Chapter 10

FILE PROCESSING

LEARNING OBJECTIVES

After reading this chapter the reader will be able to

- *understand the need of data file.*
- *learn the operations on files.*
- *use different data input/output functions on files.*
- *write simple programs for creating a data file, reading data from the file for processing and display.*

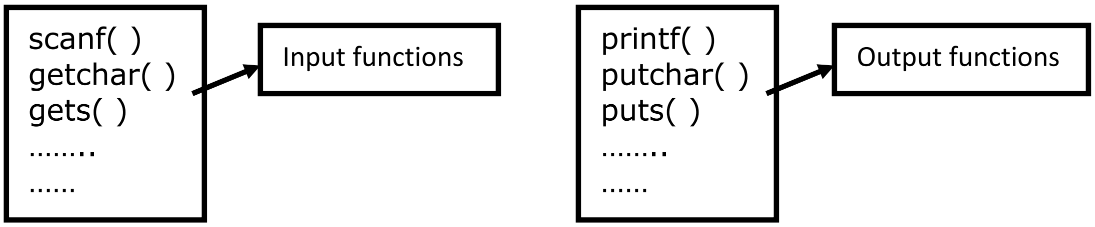
10.1 INTRODUCTION

What is a file?

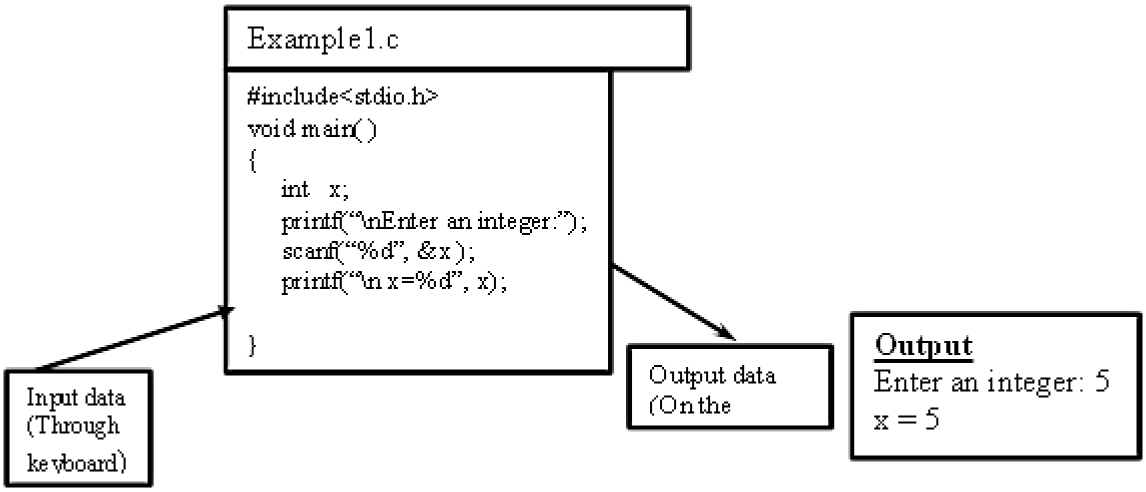
With any C program there are two major operations between the user and the program which are

- To send data (input)into the program and
- To get results data (output) from the program

So far we have done these two operation using the following input/output functions



