Unit - 4

Unit :- 4

=: File handling :=

Uptill, we have been using various input - output functions like getchar(), getch(), putch(), putchar(), scanf() & printf() to read data from the keyboard and to write data onto the monitor. These functions are used when data is small, because these functions does not have capability to store data permenant. Therefore following are the various limitations.

1) It becomes difficult to handle large amount of data.
2) It becomes time consuming, to enter large amount of data.
3) The entered data was lossed, as soon as program is terminated.

To overcome such types of limitations, file-handling is used. In file-handling data can be store on the hard-disk and can be read whenever necessary. To store data, we must want to provide file name, file mode & various types of library functions.

Following are the various types of library functions called 'file oriented I/O functions'.

*File oriented I/o functions :-

	Function name	Operation
1)	fopen () →	opens a new file or an existing file.
2)	fclose () →	closes a opened file.
3)	fgetc () →	Reads a single character from a file.
4)	fputc () →	Writes a single character into a file.
5)	fscanf () →	Reads a set of data from a file.
6)	fprintf () →	Writes a set of data into the file.
7)	getw () →	Reads an integer value from a file
8)	putw () →	Writes integer value into the file.
9)	fgets () →	Reads a string from a file
10)	fputs () →	Write a string into a file
11)	fread () →	Reads number of record from a file.

12) fwrite() → Writes a no. of records into the file.

13) fseek() → Set the position to a desir point in the file.

14) ftell() → Gives the current position in the file, in terms of bytes from start.

15) rewind() → set the position to the begining of the file.

16) feof() → Is used to test end of the file condition.

17) ferror() → Is used to check whethe the mode of the file and operation of the file is correct or not.

* Operation on a File :—

In C when we want to perform file operation then following are four types of file operations.

1) Opening a file.
2) closing a file.
3) Reading data from a file.
4) Writing data into the file.

17 Opening a file:—

If user wants to store data into the secondary memory, then we must wan

(i.e Read "r", Write "w" and Append "a");

Q. Explain the various file opening modes with example. — (5M)

पृष्ठ सं.: 4 दि.:

to open a file by using a function fopen(). In the function fopen() arguments are file name and mode of the file enclosed within paranthesis. The address of the file can be assigned to the file pointer, hence the general form of the opening a file is as follows.

```
FILE *fp;
fp = fopen("File Name", "file mode");
```

Where, FILE is a data type related to a file. fp is a file pointer which specifies address of the file.
fopen() is a function is used to open a file, which contains two arguments; File name and mode.

File name is a string of characters is used to store data into the hard disk. File name may contain two parts

[primary name . Extension]

primary name is the name of the file and length can not be greater than eight characters. Where as extension can not be greater than three characters.

Ex :- ABC.TXT
STUDENT.DAT
OUTPUT

File name must be coated within double quatation mark.

When user wants to open a file then that file must be open in one of the following mode.

1) Reading mode "r"
2) Writing mode "w"
3) appending mode "a"

1) Reading mode "r" :-

Read mode is opens a file for reading only. If you want to open a file in mode "r" then that file must be exist, to access the containts from that file, otherwise 'c' compiler gives an error message.

2) Writing mode "w" :-

Write mode "w" opens a file for writing only. If you want to open a file in "w" mode then new file will be created. Otherwise if it is exist then previous containts will be deleted and new containts will be added.

3) Appending mode "a" :-

Appending mode "a" opens a file for appending a data. If you want to open a file in mode "a" then new containts will be added at the end of the file if it is exist. Otherwise

new file will be created if it is not exist.

Ex. FILE *fp1, *fp2;
fp1 = fopen ("ABC.TXT", "r");
fp2 = fopen ("XYZ.TXT", "w");

2] Closing a file :-

A file must be close as soon as all the operations on it have been completed. The function fclose() is used to close opened file. The general form of closing a file is as follows.

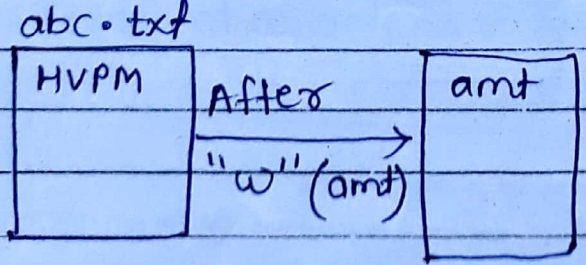
```
fclose (fp);
```

Where, fclose fp is a file pointer which we want to close, a file which is already open.

