# UNIT-3

**Iteration and loops**:

- Use of while, do while and for loops
- Multiple for loop variables
- Use of break and continue statements.

**Functions**:

- Introduction, types of functions
- Functions with array
- Passing parameters to functions
- Call by value and Call by reference
- Recursive functions

## Short Question & Answers (Module1 : Loops and Iterations)

**Ques 1: Can we use continue statement within the body of switch statement like break?**

Ans: No,a ***continue*** can appear only in or as a loop body. A switch statement is a branching statement is a branching statement and not a looping statement .

**Ques 2. Differentiate between a while loop and do –while loop.**

Ans :

| S.NO | While loop | do while loop |
|---|---|---|
| **1.** | A while loop is used to execute and repeat a statement block depending on a condition which evaluates at start of the loop. | A do while loop is used to execute and repeat a statement block depending on a condition which evaluates at the end of the loop. |
| **2.** | A variable value is initialized at the beginning or before the loop and used in condition. | A variable value is initialized before the the loop or assigned inside the loop and used in condition. |
| **3.** | The statement block will not be executed when the value of the condition is false. | The statement block will not be executed when the value of the condition is false. Block is executed at least once irrespective of the value of condition. |
| **4.** | Synatx : Statement x ; while (condition) { Block of statements under execution ; } Statements y ; | Statement x ; do { Block of statements under execution ; } while (condition); Statement y; |
| **5.** | Entry controlled loop | Exit controlled loop. |

**Ques 3**. **Can the while statement end with a semicolon?**

Ans: By definition, the while statement does not end with a semicolon. However, it's legal in C to put a Semicolon right after the while statement like this: **while (expression);** which means there is a **null Statement controlled by the while statement**. Remember that the result will be quite different from what you expect if you accidentally put a semicolon at the end of the while statement.

**Ques 4. Write a program to generate even number series from 1 to 50 , using while loop.**

**Ans :**
```
# include <stdio.h>
 # include <conio.h>
Void main()
{
    int i , n ;   i= 2, n= 50 ;
   while (i<= n)
      {
            printf (“ %d \n ” , i);
            i = i + 2 ;    //  i +=2
       }  // End of while loop.
             getch();
   }// End of main()
```

**Ques 5. For every use of a for loop, we can implement an equivalent while loop. So when should we prefer to use these in different programs?**

Ans: A ***while loop*** is preferred over ***a for loop*** when the number of iterations to be performed is not known in advance. The termination of the while loop is based on the occurrence of some particular condition, i.e specific sentinel value. Whereas the ***for loop*** is preferred when no. of iterations to performed are known beforehand .

**Ques 6. What will be the output of the following programs?**

```
 #include<stdio.h>
#include<conio..h>
  void  main()
  {
    int c=5;
   do
     {   printf( “Hello”) ;
          c++;
    }   while (c<5) ;
         return  0; }
```

**Ans : Output “ Hello” will be printed.**

**Ques 7. Explain continue statement with an example .**

**Ans : Continue** is a keyword . When we want to take control to the beginning of loop, by ignoring the statements without their execution inside the loop , then continue statement is used.

When continue is placed inside any loop, control automatically passes to the start of loop. ***Continue is generally associated with an If statement.***

**Example :**
```
void main()
        { int i=1 ;
         while ( i<=10)
           {
             if ( i = = 5)
               continue ;
            printf(“ \t %d ” , ;);
                ++ i;
           } return 0;
        }  // End of main
```

**Output : 1 2 3 4 5 6 7 8 9 10**

**Ques 8.What is the output of following code snippet?**
```
void main ()
    {
      int i , j ;
     for ( i=1 ; i<3 ; i++)
       for ( j=1 ; j < 4 ; j++ )
         {   if ( j==2)  continue ;
             printf (“%d %d ”  , i ,j ) ;
         }
    }
```

**Ans: OUTPUT: 1 1 , 1 3 , 2 1 , 3 3**

## Long Question & Answers (Module1 : Loops and Iterations)

**Ques 9. What do you mean by iteration? Explain entry and exit controlled loops with example. Give the classification of loops also.**

