Function

Function is a reusable self-contained block of code that perform any specific task. Functions divide the code and modularize the program for better and effective results. Every C program has at least one function, which is **main(),** and all the most trivial programs can define additional functions.

Advantages of function:

- It increases code reusability.
- The source code can be reduced.
- Coding will be faster.
- Debugging will be easier.
- The program can be divided into multiple parts.
- Many programmers can work simultaneously.

In 'C' programming, functions are divided into two types:

1. **Library functions**
2. **User-defined functions**
   - The **library** function is the pre-defined compiled set of code in the header file of **C** library. The programmer can use this function by simply adding header file to their program. Programmer cannot change or alter the meaning of this function. They do not have to declare and define this function, they are just allowed to use this function with their proper syntax.

     The function's name, its return type, their number of arguments and types have been already defined and cannot be changed.

     For example:

     To use printf, scanf function we have to include stdio.h header file.
     To use getch, clrscr function we have to include conio.h header file.
     To use strlen, strcpy function we have to include string.h header file.
     To use pow, sqrt function we have to include math.h header file.
     To use clock function we have to include time.h header file.
   - The block of code which are defined by program at the time of writing a program having specific task is called **user-defined** function. The job of the user-defined function is defined by the programmer. The programmer has choice to choose its name, return type and arguments.

Arguments and Parameters

- **Arguments:** The variable used in the function reference called 'arguments'. These are written within the parenthesis followed by the function name. They are also called actual parameters which they are accepted in the main program.
- **Parameters:** The variables used in the function definition are called 'parameters.' They are also referred as function parameters, because they are not accepted values. They receive value from the calling function. Parameters must be written within the parenthesis followed by the name of function in the function prototype.

Elements of function

- Function prototyping (function declaration)
- Function calling
- Function definition

Function prototyping:

The function prototyping is the blueprint of the function, which provides the basic information like function name, return type, number and types of arguments to the compiler.

It provides the following information to the compiler:

- The name of the function.
- Return type.
- Number and type of arguments.

Syntax:

returnType functionName (arg1, arg2, ….., argN);

Example:

int sum (int a, int b, int c);

Function calling:

The function calling is execution part of the function, which provides the basic information like function name and number of arguments to the compiler.

It provides the following information to the compiler:

- The name of the function.
- Number of arguments.

Syntax:

function_Name (arg1, arg2,….., argN);

Example:

sum (int a, int b, int c);

Categories of user defined function

i. Function with argument with return value.

```
#include<stdio.h>
#include<conio.h>
float sinterest(float p, float t, float r);
void main()
{
        float p, t, r, si;
        clrscr();
        printf("Enter principal, time and rate: ");
        scanf("%f%f%f", &p, &t, &r);
        si= sinterest(p, t, r);//Actual parameter
        printf("\nSimple interest is: %.2f", sinterest(p, t, r));
        getch();
}
float sinterest(float p, float t, float r)//Formal parameter
{
        float si;
        si = (p*t*r)/100;
        return si;
}
```

ii. Function with argument without return value.

```
#include<stdio.h>
#include<conio.h>
void  sinterest(float p, float t, float r);
void main()
{
        float p, t, r;
        clrscr();
        printf("Enter principal, time and rate: ");
        scanf("%f%f%f", &p, &t, &r);
        sinterest(p, t, r);
        getch();
}
void sinterest(float p, float t, float r)
{
        float si;
        si = (p*t*r)/100;
        printf("\nSimple interest is: %.2f", si);
}
```

iii. Function without argument with return value.

```
#include<stdio.h>
#include<conio.h>
float  sinterest();
void main()
{
        clrscr();
        printf("\nSimple interest is: %.2f", sinterest());
        getch();
}
float sinterest()
{
        float p, t, r, si;
        printf("Enter principal, time and rate: ");
        scanf("%f%f%f", &p, &t, &r);
        si = (p*t*r)/100;
        return si;
}
```

iv. Function without argument without return value.

```
#include<stdio.h>
#include<conio.h>
void  sinterest();
void main()
{
        clrscr();
        sinterest();
        getch();
}
void sinterest()
{
        float p, t, r, si;
        printf("Enter principal, time and rate: ");
        scanf("%f%f%f", &p, &t, &r);
        si = (p*t*r)/100;
        printf("\nSimple interest is: %.2f", si);
}
```

Workout Programs

Write C program to find sum of two numbers using function named sum().

```
#include<stdio.h>
#include<conio.h>
int sum(int a, int b);
void main()
{
        int a,b,c;
        clrscr();
        printf("Enter two numbers:");
        scanf("%d%d",&a,&b);
        c=sum(a,b);
        printf("Sum of two numbers = %d",c);
        getch();
}
int sum(int a, int b)
{
        int da;
        da=a+b;
        return (da);
}
```

Write C program to find area and perimeter of a rectangle. Use function area_perimeter().

```
#include<stdio.h>
#include<conio.h>
void area_perimeter(int l, int b);
void main()
{
        int l, b;
        clrscr();
        printf("Enter length and breadth :");
        scanf("%d%d",&l,&b);
        area_perimeter(l,b);
        getch();
}
void area_perimeter(int l, int b)
{
        int area,perimeter;
        area=l*b;
        printf("Area of rectangle = %d",area);
        perimeter=2*(l+b);
        printf("\n Perimeter of rectangle = %d",perimeter);
}
```

Write C program to know a number is even or odd using function named even_odd().

```
#include<stdio.h>
#include<conio.h>
void even_odd(int n);
int main()
{
        int n;
        clrscr();
        printf("Enter a number :");
        scanf("%d", &n);
        even_odd(n);
        getch();
}
void even_odd(int n)
{
        if(n%2==0)
                printf("Even");
        else
                printf("Odd");
}
```

Write C program to print multiplication table of a number using function table().

```
#include<stdio.h>
#include<conio.h>
void table(int n);
void main()
{
        int n;
        printf("Enter a number :");
        scanf("%d",&n);
        table(n);
        getch();
}
void table(int n)
{
        int i;
        for(i=1;i<=10;i++)
        {
                printf("%dX%d=%d\n", n, i, n*i);
        }
}
```

Write C program to print the greatest value among three numbers using a function int great(). We have to use return statement.

```
#include<stdio.h>
#include<conio.h>
int great(int a, int, int c);
void main()
 {
        int a, b, c, d;
        clrscr();
        printf("Enter three numbers : ");
        scanf("%d%d%d", &a, &b, &c);
        d=great(a, b, c);
        printf("greatest number is %d", d);
        getch();
 }

  int great(int a, int b, int c)
 {
  if(a>b && a>c)
        return a;
  else if (b>c)
        return b;
  else
        return c;
}
```

Write C program to check and display the given number is prime or not.

```
#include<stdio.h>
#include<conio.h>
void test(int n);
void main()
{
        int n;
        clrscr();
        printf("Enter a number :");
        scanf("%d", &n);
        test(n);
        getch();
}
void test(int n)
{
        int i, f=0;
        for(i=1; i<=n; i++)
        {
                if(n%i==0)
                        f++;
        }
        if(f==2)
                printf("Prime");
        else
                printf("Not prime or composite");
}
```

Write C program to find sum of series 1, 3, 5,.....200 using function.

```
#include<stdio.h>
#include<conio.h>
float sum();
void main()
{
        clrscr();
        printf("the sum is=%f", sum());
        getch();
}
float sum()
{
        int s=0, i;
        for(i=1; i<=200; i+=2)
        {
                s=s+i;
        }
        return s;

}
```

WAP to calculate and display sum of n numbers using array and function.

```
#include<stdio.h>
#include<conio.h>
void da(int a[], int n);
void main()
{
        int a[100], i, n;
        clrscr();
        printf("\nEnter value of n: ");
        scanf("%d", &n);
        printf("\nEnter %d numbers:\n");
        for(i=0; i<n; i++)
        {
                scanf("%d", &a[i]);
        }
        da(a, n);
        gech();
}
void da(int a[], int n)
{
        int sum=0, i;
        for(i=0; i<n; i++)
        {
                sum+= a[i];
        }
        printf("Sum=%d", sum);
 }
```

