File Handling Functions in C

File is a collection of data that stored on secondary memory like hard disk of a computer.

The following are the operations performed on files in the c programming language...

- **Creating (or) Opening a file**
- **Reading data from a file**
- **Writing data into a file**
- **Closing a file**

All the above operations are performed using file handling functions available in C.

Creating (or) Opening a file

To create a new file or open an existing file, we need to create a file pointer of FILE type. Following is the sample code for creating file pointer.

File *f_ptr ;

We use the pre-defined method **fopen()** to create a new file or to open an existing file. There are different modes in which a file can be opened. Consider the following code...

```
File *f_ptr ;
*f_ptr = fopen("c://aaa/abc.txt", "w") ;
```

The above example code creates a new file called **abc.txt** if it does not exists otherwise it is opened in writing mode.

In C programming language, there different modes are available to open a file and they are shown in the following table.

S. No.	Mode	Description
1	r	Opens a text file in **reading** mode.
2	w	Opens a text file in **writing** mode.
3	a	Opens a text file in **append** mode.

S. No.	Mode	Description
4	r+	Opens a text file in both **reading and writing** mode.
5	w+	Opens a text file in both **reading and writing** mode. It set the cursor position to the begining of the file if it exists.
6	a+	Opens a text file in both **reading and writing** mode. The reading operation is performed from begining and writing operation is performed at the end of the file.

Note - The above modes are used with text files only. If we want to work with binary files we use **rb, wb, ab, rb+, wb+ and ab+**.

Reading from a file

The reading from a file operation is performed using the following pre-defined file handling methods.

1. **getc()**
2. **getw()**
3. **fscanf()**
4. **fgets()**
5. **fread()**

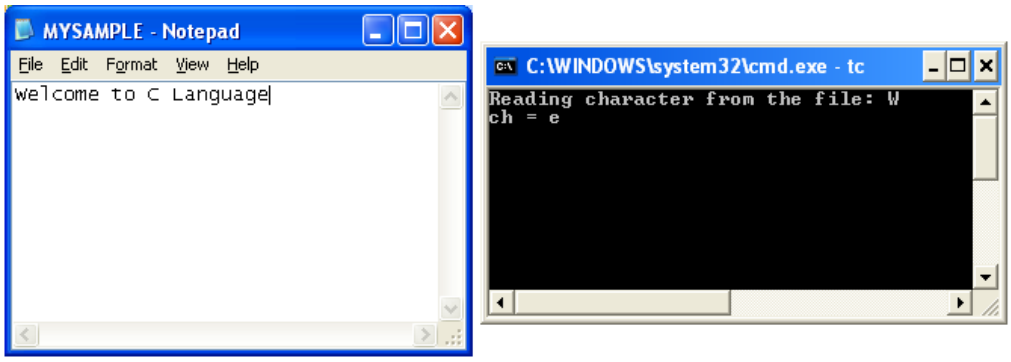
- ***getc(*file_pointer)*** - This function is used to read a character from specified file which is opened in reading mode. It reads from the current position of the cursor. After reading the character the cursor will be at next character.

Example Program to illustrate getc() in C.

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

```
    fp = fopen("MySample.txt","r");
    printf("Reading character from the file: %c\n",getc(fp));
    ch = getc(fp);
    printf("ch = %c", ch);
ch = getc(fp);
    printf("ch = %c", ch);

    fclose(fp);
    getch();
    return 0;
}
```

