FILES

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. Thus, files allow us to store information permanently on the disk; it can be accessed and can further be altered depending upon the needs. This process leads to the concept of data files.

Definition:

File is a set of records that can be accessed through a set of library functions.

File Types:

Sequential file:

- In this type data are kept sequentially.
- If we want to read last record of the file we need to read all the records before reading that record.
- It takes more time.

Random access file:

- In this type data can be read and modified randomly.
- If we want to read last record of the file, we can read it directly.
- It takes less time.

File Functions:

fopen()	-	Creates and Open a new file and returns a pointer to the file.
fclose()	-	Closes an existing opened file.
closeall()	-	Closes all opened files with fopen()
fgetc(), getc()	-	Reads a single character from the file.
fputc(), putc()	-	Write a single character to the file.
fgets(), gets()	-	Read a string from the file.

fputs(), puts()	-	Write a string to the file.
fscanf()	-	Reads all types of data values from a file.
fprintf()	-	Writes all types of data values to the file.
putw()	-	Writes an integer to the file
getw	-	Reads an integer from the file
fread()	-	Reads structured data written by fwrite() function.
fwrite()	-	Writes block of structured data to the file.
fseek()	-	Sets the file pointer to a desired position in the file.
ftell()	-	Returns the current file pointer position in terms of bytes.
ferror()	-	Reports error occurred while read / write operations.
feof()	-	Detects the end of file.

The basic operation performed on a file includes:

1. Naming a File
2. Opening a File
3. Closing a File
4. Writing data into a File
5. Reading data from a file
6. Moving to a specific location in a file (seeking)

DEFINING AND OPENING A FILE

- If we want to store data in a file into the secondary memory, we must specify certain things about the file to the operating system. They include the fielname, data structure, purpose.
- Data structure of a file is defined as FILE in the library of standard Input/Output functions.
- FILE is a defined Data type.

- FILE should be compulsorily written in Uppercase.
- Before performing any Input/Output operation in a file, it must be opened by the program.

The general format of the function used for opening a file is

FILE *fp;

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

- The first statement declares the variable fp as a pointer to the data type FILE.
- This pointer contains all the information about the file, is subsequently used as a communication link between the system and the program.

R open the file for read only.

W open the file for writing only.

A open the file for appending data to it.

Consider the following statements:

FILE *p1, *p2;

p1= fopen("data","r");

p2 = fopen("results","w");

- In these statements the p1 and p2 are created and assigned to open the files data and results respectively the file data is opened for reading and result is opened for writing.
- In case the results file already exists, its contents are deleted and the files are opened as a new file.
- If data file does not exist error will occur.

Closing a file:

A file is closed when no more Input/Output operations is to be performed on it. Closing a file is carried out by using the fclose() library functions.

Syntax

fclose(file_pointer);

Example

….

FILE *p1 *p2;

```
p1=fopen (“Input”,”w”);
p2=fopen (“Output”,”r”);
….
…
fclose(p1);
fclose(p2)
```

The above program opens two files and closes them after all operations on them are completed, once a file is closed its file pointer can be reused on other file.

Write a program to write data to text file and read it.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
FILE *fp;
char c=' ';
clrscr();
fp=fopen("data.txt","w");
if(fp==NULL)
{
printf("\nCannot open file");
exit(1);
}
printf("Write data and to stop press '.'");
while(c!='.')
{
c=getche();
fputc(c,fp);
}
fclose(fp);
printf("\ncontents read:");
```

```
fp=fopen("data.txt","r");
while(!feof(fp))
printf("%c",getc(fp));
getch();
}
```

Sample Input and Output:

```
Write data and to stop press '.'
hai how are u.
contents read:hai how are u.
```

Mode: a (append)

- When a file is opened for appending, it will be created if it does not already exist and it will be initially empty.
- If it does exist, the data input point will be positioned at the end of the present data so that any new data will be added to any data that already exists in the file.

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
void main()
{
FILE *fp;
char c;
clrscr();
printf("Contents of file before appending:\n");
fp=fopen("data.txt","r");
while(!feof(fp))
{
c=getc(fp);
printf("%c",c);
}
fp=fopen("data.txt","a");
```

```
if(fp==NULL)
{
printf("\nCannot open file");
exit(1);
}
printf("\nWrite data to append  and to - stop press '.'\n");
while(c!='.')
{
c=getche();
fputc(c,fp);
}
fclose(fp);
printf("\nContents of file after appending:\n");
fp=fopen("data.txt","r");
while(!feof(fp))
{
c=getc(fp);
printf("%c",c);
}

getch();
}
```

Sample Input and Output:

Contents of file before appending:

I am studying in RGCET.

Write data to append and to - stop press '.'

This is file concept example.

Contents of file after appending:

I am studying in RGCET.This is file concept example.

FILE I/O

getc()

This function reads a single character from the opened file and moves the file pointer . It returns EOF, if end of file is reached.

```
#include <stdio.h>
  main( )
  {
   FILE *fp;
   char c;
   fp = fopen("sample.TXT", "r");
   if (fp == NULL)
                printf("File doesn't exist\n");
   else {
    do {
     c = getc(fp);
       putchar(c);
    } while (c != EOF);
   }
  fclose(fp);
  }
```

Sample Output:

Hello!

Welcome to this world.

sample.txt (file)

Hello!

Welcome to this world.

fgetc():

This function is similar to getc() function. It also reads a character and increases the file pointer position. If any error or end of file is reached it returns EOF.

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

```
main( )
{
 FILE *fp;
 char c;
 fp = fopen("sam.TXT", "r");
 if (fp == NULL)
                printf("File doesn't exist\n");
 else {
  do {
   c = fgetc(fp);
     putchar(c);
   } while (c != EOF);
 }
fclose(fp);
}
```

sam.txt (file)

I am

Working in File concept.

Sample Output:

I am

Working in File concept.

Note:

The getc and putc functions are analogous to getchar and putchar functions and handle one character at a time. The putc function writes the character contained in character variable c to the file associated with the pointer fp1.

Example putc(c,fp1);

similarly getc function is used to read a character from a file that has been open in read mode. **c=getc(fp2).**

fputc():

- The **fputc** function is used to write a character to a stream.
- int fputc(int c, FILE *fp);
- The parameter c is silently converted to an unsigned char before being output. If successful, fputc returns the character written. If unsuccessful, fputc returns EOF.
- The standard macro **putc**, also defined in <stdio.h>, behaves in almost the same way as fputc, except that—being a macro—it may evaluate its arguments more than once.
- The standard function **putchar**, also defined in <stdio.h>, takes only the first argument, and is equivalent to putc.

Program : Copy Text From One File to Other File

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

void main() {
  FILE *fp1, *fp2;
  char ch;
  clrscr();

  fp1 = fopen("Sample.txt", "r");
  fp2 = fopen("Output.txt", "w");

  while (1) {
    ch = fgetc(fp1);

    if (ch == EOF)
      break;
    else
      putc(ch, fp2);
  }

  printf("File copied Successfully!");
  fclose(fp1);
  fclose(fp2);
}
```