WAP to find string length.

```
#include<stdio.h>
#include<conio.h>
int strLength(char st[]);
void main()
{
        char str[100];
        clrscr();
        printf(" Enter any string : ");
        scanf("%s",str);
        printf(" The length of string is %d",strLength(str));
        getch();
}
int strLength(char st[])
{
        int i;
        for(i=0;st[i]!='\0';i++);
                return (i);
}
```

WAP to sum two Metrix.

```
#include<stdio.h>
#include<conio.h>
void matrix_sum(int first[2][2],int second[2][2]);
void main()
{
        int i, j, first[2][2], second[2][2];
        clrscr();
        printf("Enter the elements of first matrix\n");
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                        scanf("%d", &first[i][j]);
                }
        }
        printf("Enter the elements of second matrix\n");
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                         scanf("%d", &second[i][j]);
                }
        }
        matrix_sum(first,second);
        getch();
}
 void matrix_sum(int first[2][2],int second[2][2])
{
        int sum[2][2], i, j;
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                         sum[i][j]= first[i][j]+ second[i][j];
                 }
        }
        printf("Sum of entered matrices:-\n");
        for(i=0;i<2;i++)
        {
                for(j=0;j<2;j++)
                {
                        printf("%d\t", sum[i][j]);
                }
                printf("\n\n");
        }
}
```

WAP to sort n integers

```
#include<stdio.h>
#include<conio.h>
void sort(int num[],int n);
void main()
{
	int num[100],n,i;
	clrscr();
	printf(" Enter the array size not more than 100 : ");
	scanf("%d",&n);
	printf("Enter %d elements for array: ", n);
	for(i=0;i<n;i++)
	{
		scanf("%d",&num[i]);
	}
	sort(num,n);
	getch();
}
void sort(int num[],int n)
{
	int i,j,temp;
	for(i=0; i<n; i++)
	{
		for(j=i+1; j<n; j++)
		{
			if(num[i]>num[j])
			{
				temp=num[i];
				num[i]=num[j];
				num[j]=temp;
			}
		}
	}
	printf(" Numbers in ascending order \n ");
	for(i=0;i<n;i++)
		printf(" %d\t ",num[i]);
}
```

WAP to sort n strings

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
void sort(char str[][20], int n);
void main()
{
	char str[50][20];
	int i,n;
	clrscr();
	printf(" How many strings :");
	scanf("%d",&n);
	printf("Enter %d string: ",n);
	for(i=0;i<n;i++)
	{
		scanf("%s",str[i]);
	}
	sort(str,n);
	getch();
}
void sort(char str[][20], int n)
{
	int i,j;
	char temp[20];
	for(i=0; i<n; i++)
	{
		for(j=i+1; j<n; j++)
		{
			if(strcmpi(str[i], str[j])>0)
			{
				strcpy(temp,str[i]);
				strcpy(str[i],str[j]);
				strcpy(str[j],temp);
			}
		}
	}
	printf(" Sorted strings are : \n");
	for(i=0;i<n;i++)
		printf("%s\n",str[i]);
 }
```

Recursive function

It is process by which function call itself repeatedly, until some condition has been satisfied. For the problem to solve recursively two conditions must be satisfied:

- The function should call itself.
- The problem statement must include a stopping condition.

Factorial using recursive function.

```
#include<stdio.h>
#include<conio.h>
long int factorial(int n);
void main()
{
    int n;
    clrscr();
    printf("Enter the number:\n\n");
    scanf("%d",&n);
    printf("Factorial of %d is %ld",n,factorial(n));
    getch();
}
long int factorial(int n)
{
   if(n==0|| n==1)
   {
        return n;
   }else{
       return(n*factorial(n-1));
   }
}
```

Sum of n numbers using recursive function.

```
#include<stdio.h>
#include<conio.h>
long int sum(int n);
void main()
{
    int n;
    clrscr();
    printf("Enter the number:\n\n");
    scanf("%d",&n);
    printf("Sum of %d is %ld",n,sum(n));
    getch();
}
long int sum(int n)
{
   if(n==0|| n==1)
   {
        return(n);
   }else{
       n=n+sum(n-1);
       return(n);
   }
}
```

Fibonacci series using recursive function.

```
#include<stdio.h>
#include<conio.h>
long int fab(int n);
void main()
{
    int i, n;
    clrscr();
    printf("Enter the number:\n\n");
    scanf("%d",&n);
printf("Fibonacci series:\n");
        for(i=0; i<n; i++)
{
        printf("%3ld", fab(i));
}
    getch();
}
long int fab(int n)
{
   if(n==0|| n==1)
   {
         return(n);
   }else{
return(fab(n-1)+fab(n-2));
   }
```

First 50 natural numbers using recursion.

```
#include<stdio.h>
int  numPrint(int);
main()
{
   int n = 1;
        printf("\n\n Recursion : print first 50 natural numbers :\n");
        printf("-------------------------------------------------\n");
        printf(" The natural numbers are :\n");
   numPrint(n);
   printf("\n\n");
}
int numPrint(int n)
{
   if(n<=50)
   {
      printf(" %d ",n);
      numPrint(n+1);
   }
}
```

Series n,…….,4,3,2,1 using recursion.

```
#include<stdio.h>
int  numPrint(int);
main()
{
   int n;
        printf("\nInput any number:\n");
        scanf("%d", &n);
   numPrint(n);
}
int numPrint(int n)
{
   if(n==0)
   {
        return 0;
   }else{
        printf(" %d ",n);
      numPrint(n-1);
   }
}
```

Series 1,2,3,4,……. n^{th} using recursion.

```
#include <stdio.h>
int num(int s, int e);
main()
{
   int s, e;
   printf("Enter lower limit: ");
   scanf("%d", &s);
   printf("Enter upper limit: ");
   scanf("%d", &e);
   printf("Series from %d to %d are: ", s, e);
   num(s, e);
}
int num(int s, int e)
{
   if(s>e){
      return 0;
   }else{
        printf("%d, ", s);
        num(s+1, e);
   }
}
```

Odd numbers 1,3,5,7……. 50^{th} using recursion.

```
#include<stdio.h>
int  numPrint(int);
main()
{
   int n = 1;
        printf("\n\n Recursion : print first 50 natural numbers :\n");
        printf("-------------------------------------------------\n");
```

```
        printf(" The natural numbers are :\n");
   numPrint(n);
   printf("\n\n");
}
int numPrint(int n)
{
   if(n<=50)
   {
      printf(" %d ",n);
      numPrint(n+2);
   }
}
```

Digit count from the given number using recursion.

```
#include<stdio.h>
int dgtcount(int n);
main()
{
        int n;
        printf("\n\n count the digits:\n");
        printf("----------------------------------------\n");
        printf(" Input  a number : ");
        scanf("%d",&n);
        printf("Number of digits :  %d",dgtcount(n));
}
int dgtcount(int n)
{
        static int ctr=0;
        if(n!=0)
        {
                ctr++;
                dgtcount(n/10);
        }
        return ctr;
}
```

Sum of digits from given number using recursion.

```
#include <stdio.h>
int digitSum(int num);
main()
{
   int n, sum;
   printf(" Input any number to find sum of digits: ");
   scanf("%d", &n);
   printf(" The Sum of digits of %d = %d\n\n", n, digitSum(n));
}
int digitSum(int n)
{
   if(n == 0)
      return 0;
   else
        return ((n % 10) + digitSum(n / 10));
}
```

What is Storage Class in C?

A storage class represents the visibility and a location of a variable. It tells from what part of code we can access a variable. A storage class in C is used to describe the following things:

- The variable scope.
- The location where the variable will be stored.
- The initialized value of a variable.
- A lifetime of a variable.
- Who can access a variable?

So, a storage class is used to represent the information about a variable.

NOTE: A variable is not only associated with a data type, its value but also a storage class.

There are total four types of standard storage classes. The table below represents the storage classes in C.

Storage class	Purpose
auto	It is a default storage class.
extern	It is a global variable.
static	It is a local variable which is capable of returning a value even when control is transferred to the function call.
register	It is a variable which is stored inside a Register.

Auto Storage Class

The variables defined using auto storage class are called as local variables. Auto stands for automatic storage class. A variable is in auto storage class by default if it is not explicitly specified.

The scope of an auto variable is limited with the particular block only. Once the control goes out of the block, the access is destroyed. This means only the block in which the auto variable is declared can access it.

A keyword auto is used to define an auto storage class. By default, an auto variable contains a garbage value.
Example, auto int age;
The program below defines a function with has two local variables

```
int add(void) {
  int a=13;
  auto int b=48;
    return a+b;
}
```

We take another program which shows the scope level “visibility level” for auto variables in each block code which are independently to each other:

```
#include <stdio.h>
#include <conio.h>
void autofun();
void main( )
{
   clrscr();
   autofun();
   getch();
}
void autofun()
{
     auto int j = 1;
  {
    auto int j= 2;
    {
   auto int j = 3;
              printf ( " %d ", j);
    }
    printf ( "\t %d ",j);
  }
  printf( "\t %d\n", j);
}
```

OUTPUT:

3 2 1

Extern Storage Class

Extern stands for external storage class. Extern storage class is used when we have global functions or variables which are shared between two or more than two blocks of a program.

The variables defined using an extern keyword are called as global variables. These variables are accessible throughout the program. Notice that the extern variable cannot be initialized it has already initialized while defining.