After closing a file, file name and its mode will be closed.

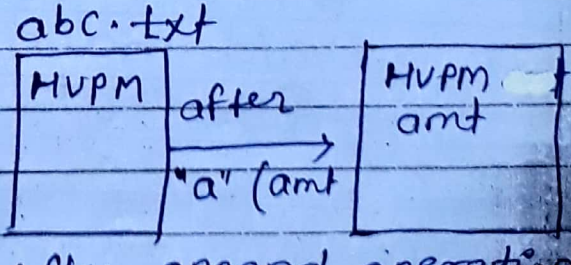
Ex:-
```
fclose (fp1);
fclose (fp2);
```

Write mode



[After write operation previous contents deleted & new contents added.]

Append mode



[After append operation previous contents remains as it is & new contents added after previous one]

3] **Reading data from a file & Writing data into the file :=**

Ones the file is already opened, then data can be read from a file using a function fgetc(), getw(), fscanf(), fread(); fgets() and data can be write into a file using functions fputc() fputs(), putw(), fprintf(), fwrite

1) **fgetc() :=** fgetc() is similar to getchar() function, but only difference is that getchar() reads a single character from the keyboard, whereas fgetc() reads a single character from the file.

The general form of fgetc() is as follows

```
ch = fgetc(fp);
```

Where, ch is a character variable of data type char. fp is a file pointer which open the file in read mode.

Ex. Write a program to read character data from the file and display onto the monitor and also count the number of character from the file.

→
```
/* Program for reading data from file and
#include<stdio.h>          writing into file */
#include<conio.h>
void main
 {
```

EOF = End of File

पृष्ठ सं.: 8

```
FILE *fp;
char ch;
int c;
fp= fopen ("ABC.TXT", "r");
while (1)
{
    ch = fgets (fp);
    if (ch == EOF)
        break;
    else
        {
          putchar (ch);
          c++;
        }
}
printf ("\n Total characters = %d", c);
fclose (fp);
getch();
}
```

ABC.TXT

AMRAVATI

Output ;-

```
AMRAVATI
Total characters = 8
```

27 **fputc():-** The fputc() function is used to write a single character into the file. fputc() is similar to putchar(), but only difference is that putchar(), writes a single character onto the monitor whereas fputc() writes a single character into the file.

The general form of fputc() is as follows.

```
fputc(ch,fp);
```

Where, 'ch' is the character which is to be write in the file, fp is the file pointer which opens the file in write mode "w".

Ex. Write a program to read a character from the keyboard and stores into the file.

→
```
/* Read character from keyboard & store into file*/
#include<stdio.h>
#include<conio.h>
void main()
{
    FILE *fp;
    char ch;
    fp=fopen("XYZ.TXT","w");
    while(1)
    {
     ch=getchar();
```

```
    if (ch==*')
        break;
    else
        fputc (ch, fp);
  fclose (fp);
  getch();
}
```

x42.TXT

AMRAVATI

Output:-

AMRAVATI *

---

3) **getw()**:- getw() function is used to read a numeric value from the file. The general form of getw() is as follows;

```
n= getw (fp);
```

where, 'n' is a numeric integer variable of data type int.

'fp' is a file pointer which opens a file in read mode "r".

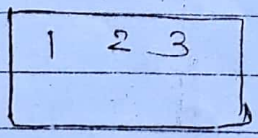
Ex. Write a program to read numeric integer values from the file and print onto the monitor.

→

```
/*
#include<stdio.h>
#include <conio.h>
void main
```

```
{
    FILE *fp;
    int n, c=0;
    fp= fopen("ABC.TXT", "r");
    while (1)
    {
      n = getw(fp);
      if (n== EOF)
          break;
      else
          {
          printf("%d", n);
          c++;
          }
    }
    printf("\n Total digits = %d", c);
    fclose (fp);
    getch();
}
```

Output :-



1 2 3
Total digits = 3

4) **putw()**:- putw() is used to write a numeric value into the file. The general form of putw() is as follows.

```
putw(n, fp);
```

Where, n is a numeric integer of data type int and 'fp' is file pointer which opens the file in write mode "w".

