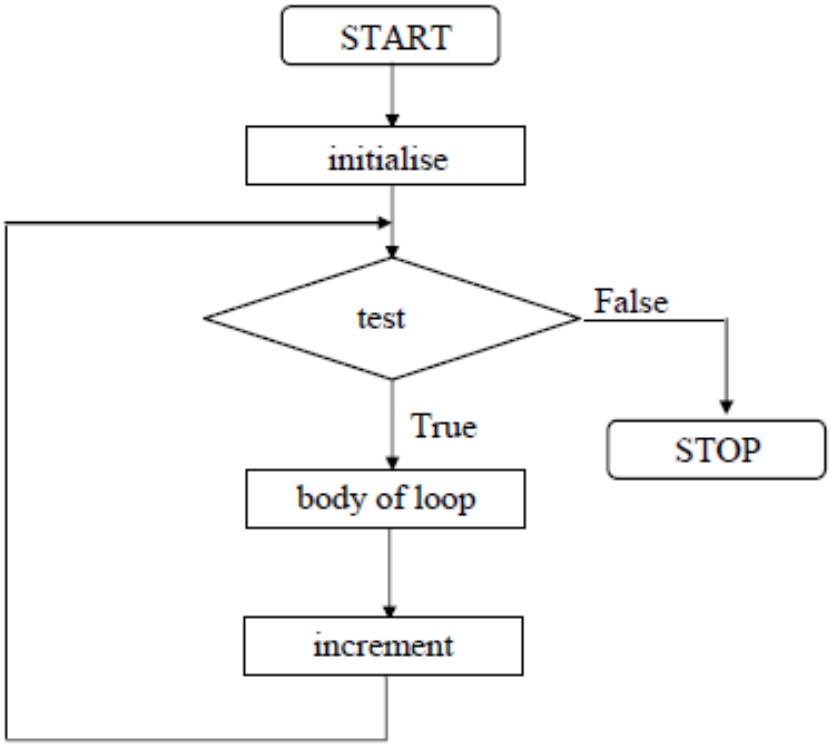
Loops

- We need to repeat some portion of the program either a specified number of times or until a particular condition is being satisfied.
- This repetition is done through loop control instructions
- A. Using a **for** statement
- B. Using a **while** statement
- C. Using a **do-while** statement

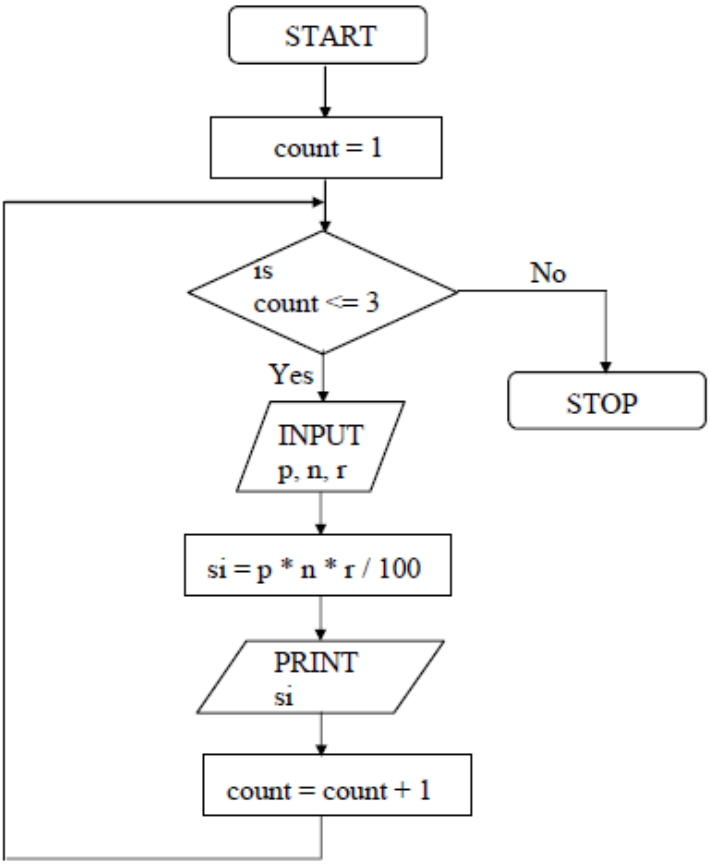
While

- initialise loop counter ;
- while (test loop counter using a condition)
- {
- do this ;
- and this ;
- increment loop counter ;
- }

While loop



While



Calculation of simple interest

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

While loops

- While must test a condition that will eventually become false, otherwise the loop would be executed forever indefinitely.

```
int main()
{
    int i=1;
    while(i<=10)
        printf("%d\n",i);
}
```

While loop

- **Correct form:**

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

}
```

While loop

- Instead of incrementing a loop counter, we can even decrement it.