```
#include<stdio.h>
#include<conio.h>
void area();
extern int a=10, c=20;
void main()
{
  clrscr();
  area();
  getch();
}
void area()
{
  int da;
  da = a*c;
  printf("\nArea is: %d", da);
}
```

OR

```
#include<stdio.h>
#include<conio.h>
void area();
int a, c;
void main()
{
  clrscr();
  area();
  getch();
}
void area()
{
  int da;
  prinf("\nEnter length and breadth: ");
  scanf("%d%d", &a, &c);
  da = a*c;
  printf("\nArea is: %d", da);
}
```

OR

```
#include<stdio.h>
#include<conio.h>
void area();
extern int a=0, c=0;
void main()
{
  clrscr();
  area();
  getch();
}
void area()
{
  int da;
  prinf("\nEnter length and breadth: ");
  scanf("%d%d", &a, &c);
  da = a*c;
  printf("\nArea is: %d", da);
}
```

Static Storage Class

The static variables are used within function/ file as local static variables. They can also be used as a global variable

Static local variable is a local variable that retains and stores its value between function calls or block and remains visible only to the function or block in which it is defined.

Static global variables are global variables visible only to the file in which it is declared.

Example: static int count = 10;

Keep in mind that static variable has a default initial value zero and is initialized only once in its lifetime.

```
#include <stdio.h>
#include <conio.h>
void next(void); /* function declaration */
static int counter = 7; /* global variable */
void main()
{
        clrscr();
       while(counter<10)
       {
           next();
          counter++;
       }
       getch();
}
void next( void )
{
       static int iteration = 13; /* local static variable */
       iteration ++;
       printf("iteration=%d and counter= %d\n", iteration, counter);
}
```

Result:

```
iteration=14 and counter= 7
iteration=15 and counter= 8
iteration=16 and counter= 9
```

Register Storage Class

You can use the register storage class when you want to store local variables within functions or blocks in CPU registers instead of RAM to have quick access to these variables. For example, “counters” are a good candidate to be stored in the register.
Example: register int age;
The keyword **register** is used to declare a register storage class. The variables declared using register storage class has lifespan throughout the program.
The variables declared using register storage class has no default value. These variables are often declared at the beginning of a program.

```
#include<stdio.h>
#include<conio.h>
void series();
void main()
{
  clrscr();
  series();
  getch();
}
void series()
{
  register int n=1;
  int i;
  for(i=1; i<10; i++)
  {
        printf("%d\t", n);
        n+=2;
  }
}
```

Structure

Structure is defined as the collection of dissimilar data items treated as single unit. Each data item is called member of structure. The keyword **struct** is used to declare structure variable and type of structure variable is defined by **tag_name** of the structure. It is collection of heterogeneous data items treated as a single unit. Each data item is called **member** of structure.

Some major features of structure:

a. It can be treated as record that is collection of interrelated data fields having different data types.
b. It is possible to copy one structure variable to another by simply assignment operator (=).
c. It allows nested structure that is one structure inside another structure.
d. It is possible to pass structure variable as parameter to any function.
e. It is possible to define structure pointer also known as link list.

Syntax for structure declaration:

```
struct tag_name
{
        datatype member1;
        datatype member2;
        --------------------
        datatype memberN;
};
struct tag_name var1,
var2,……,varN;
```

```
struct tag_name
{
        datatype member1;
        datatype member2;
        --------------------
        datatype memberN;
}var1, var2,……,varN;
```

In the above syntax member1, member2 up to Nmember are the member of the structure and var1, var2, ….. varN are the N variables with datatype tag_name.

Example for structure variable:

```
struct student
{
        int rollNo;
        char fName[20];
        char lName[20];
};
struct student s1, s2, s3;
```

```
struct student
{
        int rollNo;
        char fName[20];
        char lName[20];
} s1, s2, s3;
```

In the above example student is the structure type and int rollNo, char fName[20] and char lName[20] are the member of structure. s1, s2 and s3 are the structure variables with datatype student.

Structure initialization:

Structure variable can be initialized quite similar to array. The following example show the initiation of structure process:

```
struct student
{
        int rollNo;
        char fName[20];
        char mName[20];
        char lName[20];
}s1={1, "Ajay", "Kumar", "Chaudhary"};
```

The alternative method to initialize structure variable as:

```
struct student s2={2, "Daya", "Ram", "Yadav" };
struct student s3={3, "Ashim", "Kumar", "Chaudhary" };
```

Accessing structure member:

A structure members can be accessed by using period(.) sign between structure variable and respective member.

Syntax:

variableName.member

Example:

s1.rollNo, s1.fName, s1.mName, s1.lName;

Sample example of structure

Program to store a student records like rollNo, name and address.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    struct student
    {
int rollNo;
     char name[50];
char adr[60];
    };
    struct student st;
clrscr();
    //code to input to structure
    printf("\nEnter the roll no of the student: ");
    scanf("%d", &st.rollNo);
    printf("\nEnter the name of the student: ");
    scanf("%s", st.name);
    printf("\nEnter the address of the student: ");
    scanf("%s", st.adr);
   //code to display after sorting
    printf("\n\nThe record of the sutdent: ");
    printf("\nRoll No\tName of Student\t\tAddress");
    printf("\n\n%d \t\t\t%s\t\t\t%s",st.rollNo, st.name, st.adr);
 getch();
}
```

Array of Structure

It is possible to declare an array; called array of structure. It is also known as the collection of structure variables having same set of members. Consider the following structure variable **s** with datatype student and array size **5**

Example

```
struct student{
        int rollNo;
        char fName[30];
        char lName[30];
}
struct student s[5]
```

Sample example of structure using loop

Program to store n students records like rollNo, name and address using array and loop.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    struct student
    {
         int rollNo;
        char name[50];
        char adr[60];
    }st[100];
     int i, n;
clrscr();
     //code to ask no of record
     printf("\nEnter the no of records n: ");
     scanf("%d", &n);
    //code to input to structure
     for(i=0; i<n; i++)
     {
          printf("\nEnter the roll no of the student: ");
          scanf("%d", &st[i].rollNo);
          printf("\nEnter the name of the student: ");
          scanf("%s", st[i].name);
          printf("\nEnter the address of the student: ");
          scanf("%s", st[i].adr);
     }
   //code to display records
printf("\n\nThe record of the sutdent: ");
    printf("\nRoll No\tName of Student\t\tAddress");
for(i=0; i<n; i++)
     {
          printf("\n\n%d \t\t\t%s\t\t\t%s",st[i].rollNo, st[i].name, st[i].adr);
     }
     getch();
}
```

Program to store n students records like name and address using array and loop. Print the sorted record on the basis of name.

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    struct student
    {
        char name[50], adr[60];
    };
    struct student st[100], temp;
    int i, j, n;
     clrscr();
    printf("\nEnter a number how many records you want to input: ");
    scanf("%d",&n);
    //code to input to structure
    for(i=0;i<n;i++)
    {
        fflush(stdin);
        printf("\nEnter the name of the student: ");
        gets(st[i].name);
        printf("\nEnter the address of the student: ");
        gets(st[i].adr);
    }
    //code to arrange in alphabetic order
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(strcmpi(st[i].name, st[j].name)>0)
            {
                temp=st[i];
                st[i]=st[j];
                st[j]=temp;
            }
        }
    }
    //code to display after sorting
    printf("\n\nThe record of the sutdent after sorting: ");
    printf("\nName of Student\t\tAddress");
    for(i=0;i<n;i++)
    {
        printf("\n\n%s\t\t\t%s",st[i].name,st[i].adr);
    }
    getch();
}
```