For example the following program reads input data and prints the out data

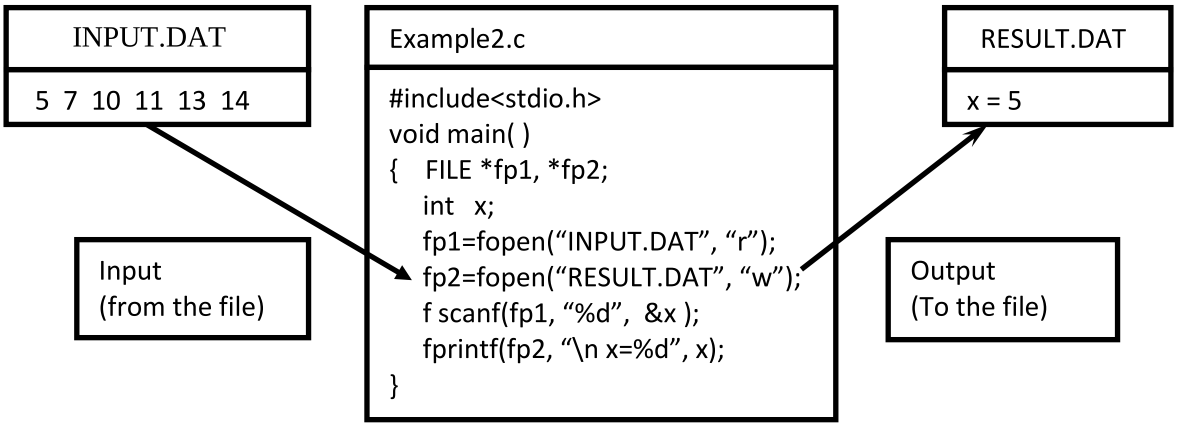


```
#include<stdio.h>
void main()
{
   int  x;
   printf("\nEnter an integer:");
   scanf("%d", &x);
   printf("\n x=%d", x);

}
```

```
Enter an integer: 5
x = 5
```

In the above program the data that is input through keyboard and that is output on the screen is temporary, because the data is stored in main memory. So whenever the application is closed or the system is switched off the entire data on main memory is lost. So it is not possible to see the data in future. Since these I/O functions use keyboard and screen, these functions are console oriented I/O functions. Using these functions it is not possible to store data permanently in the system.
But in real application like student database, employee database and bank data, it is required to store data permanently in the system. It is possible to store data permanently in the system by using secondary memory. The console I/O functions do not support to handle data with secondary memory.
It is therefore necessary to have a flexible approach to operate input and output operations between a file in secondary memory and the program. This can be achieved by using FILE concept.
For example assume that two files "INPUT.DAT" and "RESULT.DAT" in secondary memory and the input/output operations between these files and the program shown as follows.



```
#include<stdio.h>
void main( )
{   FILE *fp1, *fp2;
    int  x;
    fp1=fopen("INPUT.DAT", "r");
    fp2=fopen("RESULT.DAT", "w");
    f scanf(fp1, "%d",  &x );
    fprintf(fp2, "\n x=%d", x);
}
```

where
fscanf( ) – is file input function used to read data from the file to the program
fprintf( ) – is file output function used to write data to the file from the program

With this explanation we can define a file
A **file** is some reserved space on the disk where a group of related data is stored permanently in the system.
Example 10.1: Student data input with the file name "STUDENT.IN"

STUDENT.IN				
RNO	NAME	SUB1	SUB2	SUB3
10	Ramu	18	15	16
11	Gopi	20	18	15

Example 10.2: Student data output with the file name "OUTPUT.DAT"

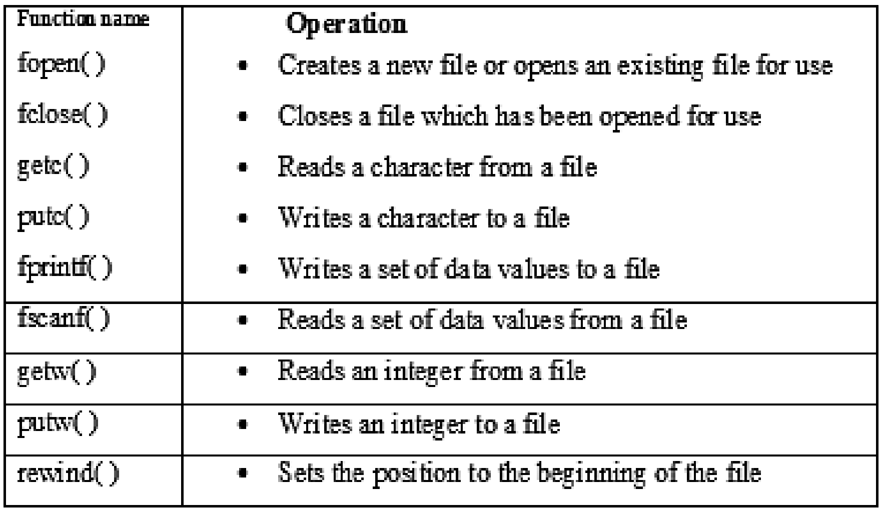
OUTPUT.DAT						
RNO	NAME	SUB1	SUB2	SUB3	TOTAL	AVG
10	Ramu	18	15	16	49	16.33
11	Gopi	20	18	15	53	17.67

