File Handling in C

A file is a place on your physical disk where information is stored.

Why files are needed?

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all.
  However, if you have a file containing all the data, you can easily access the contents of the file using few commands in C.
- You can easily move your data from one computer to another without any changes.

Types of Files

When dealing with files, there are two types of files you should know about:

1. Text files
2. Binary files

1. Text files

Text files are the normal .txt files that you can easily create using Notepad or any simple text editors.

When you open those files, you'll see all the contents within the file as plain text. You can easily edit or delete the contents.

They take minimum effort to maintain, are easily readable, and provide least security and takes bigger storage space.

2. Binary files

Binary files are mostly the .bin files in your computer.

Instead of storing data in plain text, they store it in the binary form (0's and 1's).

They can hold higher amount of data, are not readable easily and provides a better security than text files.

File Operations

In C, you can perform four major operations on the file, either text or binary:

1. Creating a new file
2. Opening an existing file
3. Closing a file
4. Reading from and writing information to a file

Working with files

When working with files, you need to declare a pointer of type file. This declaration is needed for communication between the file and program.

```
FILE *fptr;
```

Opening a file - for creation and edit

Opening a file is performed using the library function in the "**stdio.h**" header file: fopen().

The syntax for opening a file in standard I/O is:

```
ptr = fopen("fileopen","mode")
```

For Example:

```
fopen("E:\\cprogram\\newprogram.txt","w");

fopen("E:\\cprogram\\oldprogram.bin","rb");
```

- Let's suppose the file `newprogram.txt` doesn't exist in the location `E:\cprogram`. The first function creates a new file named `newprogram.txt` and opens it for writing as per the mode 'w'.
  The writing mode allows you to create and edit (overwrite) the contents of the file.
- Now let's suppose the second binary file `oldprogram.bin` exists in the location `E:\cprogram`. The second function opens the existing file for reading in binary mode 'rb'.
  The reading mode only allows you to read the file, you cannot write into the file.

Opening Modes in Standard I/O

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
W	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
A	Open for append. i.e, Data is added to end of file.	If the file does not exists, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exists, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.

Opening Modes in Standard I/O		
File Mode	Meaning of Mode	During Inexistence of file
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exists, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exists, it will be created.

Closing a File

The file (both text and binary) should be closed after reading/writing.

Closing a file is performed using library function fclose().

```
fclose(fptr); //fptr is the file pointer associated with file to be closed.
```

Reading and writing to a text file

For reading and writing to a text file, we use the functions `fprintf()` and `fscanf()`.

They are just the file versions of `printf()` and `scanf()`. The only difference is that, fprint and fscanf expects a pointer to the structure FILE.

Example 1: Write to a text file using fprintf()

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

int main()
{
   int num;
   FILE *fptr;
   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }
```

```
    printf("Enter num: ");
    scanf("%d",&num);

    fprintf(fptr,"%d\n",num);

    fclose(fptr);

    return 0;
}
```

This program takes a number from user and stores in the file `program.txt`.

After you compile and run this program, you can see a text file program.txt created in C drive of your computer. When you open the file, you can see the integer you entered.

Example 2: Read from a text file using fscanf()

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

int main()
{
    int num;
    FILE *fptr;
```

```
   if ((fptr = fopen("C:\\program.txt","r")) ==
NULL){
        printf("Error! opening file");

        // Program exits if the file pointer returns
NULL.
        exit(1);
   }

   fscanf(fptr,"%d", &num);

   printf("Value of n=%d", num);
   fclose(fptr);

   return 0;
}
```

This program reads the integer present in the `program.txt` file and prints it onto the screen.

If you successfully created the file from **Example 1**, running this program will get you the integer you entered.

Other functions like `fgetchar()`, `fputc()` etc. can be used in similar way.

Random Access To File

The C library function **int fseek(FILE *stream, long int offset, int whence)** sets the file position of the **stream** to the given **offset**.

Declaration

Following is the declaration for fseek() function.

```
int fseek(FILE *stream, long int offset, int whence)
```

Parameters

- **stream** – This is the pointer to a FILE object that identifies the stream.
- **offset** – This is the number of bytes to offset from whence.
- **whence** – This is the position from where offset is added. It is specified by one of the following constants –

Sr.No.	Constant & Description
1	**SEEK_SET** Beginning of file
2	**SEEK_CUR** Current position of the file pointer
3	**SEEK_END** End of file

Example program for fseek():

Write a program to read last 'n' characters of the file using appropriate file functions(Here we need fseek() and fgetc()).

```
01 #include<stdio.h>
02 #include<conio.h>
```

```
03 void main()
04 {
05       FILE *fp;
06       char ch;
07       clrscr();
08       fp=fopen("file1.c", "r");
09       if(fp==NULL)
10          printf("file cannot be opened");
11       else
12       {
13              printf("Enter value of n  to read
   last 'n' characters");
14              scanf("%d",&n);
15              fseek(fp,-n,2);
16              while((ch=fgetc(fp))!=EOF)
17              {
18                   printf("%c\t",ch);
19              }
20        }
21     fclose(fp);
22     getch();
23 }
```

OUTPUT: It depends on the content in the file.