Program to store n students records like name, address and age using array and loop. Print the sorted record on the basis of age.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    struct student
    {
        char name[50], adr[60];
int age;
    };
    struct student st[100], temp;
    int i, j, n;
     clrscr();
    printf("\nEnter a number how many records you want to input: ");
    scanf("%d",&n);
    //code to input to structure
    for(i=0;i<n;i++)
    {
fflush(stdin);
        printf("\nEnter the name of the student: ");
        gets(st[i].name);
        printf("\nEnter the address of the student: ");
        gets(st[i].adr);
fflush(stdin);
        printf("\nEnter the age of the student: ");
        scanf("%d", &st[i].age);
    }
    //code to sort record in ascending order on the basis of age
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(st[i].age > st[j].age)
            {
                temp=st[i];
                st[i]=st[j];
                st[j]=temp;
            }
        }
    }
    //code to display after sorting
    printf("\n\nThe record of the sutdent after sorting: ");
    printf("\nName of Student\t\tAddress\t\tAge ");
    for(i=0;i<n;i++)
    {
        printf("\n\n%s\t\t\t%s\t\t%d", st[i].name,st[i].adr,st[i].age);
    }
    getch();
}
```

Program to store n students records like name, address and age using array and loop. Print the record on the basis of age between 20 and 30.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    struct student
    {
        char name[50], adr[60];
int age;
    };
    struct student st[100], temp;
    int i, j, n;
     clrscr();
    printf("\nEnter a number how many records you want to input: ");
    scanf("%d", &n);
    //code to input to structure
    for(i=0; i<n; i++)
    {
fflush(stdin);
        printf("\nEnter the name of the student: ");
        gets(st[i].name);
        printf("\nEnter the address of the student: ");
        gets(st[i].adr);
fflush(stdin);
        printf("\nEnter the age of the student: ");
        scanf("%d", &st[i].age);
    }
    //code to display after sorting
    printf("\nName of Student\t\tAddress\t\tAge ");
    for(i=0;i<n;i++)
    {
        if(st[i].age>=20 && st[i].age<=30)
        {
        printf("\n\n%s\t\t\t%s\t\t%d",st[i].name,st[i].adr,st[i].age);
     }
    }
    getch();
}
```

Passing structure to function

A function can be called by passing structure variable as parameter. Structure variable can be passed in various ways: member can be passed individually or entire structure can be passed once.

Example:

Write a program using function to calculate sum of two distance and distance is measured in the term of feet and inch by passing structure as a parameter.

```
#include<stdio.h>
#include<conio.h>
void sum(struct distance d1, struct distance d2);
void main(){
    struct distance{
            int feet, inch;
}d1, d2;
clrscr();
    printf("Enter first distance in feet and inch format:");
    scanf("%d%d", &d1.feet, &d1.inch);
    printf("Enter second distance in feet and inch format:");
    scanf("%d%d", &d2.feet, &d2.inch);
    sum(d1, d2);
    getch();
}
void sum(struct distance d1, struct distance d2){
    struct distance d;
    d.inch = (d1.inch+d2.inch)%12;
    d.feet = ((d1.feet+d2.feet)+(d1.inch+d2.inch))/12;
    printf("\nSum of two distances: %d Feet %d inch", d.feet, d.inch);
}
```

Nested structure

WAP to store name, address, post salary using nested structure.

```
#include<stdio.h>
#include<conio.h>
struct employee {
    char ename[100];
    char eaddress[100];
};
struct salarydetails {
    char post[100];
    float salary;
    struct employee detail;
}payinfo;
void main() {
    clrscr();
    printf("Enter employee name, address, post and salary : ");
    scanf("%s%s%s%f", payinfo.detail.ename, payinfo.detail.eaddress, payinfo.post,
    &payinfo.salary);
    printf("Employee Name : %s\n", payinfo.detail.ename);
    printf("Address : %s\n", payinfo.detail.eaddress);
    printf("Post: %s\n",payinfo.post);
    printf("Salary: %f", payinfo.salary);
}
```

Workout Programs

WAP to input n employees name, post and salary. Then search a record of an employee on the basis of name.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
        char name[30];
        char post[30];
        int salary;
}e[20];
void main()
{
        int i, n;
        char search_name[30];
        clrscr();
        printf("How many record would you like to store? ");
        scanf("%d", &n);
        printf("Enter name,post and salary of %d  employee: ", n);
        for(i=0; i<n; i++)
        {
                scanf("%s%s%d", e[i].name, e[i].post, &e[i].salary);
        }
        printf("Enter name to search: ");
        scanf("%s", search_name);
        printf("\nName\t Post\t Salary\n");
        for(i=0; i<n; i++)
        {
                if(strcmpi(e[i].name,search_name)==0)
                printf("%s\t %s\t %d\n",e[i].name,e[i].post,e[i].salary);
        }
        getch();
}
```

WAP to input n students id, name, address and age. Then search record on the basis of id.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct student
{
        int id;
        char name[30];
        char address[30];
        int age;
}s[20];
void main()
{
        int i, n, ser;
        clrscr();
        printf("How many record would you like to store? ");
        scanf("%d", &n);
        printf("Enter id, name, address and age %d students", n);
        for(i=0; i<n; i++)
        {
                scanf("%d%s%s%d", &s[i].id, s[i].name, s[i].address, &e[i].age);
        }
        printf("Enter id to search: ");
        scanf("%d", ser);
        printf("\nId\t Name\t Address\t Age\n");
        for(i=0; i<n; i++)
        {
                if(e[i].id == ser)
                printf("%d %s\t %s\t %d\n", s[i].id, s[i].name, s[i].address, s[i].age);
        }
        getch();
}
```

WAP to input n employees name, post and salary. Then search a record of an employee on the basis of first letter of name.

```
#include<stdio.h>
#include<conio.h>
struct employee
{
        char name[30];
        char post[30];
        int salary;
}e[100];
void main()
{
        int i, n;
        char search_name, ch;
        clrscr();
        printf("How many record would you like to store");
        scanf("%d", &n);
        printf("Enter name,post and salary of 20  employee: ");
        for(i=0;i<n;i++)
        {
                scanf("%s%s%d",e[i].name,e[i].post,&e[i].salary);
        }
        fflush(stdin);
        printf("Enter name's first alphabet to search: ");
        scanf("%c", &search_name);
        printf("Name\t Post\t Salary\n");
        for(i=0; i<n; i++)
        {
                ch = e[i].name[0];
                if(search_name == ch)
                {
                        printf("%s\t %s\t %d\n",e[i].name,e[i].post,e[i].salary);
                }
        }
        getch();
}
```