10.2 FILE OPERATIONS

The different basic file operations are

- Naming a file
- Creating / Opening a file
- Closing a file
- Writing data to a file
- Reading data from a file

C supports a number of functions that have the ability to perform basic file operations, which includes:



Function name	Operation
fopen( )	• Creates a new file or opens an existing file for use
fclose( )	• Closes a file which has been opened for use
getc( )	• Reads a character from a file
putc( )	• Writes a character to a file
fprintf( )	• Writes a set of data values to a file
fscanf( )	• Reads a set of data values from a file
getw( )	• Reads an integer from a file
putw( )	• Writes an integer to a file
rewind( )	• Sets the position to the beginning of the file

10.2.1 Creating a new file / Opening an existing file

To create a new file or to open an existing file the function **fopen( )** is used.
fopen( ) is a file function used to create a new file or to open an existing file. Its general form is

```
fp = fopen ( "filename", "mode" );
```

where
fp – is a file pointer which is of structure data type FILE defind in stdio.h
The general syntax to declare the file pointer is

```
FILE *file-pointer;
```

For example,

```
FILE *p1, *p2, *fp;
```

- **"filename"** – is the name of the file and is a string of characters that make up a valid filename for the operating system. It may contain two parts, a primary name and an optional period with the extension. For example,

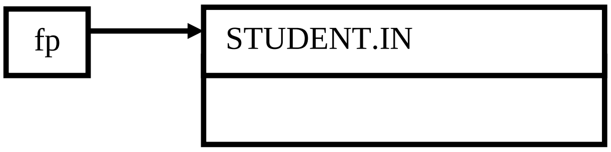
  "input.data"
  "STUDENT.DAT"
  "RESULTS.OUT"

- **"mode"** – specifies the purpose of opening the file. The mode of the file may be one of the following

"w"	-- open a new file for writing data
"r"	-- open an existing file for reading data
"a"	-- open an existing file for appending data to it
"r+"	-- open an existing file for both reading and writing data
"w+"	-- same as "w" except both for reading and writing
"a+"	-- same as "a" except both for reading and appending

Example10.3: Create a new file **"STUDENT.IN"** for writing

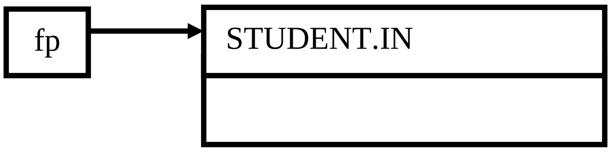
```
 #include<stdio.h>
void main( )
{
    FILE *fp;
    fp = fopen( "STUDENT.IN", "w" );
    ……………………
    ……………………
}
```



A new file with the name "STUDENT.IN" is created. If the file already exists, contents of the file are deleted.

Example10.4: Open an existing file **"STUDENT.IN"** for data reading

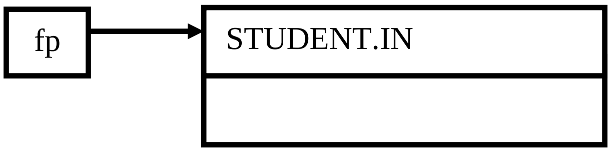
```
#include<stdio.h>
void main( )
{
   FILE *fp;
   fp = fopen( "STUDENT.IN" , "r" );
   ………………
   ……………………
 }
```



An existing file with the name "STUDENT.IN" is opened for reading data. If the file does not exist an error will occur.

Example10.5: open the file **"STUDENT.IN"** for appending data

```
#include<stdio.h>
void main( )
{
    FILE *fp;
    fp = fopen( "STUDENT.IN" , "a" );
    ……………………
    ……………………
}
```



An existing file with the name "STUDENT.IN" is opened for appending data to it .If the file does not exist a new file with the name "STUDENT.IN" is opened.

10.2.2 Closing a file

A file must be closed as soon as all operations on it have been completed. The function **fclose()** is used to close the file that has been opened earlier.

fclose( ) is the counterpart of **fopen( )**. Closing file means delinking the file from the program and saving the contents of the file. Its general form is

$$\boxed{\textbf{fclose(fp);}}$$

This would close the file associated with the file pointer **fp**.