Output



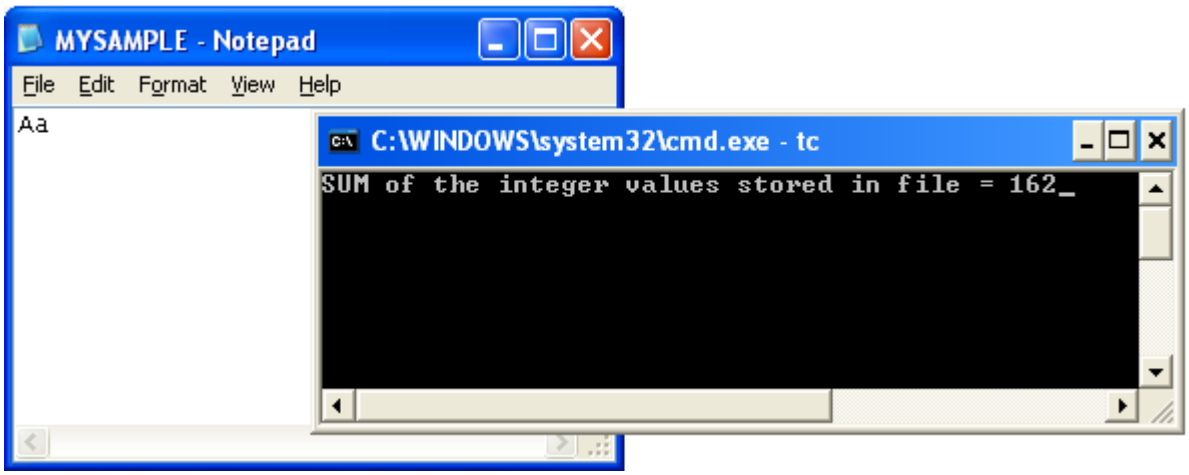
- ***getw(*file_pointer) -*** This function is used to read an integer value from the specified file which is opened in reading mode. If the data in file is set of characters then it reads ASCII values of those characters.

Example Program to illustrate getw() in C.

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

int main(){

    FILE *fp;
    int i,j;
```

```
  fp = fopen("MySample.txt","w");
  putw(65,fp);  // inserts A
  putw(97,fp);  // inserts a
  fclose(fp);
  fp = fopen("MySample.txt","r");
  i = getw(fp);   // reads 65 - ASCII value of A
  j = getw(fp);   // reads 97 - ASCII value of a
  printf("SUM of the integer values stored in file = %d", i+j);  // 65 + 97 = 162
  fclose(fp);
  getch();

  return 0;
}
```

Output



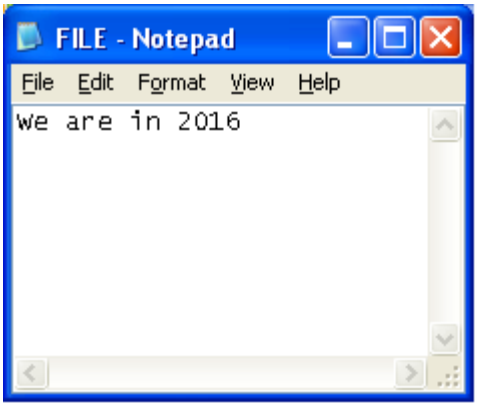
- ***fscanf(*file_pointer, typeSpecifier, &variableName) -*** This function is used to read multiple datatype values from specified file which is opened in reading mode.

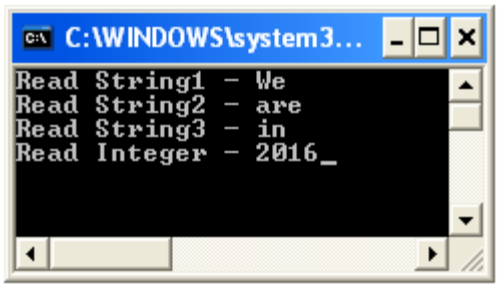
Example Program to illustrate fscanf() in C.

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

int main(){

  char str1[10], str2[10], str3[10];
```

```
    int year;
    FILE * fp;

    fp = fopen ("file.txt", "w+");
    fputs("We are in 2016", fp);
    rewind(fp);    // moves the cursor to begining of the file
    fscanf(fp, "%s %s %s %d", str1, str2, str3, &year);
    printf("Read String1 - %s\n", str1 );
    printf("Read String2 - %s\n", str2 );
    printf("Read String3 - %s\n", str3 );
    printf("Read Integer - %d", year );
    fclose(fp);
    getch();

    return 0;
}
```

