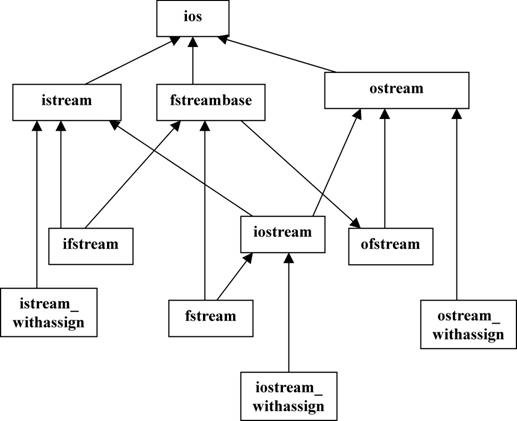
**Object Filing in C++**

**(Written By Dr. Saqib Majeed)**

**Stream Class Hierarchy:**



**Creating File Object:**

fileClassName fileObjectName (“FileName”, mode);

**C++ className for filing:**

ofstream 🡪 Stream class for writing data in files (default mode is ios::out)

ifstream 🡪 Stream class for reading data in files (default mode is ios::in)

fstream 🡪 Stream class for both above (default mode is ios::out|ios::in)

Note: if we add any mode in fstream then default overrides

**C++ mode for filing:**

|  |  |
| --- | --- |
| ios::in | Open for input operations. |
| ios::out | Open for output operations. |
| ios::binary | Open in binary mode. |
| ios::ate | Set the initial position at the end of the file. If this flag is not set, the initial position is the beginning of the file. |
| ios::app | All output operations are performed at the end of the file, appending the content to the current content of the file. |
| ios::trunc | If the file is opened for output operations and it already existed, its previous content is deleted and replaced by the new one. |

**Opening a File:**

fstream file(“D://UIIT//record.txt”, ios::out| ios::in| ios::binary| ios::app);

OR

fstream file;

file.open(“D://UIIT/record.txt”, ios::out| ios::in| ios::binary| ios::app);

**Closing a File:**

file.close();

**Data Flush to file without closing file:**

file.flush();

**File Object connection with File:**

if (!file.is\_open()) { /\* file stream not opening a file \*/ }

**File stream state flags:**

bad()

Returns true if a reading or writing operation fails. For example, in the case that we try to write to a file that is not open for writing or if the device where we try to write has no space left.

fail()

Returns true in the same cases as bad(), but also in the case that a format error happens, like when an alphabetical character is extracted when we are trying to read an integer number.

eof()

Returns true if a file open for reading has reached the end.

good()

It is the most generic state flag: it returns false in the same cases in which calling any of the previous functions would return true. Note that good and bad are not exact opposites (good checks more state flags at once).

clear()   
The member function clear() can be used to reset the state flags.

**Read Pointers:**

streampos position=**tellg()**; //return the position of read pointer

**seekg(position)**; //change the position of read pointer at a specific location

OR

seekg(position, direction); //direction may be: ios::beg

// ios::curr

// ios::end

**Write Pointers:**

streampos position= **tellp()**; //return the position of write pointer

**seekp(position)**; //change the position of write pointer at a specific location

OR

seekp(position, direction);

**Example:**

file.seekp(file.tellg() - sizeof(this));

//relocating the position of write pointer one object back from the

//current position of read pointer, where this means calling object.

**Writing Data in file for calling object:**

file.write((char \*)this, sizeof(\*this));

**Readinging Data from file for calling object:**

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

**Example Code:**

//Tested on Visual Studio 2015

#include"stdafx.h"

#include<iostream>

#include<conio.h>

#include<fstream>

#include<stdio.h> //for dos remove and rename command

using namespace std;

class Student

{

int rollno;

char name[20];

public:

void getData()

{

cout << "\nEnter Roll No;"; cin >> rollno;

cout << "Enter First Name:"; cin >> name;

}

void showData()

{

cout << "\n" << rollno << "\t" << name;

}

void writeFile()

{

fstream file("record.txt", ios::binary | ios::out | ios::app);

if (!file)

cout << "\nFile Writing Object Error....";

else

{

getData();

file.write((char\*)this, sizeof(\*this));

cout << "\nThe following record added successfully";

cout <<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout << "\nRoll No\tFirst Name:";

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

showData();

}

file.close();

}

void readFile()

{

int count = 0;

fstream file("record.txt", ios::binary | ios::in);

if (!file)

cout << "File reading error....";

else

{

cout << "\nFollowing are the Records from File";

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout << "\nRoll No\tFirst Name:";

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

file.read((char\*)this, sizeof(\*this));

while (!file.eof())

{

count++;

showData();

file.read((char\*)this, sizeof(\*this));

}

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\nTotal Records found="<<count;

file.close();

}

void searchFile()

{

fstream file("record.txt",ios::binary|ios::in);

if(!file)

cout<<"\nReading error";

else

{ cout<<"\nEnter Roll No you want to search:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

if(r==rollno)

{

showData();

count++;

}

file.read((char\*)this, sizeof(\*this));

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\nTotal Records found="<<count;

}

file.close();

}

void updateRecord()

{

fstream file("record.txt",ios::binary|ios::in|ios::out);

/\*we will not use ios::app because we need random updation,

however ios::app only allow updation after the eof

\*/

if(!file)

cout<<"\nFile Reading error....";

else

{ cout<<"\nEnter Roll No you want to update:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

if(r==rollno)

{

count++;

streampos rposition=file.tellg();

int size=sizeof(\*this);

file.seekp((int)rposition - size);

getData();

file.write((char\*)this, sizeof(\*this));

break; //removing one record at a time

}

file.read((char\*)this, sizeof(\*this));

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\n"<<count<<" Record updated successfully";