Ex:- Write a program to read numeric integer from the keyboard & store into the file

→
```
/* Read numeric value from keyboard & Store into File */
#include<stdio.h>
#include<conio.h>
void main()
{
  FILE *fp;
  int n;
fp= fopen("XYZ.TXT", "w");
  while(1)
  {
    scanf("%d", &n);
   if(n==99)
     break;
   else
   putw(n, fp);
  }
  fclose(fp);
  getch();
}
```

Output :-

1,2,3,99

x42.txt

1 2 3

5) **fscanf():-**

fscanf() is used to read set of data values from the file. fscanf() is similar to scanf(), only difference is that fscanf() reads set of value from the file whereas scanf() reads the set of values from the keyboard.

The general form of the fscanf() is as follows.

```
fscanf(fp, "control string", &v1, &v2, ... &vn);
```

Where, fp is a file pointer which opens the file in read mode "r". The control string is the format specification which may contain %s, %d, %f, %c.

&v1, &v2, ---&vn are the list of variables and corresponding address

1) Write a program to read set of data values from the file and display onto the monitor.

→
```
/*
#include<stdio.h>
#include<conio.h>
void main()
{
   FILE*fp;
   char name[20];
   int age;
   float per;
   fp = fopen ("ABC.TXT", "r");
   while (fscanf (fp, "%s %d %f", name, &age,
                  &per) != EOF)
   {
    printf("%s %d %f", name, age, per);
   }
   fclose(fp);
   getch();
}
```

Output :-

ABC.TXT

ZEROx 26 72.12
callender 17 51.16

```
ZEROx 26 72.12
callender 17 51.16
```

×

6) **fprintf()**:-

fprintf() is used to write set of data values into the file. fprintf() is similar to printf(), only difference is that printf() writes ~~data~~ onto the monitor & fprintf() writes set of data into the file.

The general form of fprintf() is

```
fprintf(*fp, "control string", arg1, arg2,...
                                    argn);
```

Where, fp is a file pointer, which opens the file in write mode "w".

The control string is a format specification which may contain %s, %c, %d or %f.

arg1, arg2, ~~are the~~ ~~constant~~ may be constant, variable or expression.

1) Write a program to (accept) read set of data values from the keyboard and write into the file.

→
```
/* Program to read values from
#include<stdio.h>
#include<conio.h>
void main()
{
```

```
FILE *fp;
char name[20];
int age;
float per;  → char ch = 'y';
fp = fopen ("ABC.TXT", "w");
while (ch == 'y')
{
 printf ("\n Enter information");
 scanf ("%s%d%f", name, &age, &per);
 printf ("\n Do you want to enter another
          entry (Y/N)");
 ch = getchar();
}
fclose(fp)
fprintf (fp "%s%d%f", name, &age, &per);
printf ("\n Do you want to enter another
         entry (Y/N)");
ch = getchar ();
}
fclose (fp);
}
```

output :-

Enter information
ZEROX 26 50.12

ZEROX 26 50.12

Do you want to enter another entry

7) fread() :-

fread() is used to read record from a file in binary mode. To read record from the file we must want to used structure variable to store information, of a record.

fread() must want to read a file in read binary mode "rb".

The general form of fread() is as follows.

```
fread(&struct_var, sizeof(struct_var),
        NOR, fp);
```

address

Where, &struct_var (is the) address of the structure variable.

'sizeof(struct_var)' is the size of structure variable in the form of bytes.

'NOR' is the number of record wants to read.

'fp' is the file pointer which opens a file in read binary mode "rb".

1) Write a program to read a structure member values from the ~~keyboard~~ file and display onto the monitor.

→
```
/*
#include<stdio.h>
#include<conio.h>
void main()
{
  FILE *fp;
  struct stud
  {
    char name[20];
    int age;
    float per;
  };
  struct stud x;
  fp = fopen("ABC.TXT", "rb");
  while((fread(&x, sizeof(x), 1, FP)) != EOF)
  {
   printf("\n %s %d %f", x.name, x.age,
                                x.per);
  }
  fclose(FP);
}
```

Output:-

AB	26	70.1
CD	17	60.2
EF	18	50.3

AB	26	70.1	26
CD	17	60.2	
EF	18	50.3	

8) fwrite():-

It is used to write record into the file ~~into~~ in write binary mode "wb". We must want to use structure variable to write record into the file in write binary mode "wb". The general form is as follow

```
fwrite(&struct_var, sizeof(struct_var),
        NOR, fp);
```

Where, '&struct_var' is the address of structure variable.

sizeof(struct_var) is the size of structure variable

'NOR' is the number of records want to read. fp is the file pointer which opens the file in write mode.

