**File Handling through C++ Classes**

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.   
**ofstream:** Stream class to write on files   
**ifstream:** Stream class to read from files   
**fstream:** Stream class to both read and write from/to files.

Diagram

Description automatically generated

**Modes :**

| Member Constant | Stands For | Access |
| --- | --- | --- |
| in \* | input | File open for reading: the internal stream buffer supports input operations. |
| out | output | File open for writing: the internal stream buffer supports output operations. |
| binary | binary | Operations are performed in binary mode rather than text. |
| ate | at end | The output position starts at the end of the file. |
| app | append | All output operations happen at the end of the file, appending to its existing contents. |
| trunc | truncate | Any contents that existed in the file before it is open are discarded. |

**Default Open Modes :**

|  |  |
| --- | --- |
| ifstream | ios::in |
| ofstream | ios::out |
| fstream | ios::in | ios::out |

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream.

* ofstream: This Stream class signifies the output file stream and is applied to create files for writing information to files
* ifstream: This Stream class signifies the input file stream and is applied for reading information from files
* fstream: This Stream class can be used for both read and write from/to files.

All the above three classes are derived from fstreambase and from the corresponding iostream class and they are designed specifically to manage disk files.  
C++ provides us with the following operations in File Handling:

* Creating a file: open()
* Reading data: read()
* Writing new data: write()
* Closing a file: close()

Moving on with article on File Handling in C++

**Opening a File**

Generally, the first operation performed on an object of one of these classes is to associate it to a real file. This procedure is known to open a file.

We can open a file using any one of the following methods:  
1. First is bypassing the file name in constructor at the time of object creation.  
2. Second is using the open() function.

To open a file use

|  |  |
| --- | --- |
| 1 | open() function |

**Syntax**

|  |  |
| --- | --- |
| 1 | void open(const char\* file\_name,ios::openmode mode); |

Here, the first argument of the open function defines the name and format of the file with the address of the file.

The second argument represents the mode in which the file has to be opened. The following modes are used as per the requirements.

|  |  |
| --- | --- |
| ***Modes*** | ***Description*** |
| in | Opens the file to read(default for ifstream) |
| out | Opens the file to write(default for ofstream) |
| binary | Opens the file in binary mode |
| app | Opens the file and appends all the outputs at the end |
| ate | Opens the file and moves the control to the end of the file |
| trunc | Removes the data in the existing file |
| nocreate | Opens the file only if it already exists |
| noreplace | Opens the file only if it does not already exist |

**Example**

|  |  |
| --- | --- |
| 1  2 | fstream new\_file;  new\_file.open(“newfile.txt”, ios::out); |

In the above example, new\_file is an object of type fstream, as we know fstream is a class so we need to create an object of this class to use its member functions. So we create new\_file object and call open() function. Here we use out mode that allows us to open the file to write in it.

Default Open Modes :

* ifstream ios::in
* ofstream ios::out
* fstream ios::in | ios::out

We can combine the different modes using or symbol | .