FILES

DATA HIERARCHY IN COMPUTER :

Bit (0,1)
▼
Byte (8-bits)
▼
Field (set of bytes)
▼
Record (set of fields)
▼
File (set of records)

INTRODUCTION TO FILES

Using the printf and scanf statements :-

- We can read / write a character
- We can read / write a string
- We can read / write a line of text

- But if we have to give a set of lines as input then we can use gets() function.
- Using the printf and scanf statements we can give input ad display output, but whenever we exit from the program the contents are automatically erased from RAM.
- If the input contains say 50 lines of text , then after execution the input cannot be stored.
- If in another program we want to use the same input the we have to type the entire text again which is very time-consuming.
- In order to avoid this problem we will be using the concept of FILES.

Definition of file : files are set of records, that are used to store large amounts of data permanently. Usually, contents from RAM are copied into files for further use.

Ex: Consider the below given table

Roll no.	Name	Percentage	Grade
200	Rohan	75	A
201	Rohit	77	A
202	Sam	69	B
203	John	86	A+

In the above table :

- **Each row is a record**

- **Each column is a field**
- **Entire table is a FILE**

WHY TO USE FILES ?

Reason-1 : to read multiple lines of data or huge amount of data.
Reason-2 : to store data permanently on secondary storage devices like hard disc.

BASIC OPERATIONS THAT CAN BE PERFORMED ON FILE

1. Creating a file
2. Opening a file
3. Writing data into a file
4. Reading data from the file
5. Closing the file

MODES IN WHICH A FILE CAN BE OPENED

Name of the mode	Purpose
w	To create a text file. If file already exists the contents are erased.
r	Opens the already existing file in read mode
a	Opens file for appending data. If the file does not exist, then it is created.
w+	To open a file both in reading and writing modes; if file already exists its contents will be erased.
r+	To open the existing file in both reading and writing modes.
a+	Opens the file in both reading and writing modes. If file is not present, a new file is created.
wb	To create a binary file in writing mode; if file already exists its contents will be erased or else a new file is created.
rb	To open the existing binary file in reading mode.
ab	Opens binary file for appending.
wb+	To open a binary file both in reading and writing modes; if file already exists its contents will be erased or new file is created.
rb+	To open the existing binary file both in reading and writing modes.
ab+	Opens a binary file for both reading and appending. If file do not exist then new file is created.

There are two ways to perform file operations in c :

i. Low –level I/O functions
ii. High-level I/O functions

Some of the high-level i/o functions in ‘ c ‘ are :

Function name	Operation
fopen()	Creates/opens file
fclose()	Closes the file
getc()	Reads character from a file
putc()	Writes character into file
fprintf()	Writes set of characters into file
fscanf()	Reads set of characters from file
getw()	Reads integer from a file
putw()	Writes an integer to a file

Creating ,opening and closing a file :

To create file syntax is :

FILE *fp; // fp means file pointer….any variable can be used

To open file syntax is :

fp = fopen(“ file name “, “mode”);

a string or set of characters with any valid file extension can given as file name.

Note : FILE is a predefined structure in stdio.h which is accessed by using a pointer- variable.

Closing a file :

To close file syntax is : fclose(fp);

<u>Program to perform read / write operations on file</u>

```
int  main()
{
FILE  *fp;
```

```
char c;
fp = fopen("abc.txt","w");

while( ( c = getchar() ) != EOF )
{
        Putc( c, fp );

 }

fclose( fp);
```

For writing data into the file.

```
fp = fopen("abc.txt","w");

while( ( c = getc(fp) ) != EOF )
{
        Putchar( c );
}

fclose( fp );

}
```

For displaying data of the file.

TYPES OF FILES

1. Sequential access files
2. Random access files

Sequential access files:-

In this records can be accessed one-by-one in order.

fprintf()…..writes data into the file
syntax: fprintf(file pointer, "format specifier" ,list of variables);

fscanf()…… reads data from the file
syntax: fscanf(file pointer, "format specifier", address of variables);

<u>program for sequential copying a file</u>

```
int main()
{
 Char ch;
```

```
int f=1;
FILE  *fp1,*fp2;
fp1= fopen("x.txt","w");
puts("enter text");
while( ( ch=getchar() )!= EOF)
{
 Putc( ch , fp1);
}
 fclose(fp1);
```

Writing data into a file

```
fp1= fopen("x.txt","r");
fp2= fopen("y.txt","w");
while( ( ch=getc(fp) )!= EOF)
{
 Putc( ch , fp2);
}
fclose(fp2);
}
```

Copying the contents of one file into another.

<u>Program for sequential comparision a file</u>

```
int  main()
{
 Char ch, k;
```

```
int f=1;
FILE  *fp1,*fp2;
fp1= fopen("x.txt","w");
puts("enter text");
while( ( ch=getchar() )!= EOF)
{
 Putc( ch , fp1);
}
fclose(fp1);
```

Writing data into a file-1

```
fp1= fopen("y.txt","w");
puts("enter text");
while( ( k=getchar() )!= EOF)
{
    Putc( k , fp2);
 }

 fclose(fp2);
```

Writing data into a file-2

```
While( !feof(fp1) || !feof(fp2) )
{
  if( getc(fp1) != getc(fp2) )
  {
     f= -1;
     break;
  }
}
if( f == 1 )
puts ("equal");
else
puts("not equal");
fclose(fp1);
fclose(fp2);
 }
```

Comparision of two files