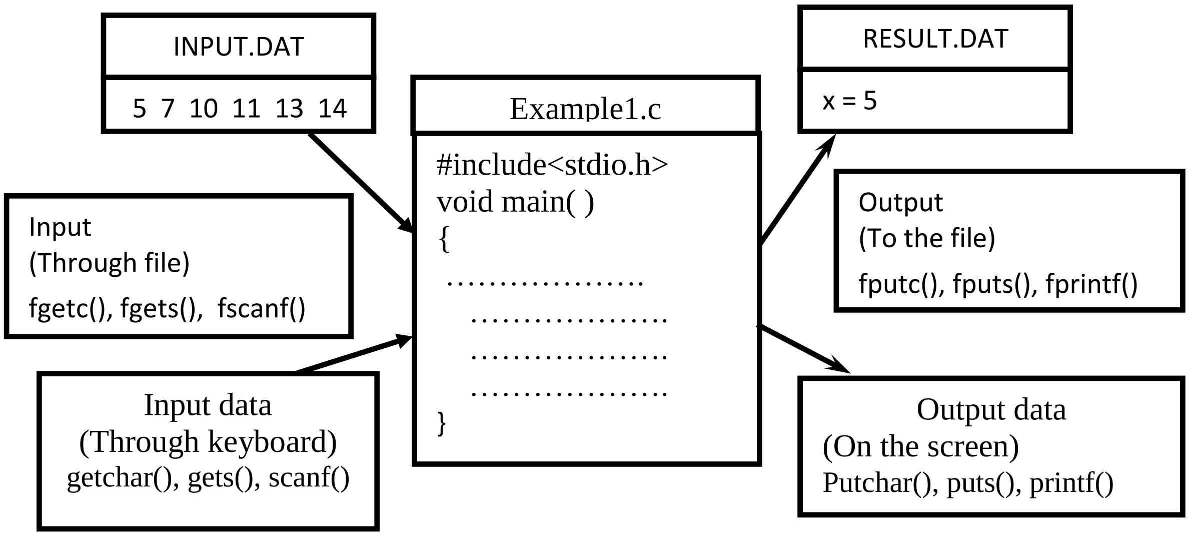
Example10.6

```
#include<stdio.h>
void main( )
{
    FILE *p1 *p2;
    p1=fopen (“Input”,”w”);
    p2=fopen (“Output”,”r”);
    ….
    …
    fclose(p1);
    fclose(p2);
 }
```

In the above example two files are opened and will be closed after all operations on them have been completed. Once a file is closed its file pointer can be used to open another file.

10.3 FILE INPUT/OUTPUT FUNCTIONS

Input to a program may be through keyboard or from a file and output is to the screen or to a file. Programs using files may involve the combinations of these different input/output operations and shown as follows:



After a file is opened, we can read data stored in the file or write new data to it depending on the mode of opening. C standard library supports a good number of functions which can be used for performing I/O operations. File I/O functions are broadly classified into two types.

1. High level file I/O functions
2. Low level file I/O functions

High level file I/O functions are basically C standard library functions and are easy to use. Most of the C programs handling files use these because of their simple nature. Low level file I/O functions are file related system calls of the underlying operating system. These are relatively more complex in nature when compared to high level file I/O functions but efficient in nature. The following are the high level file I/O functions

- Character data oriented functions
- String data oriented functions
- Mixed data oriented functions

`10.3.1 Character data oriented functions

- **fputc( )** : This function is used to write a single character to a file. Its general form is

$$\boxed{\textbf{fputc(character/char-variable, fp);}}$$

Where the character/char-variable represents a character and fp is a pointer to the FILE. The function writes the character or content of char-variable to the file pointed by fp.

Program10.1: Program to create a text file named TEXT.DAT

```
#include<stdio.h>
#include<conio.h>
void main()
{
    FILE *fp;
    char ch;
    fp=fopen("TEXT.DAT","w");
    if(fp==NULL)
     {
        printf("file is not opened\n");
        exit(0);
     }
     while((ch=getchar())!=EOF)
      fputc(ch,fp);
     fclose(fp);
 }
```

Text is read character by character from the keyboard and is written to the file. The end of the text is indicated by entering an EOF character, which is control-Z in the DOS environment.(this may be control-D on other environments).After the EOF is encountered the file is closed.