1> Write a program to read values of the structure members from the keyboard and write into the file

→

```
#include <stdio.h>
#include <conio.h>
void main()
{
  FILE *fp;
  char ch = 'y';
  struct stud
  {
    char name[20];
    int age;
    float per;
  };
  struct stud x;
  fp = fopen("ABC.TXT", "wb");
  while (ch == 'y')
  {
    printf("\n Enter Information \n");
    scanf("%s %d %f", x.name, &x.age,
                      &x.per);
    fwrite(&x, sizeof(x), 1, fp);
    printf("\n Another Record (Y/N) ?");
    ch = getchar();
  }
  fclose(fp);
}
```

9) ftell() :-

This function gives the current position in the file in terms of bytes from the starting. ftell() takes a file pointer and returns the no. of bytes at which file pointer is currently present. i.e this function is use to identify the current position of the file pointer. The general form of the ftell() is as follows

```
n = ftell(fp);
```

Where, 'n' is a numeric integer variable of data type int. 'fp' is a file pointer which opens a file either in read mode "r" or write mode "w".

i) Write a program to write character data into the file and count total number of character present in the file using ftell().

→
```
#include <stdio.h>
#include <conio.h>
void main()
{
   FILE *fp;
   char ch;
   int n;
```

```
fp= fopen ("ABC.TXT", "w");
while (1)
{
   ch = getchar ();
   if (ch == '*')
        break;
   else
      ~~print~~
      fputc (ch, fp);
}
   n= ftell (fp);
   printf ("\n ~~current~~ Total character = %d", n);
   printf ("\n current position of file
            pointer = %d", n);
   fclose (fp);
}
```

107 rewind ( ) :-

~~The orewind~~ ( )

position 0 1 2 3 4 5 6 7 8
A M R A V A T I

Output :-

AMRAVATI *

AMRAVATI

Total character = 8

current position = 8

10) rewind():-
rewind() function set the position of file pointer to the starting position '0'. This function takes a file pointer and transfer to the first position in the file. The value of first position is always zero.
The general form of rewind() is as follows

```
rewind(fp);
```

Where, fp is a file pointer which the file either in read mode "r" or write mode "w".

11) Write a program to enter character data into a file & display current position of file pointer before & after rewind.

→
```
#include<stdio.h>
#include<conio.h>
void main()
{
  FILE *fp;
  char ch;
  int n;
  fp = fopen("ABC.TXT", "w");
  while()
  {
    ch= getchar();
```

AMRAVATI –

```
if (ch == '*')
    break;
else
    fputc (ch, fp);
}
n = ftell (fp);
printf ("\n current position of file pointer
        before rewind = %d ", n);
rewind (fp);
printf ("\n current position of file pointer
        after rewind = %d", n);
fclose (fp);
}
```

Output :-

AMRAVATI *

AMRAVATI

Current position of file pointer before rewind = 8

Current – %

after rewind = 0

11) fseek( ):-

This function sets the position to the desired point in the file, fseek() is used to move the file pointer position either forward or backward in the file. The general form of the fseek() is as follows.

```
fseek( fp, offset, position);
```

Where, fp is the file pointer is used to open a file either in read mode "r" or write mode "w".

'offset' is a variable is used to indicate the current position in the form of bytes. The 'offset' specifies number of position to remove from the location specified by the position.

'position' specifies from which direction you want to transfer the file pointer and takes the one of the following value.

Value	meaning
0	Begining of the file
1	Current position
2	End of the file.

The 'offset' may be positive or negative. If positive then file pointer

moves forward & if negative then file pointer moves backward.

Ex:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
   FILE *fp;
   char ch;
   int n;
   fp=fopen("ABC.TXT","w");
   while(1)
   {
     ch=getchar();
     if(ch==='*')
          break;
     else
        fputc(ch,fp);
   }
   n=ftell(fp);
   printf("\n No of character entered=%d",n);
   fclose(fp);
   fp=fopen("ABC.TXT","r");
   n=0;
   while(1)
   {
     ch=fgetc(fp);
     fseek (fp,n,0);
```

```
if (ch==EOF)
        break;
else
    printf ("in position of %c = %d",
                ch, ftell (fp));
    n=n+2;
  }
  fclose (fp);
}
```

Output:-

AMRAVATI *

AMRAVATI

Number of character = 8

position of A = 0

R = 2

V = 4

T = 6

12) feof() :-

This function can be use to test for an end of the file condition. i.e feof() is used to check whether the file is end or not. The result of the feof function is true or false. The general form of feof() is as follows.

[feof(fp);]

Where, fp is a file pointer which opens a file in read mode "r".