Output





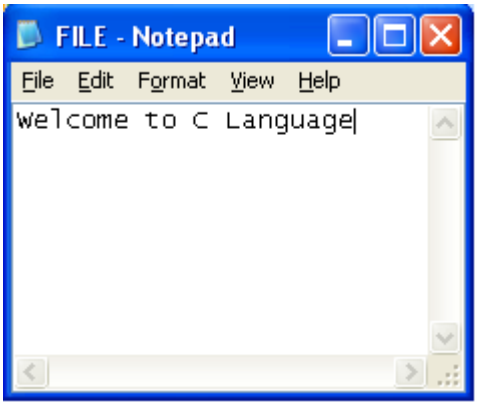
- ***fgets(variableName, numberOfCharacters, *file_pointer) -*** This method is used for reading a set of characters from a file which is opened in reading mode starting from the current cursor position. The fgets() function reading terminates with reading NULL character.

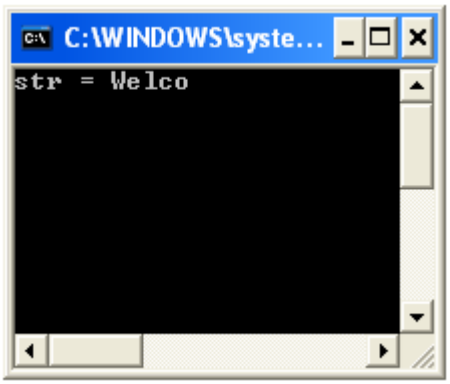
Example Program to illustrate fgets() in C.

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

```
int main(){

  FILE *fp;
  char str[] = “Welcome”;

  fp = fopen ("file.txt", "r");
  fgets(str,6,fp);
  printf("str = %s", str);
  fclose(fp);
  getch();

  return 0;
}
```

Output





- ***fread(source, sizeofReadingElement, numberOfCharacters, FILE *pointer) -*** This function is used to read specific number of sequence of characters from the specified file which is opened in reading mode.

Example Program to illustrate fgets() in C.

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


int main(){
```

```
    FILE *fp;
    char *str;

    fp = fopen ("file.txt", "r");
    fread(str,sizeof(char),5,fp);
    str[strlen(str)+1] = 0;
    printf("str = %s", str);
    fclose(fp);
    getch();

    return 0;
}
```

Output

Writing into a file

The writing into a file operation is performed using the following pre-defined file handling methods.

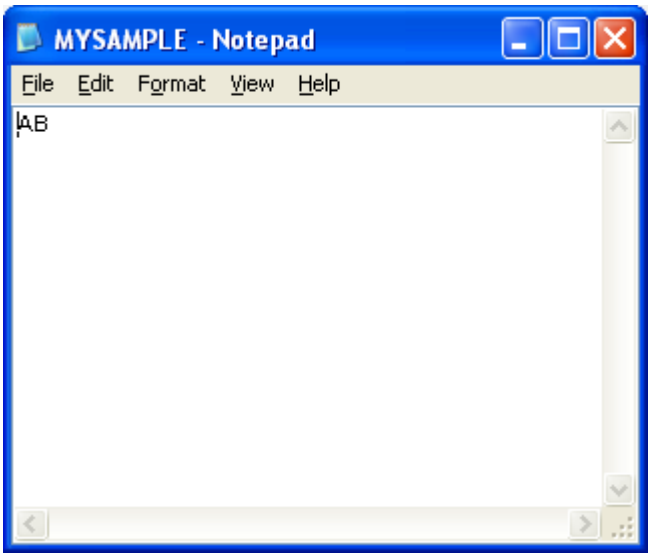
1. **putc()**
2. **putw()**
3. **fprintf()**
4. **fputs()**
5. **fwrite()**

- ***putc(char, *file_pointer) -*** This function is used to write/insert a character to the specified file when the file is opened in writing mode.

Example Program to illustrate putc() in C.

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


int main(){

  FILE *fp;
  char ch;

  fp = fopen("C:/TC/EXAMPLES/MySample.txt","w");
  putc('A',fp);
  ch = 'B';
  putc(ch,fp);
  fclose(fp);
  getch();

  return 0;
}
```