- **fgetc( )** : This function is used to read a single character from a file. Its general form is

$$\boxed{\textbf{char-variable = fgetc(fp);}}$$

The function reads a character from the file denoted by **fp** and returns the character value, which is collected by the **char-variable**.

Program10.2: A file named TEXT.DAT is on the disk. Write a program to read the text from the file and display the text on the monitor after converting to upper case.

```
#include<stdio.h>
#include<conio.h>
#include<ctype.h>
void main()
{
FILE *fp;
char ch;
fp=fopen(“TEXT.DAT”,”r”);
if(fp==NULL)
{
     printf(“Unable to open the file”);
     exit(0);
}
printf(“The text stored on the disk after converting to upper case is \n”);
while((ch==fgetc(fp))!=EOF)
      putchar(toupper(ch));
fclose(fp);
}
```

The file TEXT.DAT is opened for reading. The text is read character by character ,converted to upper case and is displayed on the screen. Reading is terminated when fgetc() encounters the end of file mark EOF.

Program10.3:A file named INTEXT contains a text .Write a program to copy the text from this file to the file OUTTEXT.

```
#include<stdio.h>
void main()
{
    File *fp1,*fp2;
    char ch;
    if((fp1=fopen(“INTEXT”,”r”))==NULLl)
 {
    prinf(“Unable to open the file INTEXT\n”);
    exit(0);
 }
if((fp2=fopen(“OUTEXT”,”w”))==NULL)
{
     printf(“Unable to open the file OUTTEXT\n”);
     exit(1);
 }
while((ch=fgetc(fp1))!=EOF)
     fputc(ch,fp2);
fclose(fp1);
fclose(fp2);
}
```

Program10.4: Program to count the number of words in a text stored in a file. Words are separated by one or more whitespace characters i.e. by a space, a tab or a new line.

```
#include<stdio.h>
void main()
{
     FILe *fp;
```

```
      char curch;infile[20];              //curch represent the current character
      int nw=0,flag=0;
      printf(“enter the name of the file containing text:\n”);
      gets(infile);
      if((fp=fopen(infile,”r”))==NULL)
      {
        printf(“Unable to open the file\n”);
        printf(“\n”);
      }
     while((curch=fgetc(fp))!=EOF)
    {
       if(curch==’ ’||curch==’\t’||curch==’\n’)
       flag=0;
       else
       {
        if(flag==0)
        {
           nw++;
           flag=1;
        }
       }
     }
      printf(“the number of words=%d\n”,nw);
}
```
The variable flag is used to identify the previous character. If flag=0,previous character is white space otherwise it is not white space. Previous character is white space and current character is non-white space indicate the beginning of a word and hence word count **nw** is incremented by one.

Program 10.5 Program to count number of characters and lines in a text stored in a file.
```
        #include<stdio.h>
        void main()
        {
          int nc=0,nl=0;
          FILE *fp;
          char  curch,prech;                        //curch represent current character
        if((fp=fopen(“TEXT”,”r”))==NULL)   //prech represent previous character
        {
           print(“unable to open the file\n”);
           exit(0);
        }
        while((curch=fgetc(fp))!=EOF)
        {
           if(curch!=’\n’)
              ++nc;
           else
              ++nl;
           prech=curch;
        }
        if(prech!=’\n’)
             ++nl;
        printf(“number of characters=%d\n”,nc);
        printf(“number of lines=%d\n”,nl);
        fclose(fp);
        }
```

The variable curch represent the character read in and prech represent the previous character. The variable prech is used to know whether the text ends with new line. If it does not end with new line after the control exits from the loop nl must be incremented by1 to count the last line of the text.
The statement

```
        if(prech!='\n')
            ++nl;
```

is written for this purpose.

