6

UNIT:6
FILES

Topics to be covered

- 6.1 Introduction
- 6.2 File Operations
  - 6.2.1 Opening a File
  - 6.2.2 Reading a File
  - 6.2.3 Closing a File
- 6.3 File Operations
- 6.4 Text Modes
- 6.5 File Functions
- 6.6 Command Line Arguments

6.1 INTRODUCTION OF FILE

- When the program is terminated, the entire data is lost in C. if u keep large volume of data, it is time consuming to enter the entire data but if file is created it can be access using few commands.
- There are large numbers of functions to handle file I/O in c language.
- High level file I/O functions can be categorized as:

  1)Text file

  2)Binary file

THE FILE I/O FUNCTIONS IN THE STDIO LIBRARY ARE:

- **fopen-**opens a text file.
- **fclose-**closes a text file.
- **feof-**detects end-of-file marker in a file.
- **fscanf-**reads formatted input from a file.
- **fprintf-**prints formatted output to a file.
- **fgets-**reads a string from a file.
- **fputs-**prints a string to a file.
- **fgetc-**reads a character from a file.
- **fputc-**prints a character to a file.
- **fseek-**set the positions to a desired point in the file.

FEATURES OF FILE:-

- It is the new concept to deal with data in file handling.
- The data is stored into the disk and whenever it can require,we can retrieve easily.
- The output of the program can be store into the file.
- It use many file function to deal with file handling.

6.2 File operations

- There are different operations that can be carried out on a file. These are
- 1. Creating a new file
- 2. Opening an existing file
- 3. Reading from a file
- 4. Writing to a file
- 5. Closing a file

6.2.1 Opening a file

- Opening a file is performed by using fopen().
- Syntax :

```
ptr=fopen (“filename”,”mode”);
```

Where,

filename is the c string containing the name of the file to be opened.

mode is the c string containing the file access mode

OPENING MODES IN STANDARD I/O FILE.

FILE MODE	MEANING OF MODES
R	To read the data in a file.
W	To write the data in a file.
A	To append or to add the data in a file.
r+	Both reading and writing to a file.
w+	Both reading and writing to a file.
a+	Both reading and append data in a file.

EXAMPLE:-

....

```
FILE  *ptr;
ptr=fopen("d:\\program.txt","w");
....
```

Here the program.txt file is opened for writing mode.

6.2.2 Closing a file:-

- Once we open a file, finished reading from the file, we need to close the file. The file should be closed after reading/writing of a file. This is done using the function fclose().

- Syntax:-

```
fclose(ptr);
```

ptr is the pointer associated with file to be closed.

EXAMPLE:-

```
#include<stdio.h>
main()
{
FILE *fp;
fp=fopen("data.txt","r");
If(fp==NULL)
{
Printf("file does not exist,please check!\n");
}
fclose(fp);
}
```

6.2.3 Reading the file:-

- C provides four functions that can be used to read files disk:
- 1)fscanf()
- 2)fgets()
- 3)fgetc()
- 4)fread()

THE BASIC STEPS FOR USING A FILE IN C ARE ALWAYS THE SAME:-

- 1)create a variable of type “FILE*”.
- 2)open the file using the “fopen” function and assign the “file” to the variable.
- 3)check to make sure the file was successful opened by checking to see if gthe variable==NULL.if it does,an error has occured (it is not compulsory to check. Without checking you can also proceed).

6.3 BINARY MODE:-

- In binary mode,your program can access every byte in the file.
- The following code fragment will open the file afile.txt for reading in binary mode:

- FILE *inputFilePtr;

- inputFileptr=fopen(“file1.txt”,”rb”);

6.4 TEXT MODE:-

- Text mode is used only for text files.when a file opened in text mode some character may be hidden from ur program.
- The following code fragment will open the file "file.txt" for reading in text mode:

- FILE *inputFilePtr;

- inputFilePtr=fopen("file1.txt","r");

6.5 File functions

- 6.5.1 fprintf() and fscanf() function
- fprintf() and fscanf() function are equal to printf and scanf function except that they work on files.
- The frist argument of these functionis a file pointer which specifies the file to be used.
- Syntax of fprint():

  fprintf(fp,“control string”,list);

  For example:

  fprintf(fp“%d,%d,%d”,a,b,c);
- Syntax of fscanf();

  fscanf(fp,“control string”,list);