Output

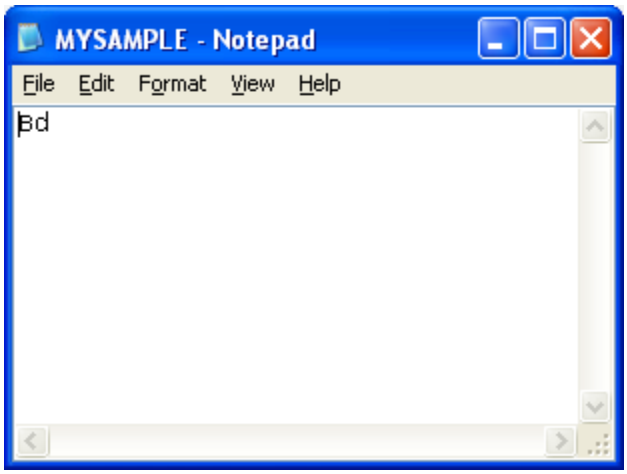


- ***putw(int, *file_pointer) -*** This function is used to writes/inserts an integer value to the specified file when the file is opened in writing mode.

Example Program to illustrate putw() in C.

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


int main(){

  FILE *fp;
  int i;

  fp = fopen("MySample.txt","w");
  putw(66,fp);
  i = 100;
  putw(i,fp);
  fclose(fp);
  getch();

  return 0;
}
```

Output

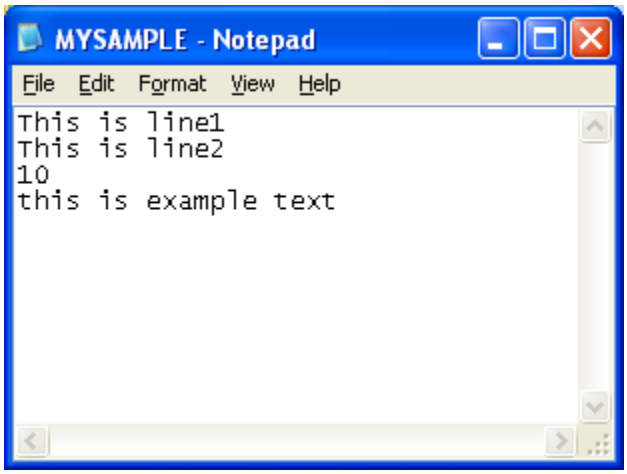


- ***fprintf(*file_pointer, "text") -*** This function is used to writes/inserts multiple lines of text with mixed data types (char, int, float, double) into specified file which is opened in writing mode.

Example Program to illustrate "fprintf()" in C.

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


int main(){

  FILE *fp;
  char *text = "\nthis is example text";
  int i = 10;

  fp = fopen("MySample.txt","w");
  fprintf(fp,"This is line1\nThis is line2\n%d", i);
    fprintf(fp,text);
  fclose(fp);
  getch();

  return 0;
}
```

Output

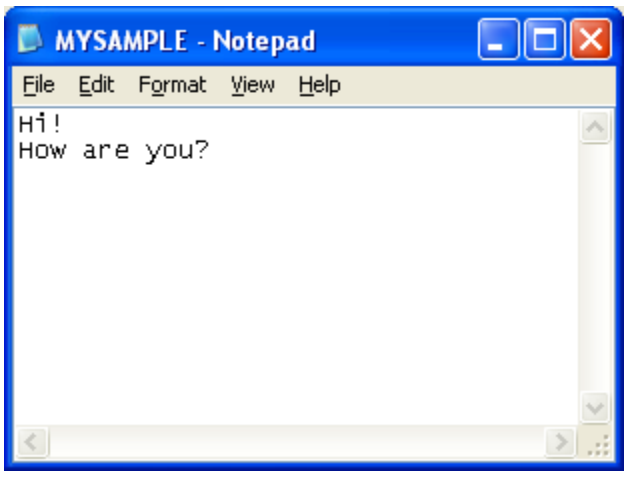


- ***fputs("string", *file_pointer) -*** TThis method is used to insert string data into specified file which is opened in writing mode.