```
/* To count total No. of character */
ex:- #include<stdio.h>
#include<conio.h>
void main()
{
   FILE *fp;
   char ch;
   int c=0;
   fp= fopen("ABC.TXT","r");
   while (feof(fp)==0)
   {
      ch= fgetc(fp);
      c++;
   }
   printf("\n Total No. of characters = %d",c);
   fclose(fp);
}
```

ABC.TXT

AMRAVATI

output :-

Total No. of characters = 8

137 ferror():-

The ferror() reports the status of the file. The ferror() is used to check whether mode of the file & operation of the file is correct or not. The result of the ferror() is either true or false.

The general form of ferror() is as follows.

```
ferror(fp);
```

Where 'fp' is a file pointer which opens a file in either read or write mode.

```
/*
#include<stdio.h>
#include<conio.h>
void main()
{
   FILE *fp;
   char ch;
   int c=0;
   fp=fopen("ABC.TXT","w");
   while (feof(fp)==0)
   {
      ch=fgetc(fp);
      if (ferror(fp)!=0)
      {printf("\n Error occurs in Reading");
       break;
      }
```

```
    else
        c++;
}
fclose(fp);
printf("\n No of character = %d", c);
fclose(fp);
}
```

output :-

Error occurs in Reading.

1) Write a program to copy the content from one file to other file.

→

```
/*
#include<stdio.h>
#include<conio.h>
void main()
  {
    FILE *fp1, *fp2;
    char ch;
    fp1= fopen("ABC.TXT", "r");
    fp2= fopen("XYZ.TXT", "w");
    while(1)
     {
       ch=fgetc(fp1);
       if (ch== EOF)
            break;
       else
         fputc(ch, fp2);
     }
```

```
    fclose (fp1);
    fclose (fp2);
}
```

2) Write a program to read a text file "TEXT.DAT" and store upper case character "UPPER.DAT" & lower case character into "LOWER.DAT".

→
```
#include <stdio.h>
#include <conio.h>
#include <c.type.h>
void main()
{
    FILE *fp1, *fp2, *fp3;
    char ch;
    fp1 = fopen ("TEXT.DAT", "r");
    fp2 = fopen ("UPPER.DAT", "w");
    fp3 = fopen ("LOWER.DAT", "w");
    while (1)
    {
        ch = fgetc (fp1);
        if (ch == EOF)
            break;
        else
        {
            if (isalpha(ch) > 0)
            {
                if (isupper(ch) > 0)
                    fputc (ch, fp2);
```

```
            else
                fputc(ch, fp3);
        }
    }
    fclose(fp1);
    fclose(fp2);
    fclose(fp3);
}
```

3) Write a program to read numeric file "NUMBER.DAT" and store even nos into "EVEN.DAT" and odd nos into "odd.DAT"

→

```
#include<stdio.h>
#include<conio.h>
void main()
{
    FILE *fp1,*fp2,*fp3;
    int n;
    fp1 = fopen("NUMBER.DAT", "r");
    fp2 = fopen("EVEN.DAT", "w");
    fp3 = fopen("odd.DAT", "w");
    while(1)
    {
        n = getw(fp1);
        if(n == EOF)
            break;
        else
            {if(n%2 ==
```

```
        r = n % 2;
        if (r == 0)
            putw (fp2);
                  n,
        else
            putw (fp3);
                  n,
    }
}
fclose (fp1);
fclose (fp2);
fclose (fp3);
}
```

4) Write a program segement to read the string by using a file & display it in reverse order.

→

```
#include <stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
    FILE *fp;
    char ch, x[50];
    fp = fopen
    int i = 0;
    while (1)
    fp = fopen ("ABC.TXT", "r");
    while (1)
    {
        ch = fgetc (fp);
```

```
        if (ch== EOF)
                break;
        else
          {
            x[i]= ch;
            i++;
          }
    }
    strrev (x);
    printf("\n Reverse String = %s", x);
    fclose (fp);
}
```