  For example:

  fscanf(fp“%d,%d,%d”,&a,&b,&c);

WRITE A PROGRAM TO USE FPRINTF()&FSCANF() FUNCTION.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   FILE *fp;
   Float total;
                fp=fopen("d:\\data.txt","w+");
                fprintf(fp, 100);
                fscanf(fp,"%f",&total);
                fclose(fp);
                printf("value of total is %f\n",total);
   Getch();
}
```

6.5.2 GETC & PUTC FUNCTION.

- 6.5.2.1 getc function:
- Getc() is a function that return the next character on the given input file and increments the file pointer to point to the next character.
- Getc() return the character,after converting it into an integer. On end-of-file,it returns EOF.
- Syntax of getc:

  getc(FILE *stream);
- 6.5.2.2 putc function:
- Putc() is a function that outputs the character ch the file
- Putc() return the character printed ch. On error, it returns EOF.
- Syntax of putc;
- putc(int ch, FILE *stream);

WRITE A PROGRAM TO USE GETC.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   FILE *fp;
   Char ch;
                fp=fopen("C:\\Myprog.txt","r");
                ch=getc(fp);
                 printf("%c",ch);
                fclose(fp);
   getch();
}
```

WRITE A PROGRAM TO USE PUTC.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   FILE *fp;
   Char ch;
                fp=fopen("C:\\Myprog.txt","w");
                ch='a'
                putc(ch,fp);
                fclose(fp);
   getch();
}
```

6.5.3 FGETC & FPUTC FUNCTION.

- 6.5.3.1 fgetc function:
- The fgetc funcation returns the character read. If an error occurs,the fgetc function will set the stream's error indicator and return EOF.
- it's equivalent to the same getc() call,only the implementation of the two function differs.
- Syntax of fgetc:

  fgetc(FILE *stream);
- 6.5.3.2 fputc function: The fputc funcation returns the character written. If an error occurs writing to the stream,the fputc function will set the stream's error indicator and return EOF.
- Syntax of fputc:

  fputc(int ch,FILE *stream);

WRITE A PROGRAM TO USE FGETC.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   FILE *fp;
   Char ch;
                fp=fopen("C:\\Myprog.txt","r");
                ch=fgetc(fp);
                 printf("%c",ch);
                fclose(fp);
   getch();
}
```

WRITE A PROGRAM TO USE FPUTC.

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   FILE *fp;
   Char ch;
                fp=fopen("C:\\Myprog.txt","w");
                ch='a'
                fputc(ch,fp);
                fclose(fp);
   getch();
}
```

6.5.4 GETS AND PUTS FUNCTION.

- Syntax of gets():

  gets(char_array_variable);

- Syntax of puts():

  puts(char_array_variablr/string);

WRITE A PROGRAM TO USE GETS AND PUTS .

```
#include<stdio.h>
#include<conio.h>
Void main()
{
   Char data[100];
                    printf("enter a string:");
                   gets(data);
                   printf("entered data is :");
                   puts(data);
   getch();
}
```

6.5.5 & 6.5.6 FSEEK & FEOF FUNCTION

- 6.5.5 fseek
- In the c programming Langauge, the fseek function changes the file position indicator for the strem pointed to by stream.
- The fseek function return zero if succes. If an error occurs, the fseek funcation will return a nonzero value.
- The syntax of fseek:

  Int fseek(FILE *stream,long int offset,int whence);
- 6.5.6 FEOF
- User to determine if the end of the file specified, has been reached. When a file is being read sequentially one line, or one piece of data at a time, this is the function you check every iteration, to see if the end of the file has come.
- Nonzero if the end of file has been reached, zero (false) otherwise.
- The syntax of FEOF:

  feof(FILE *stream)

COMMAND LINE ARGUMENTS

- The main() function have two argument as shown below:

```
int main (int argc , char *argv[ ]);
```

- The frist arg. Argc represent the number of arg.in command line.
- The second arg. Argv is an array of char type pointer that stores number of arg. At command line.
- The size of this array is equal to the value of arg. The command line can be achieved by using the arg. of main().

WRITE A PROGRAM TO USE COMMAND LINE ARGUMENTS.

```
#include<stdio.h>
main ( int argc , char *argv[ ] )
{
Printf("argc =%d",argc);
Printf("\n argv are");
For(int i=0;i,argc;i++)
printf("%s",argv [i]);
}
```