WAP to input any n student's name, grade, gender and marks in 5 subjects. Then print all students'name, grade, gender, total and percentage whose gender is female.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct student
{
	char name[30];
	int grade;
	char gender[10];
	int sub1,sub2,sub3,sub4,sub5,total;
	float per;
}s[100];
void main()
{
	int i, n;
	clrscr();
	printf("How many record would you like to store");
	scanf("%d", &n);
	printf("Enter name,grade,gender, marks in five subjects of %d students ", n);
	for(i=0;i<n;i++)
	{

	scanf("%s%d%s%d%d%d%d%d",s[i].name,&s[i].grade,s[i].gender,&s[i].sub1,&s[i].sub
2,&s[i].sub3,&s[i].sub4,&s[i].sub5);
	}
	for(i=0;i<n;i++)
	{
		s[i].total=s[i].sub1+s[i].sub2+s[i].sub3+s[i].sub4+s[i].sub5;
		s[i].per=(float)s[i].total/5;
	}
	printf("\n Female Records only \n");
	printf("Name\t Grade\t Gender\t Total\t Percentage\n");
	for(i=0;i<n;i++)
	{
		if(strcmpi(s[i].gender,"female")==0)
		{
			printf("%s\t %d\t %s\t %d\t %0.2f\n ",s[i].name, s[i].grade, s[i].gender,
s[i].total, s[i].per);
		}
	}
	getch();
}
```

Union

Union is the similar to the structure but it differs in its storage size. The union keyword is used to define **union.** In structure, each member has its own memory block whereas all members of union can share the same memory block of the largest member of the **union.** Therefore, union take less amount of memory and it can access only one member at a time.

Difference between Structure and Union

	Structure		Union
1.	It is the collection of the data item having dissimilar data types.	1.	Same as structure dissimilar data types.
2.	Memory size is determined by sum of memories allocated to all the members.	2.	Memory size is determined by largest member.
3.	Multiple members can be accessed simultaneously at a time.	3.	Only one member can be accessed at a time.
4.	The **struct** keyword is used to define structure.	4.	The **union** keyword is used to define union.
5.	**Syntax:** struct tag_name { datatype member1; datatype member2; ---------- datatype memberN; }var1,……,varN;	5.	**Syntax:** union tag_name { datatype member1; datatype member2; ---------- datatype memberN; }var1,……,varN;
6.	**Example:** struct data { char a; -> 1 byte int i; -> 4 byte float j; -> 4 byte }var1,……,varN; -> Memory size for var is 9 byte	6.	**Example:** union data { char a; -> 1 byte int i; -> 4 byte float j; -> 4 byte }var1,……,varN; -> Memory size for var is 4 byte

A program to show the size of union and structure

```
#include<stdio.h>
#include<conio.h>
struct student{
    char name[10];
    float marks;
    int roll;
} stu;
union student1{
    char name[10];
    float marks;
    int roll;
} un;
void main(){
    printf("\n Sturcture: %d", sizeof(stu));
    printf("\n Union: %d", sizeof(un));
getch();
}
```

Example

```
#include<stdio.h>
#include<conio.h>
union student {
    int id;
    char name[30];
    int grade;
} s;
void main()
{
    clrscr();
    printf("Enter student id: ");
    scanf("%d", &s.id);
    printf("Student ID: %d\n ", s.id);
    printf("Enter name : ");
    scanf("%s", s.name);
    printf("Name : %s\n", s.name);
    printf("Enter grade : ");
    scanf("%d", &s.grade);
    printf("Grade : %d\n", s.grade);
    getch();
}
```

Pointer

Pointer is a variable that points to another variable which means it contains the memory address of another variable and is declared as the pointer type. It contains only the memory location of the variable rather than its contents.

A pointer is a special type of variable that represent memory address of a variable rather than its value.

Features of Pointer

- Pointer saves memory space.
- Pointer assigns the memory space and also releases it which makes the best use of the available memory.
- The execution time is faster because data manipulation is done directly inside memory.
- Manipulation of data at different memory locations is easier.
- Dynamic memory allocation.

Declaring a pointer variable

To declare and refer a pointer variable, provides two special operators **&** and *** .**

- Address operator(ampersand): &
  This operator gives the memory address of the variable.
- Indirection operator: *
  This operator gives the value of the variable at the specified address.

Example:

i = 100

i	Memory address
100	1004

&i -> it points the memory address i.e., 1004
*i -> it points value of the variable i i.e., 100

Types of pointer variable declaration:

char *ptr, *name;	pointer to character type variable
int *ptr, *num;	pointer to integer type variable
float *ptr, *marks;	pointer to float type variable
char *address[50];	pointer to character array.

Example:

A sample program to demonstrate address and value.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int *p;
    int q=10;
    p = &q;
    clrscr();
    printf("The value of q is: %d", *p);
    printf("\nThe address of q is: %d", p);
    getch();
}
```

WAP to sum two numbers 10 and 5 using pointers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int *p, *q, a=10, b=5, sum;
    clrscr();
    p = &a;
    q = &b;
    sum = *p + *q;
    printf("Sum is: %d", sum);
    printf("\nSum is: %d", *p + *q);
    getch();
}
```

WAP to perform arithmetic calculations (sum, difference, multiplication and division) of two numbers using pointers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int n1, n2, *x, *y;
    clrscr();
    printf("Enter two numbers : ");
    scanf("%d%d",&n1,&n2);
    x=&n1;
    y=&n2;
    printf("Sum=%d\n", *x + *y);
    printf("Difference=%d\n", *x - *y);
    printf("Multiplication=%d\n", *x * *y);
    printf("Division=%d\n", *x / *y);
    printf("Reminder=%d\n", *x % *y);
    getch();
}
```

WAP to find odd or even number using pointers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num,*n;
    clrscr();
    printf("Enter any number: ");
    scanf("%d", &num);
    n=&num;
    if(*n % 2==0)
        printf("even");
    else
        printf("odd");
    getch();
}
```

Write C program to find sum and average of 'n' natural numbers using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int *p;
    int n, i, sum=0;
    float avg=0;
    clrscr();
    p=&n;
    printf("Enter value of n: ");
    scanf("%d", &n);
    p=&n;
    for(i=1; i<=*p; i++)
    {
        sum=sum+i;
    }
    avg=sum/(*p);
    printf("Sum=%d\n", sum);
    printf("Average=%f\n", avg);
    getch();
}
```

WAP to print hailstone (7, 22, 11, 34, 17 ..... 10th) terms using function. Pass pointer to function.

```
#include<stdio.h>
#include<conio.h>
void series (int *p);
void main()
{
    int *p;
    int n=10;
    p=&n;
    series(&n);
    getch();
}
void series(int *p)
{
    int a=7,i;
    for(i=1;i<=*p;i++)
    {
            printf("%5d",a);
            if(a%2==0)
                    a=a/2;
            else
                    a=a*3+1;
    }
}
```

Write C program to find factorial of a number using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int *p, n, i, fact=1;
    clrscr();
    printf("enter a number\n");
    scanf("%d", &n);
    p=&n;
    for(i=1; i<=*p; i++)
    {
        fact= fact*i;
    }
    printf("Factorial =%d", fact);
    getch();
}
```

WAP to Use array as a pointer to input 10 elements and print them.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int i, a[10], *n[10];
    clrscr();
    printf("Enter 10 numbers: ");
    for(i=0;i<10;i++)
    {
            scanf("%d", &a[i]);
            n[i] = &a[i];
    }
    for(i=0;i<10;i++)
    {
            printf("%d\n", *n[i]);
    }
    getch();
}
```

WAP to print factors of a number using pointer.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num,i;
    int  *p ;
    clrscr();
  printf(" Enter any number : ") ;
  scanf("%d",&num);
  p = &num;
  for(i=1;i<=*p;i++)
    {
            if(*p%i==0)
                    printf("%d\n ", i) ;
    }
 getch();
}
```

WAP to sort n numbers.

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int num[100], *num1[100], *n1, n,i, j, temp;
    clrscr();
    printf(" Enter the array size not more than 100 : ");
    scanf("%d",&n);
    n1 = &n;
    printf("Enter %d elements for array: ", n);
    for(i=0;i<*n1;i++)
    {
        scanf("%d",&num[i]);
        num1[i] = &num[i];
    }
    for(i=0; i<*n1; i++)
    {
        for(j=i+1; j<*n1; j++)
        {
            if(*num1[i] > *num1[j])
            {
                temp = *num1[i];
                *num1[i] = *num1[j];
                *num1[j] = temp;
            }
        }
    }
    printf(" Numbers in ascending order \n ");
    for(i=0;i<*n1;i++)
        printf(" %d\t ",*num1[i]);
    getch();
}
```

Pointer and Function

As we have studied about the function's arguments and parameters passing in function. They are passed in two ways:

1. **Call by value**

   In this method values of the variables are passed. By using this method values of variable are not affected by changing the values of **formal parameter.** When an argument is passed by value the data item is copied to the function. So, any alternation made to the data item within the function is not carried in to the calling function.