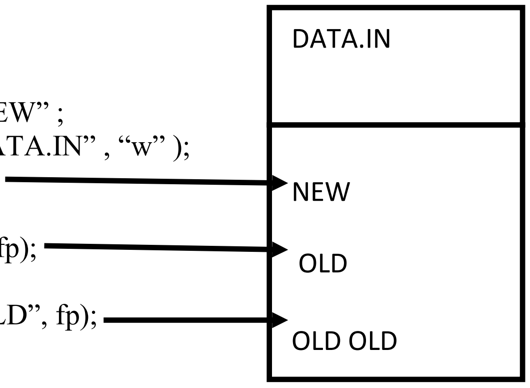
10.3.2 String data oriented functions

- **fputs( ) :** This function is used to write a string to a file. Its general form is.

fputs(string/string-variable, fp);

Where the string/string-variable represents a string constant and fp is a pointer to FILE. The function writes the string or content of string-variable to the file pointed by fp. Also, fputs() does not add a new line to the end of the string. It must be included explicitly.
Program10.6: Program to create a file for writing strings to it using **fputs( )**

```
#include<stdio.h>
void  main( )
{
    FILE *fp;
    char str[] = "NEW" ;
    fp = fopen("DATA.IN" , "w" );
    fputs(str , fp);
    fputc("\n",fp);
    fputs("OLD" , fp);
    fputc("\n",fp);
    fputs("OLD OLD", fp);
    fputc("\n",fp);
    fclose(fp);
}
```



The function fputs() writes the content of string variable **str** first and then the strings "OLD" and "OLD OLD" to the file

- **fgets( )** : This function is used to read a string from a file. Its general form is

fgets(string-variable , n, fp);

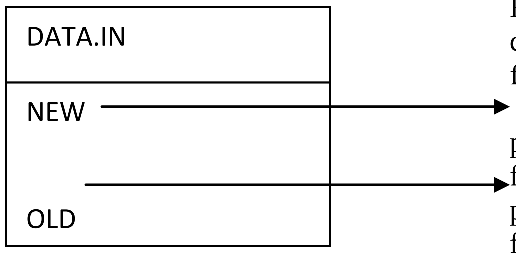
The function reads a string of n-1 characters from the file pointed by **fp** and returns the string value, which is collected by the string-variable.

It will stop reading when any one of the following conditions is satisfied

- It has read(n-1)characters
- It encounters a new line character
- It reaches the end of file
- A read error occurs

fgets() automatically appends a null character to the string read.

Program10.7 Program to read strings from a file using **fgets( ).**



```
#include<stdio.h>
void  main( )
{
  FILE *fp;
  char str1[ ], str2[ ];
  fp = fopen("DATA.IN" , "r" );
   fgets(str1,  fp);
  puts(str1);
  fgets(str2, fp);
  puts(str2);
  fclose(fp);
}
```

Initially the string "NEW" is read and stored in string variable str1 and next the string "OLD" is read and stored in string variable **str2.**then theyare displayed on the screen.

10.3.4 Mixed data oriented functions

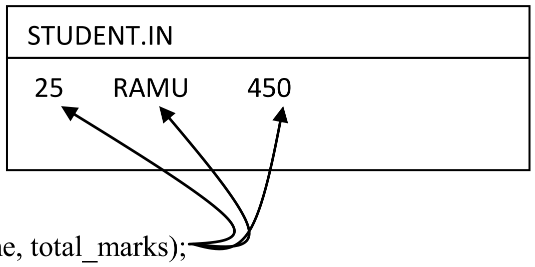
- **fprintf( ) :**The function is used to write multiple data items which may (or may not) be of different types to a file. Its general form is.

fprintf(fp,"control-string", arg1, arg2, arg3,…);

Where fp is a pointer to the file, "control-string" contains output specifications for the data to be written and arg1, arg2 … are the arguments list. The function writes the data values of arg1, arg2… to the file pointed by fp.

P**rogram10.8:** Program to write student details to a file using fprintf( )

```
#include<stdio.h>
void  main( )
{
FILE *fp;
    char name[30] = "RAMU";
    int rollno = 25;
    float total_marks = 450;
    fp = fopen("STUDENT.IN" , "w" );
    fprintf( fp, "\n%d\t%s\t%f", rollno, name, total_marks);
    fclose(fp);
}
```



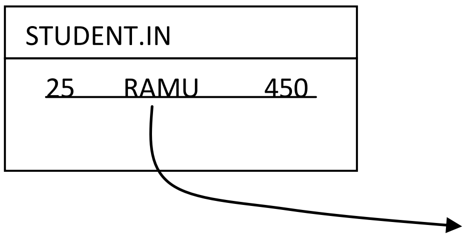
The contents of rollno, name and total_marks are written to the file pointed by fp.

