Decision Making And Branching

- for STATEMENT

The ***for*** loop can execute a block of code for a fixed or given number of times. i.e.
A ***for*** loop construct executes a group of statements a specified number of times.
An optional keyword can be used to prematurely exit the loop before specific numbers of times.
The syntax of ***for*** loop statement is

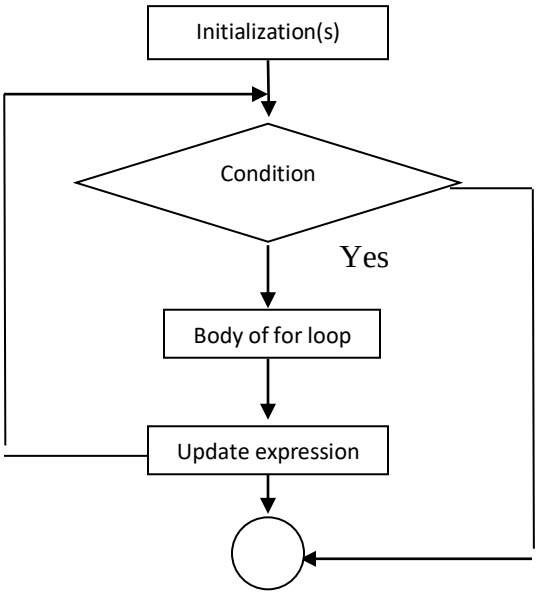
```
for(initialization_expression; condition_check_expression; update_expression)
{
    statement sequence;
  }
```

// General form

```
for(i=0;i<=n;i++)
{
Statement 1;
}
```

Here i is an integer, loop starting form 0 to check the condition up to n and increment by 1. Using for loop, a block of statement code is executed an infinite or finite time depending on condition_check_expression.

Flow chart of *for* loop



i) Print the ineteger numbers from 1 to 10 using for loop

```
#include<stdio.h>
#include<conio.h>
int main()
{
int i;
for(i=1; i<=10;i++)
printf("%d \n", i);
return 0;
}
```

OUTPUT

1
2
3
4

5
6
7
8
9
10

ii) **write a program to read 5 numbers and print their sum**

```
#include <stdio.h>
#include<conio.h>
int main()
{
int n,i,sum=0;
printf("Enter the value of n.\n");
scanf("%d",&n);
for(i=0;i<=n;i++)  //for loop terminates if i>n
{
sum=sum+i;    /* this statement is equivalent to sum=sum+i */
}
printf("Sum=%d",sum);
return 0;
}
```

output:
entered number:5
sum =15

iii) The number is perfect or not.

Any number can be the perfect number in C if the sum of its positive divisors excluding the number itself is equal to that number.
For example, 6 is a perfect number in C because 6 is divisible by 1, 2, 3, and 6. So, the sum of these values is 1+2+3 = 6 (Remember, we have to exclude the number itself. That's why we haven't added 6 here). Some of the perfect numbers are 6, 28, 496, 8128, and 33550336, etc.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n,i,per=0;
printf(" Enter the no to be checked ");
```

```
scanf("%d",&n);
for(i=1;i<n;i++)
{
if(n%i==0)
per=per+i;
}
if(per==n)
printf("\n It is a Perfect no");
else
printf("\n It is not a perfect no");
return 0;
}
```

Output

Enter the no- 6
This is perfect number.
Enter the number – 9
This is not perfect number

iv) Calculate Factorial of a number in C.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int i,fact=1,number;
printf("Enter a number: ");
scanf("%d",&number);
for(i=1;i<=number;i++)
{
fact=fact*i;
}
printf("Factorial of %d is: %d",number,fact);
return 0;
}
```

OUTPUT
Enter a number: 5Factorial of 5 is: 120

v) Check whether a number is prime or not.

A prime number is a positive integer that is divisible only by 1 and itself. For example: 2, 3, 5, 7, 11, 13, 17

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n,i;
printf(" Enter the no to be Checked");
scanf("%d",&n);
for(i=2; i<n;i++)
{
if(n%i==0)
break;
}
if(i==n)
printf("\n Entered no is Prime no. ");
else
printf("\n Enterd no is not a Prime no. ");
return 0;
}
```

Output

```
Enter the number- 8
This is not prime number
Enter the number- 5
This is prime number
```

vi) Reverse of a number

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n,rev=0;
printf("\n Enter the no to be reversed ");
scanf("%d",&n);
while(n!=0)
{
rev=rev*10;
rev=rev+(n%10);
n=n/10;
}
printf("\n The Reverse of the no is %d", rev);
return 0;
}
```

Output
Enter the number-568
Reverse number is-865

vii) Check whether a number is palindrome or not in c.