Error Handling in C

feof(), ferror(), clearerr()

Determine if a file has reached end-of-file or if an error has occurred.

Prototypes

```
#include <stdio.h>

int feof(FILE *stream);
int ferror(FILE *stream);
void clearerr(FILE *stream);
```

Description

Each `FILE*` that you use to read and write data from and to a file contains flags that the system sets when certain events occur. If you get an error, it sets the error flag; if you reach the end of the file during a read, it sets the EOF flag. Pretty simple really.

The functions **`feof()`** and **`ferror()`** give you a simple way to test these flags: they'll return non-zero (true) if they're set.

Once the flags are set for a particular stream, they stay that way until you call **`clearerr()`** to clear them.

Return Value

`feof()` and **`ferror()`** return non-zero (true) if the file has reached EOF or there has been an error, respectively.

Example

```
// read binary data, checking for eof or error
int main(void)
{
    int a;
    FILE *fp;
```

```
    fp = fopen("binaryints.dat", "rb");

    // read single ints at a time, stopping on
EOF or error:

    while(fread(&a, sizeof(int), 1, fp),
!feof(fp) && !ferror(fp)) {
        printf("I read %d\n", a);
    }

    if (feof(fp))
        printf("End of file was reached.\n");

    if (ferror(fp))
        printf("An error occurred.\n");

    fclose(fp);

    return 0;
}
```

Graphics in C

Graphics.h

graphics.h library is used to include and facilitate graphical operations in program. graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h you can make graphics programs, animations, projects and games. You can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them.

Graphics Mode Initialization

Sample graphics code

```
1.#include<graphics.h>
2.#include<conio.h>
3.
4.int main()
5.{
6.    int gd = DETECT, gm;
7.
8.    initgraph(&gd, &gm, "C:\\TC\\BGI");
9.
10.        getch();
11.        closegraph();
12.        return 0;
13.    }
```

Let me tell you what the output of this program is, this program initializes graphics mode and then closes it after a key is pressed. To begin with we have declared two variables of int type gd and gm for graphics driver and graphics mode respectively, you can choose any other variable name as well. DETECT is a macro defined in "graphics.h" header file, then we have passed three arguments to initgraph function first is the address of gd, second is the address of gm and third is the path where your BGI files are present (you have to adjust this accordingly where you Turbo C compiler is installed). Initgraph function automatically decides an appropriate graphics driver and mode such that maximum screen resolution is set, getch helps us

to wait until a key is pressed, closegraph function closes the graphics mode, and finally return statement returns a value 0 to main indicating successful execution of the program. After you have understood initgraph function then you can use functions to draw shapes such as circle, line , rectangle, etc., then you can learn how to change colors and fonts using suitable functions, then you can go for functions such as getimage, putimage, etc., for doing animation.

Graphics.h Functions

Drawing Line

line(int x1, int y1, int x2, int y2);
line function is used to draw a line from a point(x1,y1) to point(x2,y2) i.e. (x1,y1) and (x2,y2) are end points of the line.

Drawing Circle

circle() function draws a circle with center at (x, y) and given radius.

Syntax :

```
circle(x, y, radius);


where,
(x, y) is center of the circle.
'radius' is the Radius of the circle.
```

Drawing Arc

```
       arc(int x, int y, int start_angle,
            int end_angle, int radius);


where,
(x, y) is center of the arc.
start_angle is the starting angle and
```

```
end_angle is the ending angle.
'radius' is the Radius of the arc.
```

Example: `arc(100, 100, 0, 135, 50);`

Filling An OBJECT

Floodfill algorithm

```
// A recursive function to replace previous
// color 'oldcolor' at  '(x, y)' and all
// surrounding pixels of (x, y) with new
// color 'newcolor' and
floodfill(x, y, newcolor, oldcolor)
1) If x or y is outside the screen, then
   return.
2) If color of getpixel(x, y) is same as
   oldcolor, then
3) Recur for top, bottom, right and left.
    floodFill(x+1, y, newcolor, oldcolor);
    floodFill(x-1, y, newcolor, oldcolor);
    floodFill(x, y+1, newcolor, oldcolor);
    floodFill(x, y-1, newcolor, oldcolor);
```

getmaxx() and getmaxy()

Declarations :-

int getmaxx();
int getmaxy();

getmaxx function returns the maximum X coordinate for current graphics mode and driver.
getmaxy function returns the maximum Y coordinate for current graphics mode and driver.

C programming code

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

main()
{
   int gd = DETECT, gm, maxx, maxy;
   char a[100];

      initgraph(&gd,&gm,"C:\\TC\\BGI");

      maxx = getmaxx();
      maxy = getmaxy();

      sprintf(a,"Maximum X and Y coordinates for current graphics mode and driver are %d and %d respectively.",maxx,maxy);
      outtext(a);

      getch();
      closegraph();
      return 0;
   }
```