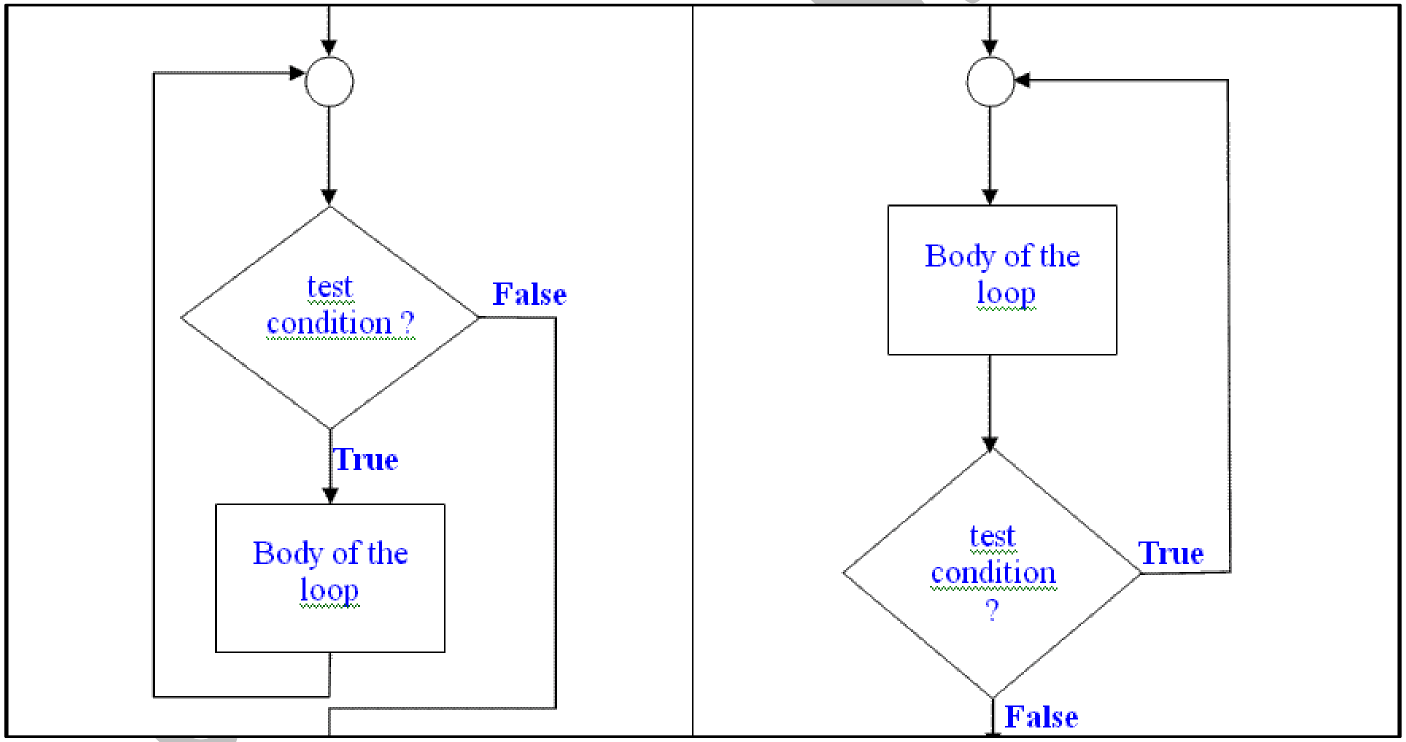
Ans : The versatility of the computer lies in its ability to perform a set of instructions repeatedly. Some specified portion of code is repeated n times , till a particular condition is satisfied. This repetitive operation is done through a loop control instruction.

A program loop is the combination of body of the loop and a control statement. Control statement tests certain conditions and then directs the repeated execution of the statements contained in the body of the loop.

Depending on the position of control statement inside the loop. Two types of control structures are defined

(a) **Entry controlled loop** **(b) Exit controlled loop**.

In Entry controlled loop pre testing is done. In exit controlled loop post testing is done.



**Entry controlled loop** **Exit controlled loop**

**Steps of the Looping Process:**

(a) Declaration and initialization of a condition variable.
(b) Execution of the statements inside the loop.
(c) Test for a specified value of condition variable for loop execution.
(d) Updating (increment/decrement) of condition variable.

**Types of loop statements in C language are : (i) while loop (ii) do while loop (iii) for loop.**

**Classification of Loops :** Based on the nature of control variable and the kind of value assigned to it f**or testing the control expression, the loops may be classified as :**

**(A) Counter controlled loops (B) Sentinel controlled loops.**

**Counter controlled loops:**
(i)In this the number of iterations to be performed is known in advance.
(ii)These loops used a counter variable called loop counter to keep track of iterations inside the loop.
(iii)It starts with the initial value of the loop counter and terminates when the final value of loop counter is reached.
(iv)Also known as definite repetition loops , because iteration is in fixed number of times.

**Sentinel controlled loops:**

(i) In this number of iterations to be performed are not known in advance
(ii)The execution and termination of loop depends on a sentinel value.
(iii) If sentinel value is true loop body will be executed otherwise not.
(iv)Also known as indefinite controlled loop.

**Ques10. How is for loop executed? Give its syntax and example. What are additional features of for loop?**
Ans : for loop is a entry controlled loop. It provides a most concise structure in program as compared to while loop and do while- loop. It is used when number of iterations are known before hand in the program.

```
Syntax of for loop :     for  ( initialization  ;  test condition ; update  variable)
                              {    body of the loop }
```
The execution is as given below:
(a) Initialization of the control variables is done first.(loop control variable).
(b) The test condition can be any expression with relational operators( x<= 10, m= = n etc) or any logical expression (x || y , !a && t etc.). This determines when the loop will terminate.
(c) If test condition is true body of loop executes, else loop is terminated.

(d) After execution the control is transferred back to the for statement after evaluating the last statement in the loop.
(e) Now control variable is updated again, and test test condition is checked for true/false.
(f) If test condition is true body of loop is executed again , otherwise exit from the loop occurs.

## Additional Features of for loop

(A) **Multiple Initialization**: In this more than one variable can be initialized at a time in the for statement. E.g : p = 1;
for ( n = 0 ; n < 17 ; n++), in this loop we can reconstruct as follows:

: **for ( p = 1 , n = 0 ; n < 17 ; n++) // variable p is also initialized inside for statement.**

(B) **Multiple Increment:** In this the increment section may also have more than one part.

```
for( n=1 , m= 50 ; n <= m ; n ++ , m- -)
   {
       p = m/n ;
       printf ( " %d %d %d \n " , n , m , p) ;
   }
```

(C) **Variation in test condition:** for loop may have any compound relational expression or logical expression for testing the loop condition, rather than depending on a single loop control variables.

```
sum = 0 ;
for( i=1 ; i <20  && sum < 100 ; ++ i)
 {
      sum = sum + i ;
      printf(" %d " , sum) ;       // Here i is counter variable
  }                                //  sum is sentinel variable.
```

**Loop will execute as long as both the conditions are true**

**Ques 10. (a) Explain break and continue statement with examples.**
**(b) What are the properties of nested loops?**

**Ans : (a) Break statement:**

(i) In C break statement is used to terminate the execution of the nearest enclosing loop in which it appears.
(i) Used with for , while and do-while loops.
(ii) When compiler encounters the break statement , the control passes to the statement That follows the loop in which break statement appears

