

SALE!

✕

GeeksforGeeks Courses Upto 25% Off Enroll Now!

[Save 25% on Courses](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [T](#)

File Handling through C++ Classes

Difficulty Level : Medium • Last Updated : 02 Nov, 2022

[Read](#) [Discuss](#) [Courses](#) [Practice](#) [Video](#)

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).

How to achieve the File Handling

For achieving file handling we need to follow the following steps:-

STEP 1-Naming a file

STEP 2-Opening a file

STEP 3-Writing data into the file

STEP 4-Reading data from the file

STEP 5-Closing a file.

Streams in C++ :-

We give input to the executing program and the execution program gives back the output. The sequence of bytes given as input to the executing program and the sequence of bytes that comes as output from the executing program are called stream. In other words, streams are nothing but the flow of data in a sequence.

The input and output operation between the executing program and the devices like keyboard and monitor are known as "console I/O operation". The input and output operation between the executing program and files are known as "disk I/O operation".

Classes for File stream operations :-

The I/O system of C++ contains a set of classes which define the file handling methods. These include ifstream, ofstream and fstream classes. These classes are derived from fstream and from the corresponding iostream class. These classes, designed to manage

the disk files, are declared in `fstream` and therefore we must include this file in any program that uses files.

1. `ios`:-

AD

- `ios` stands for input output stream.
- This class is the base class for other classes in this class hierarchy.
- This class contains the necessary facilities that are used by all the other derived classes for input and output operations.

2. `istream`:-

- `istream` stands for input stream.
- This class is derived from the class '`ios`'.
- This class handle input stream.
- The extraction operator(`>>`) is overloaded in this class to handle input streams from files to the program execution.
- This class declares input functions such as `get()`, `getline()` and `read()`.

3. `ostream`:-

- `ostream` stands for output stream.
- This class is derived from the class '`ios`'.
- This class handle output stream.
- The insertion operator(`<<`) is overloaded in this class to handle output streams to files from the program execution.
- This class declares output functions such as `put()` and `write()`.

4. `streambuf`:-

- This class contains a pointer which points to the buffer which is used to manage the input and output streams.

5. `fstreambase`:-

- This class provides operations common to the file streams. Serves as a base for fstream, ifstream and ofstream class.
- This class contains open() and close() function.

6. ifstream:-

- This class provides input operations.
- It contains open() function with default input mode.
- Inherits the functions get(), getline(), read(), seekg() and tellg() functions from the istream.

7. ofstream:-

- This class provides output operations.
- It contains open() function with default output mode.
- Inherits the functions put(), write(), seekp() and tellp() functions from the ostream.

8. fstream:-

- This class provides support for simultaneous input and output operations.
- Inherits all the functions from istream and ostream classes through iostream.

9. filebuf:-

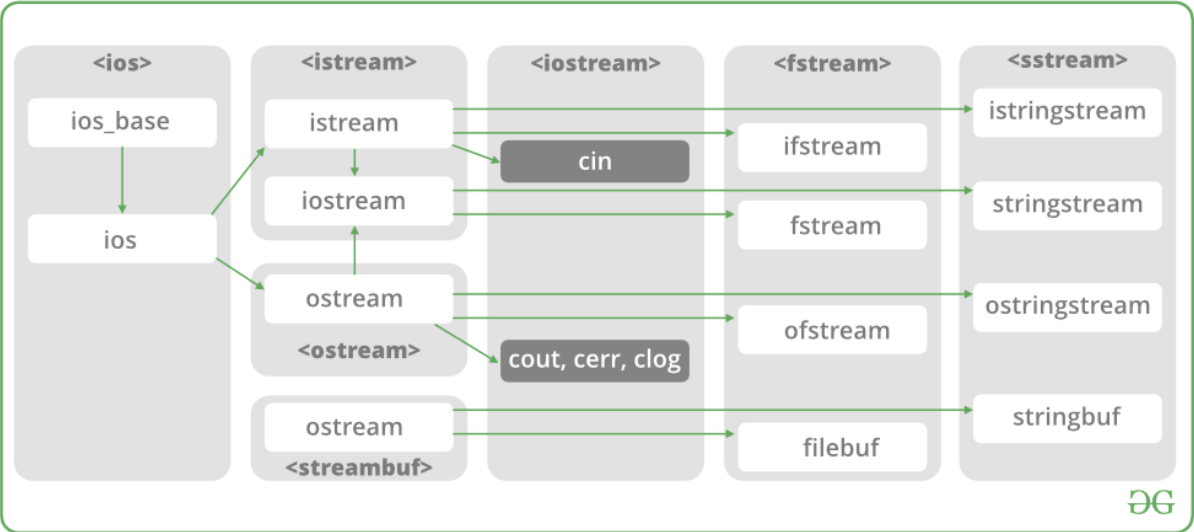
- Its purpose is to set the file buffers to read and write.
- We can also use file buffer member function to determine the length of the file.