```
int main()
{
    int i=5;
    while(i>=1)
    {
        printf("%d\n",i);
        i=i-1;
    }

}
```

Quiz time

- What will be output of the following program?

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

}
```

While program

- **WAP to print sum of squares of all integers from 1 to 10**

```
int main()
{
    int num =1;
    int sum =0;
    while(num<=10)
    {
        sum =sum+num*num;
        num =num+1;
    }
    printf("Sum = %d\n",sum);
}
```

Do while

- In real life programming one comes across a situation when it is not known beforehand how many times the statements in the loop are to be executed.

```
do
{
this ;
and this ;
and this ;
and this ;
} while ( this condition is true ) ;
```

Do while

- There is a minor difference between the working of **while** and **do-while** loops.
- This difference is the place where the condition is tested.
- The **while** tests the condition before executing any of the statements within the **while** loop.
- The **do-while** tests the condition after having executed the statements within the loop.

Quiz time

- main( )
- {
- while (4 < 1)
- printf ("Hello there \n") ;
- }

```
main( )
{
do
{
printf ( "Hello there \n") ;
} while ( 4 < 1 ) ;
}
```

Do while

```
int main( )
{ char another ;
int num ;
 do
{
printf ( "Enter a number " ) ;
scanf ( "%d", &num ) ;
printf ( "square of %d is %d", num, num * num ) ;
printf ( "\nWant to enter another number y/n " ) ;
scanf ( " %c", &another ) ;
} while ( another == 'y' ) ;
}
```

Do while

- Enter a number 5
- square of 5 is 25
- Want to enter another number y/n y
- Enter a number 6
- square of 6 is 36
- Want to enter another number y/n y
- Enter a number 7
- square of 7 is 49
- Want to enter another number y/n n

While vs Do-While

Condition tested before executing any of the statements within the while loop.	Condition tested after having executed the statements within the do-while loop.
Will not execute the statements of the condition fails for the first time.	Executes its statements at least once even if the condition fails for the first time itself.

void main( )
{
while(4<1)
printf("Hello Here\n");
}
	void main( )
	{
	do
	printf("Hello Here\n");
	while(4<1);
	}

Practice time

- Program to add numbers until user enters zero

```
#include <stdio.h>
int main()
{
    double number, sum = 0;

do
    {
        printf("Enter a number: ");
        scanf("%lf", &number);
        sum += number;
    }
    while(number != 0.0);

    printf("Sum = %.2lf",sum);

    return 0;
}
```

For loops

- For loops allows us to specify following in single line
- (a) Setting a loop counter to an initial value.
- (b) Testing the loop counter to determine whether its value has reached number of repetition desired
- (c) Increasing the value of loop counter each time program segment within the loop has been executed.

For loops

- The general form is
- for(initialise counter; test counter; increment counter)
- {
- Body of for loop
- }

For loops

- *Initialization* of the *control variables* is done first.
- The value of the control variable is tested using the *test-condition.* If the condition is *true*, the body of the loop is executed; otherwise the loop is terminated and the execution continues with the statement that immediately follows the loop.
- When the body of the loop is executed, the control is transferred back to the **for** statement after evaluating the last statement in the loop.
- Now, the control variable is either incremented or decremented as per the condition.