(iii) Generally break is associated with if statement.

| **Syntax of break with while loop** | **Syntax of break with for loop** |
|---|---|
| while (test condition)<br>{<br>……..<br>………<br>if( condition)<br>break ;<br>…….<br>……..<br>}<br>Transfers control out of the loop while. | for( ……)<br>{<br>……..<br>………<br>if( condition)<br>break ;<br>…….<br>……..<br>}<br>Transfers control out of the for loop. |
| Example :<br># include<stdio.h><br>void main()<br>{<br>int i=1;<br>while( i<=10)<br>{<br>if(i= = 5)<br>break;<br>printf( " \n %d " , i);<br>++i;<br>}<br>} | Example :<br># include <stdio.h><br>void main()<br>{<br>int i=1;<br>for ( ; ;)<br>{<br>printf ( " \n %d " , i);<br>if( i= = 5)<br>break;<br>}<br>} |

(B) **Continue statement :**

(i) This keyword is used inside the loop body

(ii) When compiler encounters a continue statement, then rest of the statements in loop are skipped and control is unconditionally transferred to the start statement of the loop.

(iii) When it is present inside the nested loop, it only terminates current iteration of the nearest enclosing loop.

(iv) There is no limitation on the number of continue statements that can be present inside the loop

| **Syntax of continue with while loop** | **Syntax of continue with for loop** |
|---|---|
| while (test condition) { …….. :……… if( condition) continue ; ……. …….. }an be udes to control Transfers control out of the loop while. | for( ……) { …….. ……… if( condition) continue ; ……. …….. } Transfers control out of the for loop. |
| Example : # include<stdio.h> void main() { int i=1; while( i<=10) { if(i%2 = = 0) continue ; printf( " \n %d " , i); ++i; } } | Example: # include<stdio.h> void main() { int i; for ( i=1; i<=10 ; i++) { if(i%2 = = 0) continue; printf( " \t %d " , i); } } **Output : 1 3 5 7 9** |

Nested Loops :. Another outer loop is used to control the number of times that a whole loop is repeated. Loops that can be placed inside other loop. This feature can work with any loop but it is mostly useful in for loop because of simplicity of for loops. A for loop can be used to control the number of times that a particular set of statements will be execute . Indentation of nested loops should be proper. Loops can be nested to any level .

| **Example : void main()** | **Output :** |
|---|---|
| **{ int i,j; for(i=1; i<3; i++) for(i=1; j<4; j++) { if(j= = 2 ) continue ; printf ( " %d %d \n" , i,j ) ; }}** | **1 1<br>1 3<br>2 1<br>2 3** |

## Module 1: Programs on Iterations & Loops

1. **Write a C program to find the factorial of a given number.**

```
         #include<conio.h>
         #include <stdio.h>
        void main()
     {
        long int  i,fact=1,num;
        clrscr() ;
        printf("Enter the number\n");
        scanf("%ld",&num);
        if( num <= 0)
                fact = 1;
                else
        {
            for(i = 1; i <= num; i++)
                 {  fact = fact * i ;}
          }                        /* End of else */
   printf("Factorial of %ld =%ld\n", num, fact );
}               \* End of main() *\
```

2. **Program to generate the table of any number using loop**

```
#include<stdio.h>
#include<conio.h>
void main()
  {
       int  t , num , t ;
       clrscr() ;
  printf (“Enter the number whose table is to be
generated : \n” ) ;
              scanf(“%d”, & num ) ;
              printf ( “ Table of %d is :” , num) ;
                for ( i=1 ; i<= 10 ; i++ )
                {
                   t = num * i ;
             printf( “\n %d x %d = %d ” , num , i , t ) ;
                     }
            getch () ;
            }
```

**3. Program to check whether a given number is prime or not and output the given number with suitable message.**

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

   int i, num;
   int flag = 1;
   clrscr();
   printf(“ Enter the number”) ;
   scanf(“ %d” , & num) ;
   for( i = 2; i< num; i++)
  {
      if((num % i) = = 0)
     {
        prime = 0;
      }
   }

   if (prime = = 1)
printf("%d is prime number.", num);
   else
 printf("%d is not a prime number.", num);
 getch();
}
```

**4. Program to generate Fibonacci sequence Fibonacci sequence is 0 1 1 2 3 5 8 13 21.**

```
.
      #include <stdio.h>
      #include <stdio.h>
      void main()
      {
        int   fib1=0, fib2=1, fib3, limit, count=0;
        printf("Enter the limit to generate the
fibonacci sequence\n");
scanf("%d", &limit);
 printf("Fibonacci sequence is ...\n");
 printf("%d",fib1);
 printf("%d",fib2);
count = 2;  /* fib1 and fib2 are already used */
while( count < limit)
 {
      fib3 = fib1 + fib2;   count ++ ;
       printf("%d\n",fib3);
        fib1 = fib2;
        fib2 = fib3;
        }
      getch();
  }     /* End of main() */
```

**5. Program to accept an integer and reverse it.**

```
#include<conio.h>
#include <stdio.h>
void main()
{
  long  num, rev = 0, temp, digit;
  printf("Enter the number\n");
scanf("%ld", &num);
/*For better programming, choose
     'long int' */
```

**6. Program to calculate the sum of the series up to first 100 terms: $1^4 + 3^4 + 5^4 + 7^4 + \ldots + n$ terms.**