- **fscanf( )** :- This function is used to read multiple data items which may be of different types from a file. Its general form is

fscanf(fp,"control-string", v1, v2, v3,…);

The function reads data values in a given format from the file pointed by fp and returns the data values, which are collected by the variable list.

Program 10.9: Program To read student data from the file ”STUDENT.IN” using **fscanf()**



```
#include<stdio.h>
void  main( )
{
  FILE *fp;
  char name[30];
  int rollno;
  float total_mark ;
  fp = fopen(“STUDENT.IN” , “r” );
  fscanf( fp,“%d%s%f”, &rollno,name,&total_marks);
  fclose(fp);
}
```

The data values from the file “STUDENT.IN” are read and the values are collected by rollno, name and total_marks respectively.

10.4 PROGRAMS USING FILES

- Example program on getw() and putw() functions

Program10.10: Program to read integers through key board and write to the file “DATA”. Later read the integers from the file “DATA”, and write even numbers to the file “EVEN” and odd numbers to the file “ODD”. Read data from the files “EVEN”, “ODD” and display on the screen

```
#include< stdio.h >
main()
{
    FILE *f1,*f2,*f3;
    int number,i;
    printf(“Contents of the data file\n\n”);
    f1=fopen(“DATA”,”w”);
    for(i=1;i< 30;i++)
      {
          scanf(“%d”,&number);
          if(number==-1)
                    break;
          putw(number,f1);
      }
    fclose(f1);
    f1=fopen(“DATA”,”r”);
    f2=fopen(“ODD”,”w”);
    f3=fopen(“EVEN”,”w”);
    while((number=getw(f1))!=EOF)  /* Read from data file*/
      {
        if(number%2==0)
              putw(number,f3);       /*Write to EVEN file*/
        else
              putw(number,f2);       /*write to ODD file*/
      }
    fclose(f1);
    fclose(f2);
    fclose(f3);
```

```
        f2=fopen("ODD", "r");
        f3=fopen("EVEN", "r");
        printf("\n\nContents of the ODD file\n\n");
        while((number=getw(f2))!=EOF)
          printf("%d",number);
        printf("\n\nContents of the EVEN file");
        while((number=getw(f3))!=EOF)
          printf("%d",number);
        fclose(f2);
        fclose(f3);
    }
```

Program10.11: Program to read inventory data; item, number, price and quantity for three items and writes the data to the file. Later read the data from the file and display on the screen with total value.

```
/*Program to handle mixed data types*/
#include< stdio.h >
main()
{
    FILE *fp;
    int num,qty,i;
    float price,value;
    char item[10],filename[10];
    printf("Input filename");
    scanf("%s",filename);
    fp=fopen(filename,"w");
    printf("Input inventory data\n\n");
    printf("Item_name, number,  price and  quantity\n");
    for i=1; i< =3; i++)
    {
     scanf("%s%d%f%d", item, &number, &price, &quality);
      fprintf(fp,"%s\t%d\t%f\t%d", item, number, price, quality);
     }
    fclose (fp);
    fp=fopen(filename,"r");
    printf("\nItem_ name \t Number \t Price \t Quantity \t Value");
    for(i=1;i< =3;i++)
   {
    fscanf(fp,"%s%d%f%d",item,&number,&prince,&quality);
   value=price*quantity;
   printf("\n%s \t %d \t %f \t %d \t %d",item, number, price, quantity, value);
   }
  fclose(fp);
}
```

Program10.12: Define a structure emp with members as empno, name and salary. Write a program to read information for n number of employees and write this data to the file 'EMP.DAT". Later read the data from the file and display on standard output