   Program to swap any two integers using call by value

```
#include<stdio.h>
#include<conio.h>
void swap(int a, int b);
void main()
{
        int a = 10, b= 20;
        clrscr();
        swap(a,b); // Actual parameter
        printf("Call by value\n");
        printf(" After swapping a = %d and b= %d ", a, b);
        getch();
}
void swap(int a, int b) //Formal parameter
{
        int t;
        t = a;
        a = b;
        b = t;
}
```

2. Call by reference

In this method address of the variables are passed. The content of that address can be accessed freely either within the function or within the calling function so, by using this method values of variable are by changing the values of **formal parameter.** Moreover, any changes that is made to the data items will be recognized in both the function and calling routine.

Program to show call by reference

```
#include<stdio.h>
#include<conio.h>
void swap(int *a, int *b);
void main()
{
        int a = 10, b= 20;
        clrscr();
        swap(&a, &b);
        printf("Call by reference\n");
        printf(" After swapping a = %d and b= %d ", a, b);
        getch();
}
void swap(int *a, int *b)
{
        int t;
        t = *a;
        *a = *b;
        *b = t;
}
```

File Handling

Concept of data file

In previous unit we use the function scanf( ) to take data from keyboard and similarly the function printf( ) to display data on screen. In such programs, data is temporarily stored in main memory during the program execution. When the program is recompiled, the entire data have been lost. Therefore, in order to store data permanently we need data file. A data file is simply a file in hard disk which is used to store data permanently.

A data file is defined as a collection of data or information which is permanently stored inside secondary memory as a single unit. We can read, write, append and delete data in data file as per our requirement.

C language support various functions to perform various operations such as: defining file, opening file, writing file, reding file, appending file and closing file.

File Handling Functions:

fopen( ): to create new data file.
fclose( ): to close data file.
getc( ): to read a character from data file.
putc( ): to write a character to data file.
getw( ): to read an integer from data file.
putw( ): to write an integer to data file.
fscanf( ): to read formatted data from data file.
fprintf( ): to write formatted data to data file.
fread( ): to read record from data file.
fwrite( ): to write record to data file.

Basic Input/Output Functions:

scanf( ) : to input formatted data from the keyboard.
printf( ) : to display formatted data on screen.
getchar( ) : to input a single character from the keyboard.
putchar( ) : to display a character on the screen.
gets( ) : to input string from the keyboard.
puts( ) : to display string on the screen.

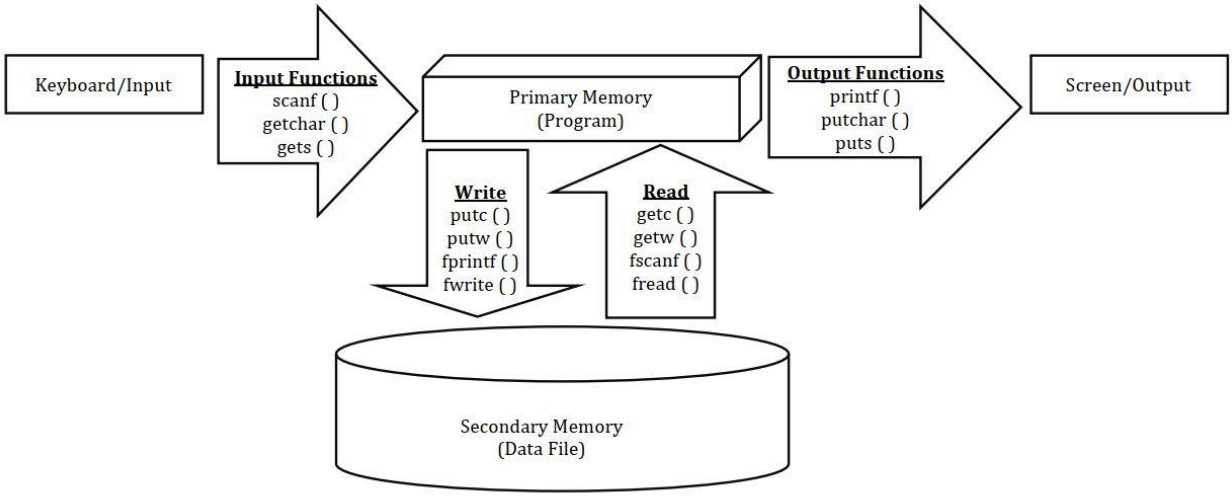


Figure: Input/Output functions and read/write function

Defining and opening Data File

Syntax:

FILE *fp;

fp = fopen("filename.extension", "mode");

The keyword **FILE** is used to declare file data type. The first statement defines file pointer variable with data type **FILE**. The second statement create file with name 'filename' and returns memory address that is assigned to the file pointer **fp**. The mode defines the purpose of the file which can be one of the followings:

- r : open a file in read mode
- w : opens or create a text file in write mode
- a : opens a file in append mode
- r+ : opens a file in both read and write mode
- a+ : opens a file in both read and write/append mode
- w+ : opens a file in both read and write mode

Compiler automatically assigns special character at the end of the file EOF.

Closing Data File

Syntax:

fclose(fp);

General guidelines for solving file related problems:

Step 1: Define file pointer variable with type FILE either inside or outside void main function.
FILE *fp;

Step 2: For the first time create new data file using fopen function in write mode.
fp = fopen("filename.extension", "w");

Step 3: Take data from keyboard using input functions: **scanf ( ), gets ( ), getchar ( )** and store data to the file using write functions: **fprintf ( ), putc ( ), putw ( ) and fwrite ( )**.

Step 4: During read operations, first open the data file in read mode.

fp = fopen("filename.extension", "r");

Step 5: Read the data from data file using functions**: fscanf ( ), getc ( ), getw ( )** and **fread ( )** and display the data on the screen using output functions: **printf ( ), puts ( )** and **putchar ( ).**

Step 6: Finally close the data file using function fclose( ).

fclose (fp);

putc( ) and getc( ) functions

putc( ) function is used for writing a character into a file.

Syntax:

putc (char ch, FILE *fp);

Example:

```
FILE *fp;
char ch;
putc(ch, fp);
```

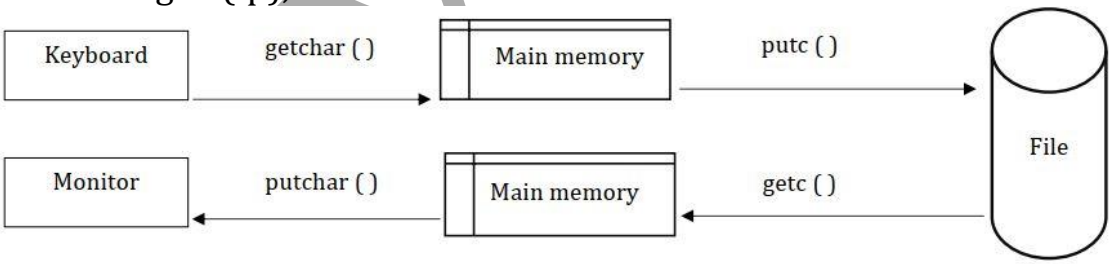
getc( ) function is used to read a character from file.

Syntax:

char getc (FILE *fp);

Example:

```
FILE *fp;
char ch;
ch = getc(fp);
```



Program using getc ( ) and putc ( ).

```
#include<stdio.h>
#include<conio.h>
void main()
{
  char ch;
  FILE *fp;
  clrscr();
  fp=fopen("std1.txt","w"); //opening file in write mode
  printf("enter the text press cntrl z to stop");
  while((ch = getchar())!=EOF)
  {
     putc(ch,fp); // writing each character into the file
```

```
  }
  fclose(fp);
  fp=fopen("std1.txt","r");
  printf("text on the file:");
  while ((ch=getc(fp))!=EOF)
   { // reading each character from file
      putchar(ch); // displaying each character on to the screen
   }
  fclose(fp);
  getch();
}
```

putw( ) and getw( ) functions

putw( ) function is used for writing a number into file.