Example Program to illustrate fputs() in C.

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


int main(){

  FILE *fp;
  char *text = "\nthis is example text";

  fp = fopen("MySample.txt","w");
  fputs("Hi!\nHow are you?",fp);
  fclose(fp);
  getch();

  return 0;
}
```

Output

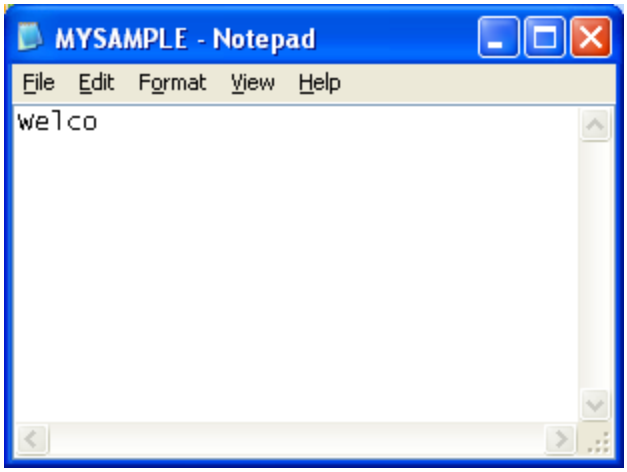


- ***fwrite("StringData", sizeof(char), numberOfCharacters, FILE *pointer) -*** This function is used to insert specified number of characters into a binary file which is opened in writing mode.

Example Program to illustrate fwrite() in C.

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


int main(){

  FILE *fp;
  char *text = "Welcome to C Language";

  fp = fopen("MySample.txt","wb");
  fwrite(text,sizeof(char),5,fp);
  fclose(fp);
  getch();

  return 0;
}
```

Output



Closing a file

Closing a file is performed using a pre-defined method fclose().

fclose(*f_ptr)

The method fclose() returns '0'on success of file close otherwise it returns EOF (End Of File).

Cursor Positioning Functions in Files

C programming language provides various pre-defined methods to set the cursor position in files. The following are the methods available in c, to position cursor in a file.

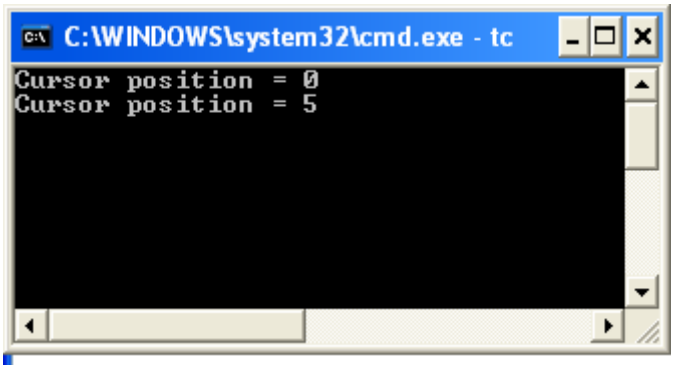
1. **ftell()**
2. **rewind()**
3. **fseek()**

- ***ftell(*file_pointer) -*** This function returns the current position of the cursor in the file.

Example Program to illustrate ftell() in C.

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


int main(){

  FILE *fp;
  int position;

  fp = fopen ("file.txt", "r");
  position = ftell(fp);
  printf("Cursor position = %d\n",position);
  fseek(fp,5,0);
  position = ftell(fp);
  printf("Cursor position = %d", position);
  fclose(fp);
  getch();