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

```
struct  emp
{
int empno;
char name[30];
float salary;
}
void main( )
{
int n,i;
FILE *fp;
struct emp e;
fp = fopen("EMP.DAT", "w");
printf("\nEnter number of employees:");
scanf("%d", &n);
printf("\n Enter empno, name and salary for %d number of employees:\n", n);
for(i=0; i<n; ++i)
{
scanf("%d%[^\n]%f", &e.empno, e.name, &e.salary);
fprintf(fp, "\n%d\t%s\n%f", e.empno, e.name, e.salary);
}
fclose(fp);
fp=fopen("EMP.DAT", "r");
printf("\nContents of the file EMP.DAT");
while(!feof(fp))
{
fscanf(fp, "%d%[^\n]%f", &e.empno, e.name, &e.salary);
printf("\n%d\t%s\t%f", e.empno, e.name, e.salary);
}
fclose(fp);
}
```

SUMMARY

Real life situations involve large volume of data and in such cases the console oriented I/O operations pose two major problems. It becomes cumbersome and time consuming to handle large volumes of data through terminals. The entire data is lost when either the program is terminated or computer is turned off. Therefore, it is necessary to have more flexible approach where data can be stored on the disks and read whenever necessary, without destroying the data. This method employs the concept of files to store data.

The concept of files enables us to store large amount of data permanently on the secondary storage devices. A defined data type FILE is used to represent a file within program. It acts as an internal file name for a file. Before performing any operation over a file, it has to be opened. The purpose of the opened file may be for writing data or reading data or may be for appending data to the file. Once the intended operation is over the file has to be closed.
The file input/output operations are by using the functions fgetc( ), fputc( ) which are the character oriented file I/O functions, fgets( ),fputs( ) are the string oriented functions and fscanf( ) and fprintf() are the mixed data oriented functions.

Suggested Reading:

1. Chapter-14 : C for Engineers and Scientists by Harry H.Cheng.
2. Chapter-2 : Chapters-13 : Computer Science- A Structured Programming approach using C by B.A.Forouzan & Ritchard F.Gilberg.

EXERCISES

Review Questions

10. 1 What is a file? Why do we need to store data in files?
10. 2 What are the most commonly performed operations on files?
10. 3 What are the different modes of opening a file?
10. 4 What is the difference between “w” and “a”?
10. 5 Mention character oriented file I/O functions and syntax of their usage
10. 6 Explain the syntax and usage of fprintf( ) and fscanf( )
10. 7 File is a named storage on the secondary storage device (TRUE / FALSE)
10. 8 A file should be first opened before performing any operation (TRUE / FALSE)
10. 9 When a file is opened in “w” mode, the previous contents remain intact (TRUE / FALSE)
10. 10 When a file is opened in “a” mode, the previous contents are lost(TRUE / FALSE)
10. 11 Identify the error

```
a)  FILE   fp;
    fp = fopen(“text.dat”, “w”);
b)  FILE *fin;
    fin = fopen(“text.dat”, r );
c)  FILE  *fp1, *fp2;
     fp1 = fopen(“text.dat”, “w”); fp2 = fopen(“text.dat”, “r”);
d)   #include<stdio.h>
  main()
     {
       FILE  *fp1;
      char c;
      fp=fopen(“infile.txt”,”r”);  fclose(infile);
    }
```

10. 12 Distinguish between fgetc()and getchar() functions
10. 13 Distinguish between printf() and fprintf() functions
10. 14 What is the significance of EOF

Comprehensive Questions

10. 1 Define a file.Explain any five operations on files.
10. 2 Write a program using a **for** loop to put the following numbers to a file named **record.dat**.
1 2 3 4 5 6 7 8 9 10
10. 3 Write a program to read the file **record.dat** created in the above problem and print its contents to the standard output.
10. 4 Write a program to count the number of characters in a text file.
10. 5 A file contains a list of names. Write a program to sort the names read from the file and display them.
10. 6 What are the three steps followed while accessing a file? Write a program to convert the case of alphabets in a text file. Upper case letters should be converted to lower case and vice versa.
10. 7 Write a program to read student data consisting of Id.No, Name and marks in six subjects from the keyboard , write it to a file called INPUT, again read the same data from the file INPUT, and find the total and percentage marks of each student. Display the output in tabular form.
10. 8 Write a program that accepts inventory details item –name, number, price and quantity, stores them in a file and display the data on the screen.
10. 9 Write a program that reads an input text file and tabulate the number of times that each of the 128 ASCII characters appears.