Syntax

putw (int num, FILE *fp);

Example

```
FILE *fp;
int num;
putw(num, fp);
```

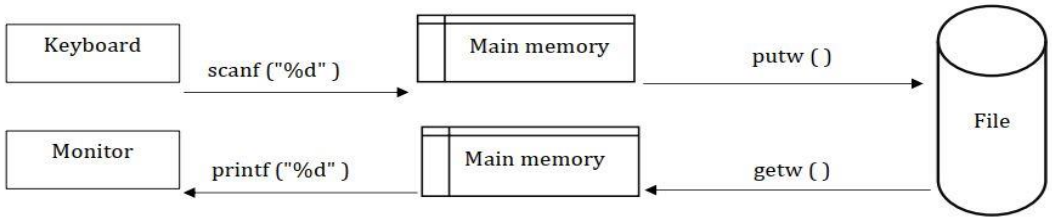
getw( ) function is used for reading a number from a file.

Syntax

int getw (FILE *fp);

Example

```
FILE *fp;
int num;
num = getw(fp);
```



Program using getw ( ) and putw ( ).

```
#include<stdio.h>
#include<conio.h>
void main( ){
  FILE *fp;
  int i;
  clrscr();
  fp = fopen ("num.txt", "w");
  for (i =1; i<= 10; i++)   {
    putw (i, fp);
   }
  fclose (fp);
  fp =fopen ("num.txt", "r");
  printf ("file content is");
  for (i =1; i<= 10; i++){
    //i= getw(fp);
    printf ("%d\t", getw(fp));
   }
  fclose (fp);
  getch();
```

```
}
```

fprintf( ) and fscanf( ) functions

fprintf() function is used to write set of characters into file. It sends formatted output to a stream.

Syntax

fprintf(fp, "format specifier", variable)

Example

```
FILE *fp;
fp = fopen("file.txt", "w");//opening file
fprintf(fp, "Hello file by fprintf...\n");//writing data into file
```

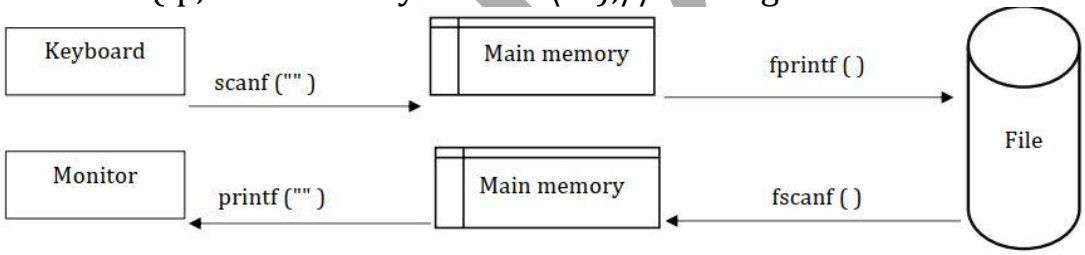
fscanf() function is used to read formatted data from file. It reads a data from the file and returns EOF at the end of file.

Syntax

fscanf(fp, "format specifier", variable)

Example

```
FILE *fp;
fp = fopen("file.txt", "r");//opening file
fscanf(fp, "Hello file by fscanf...\n");//reading data from file
```



Program using fscanf ( ) and fprintf ( ).

```
#include <stdio.h>
#include <conio.h>
void main()
{
  FILE *fp;
  int id;
  char name[30];
  float salary;
  clrscr();
  fp = fopen("emp.txt", "w");/*  open for writing */
  printf("Enter the id, Name and Salary \n");
  scanf("%d %s %f", &id, name, &salary);
  fprintf(fp, "%d %s %f", id, name, salary);
  fclose(fp);
  fp = fopen("emp.txt", "r");/*  open for reading */
  fscanf(fp, "%d %s %f", &id, name, &salary);
  printf("Id: \t Name \t Salary \n");
  printf("%d \t %s \t %f ", id, name, salary);
```

```
  fclose(fp);
  getch();
}
```

The **fread() function** reads the entire record at a time.

Syntax

fread(&structure variable, sizeof (structure variable), no of records, file pointer);

Example

```
struct emp{
  int eno;
  char ename [30];
  float sal;
} e;
FILE *fp;
fread (&e, sizeof (e), 1, fp);
```

The **fwrite() function** writes an entire record at a time.

Syntax

fwrite(&structure variable , size of structure variable, no of records, file pointer);

Example

```
struct emp{
  int eno:
  char ename [30];
  float sal;
} e;
FILE *fp;
fwrite (&e, sizeof(e), 1, fp);
```

Program using fread ( ) and fwrite ( ).

```
#include<stdio.h>
#include<conio.h>
struct student{
  int sno, age;
   char sname [30];
};
void main()
{
  struct student s;
  int i;
  FILE *fp;
  char ch='y';
  clrscr();
  fp = fopen ("student1.txt", "w");
  while(ch=='y' || ch=='Y')
  {
    printf("Serial number:");
    scanf("%d", &s.sno);
    printf("student name:");
    fflush(stdin);
    gets(s.sname);
    printf("student age:");
    fflush(stdin);
    scanf("%d", &s.age);
    fwrite(&s, sizeof(s), 1, fp);
    printf("Do you want to add more: y/n");
    fflush(stdin);
    ch = getche();
  }
  fclose (fp);
  fp = fopen ("student1.txt", "r");
 printf ("SN\t Name\t Age\n");
  while(fread(&s, sizeof(s), 1, fp)==1)
  {
      printf("%d\t %s\t %d\n", s.sno, s.sname, s.age);
  }
  fclose(fp);
  getch( );
}
```

WAP to store book's name, edition and price in a data file "book.txt" using user choice.

```
#include<stdio.h>
#include<conio.h>
void main()
{
       char name[30];
       int edition;
       float price;
       char ch='y';
       FILE *fp;
       clrscr();
       fp=fopen("book.txt","w");
       while(ch=='y')
      {
           printf("Enter book name : " );
           scanf("%s", name);
           printf("Enter book edition : " );
           scanf("%d", &edition);
           printf("Enter book price : " );
          scanf("%f", &price);
          fprintf(fp,"%s\t %d\t %f\n", name, edition, price);
          printf("Add more records (y/n) : ");
          scanf(" %c",&ch);
     }
     fclose(fp);
     fp=fopen("book.txt","r");
    printf("Book Name \t Edition \t Price\n");
    while(fscanf(fp,"%s%d%f", name, &edition, &price)!=EOF)
    {
              printf("%s\t\t %d\t %.2f\n ", name, edition, price);
     }
    fclose(fp);
    getch();
}
```

WAP to input students' name, grade and marks in five subjects, store records with total and percentage in a data file "student.dat". Print records of students who have percentage >=60.

```
#include<stdio.h>
#include<conio.h>
struct student
{
	char name[30];
	int grade;
	int sub1,sub2,sub3,sub4,sub5,total;
	float per;
}s[10];
void main()
{
	int i,n;
	FILE *fp;
	clrscr();
	printf("How many records : ");
	scanf("%d",&n);
	fp=fopen("student.dat","w");
	for(i=0;i<n;i++)
	{
		printf("Enter name : ");
		scanf("%s",s[i].name);
		printf("Enter grade : ");
		scanf("%d",&s[i].grade);
		printf("Enter sub1 mark: ");
		scanf("%d",&s[i].sub1);
		printf("Enter sub2 mark: ");
		scanf("%d",&s[i].sub2);
		printf("Enter sub3 mark: ");
		scanf("%d",&s[i].sub3);
		printf("Enter sub4 mark: ");
		scanf("%d",&s[i].sub4);
		printf("Enter sub5 mark: ");
		scanf("%d",&s[i].sub5);
		s[i].total=s[i].sub1+s[i].sub2+s[i].sub3+s[i].sub4+s[i].sub5;
		s[i].per=(float)s[i].total/5;
		fprintf(fp,"%s %d %d %d %d %d %d %d %f\n",s[i].name, s[i].grade, s[i].sub1,
s[i].sub2, s[i].sub3, s[i].sub4, s[i].sub5, s[i].total, s[i].per);
	}
	fclose(fp);
	printf("\n-----output-----\n");
	fp=fopen("student.dat","r");
	printf("Name\t Grade\t Sub1\t Sub2\t Sub3\t Sub4\t Sub5\t Total\t Per\n");
	while(fscanf(fp,"%s %d %d %d %d %d %d %d %f\n ",s[i].name, &s[i].grade, &s[i].sub1,
&s[i].sub2, &s[i].sub3, &s[i].sub4, &s[i].sub5, &s[i].total, &s[i].per)!=EOF)
	if(s[i].per>=60)
	{
```

```
                printf("%s\t %d\t %d\t %d\t %d\t %d\t %d\t %d\t %.2f\n",s[i].name, s[i].grade,
s[i].sub1, s[i].sub2, s[i].sub3, s[i].sub4, s[i].sub5, s[i].total, s[i].per);
        }
 getch();
}
```

Write C program to show the concept of rename() and remove() functions.