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.



Now the first step to open the particular file for read or write operation. We can open file by

1. passing file name in constructor at the time of object creation
2. using the open method

For e.g.

Open File by using constructor

```
ifstream (const char* filename, ios_base::openmode mode = ios_base::in);  
ifstream fin(filename, openmode) by default openmode = ios::in  
ifstream fin("filename");
```

Open File by using open method

```
Calling of default constructor  
ifstream fin;  
fin.open(filename, openmode)  
fin.open("filename");
```

Modes :

Member Constant	Stands For	Access
in *	input	File open for reading: the internal stream buffer supports input operations.

Member Constant	Stands For	Access
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

Problem Statement : To read and write a File in C++.

Examples:

Input :

Welcome in GeeksforGeeks. Best way to learn things.

-1

Output :

Welcome in GeeksforGeeks. Best way to learn things.

Recommended: Please try your approach on **{IDE}** first, before moving on to the solution.

Below is the implementation by using **ifstream & ofstream classes**.

C++

```
/* File Handling with C++ using ifstream & ofstream class object*/
/* To write the Content in File*/
/* Then to read the content of file*/
#include <iostream>

/* fstream header file for ifstream, ofstream,
   fstream classes */
#include <fstream>

using namespace std;

// Driver Code
int main()
{
    // Creation of ofstream class object
    ofstream fout;

    string line;

    // by default ios::out mode, automatically deletes
    // the content of file. To append the content, open in ios::app
    // fout.open("sample.txt", ios::app)
    fout.open("sample.txt");

    // Execute a loop If file successfully opened
    while (fout) {

        // Read a Line from standard input
        getline(cin, line);

        // Press -1 to exit
        if (line == "-1")
            break;

        // Write line in file
        fout << line << endl;
    }

    // Close the File
    fout.close();

    // Creation of ifstream class object to read the file
    ifstream fin;

    // by default open mode = ios::in mode
    fin.open("sample.txt");

    // Execute a loop until EOF (End of File)
    while (getline(fin, line)) {

        // Print line (read from file) in Console
        cout << line << endl;
    }
}
```

```
    // Close the file
    fin.close();

    return 0;
}
```

Below is the implementation by using **fstream class**.

C++

```
/* File Handling with C++ using fstream class object */
/* To write the Content in File */
/* Then to read the content of file*/
#include <iostream>

/* fstream header file for ifstream, ofstream,
   fstream classes */
#include <fstream>

using namespace std;

// Driver Code
int main()
{
    // Creation of fstream class object
    fstream fio;

    string line;

    // by default openmode = ios::in|ios::out mode
    // Automatically overwrites the content of file, To append
    // the content, open in ios::app
    // fio.open("sample.txt", ios::in|ios::out|ios::app)
    // ios::trunc mode delete all content before open
    fio.open("sample.txt", ios::trunc | ios::out | ios::in);

    // Execute a loop If file successfully Opened
    while (fio) {

        // Read a Line from standard input
        getline(cin, line);

        // Press -1 to exit
        if (line == "-1")
            break;

        // Write line in file
        fio << line << endl;
    }

    // Execute a loop until EOF (End of File)
    // point read pointer at beginning of file
    fio.seekg(0, ios::beg);
```

```
while (fio) {  
  
    // Read a Line from File  
    getline(fio, line);  
  
    // Print line in Console  
    cout << line << endl;  
}  
  
// Close the file  
fio.close();  
  
return 0;  
}
```

C++

Q: write a single file handling program in c++ to reading and writing data on a file.

```
#include<iostream>  
#include<fstream>  
  
using namespace std;  
main()  
{  
    int rno,fee;  
    char name[50];  
  
    cout<<"Enter the Roll Number:";  
    cin>>rno;  
  
    cout<<"\nEnter the Name:";  
    cin>>name;  
  
    cout<<"\nEnter the Fee:";  
    cin>>fee;  
  
    ofstream fout("d:/student.doc");  
  
    fout<<rno<<"\t"<<name<<"\t"<<fee;    //write data to the file student  
  
    fout.close();  
  
    ifstream fin("d:/student.doc");  
  
    fin>>rno>>name>>fee;    //read data from the file student  
  
    fin.close();  
  
    cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;  
  
    return 0;  
}
```


