**Data File Handling In C++**

**File.**The information / data stored under a specific name on a storage device, is called a file.

his requires another standard C++ library called **fstream**, which defines three new data types:

1. Ofstream: This file handling class in C++ signifies the output file stream and is applied to create files for writing information to files
2. Ifstream: This file handling class in C++ signifies the input file stream and is applied for reading information from files
3. Fstream: This file handling class in C++ signifies the file stream generally, and has the capabilities for representing both ofstream and ifstream

Opening a File

A file must be opened before you can read from it or write to it. Either the **ofstream** or **fstream** object may be used to open a file for writing and ifstream object is used to open a file for reading purpose only.

Following is the standard syntax for open() function, which is a member of fstream, ifstream, and ofstream objects.

void open(const char \*filename, ios::openmode mode);

Here, the first argument specifies the name and location of the file to be opened and the second argument of the **open()** member function defines the mode in which the file should be opened.

* ios::app: append mode
* ios::ate: open a file in this mode for output and read/write controlling to the end of the file
* ios::in: open file in this mode for reading
* ios::out: open file in this mode for writing
* ios::trunk: when any file already exists, its contents will be truncated before file opening

You can combine two or more of these values by **OR**ing them together. For example if you want to open a file in write mode and want to truncate it in case it already exists, following will be the syntax:

ofstream outfile;

outfile.open("file.dat", ios::out | ios::trunc );

Similar way, you can open a file for reading and writing purpose as follows:

fstream afile;

afile.open("file.dat", ios::out | ios::in );

## Closing a File

When a C++ program terminates it automatically closes flushes all the streams, release all the allocated memory and close all the opened files. But it is always a good practice that a programmer should close all the opened files before program termination.

Following is the standard syntax for close() function, which is a member of fstream, ifstream, and ofstream objects.

void close();

General functions used for File handling

1. open(): To create a file
2. close(): To close an existing file
3. get(): to read a single character from a file
4. put(): to write a single character in file
5. read():to read data from file
6. write(): to write data into file

## Writing to a File

While doing C++ programming, you write information to a file from your program using the stream insertion operator (<<) just as you use that operator to output information to the screen. The only difference is that you use an **ofstream** or **fstream** object instead of the **cout** object.

#include <iostream>

#include<fstream>

using namespace std;

int main()

{

ofstream fout;

fout.open("myfile.txt" ,ios::trunc);

if(fout.is\_open()){

fout<<"hello";

}

else{

cout<<"error";

}

fout.close();

return 0;

}

Reading from file:

## Reading from a File

You read information from a file into your program using the stream extraction operator (>>) just as you use that operator to input information from the keyboard. The only difference is that you use an **ifstream** or **fstream** object instead of the **cin** object.

#include <iostream>

#include<fstream>

using namespace std;

int main()

{

ifstream file;

file.open("myfile.txt" );

if(file.is\_open()){

string line;

getline(file,line);

cout<<line;

}

return 0;

}

Read multiple line:

#include <iostream>

#include<fstream>

using namespace std;

int main()

{

ifstream file;

file.open("myfile.txt" );

if(file.is\_open()){

string line;

while(file.good()){

getline(file,line);

cout<<line<<endl;

}

}

return 0;

}

std::ifstream is ("test.txt", std::ifstream::binary);

if (is) {

// get length of file:

is.seekg (0, is.end);

int length = is.tellg();

is.seekg (0, is.beg);

char \* buffer = new char [length];

std::cout << "Reading " << length << " characters... ";

// read data as a block:

is.read (buffer,length);

if (is)

std::cout << "all characters read successfully.";

else

std::cout << "error: only " << is.gcount() << " could be read";

is.close();

// ...buffer contains the entire file...

delete[] buffer;

}

Read and write function:

#include <iostream>

#include<fstream>

#include <string.h>

using namespace std;

class Person{

char name[30];

int age;

public:

Person(){

strcpy(name,"noname");

age=0;

}

Person(char \*name,int age){

strcpy(this->name,name);

this->age=age;

}

void display(){

cout<<"hi am "<<name<<"and i m "<<age<<"years old"<<endl;

}

};

int main(){

Person p =Person("Amit",23);

fstream file("person.bin",ios::binary | ios::in | ios::out | ios::trunc);

if(!file.is\_open()){

cout<<" error while opening";

}else{

file.write((char \*)&p , sizeof(p));

file.seekg(0);

Person pe;

file.read((char \*)&pe , sizeof(p));

p.display();

pe.display();

}

return 0;

}