```
  #include<stdio.h>
  #include<conio.h>
 #include<math.h>
    void main ()
  {   int n , i , sum = 0 ;
      clrscr();
      printf(“ENTER THE NUMBER OF
                   TERMS\n”) ;
      scanf(“%d” , &n) ;
```

| | |
|---|---|
| **/*reverse of a five digit no. can be out of range of integer*/**<br>`temp = num;`<br>`while(temp > 0)`<br>`{`<br>`digit = temp % 10;`<br>`rev = rev * 10 + digit;`<br>`temp = temp/10 ;`<br>`}`<br>`printf("Given number   = %ld\n", num);`<br>`printf("Its reverse is = %ld\n", rev);`<br>`getch();`<br>`}` | `for (i=1 ; i<=n ; i++)`<br>`{`<br>`sum = sum + pow (i, 4) ;`<br>`}`<br>`printf("Sum of the series = %d" , sum) ;`<br>`getch() ;`<br>`}` |

| | |
|---|---|
| **7 .Program to generate the sum of the following series:- sum = x + x$^2$ / 2! + x$^4$ / 4! + …… + x $^n$ / n!** | **8.Printing the pattern of program**<br>*<br>* *<br>* * *<br>* * * *<br>* * * * * |
| `#include<stdio.h>`<br>`#include<conio.h>`<br>`#include<math.h>`<br><br>`void main ()`<br>`{`<br>`int , x , n , , j ,fact =0,`<br>`float sum  ;`<br>`clrscr();`<br>`printf(Enter the no. whose series is to be found: ") ;`<br>`scanf("%d", &x) ;`<br>`printf("Enter the no. of terms up to which series is to be generated") ;`<br>`scanf("%d", &n) ;`<br>`sum = x ;`<br>`for (i=1 ; i<=n ; i++)`<br>`{`<br>`if (i%2 = = 0)`<br>`{`<br>`fact =1;`<br>`for (j=1 ; j<=i ; j++)` | `#include<stdio.h>`<br>`#include<conio.h>`<br>`void main ()`<br>`{`<br>`int row, col;`<br>`clrscr();`<br>`for (row=1;row<=5; row++)`<br>`{`<br>`for (col=1 ; col <=row; col ++)`<br>`printf( "  *  ");`<br>`printf( "\n");`<br>`}`<br>`getch();`<br>`}` |

| | |
|---|---|
| <pre>                  { fact = fact * j ;<br>                    }<br><br>                    sum = sum + pow (x ,i)  / fact ;<br>                }                 // End of IF BLOCK<br>            }                     // End of outer for loop<br>            printf ("Sum of the series = %f", sum) ;<br>            getch() ;<br>        }                         // End of main ()</pre> | |

| 9.Program to print the following pattern : | 10. Program to print the following pattern (FLOYD'S TRAINGLE) : |
|---|---|
| **5 4 3 2 1**<br>**4 3 2 1**<br>**3 2 1** | **1**<br>**2 3**<br>**4 5 6**<br>**7 8 9 10** |
| <pre>#include<stdio.h><br> #include<conio.h><br><br>      void main()<br>       { int i , j ;<br>         clrscr () ;<br>         for ( i = 5 ; i > = 1 ; i -- )<br>            { for (j = 1 ; j < = 1 ; j --)<br>                 { printf ("%d" , j ) ; }<br>               printf ("\n" ) ;<br>            }<br>    getch();<br>}</pre> | <pre>#include<stdio.h><br> #include<conio.h><br><br>void main()<br>{<br>       int n  , r , val =1, j ;<br>      clrscr () ;<br>      printf ( "Enter the number of rows in the<br>                   Floyd's triangle\t : " )<br>       scanf ("%d" , &n ) ;<br>              for ( r = 1 ; r <= n ; r ++ )<br>                                     // print n rows<br>                {<br>                     for (j = 1 ; j <=r ; j ++)<br>                       {   printf ("%d" , val ) ;<br>                            val ++ ;<br>                        }<br>                            printf ("\n" ) ;<br>                }   getch();<br>  }</pre> |

**11. WAP to detect Armstrong Numbers in three digits between 100 to 999.**

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int d=3 , n , x=0 ;
   int k , i , cube=0;

  clrscr();
   printf( " \n Following number are
              Armstrong numbers ") ;
     for (k =100 ; k<=999 ; k++)
       {
          n=k ;
          while (x<=d)
        {
               i=n %d ;
               cube = cube + pow (i, 3) ;
                n = n/10;
                x++;
           }
   if(cube = = k)
         printf ( " \n \t  %d " , k) ;
    }
  getch();
}                // End of main()
```

**12. Program to find binary equivalent of a decimal number entered by user.**

```
# include<stdio.h>
# include<conio.h>
void main()
{
    int r , n ;
    clrscr() ;
   printf ( "Enter the value of n :") ;
   scanf ( " %d " , &n) ;
   printf ( " Binary number") ;
   while ( n!=0)
   {
        r = n % 2 ;
        n = n / 2 ;
       printf ( " %d " , r ) ;
   }
getch();
}
```

**13. Program to calculate m to power n.**

```
    #include<conio.h>
   #include<stdio.h>
     void main()
     { int m , n , i=1;
        long int pow =1 ;
       clrscr();
       printf( " Enter two numbers : ") ;
       scanf ( " %d %d " , &m , &n);
     while ( i<=n)
        {
             pow = pow * m ;
             i++ ;
        }
   printf ( " \n %d to power %d = %ld " , m ,
              n, pow);
```

**14. Program to add first seven terms of following series using for loop :**
**1 / 1! + 2 / 2! + 3 / 3! +…….7 / 7!.**

```
#include<conio.h
 #include<stdio.h>
 void main()
{
  int i=1, j ;  float fact , sum = 0.0;
clrscr();
while(i<=7)
 {
   fact = 1.0 ;
    for (j=1 ; j <=i ; j++)
           fact = fact * j;
           sum = sum + i/ fact; }
printf( " Sum of series = %f " , sum);
```

```
   getch();
}
```

```
getch() ; }
```