C++

// Q: WA C++ file handling program to read data from the file called student.doc

```
#include<iostream>
#include<fstream>

using namespace std;

main()
{
    int rno,fee;
    char name[50];

    ifstream fin("d:/student.doc");

    fin>>rno>>name>>fee;    //read data from the file student

    fin.close();

    cout<<endl<<rno<<"\t"<<name<<"\t"<<fee;

    return 0;
}
```

193

Related Articles

1. Exception Handling using classes in C++
2. Set Position with seekg() in C++ File Handling
3. tellp() in file handling with c++ with example
4. How to work with file handling in C++
5. Error handling during file operations in C/C++
6. Contact Book in C++ using File Handling
7. ATM using file handling in C++

SALE!

✕

GeeksforGeeks Courses Upto 25% Off Enroll Now!

[Save 25% on Courses](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [T](#)

Read/Write Class Objects from/to File in C++

Difficulty Level : Medium • Last Updated : 24 Jan, 2023

[Read](#) [Discuss](#) [Courses](#) [Practice](#) [Video](#)

Given a file "Input.txt" in which every line has values same as instance variables of a class. Read the values into the class's object and do necessary operations.

Theory :

The data transfer is usually done using '>>' and '<<' operators. But if you have a class with 4 data members and want to write all 4 data members from its object directly to a file or vice-versa, we can do that using following syntax :

To write object's data members in a file :

```
// Here file_obj is an object of ofstream
file_obj.write((char *) & class_obj, sizeof(class_obj));
```

To read file's data members into an object :

```
// Here file_obj is an object of ifstream
file_obj.read((char *) & class_obj, sizeof(class_obj));
```

Examples:

Input :

Input.txt :

Michael 19 1806

Kemp 24 2114

Terry 21 2400

Operation : Print the name of the highest

rated programmer.

Output :

Terry

C++

```
// C++ program to demonstrate read/write of class
// objects in C++.
#include <iostream>
#include <fstream>
using namespace std;

// Class to define the properties
class Contestant {
public:
    // Instance variables
    string Name;
    int Age, Ratings;

    // Function declaration of input() to input info
    int input();

    // Function declaration of output_highest_rated() to
    // extract info from file Data Base
    int output_highest_rated();
};

// Function definition of input() to input info
int Contestant::input()
{
    // Object to write in file
    ofstream file_obj;

    // Opening file in append mode
    file_obj.open("Input.txt", ios::app);

    // Object of class contestant to input data in file
    Contestant obj;

    // Feeding appropriate data in variables
    string str = "Michael";
    int age = 18, ratings = 2500;

    // Assigning data into object
    obj.Name = str;
    obj.Age = age;
    obj.Ratings = ratings;

    // Writing the object's data in file
    file_obj.write((char*)&obj, sizeof(obj));

    // Feeding appropriate data in variables
```

```
    str = "Terry";
    age = 21;
    ratings = 3200;

    // Assigning data into object
    obj.Name = str;
    obj.Age = age;
    obj.Ratings = ratings;

    // Writing the object's data in file
    file_obj.write((char*)&obj, sizeof(obj));

    //close the file
    //It's always a good practice to close the file after opening them
    file_obj.close();

    return 0;
}

// Function definition of output_highest_rated() to
// extract info from file Data Base
int Contestant::output_highest_rated()
{
    // Object to read from file
    ifstream file_obj;

    // Opening file in input mode
    file_obj.open("Input.txt", ios::in);

    // Object of class contestant to input data in file
    Contestant obj;

    // Reading from file into object "obj"
    file_obj.read((char*)&obj, sizeof(obj));

    // max to store maximum ratings
    int max = 0;

    // Highest_rated stores the name of highest rated contestant
    string Highest_rated;

    // Checking till we have the feed
    while (!file_obj.eof()) {
        // Assigning max ratings
        if (obj.Ratings > max) {
            max = obj.Ratings;
            Highest_rated = obj.Name;
        }

        // Checking further
        file_obj.read((char*)&obj, sizeof(obj));
    }

    // close the file.
    //It's always a good practice to close the file after opening them
    file_obj.close();
}
```

```
// Output is the highest rated contestant
cout << Highest_rated;
return 0;
}

// Driver code
int main()
{
    // Creating object of the class
    Contestant object;

    // Inputting the data
    object.input();

    // Extracting the max rated contestant
    object.output_highest_rated();

    return 0;
}
```

Output:

Terry

This article is contributed by [Rohit Thapliyal](#). If you like GeeksforGeeks and would like to contribute, you can also write an article using [write.geeksforgeeks.org](https://www.geeksforgeeks.org/write-geeksforgeeks.org) or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

AD

Related Articles

1. How to make a C++ class whose objects can only be dynamically allocated?

SALE!