The C library function **rename(old_filename, new_filename)** causes the filename referred to by old_filename to be changed to new_filename.

The C library function **remove(filename)** removes the file created previously.

```
#include<stdio.h>
#include<conio.h>
void main()
{
 FILE *fp;
 char name[30];\
clrscr();
 fp=fopen("oldfile.txt","w");
 rename("oldfile.txt","newfile.txt");
 remove("newfile.txt");
}
```

Write C program to show concept of ftell(), and rewind() functions.

ftell() - It tells the byte location of current position in file pointer.

rewind() - It moves the control to beginning of a file.

```
#include<stdio.h>
#include<conio.h>
 void main ()
{
  char name [30];
  int age, length;
  FILE *fp;
  fp = fopen ("trinity.txt","w");
  printf("Enter your name: \n");
  scanf("%s",name);
  printf("Enter your age: \n");
  scanf("%d",&age);
  fprintf (fp, "%s %d", name, age);
  length = ftell(fp);
  rewind (fp);
  fscanf (fp, "%s", name);
  fscanf (fp, "%d", &age);
  fclose (fp);
  printf ("Name= %s \n Age= %d \n",name,age);
  printf ("Total number of characters in file is %d \n", length);
 getch();
}
```

Write C program to show concept of fseek() functions.

fseek() - It is used to moves the reading control to different positions using fseek function.

```
#include<stdio.h>
#include<conio.h>
void main ()
 {
        FILE *fp;
        clrscr();
        fp = fopen("test.txt", "r");
        // Moving pointer to end
        fseek(fp, 0, SEEK_END);
        // Printing position of pointer
        printf("%ld", ftell(fp));
        getch();
 }
```

WAP to accept any to numbers, store in sorted form and display it.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
  FILE *fp;
  int i, j, t, a[10];
  clrscr();
  printf ("Input any 10 numbers: ");
  for (i =0; i< 10; i++)
  {
        scanf("%d", &a[i]);
  }
  for (i =0; i< 10; i++)
  {
        for(j=i+1; j<10; j++)
        {
              if(a[i]>a[j])
              {
                    t = a[i];
                    a[i] = a[j];
                    a[j] = t;
              }
        }
  }
  fp = fopen ("num.txt", "w");
  for (i =1; i<= 10; i++)
   {
    fprintf(fp, "%d", a[i]);
   }
  fclose (fp);
  fp =fopen ("num.txt", "r");
  printf ("file content is");
  for (i =0; i< 10; i++)
   {
    fscanf(fp,"%d", a[i]);
    printf ("%d\t",a[i]);
   }
  fclose (fp);
  getch();
}
```

WAP to store id, name, post and salary of n employee in a data file "emp.txt". Then search the record on the basis of name.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
	char name[30];
	char post[30];
	int salary;
}e[20];
void main()
{
	int i, n;
	FILE *fp;
	char search_name[30];
	clrscr();
	fp = fopen("emp.txt", "w");
	printf("How many record would you like to store? ");
	scanf("%d", &n);
	printf("Enter name,post and salary of %d  employee: ", n);
	for(i=0; i<n; i++)
	{
		scanf("%s%s%d",e[i].name,e[i].post,&e[i].salary);
		fprintf(fp, "%s%s%d",e[i].name,e[i].post,e[i].salary);
	}
	fclose(fp);
	fp = fopen("emp.txt", "r");
	printf("Enter name to search: ");
	scanf("%s", search_name);
	printf("\nName\t Post\t Salary\n");
	for(i=0; i<n; i++)
	{
		fscanf(fp, "%s%s%d", e[i].name, e[i].post, &e[i].salary);
		if(strcmpi(e[i].name,search_name)==0)
		printf("%s\t %s\t %d\n",e[i].name,e[i].post,e[i].salary);
	}
	getch();
}
```

WAP to store id, name, post and salary of n employee in a data file "emp.txt". Then search the record on the basis of id.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
	int id;
	char name[30];
	char post[30];
	int salary;
}e[20];
void main()
{
	int i, n, ser;
	FILE *fp;
	clrscr();
	fp = fopen("emp.txt", "w");
	printf("How many record would you like to store? ");
	scanf("%d", &n);
	printf("Enter id, name, post and salary of %d  employee: ", n);
	for(i=0; i<n; i++)
	{
		scanf("%d%s%s%d",&e[i].id, e[i].name,e[i].post,&e[i].salary);
		fprintf(fp, "%d%s%s%d",e[i].id, e[i].name,e[i].post,e[i].salary);
	}
	fclose(fp);
	fp = fopen("emp.txt", "r");
	printf("Enter id to search: ");
	scanf("%d", &ser);
	printf("\nId\tName\t Post\t Salary\n");
	for(i=0; i<n; i++)
	{
		fscanf(fp, "%d%s%s%d", &e[i].id, e[i].name, e[i].post, &e[i].salary);
		if(e[i].id == ser)
		printf("%d\t%s\t%s\t%d\n",e[i].id, e[i].name,e[i].post,e[i].salary);
	}
	getch();
}
```

WAP to input n employees name, post and salary in a data file 'emp.txt'. Then search a record of an employee on the basis of first letter of name.

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
struct employee
{
        char name[30];
        char post[30];
        int salary;
}e[20];
void main()
{
        int i, n;
        FILE *fp;
        char ch, search_name;
        clrscr();
        fp = fopen("emp.txt", "w");
        printf("How many record would you like to store? ");
        scanf("%d", &n);
        printf("Enter name,post and salary of %d  employee: ", n);
        for(i=0; i<n; i++)
        {
                scanf("%s%s%d",e[i].name,e[i].post,&e[i].salary);
                fprintf(fp, "%s%s%d",e[i].name,e[i].post,e[i].salary);
        }
        fclose(fp);
        fp = fopen("emp.txt", "r");
        fflush(stdin);
        printf("Enter first alphabet of name to search: ");
        scanf("%c", &search_name);
        printf("\nName\t Post\t Salary\n");
        for(i=0; i<n; i++)
        {
                fscanf(fp, "%s%s%d", e[i].name, e[i].post, &e[i].salary);
                ch = e[i].name[0];
                if(search_name == ch)
                {
                        printf("%s\t %s\t %d\n",e[i].name,e[i].post,e[i].salary);
                }
        }
        getch();
}
```

WAP to implement random file handling functions.

```
#include<stdio.h>
main()
{
	FILE *fp;
	char data[100];
	fp = fopen("newfile.txt","w+");
	fputs("Hello there, I am Dayaram Yadav.", fp);
	printf("\nTotal Characters: %d", ftell(fp));
	rewind(fp);
	fgets(data, sizeof(data),fp);
	printf("\nFrom 1st position: %s", data);
	fseek( fp, 12, SEEK_SET);
	printf("\nCurrent position: %d", ftell(fp));
	fgets(data, sizeof(data),fp);
	printf("\nAfter 12th position: %s", data);
	fseek( fp, -2, SEEK_CUR);
	printf("\nCurrent position: %d", ftell(fp));
	fgets(data, sizeof(data),fp);
	printf("\n+2 more position: %s", data);
	fseek( fp, -25, SEEK_END);
	printf("\nCurrent position: %d", ftell(fp));
	fgets(data, sizeof(data),fp);
	printf("\nBack from last position: %s", data);
	rewind(fp);
	fseek( fp, 10, SEEK_SET);
	printf("\nCurrent position: %d", ftell(fp));
	fgets(data, 20,fp);
	printf("\nAccessing substring : %s", data);
	fclose(fp);
}
```

ꕥ THE END ꕥ