**15. WAP to find the number and their sum between 100 and 200 divisible by 7.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int num , sum = 0 ;
  printf( " Numbers divisible bt 7 b/w 100 &
               200 are : \n ") ;
   for (num = 101 ; num < 200 ; num++)
     {
            if (num % 7 = = 0 )
              {   printf ( " %d " , num ) ;
                 sum = sum + num ;
              }
       }
   printf(" Sum of all integers divisible by 7 b/w
             100 & 200  is = %d  ", sum);
    getch() ;
}
```

**16. WAP to find the average of first n natural numbers using while loop.**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n , i=1, sum = 0 ;
    float avg = 0.0;
    printf( " Enter the value of n :  ") ;
    scanf( " %d " , &n) ;
    while( i <=n)
      {
          sum = sum + i ;
            i++;
       }
  avg= (float) sum / n ;
  printf("\n  Sum of first %d  numbers = %d ",
                                           n ,sum ) ;
  printf( " \n The average of first %d numbers
                                    = % f " , n , avg );
     getch() ;
}
```

**17. Program to print the sum of odd numbers and even numbers separately from 1 to 50.**

```
void main()
{
    int i; m , sumeven =0 ,sumodd=0;
    clrscr();
    for ( i = 1 ; i<=50 ;++i)
         {
           if ( i % 2 = = 0)
              sumeven = sumeven + i;
          else
              sumodd = sumodd + i;
        }
printf( " Sum of even numbers = %d \n",
                              sumeven) ;
printf(" Sum of odd numbers = %d",
                               sumodd);
```

**18. WAP to find the sum of following series:**
$\sqrt{1} + \sqrt{2} + \sqrt{3} + \ldots\ldots.. + \sqrt{n}$

```
void main()
{
    int c , n ;
    float sum = 0.0 ;
    printf(" Enter the value of n : \n") ;
    scanf (" %d " , &n) ;
      for (c=1 ; c < n ; c++)
         {
             sum = sum + sqrt(c ) ;
          }
 printf( " Sum of square root series = %f",
                                       sum ) ;
getch();

}
```

```
getch();
```

**19. WAP in C to check whether an enterd number through the keyboard is palindrome or not .**

```
#include<stdio.h>
#include<conio.h>
void main()
{
   int i , n , d , rev=0 , num;
   clrscr();
    printf( " Enter the number \n") ;
    scanf( " %d " , &num);
    n= num ;
    while(n >0)
     {
        d=num %10;
        rev=rev *10 +d;
        n = num/10;
      }

printf(" Reverse of %d = %d \n" , num , rev);

if(rev = = num)
      printf( " \n %d is a palindrome" , num);
else
      printf(" %d is not a palindrome", num);
getch();
}
```

**20. Program to compute the sum of following series : $x - x^3/(3!) + x^5/5! - x^7/7! + \ldots\ldots\ldots +$ upto n terms.**

```
void main()
{
     int p =1, n  ;
     float sum=0.0, term;
     printf(" Enter the value of x \n");
     scanf( " %d" , &x) ;
    printf("Enter the power of nth term\n ");
    scanf(" %d ", &n);
    while( p<=n)
    {
          sum = sum + term;
term=(term *x *x -1) / (( i + 1) * (i +2));
i = i +2 ;

}

printf( " Sum of Series = %f ", sum) ;

getch();
}
```

| | |
|---|---|
| | |

| | |
|---|---|
| **21. WAP to check whether a given number is perfect or not ?**<br><br>**void main()**<br>**{**<br>**  int num , sum =0 , i;**<br>**  clrscr();**<br>** printf(“ Enter the number \n”) ;**<br>** scanf(“ %d ”, &num) ;**<br>** for( i=1 ; i<num ; i++)**<br>**    {**<br>**        if(num %i = = 0)**<br>**           sum = sum + i;**<br>**     }**<br><br>**  if( num = = sum)**<br>**      printf(“ %d is a perfect number”, num);**<br>**  else**<br>**      printf(“ %d is not a perfect number”,**<br>**                num);**<br>**getch();**<br><br>**}** | **22. WAP to generate the following pattern:**<br><br>**A**<br>**A B**<br>**A B C**<br>**A B C D**<br><br>**#include<stdio.h>**<br>**#include<conio.h>**<br>**void main()**<br>**{**<br>**  char i, j ;**<br>**  for ( i=65 ; i<=68 ; i++)**<br>**     {**<br>**        for( j=65;j<=i; j++)**<br>**         {**<br>**             printf( “ %c” , j);**<br>**          }**<br>**        printf(“ \n”);**<br>**     }**<br>**  getch();**<br>**}** |

## Short Question & Answers (Module 2 : Functions)

**Ques 1. What is the purpose of main ( ) function?**

**Ans:** In our c program, every program starts with main () because it tells the C compiler where the program starts its execution. It always returns an integer value. It is a well-defined user defined function, because In different programs the body of main() contains different code written by the programmer.

**Ques 2. What are the local variables with respect to a function?**

**Ans**:The local variables are defined within the body of the function or the block. The variable defined is local to that function or block only. Other functions can access these variables. The compiler shows errors in case other functions try to access the variables.

**Ques 3. What are library functions and user defined functions?**

Ans : Library functions are those which are pre defined in the library of C compiler. The definitions are given inside the header files . E.g : printf(),scanf(), getch(), sqrt(), clrscr() etc.

User defined functions are those in which definition is written by the programmer inside the function body.E.g A function named add(int a , int b) to add two integer numbers a and b .