An integer is a palindrome if the reverse of that number is equal to the original number.

```
#include<stdio.h>
#include<conio.h>
int main()
{
int n, rem, temp, pal=0;
printf("enter the number");
scanf("%d",&n);
temp=n;
while(temp!=0)
{
rem=temp%10;
pal=pal*10+rem;
temp=temp/10;
}
if(pal==n)
printf("\n the number is a palindrom",n);
else
printf("\n the number is not palindrom",n);
return 0;
}
```

Output
Enter the number- 121
This is palindrome number
Enter the number- 345
This is not palindrome number

viii) check wheather a number is armstrong or not.

In the case of an Armstrong number of 3 digits, the sum of cubes of each digit is equal to the number itself. For example, 153 is an Armstrong number because

```
153 = 1*1*1 + 5*5*5 + 3*3*3
```

```
#include <stdio.h>
#include<conio.h>
int main()
{
int n, temp, rem, arm=0;
printf("Enter a positive  integer: ");
scanf("%d",&n);
temp=n;
while(temp!=0)
{
rem=temp%10;
arm+=rem*rem*rem;
temp=temp/10;
 }
if(arm==n)
printf("%d is an Armstrong number.",n);
else
printf("%d is not an Armstrong number.",n);
}
```

Output

```
Enter the number- 153
This is Armstrong number
Enter the number-235
This is not Armstrong number.
```

ix) Write a program of sum of digits.

For example, if the input is 98, the variable sum is 0 initially
98%10 = 8 (% is modulus operator, which gives us the remainder when 98 is divided by 10).
sum = sum + remainder
so sum = 8 now.
98/10 = 9 because in C language, whenever we divide an integer by another one, we get an integer.

9%10 = 9
sum = 8 (previous value) + 9
sum = 17
9/10 = 0.
So finally, n = 0, the loop ends; we get the required sum

```
#include<stdio.h>
 #include<conio.h>
int main()
{
int n,sum=0,r;
printf("Enter a number: ");
scanf("%d",&n);
while(n!=0)
{
r=n%10;
sum=sum+r;
n=n/10;

}
printf("Sum of digits of number:  %d",sum);
return 0;
}
```

Output

Enter the number 78
The sum of digits is-15

ix) a. Compute the summation for the following series

2+4+6+8……+n

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,sum=0,i;
clrscr();
printf("Enter the last term------->");
scanf("%d",&n);
```

```
printf("\n");
for(i=1;i<=n;i++)
{
if(i%2==0)
{
printf("%d\t",i);
sum=sum+i;
}
}
printf("\n\nSum of the series-----> %d",sum);
getch();
}
```

Output

```
Enter the last term-------->10
2       4       6       8       10
Sum of the series-----> 30
```

x) b. To compute the summation for the following series

1!+ 2!+3!+4!……+n!

```
#include<stdio.h>
#include<conio.h>
void main()
{
int n,sum=0,fact=1,i;
clrscr();
printf("Enter the number term less than 8-------->");
scanf("%d",&n);
printf("\n");
for(i=1;i<=n;i++)
{
fact=fact*i;
printf("\nFactorial of--->%d is---->%d",i,fact);
sum=sum+fact;
}
printf("\n\nSum of the series-----> %d",sum);
getch();
}
```

Output

```
Enter the number term less than 8-------->7
Factorial of--->1 is---->1
```

```
Factorial of--->2 is---->2
Factorial of--->3 is---->6
Factorial of--->4 is---->24
Factorial of--->5 is---->120
Factorial of--->6 is---->720
Factorial of--->7 is---->5040

Sum of the series-----> 5913
```

xi) c. To compute the summation for the following series
$1+x+x^2/2!+x^3/3!.....+x^n/n!$

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int n,sum=0,fact=1,i,x,p;
clrscr();
printf("Enter the number term less than 8------->");
scanf("%d",&n);
printf("\n Enter the value of X----->");
scanf("%d",&x);
printf("\n");
for(i=1;i<=n;i++)
{
p=pow(x,i);/*This is a standrad library function calculates a value raised to a power*/
fact=fact*i;
printf("\nFactorial of--->%d is---->%d   p/fact---->%d",i,fact,p/fact);
sum=sum+p/fact;
}
printf("\n\nSum of the series-----> %d",sum);
getch();
}
```

Output

```
Enter the number term less than 8------->7
 Enter the value of X----->3
Factorial of--->1 is---->1   p/fact---->3
Factorial of--->2 is---->2   p/fact---->4
Factorial of--->3 is---->6   p/fact---->4
```

Factorial of--->4 is---->24 p/fact---->3
Factorial of--->5 is---->120 p/fact---->2
Factorial of--->6 is---->720 p/fact---->1
Factorial of--->7 is---->5040 p/fact---->0

Sum of the series-----> 17

- ***NESTED for LOOP***

Using one or more for loops within a for loop is called nested for loop. Nested for loop is used whenever handling multidimensional arrays which are often the elements of matrices. For any matrix operation one will have to use nested for loop.Following are the rule of handling a nested for loop.

