Files handling in C++

REVISION OF C++ LANGUAGE

Week-09

WHAT IS A FILE?

- A named collection of data, stored in secondary storage (typically).
- Typical operations on files:
  - Open
  - Read
  - Write
  - Close
- How is a file stored?
  - Stored as sequence of bytes, logically contiguous (may not be physically contiguous on disk).
  - The last byte of a file contains the end-of-file character (EOF).
  - While reading a text file, the EOF character can be checked to know the end.

MANIPULATING USER FILES

- Step 1: open a stream connected to the file

  fopen command
- Step 2: read data from the file or write data to the file using the stream input/output commands
- Step 3: close the connection to the file

  fclose command

FILE MANAGEMENT

- Following are the most important file management functions available in 'C++' Opening a file

Function	Purpose
fopen ()	Creating a file or opening an existing file
fclose ()	Closing a file
getc ()	Reads a single character from a file
putc ()	Writes a single character to a file
getw ()	Reads an integer from a file
putw ()	Writing an integer to a file

FILE POINTERS

- A file pointer is a pointer which is used to handle and keep track on the files being accessed. A new data type called "FILE" is used to declare file pointer. File pointer is declared as
  - ❖FILE *fp.
  - ❖This says that fp is the file pointer that points to a FILE structure.

FOPEN COMMAND

- Following syntax is used
  - FILE *fp; → File must be written in Capital letters
  - fp = fopen ("file_name", "mode");
  - fp = fopen ("file.txt", “w");

- fopen returns a value of type FILE * that is a stream connected to the specified file
- In the above syntax, the file is a data structure which is defined in the standard library.
- fopen is a standard function which is used to open a file.
- If the file is not present on the system, then it is created and then opened.
- If a file is already present on the system, then it is directly opened using this function.

MODE FOR OPENING FILES

- Following syntax is used
  - FILE *fp;
  - fp = fopen ("file_name", "mode");
- A mode is used to specify whether you want to open a file for any of the below-given purposes.

File Mode	Description
r	Open a file for reading. If a file is in reading mode, then no data is deleted if a file is already present on a system.
w	Open a file for writing. If a file is in writing mode, then a new file is created if a file doesn't exist at all. If a file is already present on a system, then all the data inside the file is truncated, and it is opened for writing purposes.
a	Open a file in append mode. If a file is in append mode, then the file is opened. The content within the file doesn't change.

FCLOSE COMMAND

Syntax: fclose (FilePointer)

- The file pointer must be a stream opened using fopen (that remains open)
- fclose returns
  - 0 if the fclose command is successful
  - special value EOF if the fclose command is unsuccessful

READING MODE

Fgetc( )

- Read and returns the next character from the file pointed to.
- Note: The operation of the file pointer passed in as a parameter must be "r" for read, or you will suffer an error.
    - ❖Char ch = fgetc(<filepointer>);
- The ability to get single character from files, if wrapped in a loop, means we could read all the characters from a file and print them to the screen, one-by-one, essentially

```
Char ch;
While ((ch=fgetc(<filepointer>)) !=EOF)
```

WRITING MODE

Fputc( )

- Write or append the specified character to the pointed-to- file.
- Note: The operation of the file pointer passed in as a parameter must be "w" for a write, or "a" for append , or you will suffer an error.

❖fputc(<character> ,<filepointer>);

```
Char ch;
 While ((ch=fgetc(filepointer)) !=EOF)
  fputc(ch, ptr2);
```

EXAMPLE-1-

```
# include<iostream>
using namespace std;
int main ( ) {
FILE *myfile;
char ch;
myfile= fopen("Test.txt", "w");
cin>>ch;
putc(ch, myfile);
fclose (myfile);
return 0;
}
```

EXAMPLE-2-

Write multiple letters into a file until press ‘a’ letter to stop.

```
# include<iostream>
using namespace std;
int main ( ) {
FILE *myfile;
char ch;
myfile = fopen("test.txt", "w");
do{
cin>>ch;
if (ch!= 'a')
putc(ch, myfile);
} while (ch!='a');
fclose(myfile);
return 0;
}
```

Use **cin.get(ch)** to take the spaces in the consideration during typing the letters.

EXAMPLE-3-

Writing string to a file.

```
#include <iostream>
using namespace std;
int main()
{
    FILE *fpw;
    char str[100]; //*Char array to store strings */
    fpw = fopen("Test.txt", "w");
    if (fpw== NULL) //*Error handling for output file*
    {
        puts("Issue in opening the Output file");
    }
    cout <<"Enter your string:";
    gets(str); //*Stored the input string into array str*
    fputs(str, fpw); //* Copied the content of str into file
    fclose(fpw);
    return 0;
}
```

EXAMPLE-4-

Read multiple letters from a file and stop when the file gets end.

```
# include<iostream>
using namespace std;
int main ( ) {
FILE *myfile;
char ch;
myfile = fopen("Test.txt", "r");
while ((ch= getc (myfile)) !=EOF) {
cout <<ch;
};
cout<< endl;
fclose(myfile);
system("pause");
return 0;
}
```

EXAMPLE-5-

Write a program to backup a text (“b.text”) file from the original one (“o.text”).

```
# include<iostream>
using namespace std;
int main ( ) {
FILE *original;
FILE *backup;
char ch;
original = fopen (“o.txt", "r");
backup = fopen ("b.txt", "w");
while ((ch = getc (original)) !=EOF) {
putc (ch, backup);
cout <<ch;
}
fclose(original);
fclose(backup);
return 0;
}
```

SOME IMPORTANT NOTES

- In case the file was locates in another drive(i.e D:\new\one.txt), thus the path should be written as follow:

myfile = fopen(**D:\\new**\\one.txt, "w"); // this is because \n is considered **an escape sequence,** to represent a newline character.

- In read mode, the file should be exist with known saved location.

THE END