**Ques 4**. **What is the meaning of keyword void before the name of a function?**

**Ans**: Keyword void means that the function should not return any value.

**Ques5. What is an argument? Differentiate between formal arguments and actual arguments?**

**Ans:** An argument is an entity used to pass the data from calling function to the called function. Formal arguments are the arguments available in the function definition. They are preceded by their own data types. Actual arguments are available in the function call.

**Ques 6. What is the purpose of return keyword while programming with functions?**

**Ans :** When a function calculates some value , then after that control returns from a function***, values returned are collected in calling function using the keyword 'return'.*** A function can return only one value at a time. Example : Following return statements are incorrect : return (a , b) , return ( x , 12), Because both are trying to return two values. ***In case if we want to use more than one values in return keyword syntax is :  return value 1, value 2 , value 3…….value n. Here left to right evaluation is done And the right most value is returned to calling function.***

So correct statement are : return (a) , return (x).

But a function body can contain more han one return statements.

```
E.g : if ( x >= 5)
           return (x) ;
      else return ( 0) ;
```

**Ques 7. Define exit() function with an example.**

**Ans :** C provides a way to leave the program before it is actually terminated using exit() function.

(i)Syntax : exit(status)  , where status is an int variable / constant.

(ii)This function is defined in **stdlib.h** header file

Example :

```
# include<stdio.h>
#include<conio.h>
# include <tdlib.h>
void main()
{
   printf( " C programming is logical and interesting ") ;
  exit (0);
  printf ( " C can be easily learned") ;
  printf ( " C is a powerful language invented by Denis Ritchie in AT & T labs Bell");
}
```

## Long Questions & Answers (Module 2 : Functions)

**Ques 8. What are functions and their types ? What are the elements of user defined functions , explain?**

Ans : Functions break large computing tasks into smaller ones, and make the task of coders easy.

A function is a self-contained block of statements that perform a coherent task of some kind.

In C language we use many small functions rather than few big functions. Functions support modular programming , in which a software is divided into small modules and each module consists of several fumctions/sub modules.

In a C program, execution starts from the function main(). A function can activate some another function. It can also call itself recursively , such functions are called recursive functions.

**Types of functions: (a) Library functions (b) User defined functions.**

**Library Functions**: Library functions are need not to be written by the programmers. Their definitions are already given by the compiler itself in the header files. Whenever we require to use any library function , the respective header file is included before the function main() at top of the program. ***E.g : In stdio.h : printf(), scanf() functions , In conio.h : getch(), clrscr();***

**User Defined functions**: Definition of these functions are written by the programmers itself .These are not already defined in the library of C compiler. Advantages of user defined functions are as follows :

(a) It facilitates top down approach of modular programming.

(b) Provides more structured and easy way to read and understand the large & complex codes.

(c) It is easy to find faulty functions and their testing and debugging.

(d) User defined functions may be used by many other programs (with the help of user defined Header Files). So repetition of same code again and again can be avoided, which makes less execution time and reduced code size.

***Elements of user defined functions:***

***(a) Function declaration(eclaration )***

***(b) Function call***

***(c) Function definition.***

**Function declaration**: Like variablesin C , all the functions in a C program must be declared before they are invoked . A function declaration consists of four parts :

(i) Function return type. E.g : int , float , char, void etc.

(ii) Function name (Follows the rules of an identifier.)

(iii) Parameter or argument list. A function can contain one or more parameters with data type of arguments.

(iv) Termination semicolon.

(v) Prototype definition and declaration of library functions are inside the header files.

**Syntax : function-type function name (parameter list) ;**

Parameter list must be separated by commas. Names need not to be same in the prototype declaration & function definition.

Use of parameter names is optional in prototype declaration, but their data type is compulsory.

If a function has no formal parameters , the list is written as (void)

**Parameters :** These are the communication medium to pass in formation between calling and called functions. ***These are used in three places: prototype, function call , function definition***.

**Formal Parameters** : The parameters used in prototypes and function definitions.
**Actual Parameters** : The parameters used in function calls.

**Function Call:** A function can be called by simply using the function name followed by a list of actual parameters if any required.

(i) When the compiler encounters a function call, the control is transferred to the function definition.

(ii) This function is then executed line by line and a value is returned when return statement is executed.

**Function definition**: It is the actual implementation of function, where actual logic of program is written. A function can only be defined once but can be declared many times.

A function can be declared within the body of some other functions but cannot be defined within the body of other function.

**Elements of Function definition**: (i) Function name (ii) Function name (iii) List of parameters. (iv) Local variable declarations (v) function statements (vi) return statements.

Example :

```
#include <stdio.h>
#include<conio.h>
 int cal_sum(int , imt , int);
  void main()
  {
      int a , b, c , sum=0 ;
    printf(" Enter the three values :") ;
    scanf( " %d %d %d", &a , &b , &c);
    sum=cal_sum(a,b,c) ;
    printf( "\n Sum = %d ",sum) ;
}
   int cal_sum (int x , int y , int z ) ;
   {
       int d ;
       d= x + y + z ;
       return (d) ;
    }
```

**Ques 9: Explain Call by Value and Call by Reference methods of parameter passing. In support give the example programs also.**

**Ans : Call by Value :** In this method values of variables are passed by the calling function to the called function. New variables are created by the called function to store values of arguments passed to it.
So called function uses a copy of the actual arguments .Changes are reflected only in the called function , not the calling function.
**Advantage:** Arguments passed can be variables (e.g x) , literals (e.g : 6 ) , or expressions (e.g x + 1).

**Disadvantage:** Copying data consumes additional storage. It can take lot of time to copy , so performance of function may decrease if it is called several times .