1. Inner most loops are first executed under the condition of outer loop.
2. After completing inner most loops the control goes to immediate outer loop and so on.
3. Control may be asked to come out of a loop at any stage, but it can not be entered again at the same point where it has come out.
4. In a nested for loop two loops can not cross each other.
5. Loops may end on the same line or inner loop may be completely ended inside the outer loop.

C supports nesting of loops in C. **Nesting of loops** is the feature in C that allows the looping of statements inside another loop.
Any number of loops can be defined inside another loop, i.e., there is no restriction for defining any number of loops. The nesting level can be defined at n times. You can define any type of loop inside another loop; for example, you can define '**while**' loop inside a '**for**' loop.

Syntax of Nested loop

```
Outer_loop
{
   Inner_loop
  {
       // inner loop statements.
  }
     // outer loop statements.
}
```

Nested for loop

The nested for loop means any type of loop which is defined inside the 'for' loop.

```
for (initialization; condition; update)
{
   for(initialization; condition; update)
   {
        // inner loop statements.
```

```
    }
   // outer loop statements.
}
```

Program

```
#include <stdio.h>
int main()
{
  int n;// variable declaration
  printf("Enter the value of n :");
  // Displaying the n tables.
  for(int i=1;i<=n;i++)  // outer loop
  {
     for(int j=1;j<=10;j++)  // inner loop
     {
        printf("%d\t",(i*j)); // printing the value.
     }
     printf("\n");
}
```

Explanation of the above code

- First, the 'i' variable is initialized to 1 and then program control passes to the i<=n.
- The program control checks whether the condition 'i<=n' is true or not.
- If the condition is true, then the program control passes to the inner loop.
- The inner loop will get executed until the condition is true.
- After the execution of the inner loop, the control moves back to the update of the outer loop, i.e., i++.
- After incrementing the value of the loop counter, the condition is checked again, i.e., i<=n.
- If the condition is true, then the inner loop will be executed again.
- This process will continue until the condition of the outer loop is true.

vi) a. Print the following pattern

```
*
* *
* * *
* * * *
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,i,j;
clrscr();
printf("Enter the number row ------->");
scanf("%d",&r);
printf("\n");
for(i=1;i<=r;i++)
{
for(j=1;j<=i;j++)
{
printf(" * ");
}
printf("\n");
}
getch();
}
```

Output

```
Enter the number row ------->4
 *
 * *
 * * *
 * * * *
```

vi) b. Print the following pattern

```
# # # #
  # # #
    # #
      #
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,i,j,k;
```

```
clrscr();
printf("Enter the number row ------->");
scanf("%d",&r);
printf("\n");
for(i=1;i<=r;i++)
{
for(j=0;j<=i-1;j++)
{
printf(" ");
}
for(k=0;k<=r-i;k++)
{
printf("#");
}
printf("\n");
}
getch();
}
```

Output

```
Enter the number row ------->5
 #####
  ####
   ###
    ##
     #
```

vi) c. Print the following pattern

10 1 0
0 1 0
1 0
1

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,i,j,k;
clrscr();
printf("Enter the number row ------->");
```

```
scanf("%d",&r);
printf("\n");
for(i=1;i<=r;i++)
{
for(j=0;j<=i-1;j++)
{
printf(" ");
}
for(k=0;k<=r-i;k++)
{
if((i+k+1)%2==0)
printf("1 ");
else
printf(“0 “);
}
printf("\n");
}
getch();
}
```

Output

```
Enter the number row ------->4
 1 0 1 0
   0 1 0
     1 0
       1
```

vi) d. Print the following pattern

```
      *
     * *
    * * *
   * * * *
  * * * * *
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,i,j,k;
clrscr();
printf("Enter the number row ------->");
scanf("%d",&r);
```

```
printf("\n");
for(i=1;i<=r;i++)
{
for(j=1;j<=r-i;j++)
{
printf(" ");
}
for(k=1;k<=i;k++)
{
printf(" *");
}
printf("\n");
}
getch();
}
```

Output