  return 0;
}
```

Output

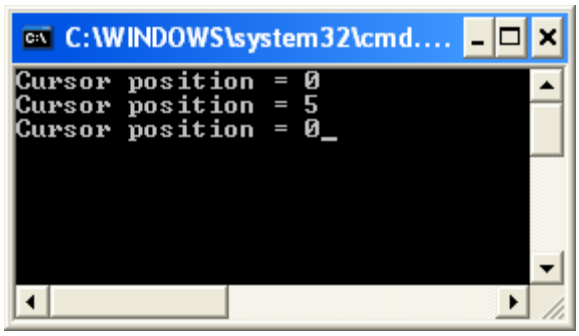


- ***rewind(*file_pointer) -*** This function is used reset the cursor position to the beginning of the file.

Example Program to illustrate rewind() in C.

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


int main(){

  FILE *fp;
  int position;

  fp = fopen ("file.txt", "r");
  position = ftell(fp);
  printf("Cursor position = %d\n",position);
  fseek(fp,5,0);
  position = ftell(fp);
  printf("Cursor position = %d\n", position);
  rewind(fp);
  position = ftell(fp);
  printf("Cursor position = %d", position);
  fclose(fp);
  getch();

  return 0;
}
```

Output



- ***fseek(*file_pointer, numberOfCharacters, fromPosition) -*** This function is used to set the cursor position to the specific position. Using this function we can set the cursor position from three different position they are as follows.
    - from beginning of the file (indicated with 0)
    - from current cursor position (indicated with 1)
    - from ending of the file (indicated with 2)

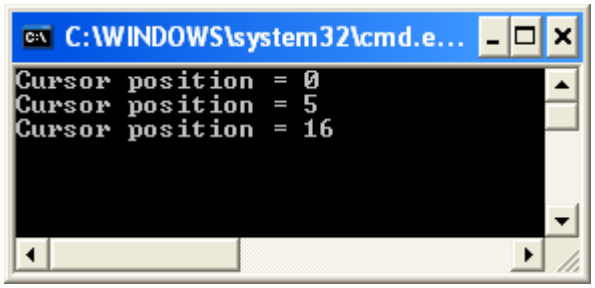
- #define SEEK_SET 0 /* set file offset to offset */
- #define SEEK_CUR 1 /* set file offset to current plus offset */
- #define SEEK_END 2 /* set file offset to EOF plus offset */

Example Program to illustrate fseek() in C.

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


int main(){

  FILE *fp;
  int position;