**Call by Reference** : When a function wants to modify the value of the argument , then we pass arguments using address of variables. Function parameters are declared as a reference/address. When this is done then any changes made by the function to the arguments it receives, are visible in the calling function.
For reference an asterisk(*) is placed after the data type in parameter list.

**Advantage** : Since arguments passed are not copied into new variables , it provides greater time and space efficiency. We can return multiple values using this method.
**Disadvantage** : Side effect of this method is whrn an argument is passed using call by address, it becomes difficult to tell whether that argument is meant for input , output or both.

| **Example of Call by Value** :<br><br>**/* Swapping the values of two variables using call by value technique*/.** | **Example of Call by Reference** :<br><br>**/* Swapping the values of two variables using call by reference method*/.** |
|---|---|
| #include<stdio.h><br>#include<conio.h><br> void swap (int x , int y) ;<br>void main()<br>{ int a , b ;<br>  clrscr();<br>  printf ( “ Enter the values of two numbers \n”) ;<br>  scanf( “% d  %d “ , &a, &b);<br> printf( “Values of a and b before swapping : \n”);<br>  printf ( “ a= %d\t b=%d” , a ,b);<br>  swap (a , b) ;<br>printf( “Values of a and b after swapping : \n”);<br>printf ( “ a= %d\t b=%d” , a ,b);<br>getch();<br>}<br>void swap(int x , int y)<br> {<br>     int t ;<br>    t = x;<br>    x = y;<br>    y = t ;<br>  printf(“ In swap function values are : \n” );<br>  printf( \n x = %d \t y = %d, x ,y);<br>}<br><br>**Output :**<br>**Enter the values of two numbers: a=10 , b=20**<br>**Values of a and b before swapping :a=10, b=20**<br>**In swap function values are : x= 20, y=10**<br>**Values of a and b after swapping : a =10, b=20** | #include<stdio.h><br>#include<conio.h><br> void swap (int *x , int *y) ;<br>void main()<br>{ int a , b ;<br>  clrscr();<br>  printf ( “ Enter the values of two numbers \n”) ;<br>  scanf( “% d  %d “ , &a, &b);<br> printf( “Values of a and b before swapping : \n”);<br>  printf ( “ a= %d\t b=%d” , a ,b);<br>  swap (a , b) ;<br>printf( “Values of a and b after swapping : \n”);<br>printf ( “ a= %d\t b=%d” , a ,b);<br>getch();<br>}<br>void swap(int *x , int *y)<br> {<br>     int t ;<br>    t = *x;<br>    *x = *y;<br>    *y = t ;<br>  printf(“ In swap function values are : \n” );<br>  printf( \n x = %d \t y = %d, x ,y);<br>}<br><br>**Output :**<br>**Enter the values of two numbers: a=10 , b=20**<br>**Values of a and b before swapping :a=10, b=20**<br>**In swap function values are : x= 20, y=10**<br>**Values of a and b after swapping : a =20, b=10** |

**Ques 10. (i) Explain recursion and recursive function.**

**(ii)WAP using recursive function to compute the sum of following series upto n terms :**

**1 + 2 + 3 + ………+(n-1) + n .**

**Ans : A recursive function is** defined as a function that calls itself again and again to solve a smaller version of its task until a final call is made. Every recursive solution has two major cases :

(a) Base Case : In this the problem is simple enough to be solved directly without making any furthrt calls to the same function.

(b) Recursive Case : In this first the given problem is divided into simpler sub parts. Then secondly , function calls itself but to handle sub parts of first step.

Basic steps of recursion are analyzed as follows :

(i) Initially Specify the base case which will terminate the function call itself.

(ii) Check whether the current value processed matches with the value of the base case. If yes, then process and return the value.

(iii) Divide the problem into smaller sub parts and call the function from sub part.

(iv) Combine the results of sub parts.

(v) Return the result of the entire problem.

| **Recursive function for adding 1 + 2 + 3 …n terms.** | |
|---|---|
| <pre>#include<stdio.h><br>#include<conio.h><br>int sum (int) ;           // function prototype.<br>void main()<br>{<br> int n , s ;<br>clrscr();<br>printf (“Enter the value of n terms :” );<br>scanf (%d “ , &n) ;<br>s = sum( n) ;     // function call in main()<br>printf (“ Sum of %d natural numbers = %d” , n,s) ;</pre> | <pre>getch();<br>}<br>  int sum (int x)        // function definition<br> {<br>    int a = 0 ;<br>       if ( x = = 0)<br>              return 0 ;<br>        else<br>              a = x + sum(x-1) ;  // recursive case.<br>         return(a);<br>  }</pre> |

**Ques 11. Differentiate between the following :**

**(i) Call by value and call by reference method of parameter passing.**

**(ii) Iteration and recursion.**

Ans :

| Call by Value method | Call by Reference method |
|---|---|
| 1. This is a usual method to call a function in which only the value of variable is passed as an argument. | In this method the address of variable is passed as an argument. |
| 2. A copy of actual arguments is passed in formal arguments in function definition. | Formal parameters are pointers to actual arguments. |
| 3. Any changes in the value of formal arguments does not affect in value of actual arguments. | Changes made in formal arguments are reflected back to the values of actual arguments. |
| 4. By this method a function can return only a single value. | By this method a function can return more than one values at a time. |
| 5. Memory locations occupied by formal and actual arguments are different. | Memory locations occupied by formal and actual arguments are same , so less memory is occupied as compared to Call by Value. |
| 6. This is a slow process. | This is more efficient and fast method. |
| 7. Function prototype contains data type as argument. | Function prototype is a pointer prefixed to arguments inside it. |

| Iteration | Recursion |
|---|---|
| 1. This uses a set of statements which are repeated. | This involves selection of a particular structure in a function. |
| 2. Repetition is applied to the loop constructs. | Function calls itself again and again. |
| 3. Iteration terminates when loop test condition becomes false. | Recursion stops when base case is encountered in function call , within its definition. |
| 4. Iteration requires a counter variable or a sentinel value. | In this base case and recursive case is required. |
| 5. Less memory consumed | More memory is consumed , because each time a function calls itself , a copy of variables is created. |