}

file.close();

}

void deleteRecord()

{

fstream file("record.txt",ios::binary|ios::in|ios::out);

fstream tempfile("record\_t.txt", ios::binary | ios::out);

if(!file)

cout<<"\nFile Reading error";

else

{

cout<<"\nEnter Roll No you want to delete:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while (!file.eof())

{

if (r == rollno)

{

count++;

}

else

{

tempfile.write((char\*)this, sizeof(\*this));

}

file.read((char\*)this, sizeof(\*this));

}

if (count == 0)

cout << "\nRecord not found";

else

{

cout << "\n" << count << " Record deleted successfully";

tempfile.close();

file.close();

remove("record.txt");

rename("record\_t.txt", "record.txt");

}

}

file.close();

}

};

int main()

{

int option;

Student s1;

do

{

system("cls");

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout << "\n\tWelcome to the UIIT Filing Program";

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout << "\n\tDeveloped by Dr. Saqib Majeed";

cout << "\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout << "\n\tPress 1 for write data into file";

cout << "\n\tPress 2 for read data from file";

cout << "\n\tPress 3 for search data from file";

cout << "\n\tPress 4 for update record in file";

cout << "\n\tPress 5 for delete record from file:";

cout << "\n\tEnter Option:";

cin >> option;

switch (option)

{

case 1: s1.writeFile(); break;

case 2: s1.readFile(); break;

case 3: s1.searchFile(); break;

case 4: s1.updateRecord(); break;

case 5: s1.deleteRecord(); break;

}

cout << "\n\n\tDo you want to continue...y/n:";

} while (\_getche() == 'y');

cout << "\n\tThanks for using this program, Bye!";

\_getch();

return 0;

}

**Example Code:**

//Tested on Turbo C++

#include<iostream.h>

#include<conio.h>

#include<fstream.h>

#include<stdio.h> //for dos remove and rename command

class Student

{

int rollno;

char name[20];

public:

void getData()

{

cout<<"\nEnter Roll No;";cin>>rollno;

cout<<"Enter First Name:";cin>>name;

}

void showData()

{

cout<<"\n"<<rollno<<"\t"<<name;

}

void writeFile()

{

fstream file("record.txt",ios::binary|ios::out|ios::app);

if(!file)

cout<<"\nFile Writing Object Error....";

else

{ getData();

file.write((char\*)this, sizeof(\*this));

cout<<"\nThe following record added successfully";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout<<"\nRoll No\tFirst Name:";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

showData();

}

file.close();

}

void readFile()

{

int count=0;

fstream file("record.txt",ios::binary|ios::in);

if(!file)

cout<<"File reading error....";

else

{ cout<<"\nFollowing are the Records from File";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout<<"\nRoll No\tFirst Name:";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

count++;

showData();

file.read((char\*)this, sizeof(\*this));

}

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\nTotal Records found="<<count;

file.close();

}

void searchFile()

{

fstream file("record.txt",ios::binary|ios::in);

if(!file)

cout<<"\nReading error";

else

{ cout<<"\nEnter Roll No you want to search:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

if(r==rollno)

{

showData();

count++;

}

file.read((char\*)this, sizeof(\*this));

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\nTotal Records found="<<count;

}

file.close();

}

void updateRecord()

{

fstream file("record.txt",ios::binary|ios::in|ios::out);

/\*we will not use ios::app because we need random updation,

however ios::app only allow updation after the eof

\*/

if(!file)

cout<<"\nFile Reading error....";

else

{ cout<<"\nEnter Roll No you want to update:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

if(r==rollno)

{

count++;

streampos rposition=file.tellg();

int size=sizeof(\*this);

file.seekp(rposition-size);

getData();

file.write((char\*)this, sizeof(\*this));

}

file.read((char\*)this, sizeof(\*this));

}

if(count==0)

cout<<"\nRecord not found";

else

cout<<"\n"<<count<<" Record updated successfully";

}

file.close();

}

void deleteRecord()

{

fstream file("record.txt",ios::binary|ios::in|ios::out);

if(!file)

cout<<"\nFile Reading error";

else

{ fstream tempfile("temp.txt",ios::binary|ios::in|ios::out);

cout<<"\nEnter Roll No you want to delete:";

int r; int count=0;

cin>>r;

file.read((char\*)this, sizeof(\*this));

while(!file.eof())

{

if(r==rollno)

{

count++;

}

else

{

tempfile.write((char\*)this, sizeof(\*this));

}

file.read((char\*)this, sizeof(\*this));

}

if(count==0)

cout<<"\nRecord not found";

else

{ cout<<"\n"<<count<<" Record deleted successfully";

tempfile.close();

file.close();

remove("record.txt");

rename("temp.txt", "record.txt");

}

}

file.close();

}

};

void main()

{

int option;

Student s1;

do

{ clrscr();

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout<<"\n\tWelcome to the UIIT Filing Program";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout<<"\n\tDeveloped by Dr. Saqib Majeed";

cout<<"\n&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&";

cout<<"\n\tPress 1 for write data into file";

cout<<"\n\tPress 2 for read data from file";

cout<<"\n\tPress 3 for search data from file";

cout<<"\n\tPress 4 for update record in file";

cout<<"\n\tPress 5 for delete record from file:";

cout<<"\n\tEnter Option:";

cin>>option;

switch(option)

{

case 1: s1.writeFile(); break;

case 2: s1.readFile(); break;

case 3: s1.searchFile(); break;

case 4: s1.updateRecord(); break;

case 5: s1.deleteRecord(); break;

}

cout<<"\n\n\tDo you want to continue...y/n:";

}while(getche()=='y');

cout<<"\n\tThanks for using this program, Bye!";

getch();

}