  fp = fopen ("file.txt", "r");
```

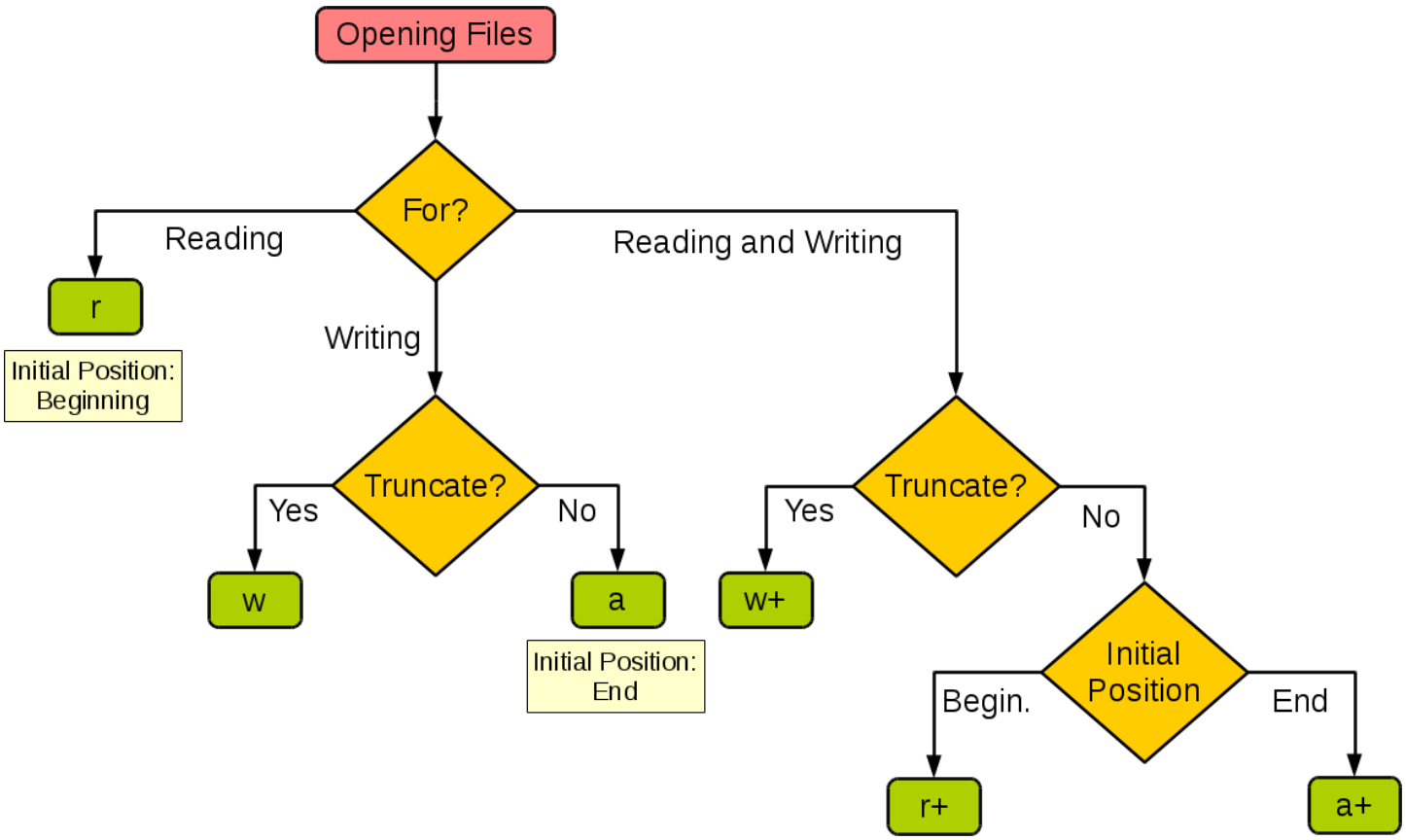
```
    position = ftell(fp);
    printf("Cursor position = %d\n",position);
    fseek(fp,5,0);
    position = ftell(fp);
    printf("Cursor position = %d\n", position);
    fseek(fp, -5, 2);
    position = ftell(fp);
    printf("Cursor position = %d", position);
    fclose(fp);
    getch();

    return 0;
}
```

Output



Summary

Both `r+` and `w+` can read and write to a file. However, `r+` doesn't delete the content of the file and doesn't create a new file if such file doesn't exist, whereas `w+` deletes the content of the file and creates it if it doesn't exist.



Error Handling in C

C programming language does not support error handling that are occured at program execution time. However, C provides a header file called **error.h**. The header file **error.h** contains few methods and variables that are used to locate error occured during the program execution. Generally, c programming function returns **NULL** or **-1** in case of any error occured, and there is a global variable called **errno** which stores the error code or error number. The following table lists few errno values and thier meaning.

Error Number	Meaning
1	Specified operation not permitted
2	No such file or directory.
3	No such process.

Error Number	Meaning
4	Interrupted system call.
5	IO Error
6	No such device or address
7	Argument list too long
8	Exec format error
9	Bad file number
10	No child processes
11	Try again
12	Out of memory
13	Permission denied

C programming language provides the following two methods to represent errors occured during program execution.

- **perror( )**
- **strerror( )**

perror( ) - The perror() function returns a string passed to it along with the textual representation of current errno value.

strerror( ) - The strerror() function returns a pointer to the string representation of the current errno value. This method is defined in the header file **string.h**

Consider the following example program...

Example Program to illustrate error handling in C.

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


int main(){

   FILE *f_ptr;

   f_ptr = fopen("abc.txt", "r");

   if(f_ptr == NULL){
      printf("Value of errno: %d\n ", errno);
      printf("The error message is : %s\n", strerror(errno));
      perror("Message from perror");
   }
   else{
      printf("File is opened in reading mode!");
      fclose(f_ptr);
   }

   return 0;
}
```

Output

```
return 0;
}
```

Output