## Programmes (Module 2 : Functions)

**1. WAP in C for finding the sum of three no. and their average using user defined functions sum() and avg().**

```
 #include<stdio.h>
#include<conio.h>
int  sum (int , int , int ) ;
float avg (int , int , int ) ;
// Prototype or declaration of function
void  main()
{
    int a , b , c, sum=0 ;
    float av =o ;
   s= sum (a , b , c);      // function call
  av = avg (a , b , c) ;
scanf ("%d%d%d", &a, &b, &c) ;
                              // actual parameters
printf ("Sum= %d\t Average = %f " , s , av) ;
  getch() ;
    }      // End of main ()

// Definition of  user defined function sum ()
    int  sum (int  x , int  y  , int  z )
//   x , y and z are formal parameters
          {
             return (x + y + z) ;
          }
/ Definition of  user defined function avg ()
        float  avg  (int x , int y , int z )
        {
            float  m ;
            m = (float) (x + y + z ) / 3;
          return m ;
        }
```

**2. Program to compute factorial of a number using recursive function.**

```
#include<stdio.h>
#include<conio.h>
long  int  rec_fact(long int) ;
void main ()
 {
    int num , fact ;
    clrscr() ;
    printf("Enter the number") ;
    scanf ("%ld", &num) ;
   fact = rec_fact (num) ;
printf("Factorial of %ld = %ld", num , fact) ;
    getch() ;
  }
     //  recursive definition of factorial

long  int  rec_fact(long  int  x) ;
  {    int f ;
       if(x = = 1 )
           return 1;                 // base case
        else
        {
        f =  x * rec_fact(x-1) ;   // Recursive case
          return (f) ;
          }
  }
```

| 3. **Program to generate the Fibonacci series using recursive function** | 4. **WAP to find largest number among the given two numbers using a user defined function large().** |
|---|---|
| <pre>#include<stdio.h><br>#include<conio.h><br> int  fibo( int) ;<br>   void main ()<br>  {<br>      int  n  , f ;<br>      clrscr() ;<br>     printf(“Enter the number upto which series is<br>                  to be generated \n”) ;<br>     scanf (“%d”, &n) ;<br>  printf(“Fibonacci series up to %d terms is : \n<br>               \n”, n ) ;<br><br>for (int i=1 ; i< = n ; i++) ;<br>              {<br>                 f = fibo (i) ;<br>               }<br><br>      int fibo (int  n)         // Recursive definition<br>{<br>      if (n = = 1)<br>          return 0 ;                // Base case<br>      if ( n = = 2)<br>           return 1 ;<br>      return ( fibo(n - 1) + fibo(n - 2) ) ;<br>// At a time two recursive function called<br>     so binary<br>}</pre> | <pre>#include<stdio.h><br>#include<conio.h><br>float large (float m , float n) ;<br> void main()<br>{<br>    float x , y , max ;<br>   float large (float m , float n) ;<br>   printf(“ Enter the two numbers \n”);<br>   scanf( “ %f %f ” , &x , &y) ;<br>   printf( “ x = %f  and y = %f \n” , x , y ) ;<br>   max = large(x , y) ;<br>   printf(“ The largest number = %f \n ”, max);<br>}<br>    float large (float m , float n)<br>     {<br>         if (m > n)<br>              return m ;<br>         else<br>               return n ;<br>     }</pre> |

| 5. **WAP that accepts the length and width of a rectangle and print the area. The area of a rectangle is calculated by a user defined function and returns it value to main program where it is printed. The values length and width are accepted from the keyboard as an integer value. Also draw the flow chart of program.** | 6. **WAP that accepts the three sides of a triangle and print the area. The area of a triangle is calculated by a user defined function and returns it value to main program where it is printed. The values of three sides are accepted from the keyboard as an integer value** |
|---|---|
| <pre># include<stdio.h><br>#include<conio.h><br>int rect_area(int , int);<br>void main()<br>{<br>    int length , width ,area=0 ;<br>   clrscr() ;<br>   printf(" Enter the length and width \n ");<br>   scanf( " %d %d ", length , width);<br>  printf(" Length =%d  width= %d \n",<br>             &length,  & width) ;<br> area= rect_area (length , width) ;<br>printf(" Area of rectangle = %d ", area);<br>getch();<br>    }<br> int rect_area(int l , int w)<br>{<br>    return (l *w)<br>}</pre> | <pre># include<stdio.h><br>#include<conio.h<br>#include <math.h><br>float tri_area(float , float, float);<br>void main()<br>{<br>    float s1 ,s2 ,s3, area=0.0 ;<br>   clrscr() ;<br>   printf(" Enter the three sides of triangle \n ");<br>   scanf( " %f %f %f", s1 , s2, s3) ;<br>  printf(" side1 =%f  side2= %f  side3=%f \n",<br>             &s1,  & s2, &s3) ;<br> area= tri_area (s1 , s2 ,s3) ;<br>printf(" Area of triangle = %f ", area);<br>getch();<br>    }<br> float tri_area (float a , float  b , float  c)<br>{<br>     perim=0.0 , a=0.0 ;<br>      perim =  ( a + b + c) /2 ;<br>     a= sqrt( perim * (perim-a) *(perim-b)<br>                *(perim-c)) ;<br>      return (a);<br>}</pre> |

**[ END of 3rd UNIT ]**