```
Enter the number row ------->5

      *
     * *
    * * *
   * * * *
  * * * * *
```

vi) e.Print the following Pattern

```
A B C D E E D C B A
A B C D       D C B A
A B C           C B A
A B               B A
A                   A
```

```
#include<stdio.h>
#include<conio.h>
void main()
{
int r,i,j,k,t=65,m;
printf("Enter the number row ------->");
scanf("%d",&r);
printf("\n");
for(i=1;i<=r;i++)
```

```
{
for(j=r;j>=i;j--)
{
printf("%2c",t);
t++;
}
m=t-1;
t=65;
for(k=2;k<=2*i-1;k++)
{
printf("  ");
}
for(j=r;j>=i;j--)
{
printf("%2c",m);
m--;
}
printf("\n");
}
getch();
}
```

Output

```
Enter the number row ------->5
A B C D E E D C B A
 A B C D      D C B A
 A B C          C B A
 A B              B A
 A                  A
```

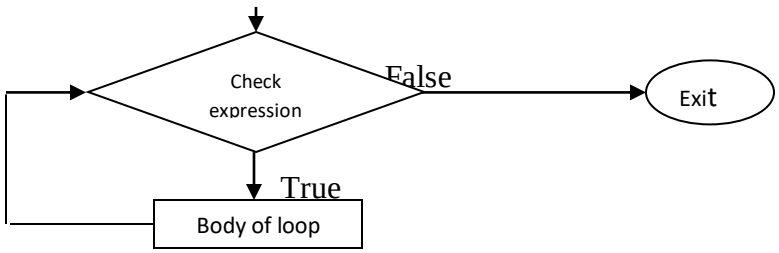
- ***while STATEMENT***

  while loop is similar type of **for** loop. It contains a check expression but no initialization and updated expressions. Loop variable should be updated inside the body of the **while** automatically.

The syntax of **while** loop

```
while(check expression)
{
statement sequence;
}
```

Flow chart of *while* statement



// Using WHILE LOOP

```
#include<stdio.h>
#include<conio.h>
void                                                                    main()
{
int i=1;
while(i<=10)
{
printf("%d                                    \n",                    i);
i++;
}
getch();
}
```

OUTPUT

1
2
3
4
5
6
7
8
9
10

//A program to reverse a number

```
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
int n,i,b,rev=0;
clrscr();
printf("Enter the Number");
scanf("%d",&n);
for(i=n;i>=1;i=i/10)
{
b=i%10;
rev=rev*10+b;
}
printf("The reverse number of the given number--->%d",rev);
getch();
}
```

Output

Enter the Number------>456
The reverse number of the given number--->654

- ***do while STATEMENT***

In **while** and **for** loops the condition for checking is given before entering in the loop body, where as, in **do while** loop the condition for checking is given after the **do** loop body. For this reason in case of **while** and **for** loop, loop body would not be executed at all if the check expression is false. But in **do while** loop, loop body will be executed at least once and then checking of condition is made. If given condition is false loop body would not be repeated. **while(1)** is called true loop and looping will continue till any instruction for termination is encountered in the program.

Syntax of the ***do while***

```
do
{
statement sequence;
}
while(check expression);
```

//Using DO-WHILE LOOP

```
#include<stdio.h>
#include<conio.h>
void                                                                    main()
{
int                                                                       i=1;
do
{
printf("%d                                    \n",                        i);
```

```
i++;
}
while(i<=10);
getch();
}
```

Output:

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

// A menu program using do-while

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
void main()
{
int ch,ch1;
int a,b,result;
clrscr();
printf("\nEnter the value of A and B--------");
scanf("%d%d",&a,&b);
do
{
printf("\n**********MENU**********");
printf("\n        1.ADDITION");
printf("\n        2.SUBTRACTION");
printf("\n        3.MULTIPLICATION");
printf("\n        4.DIVISION");
printf("\n          5.EXIT");
printf("\nEnter your choice");
scanf("%d",&ch);
switch(ch)
```

```
{
case 1: result=a+b;
        break;
case 2: result=a-b;
        break;
case 3: result=a*b;
        break;
case 4: result=a/b;
        break;
case 5: return;
default: printf("\n Wrong choice");
}
printf("\nThe Result is-------%d",result);
}while(1);
getch();
}
```

Output

```
Enter the value of A and B--------12
**********MENU**********
    1. ADDITION
    2. SUBTRACTION
    3. MULTIPLICATION
    4. DIVISION
    5. EXIT
Enter your choice--1
The Result is-------18
```