For Loop

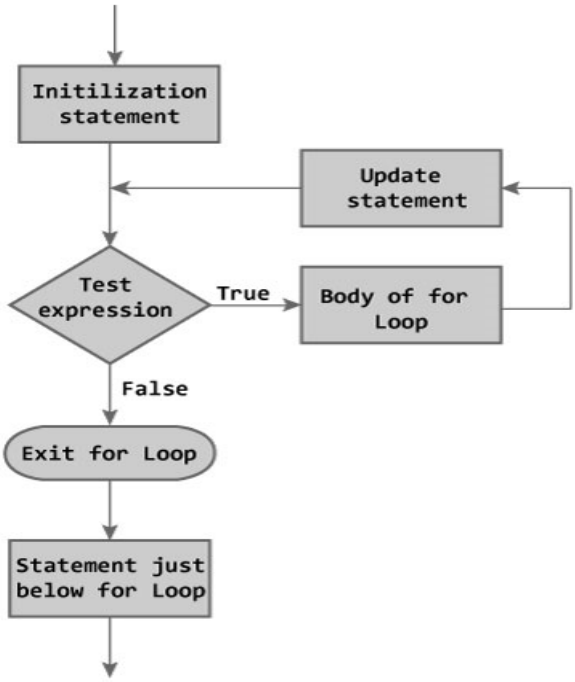


Figure: Flowchart of for Loop

Calculation of simple interest

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

Practice time

- **WAP to print integer from number 1 to 10**

```
#include <stdio.h>
int main()
{
  int Count;
  // display the numbers 1 to 10
  for(Count = 1; Count <= 10; Count++)
          printf("%d ", Count);
  printf("\n");
return 0;
}
```


Practice time: C program to print table of any number

Program to print table

```
/**
 * C program to print table of any number
 */

#include <stdio.h>

int main()
{
    int i, num;

    /* Reads number to print table */
    printf("Enter number to print the table: ");
    scanf("%d", &num);

    for(i=1; i<=10; i++)
    {
        printf("%d * %d = %d\n", num, i, (num*i));
    }

    return 0;
}
```

Output :C program to print table of any number

Output

```
Enter number to print the table of: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

Practice Problems

- Write a C program to read a positive integer n and print sum of all odd integers between 1 and n
- Write a program to compute factorial of a number
- Write a program to compute sum of square of digits.

Practice Problems

- Write a C program to read a positive integer n and print sum of all odd integers between 1 and n

```
int main()
{
    int n,i;
    printf("Enter any integer");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        if((i%2) !=0)
            printf("\n%d",i);
    }

}
```

Practice Problems

- Write a program to compute factorial of a number

```
int main()
 {
    int n,i,prod;
    printf("Enter number");
   scanf("%d",&n);
   prod = 1;
   for(i =1;i<=n;i++)
   {
      prod = prod * i;
   }
   printf("Factorial of %d is %d",n,prod);
 }
```

Practice Problems

- Write a program to compute sum of square of digits.

```
int main()
{
    int n,sum,digit;
    printf("Enter number");
    scanf("%d",&n);
    sum =0;
    while(n!=0)
    {
        digit = n%10;
        sum = sum + digit*digit;
        n=n/10;

    }
    printf("%d",sum);
}
```

- A for loop inside another for loop is called nested for loop

Nested loops

```
int main()
{
    int r, c, sum ;
 for( r = 1 ; r <= 3 ; r++ ) /* outer loop */
    {
        for( c = 1 ; c <= 2 ; c++ ) /* inner loop */
            {
                sum = r + c ;
               printf ( "r = %d c = %d sum = %d\n", r, c, sum ) ;
            }
    }
}
```

Nested loops

- r = 1 c = 1 sum = 2
- r = 1 c = 2 sum = 3
- r = 2 c = 1 sum = 3
- r = 2 c = 2 sum = 4
- r = 3 c = 1 sum = 4
- r = 3 c = 2 sum = 5

Practice time

- Write a program to display
- 1
- 11
- 111
- 1111
- 11111

Practice time

```
int main()
{
    int i,j,n;
    printf("Enter number");
    scanf("%d",&n);
    for(i =1;i<=n;i++)
    {
        for(j=1;j<=i;j++)
        {
            printf("1");

        }
        printf("\n");
    }
}
```

Nested loops

- WAP to display
- A
- A B
- A B C
- A B C D
- A B C D E

```
nt main()
{
    int i,j,n;
     printf("Enter number");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        for(j=0;j<=i;j++)
        {
             printf("%c",j+65);
        }

            printf("\n");
    }

}
```