✕

GeeksforGeeks Courses Upto 25% Off Enroll Now!

[Save 25% on Courses](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [T](#)

C++ program to create a file

Difficulty Level : Medium • Last Updated : 28 Sep, 2018

[Read](#) [Discuss](#) [Courses](#) [Practice](#) [Video](#)

Problem Statement: Write a C++ program to create a file using file handling and check whether the file is created successfully or not. If a file is created successfully then it should print "File Created Successfully" otherwise should print some error message.

Approach: Declare a stream class file and open that text file in writing mode. If the file is not present then it creates a new text file. Now check if the file does not exist or not created then return false otherwise return true.

Below is the program to create a file:

```
// C++ implementation to create a file
#include <bits/stdc++.h>
using namespace std;

// Driver code
int main()
{
    // fstream is Stream class to both
    // read and write from/to files.
    // file is object of fstream class
    fstream file;

    // opening file "Gfg.txt"
    // in out(write) mode
    // ios::out Open for output operations.
    file.open("Gfg.txt",ios::out);

    // If no file is created, then
    // show the error message.
    if(!file)
    {
        cout<<"Error in creating file!!!";
        return 0;
    }
}
```

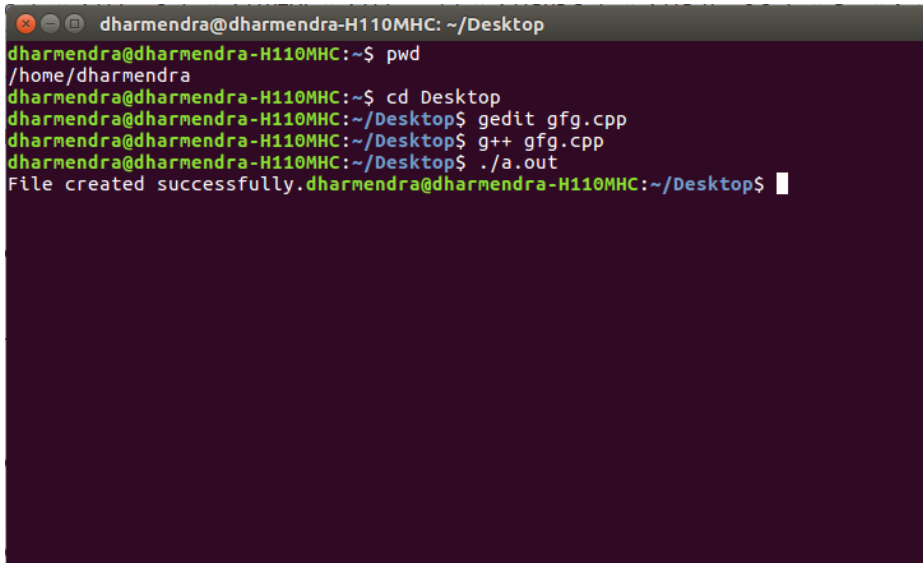
```
}

cout<<"File created successfully.";

// closing the file.
// The reason you need to call close()
// at the end of the loop is that trying
// to open a new file without closing the
// first file will fail.
file.close();

return 0;
}
```

Output:

A terminal window with a dark purple background. The title bar reads 'dharmendra@dharmendra-H110MHC: ~/Desktop'. The terminal shows the following commands and output:

```
dharmendra@dharmendra-H110MHC:~$ pwd
/home/dharmendra
dharmendra@dharmendra-H110MHC:~$ cd Desktop
dharmendra@dharmendra-H110MHC:~/Desktop$ gedit gfg.cpp
dharmendra@dharmendra-H110MHC:~/Desktop$ g++ gfg.cpp
dharmendra@dharmendra-H110MHC:~/Desktop$ ./a.out
File created successfully.dharmendra@dharmendra-H110MHC:~/Desktop$
```

AD

24

Related Articles

1. How to create Binary File from the existing Text File?
2. C program to copy contents of one file to another file
3. C++ Program to Copy One File into Another File

SALE!

✕

GeeksforGeeks Courses Upto 25% Off Enroll Now!

[Save 25% on Courses](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [T](#)

CSV file management using C++

Difficulty Level : Medium • Last Updated : 25 Jun, 2020

[Read](#) [Discuss](#) [Courses](#) [Practice](#) [Video](#)

CSV is a simple file format used to store tabular data such as a spreadsheet or a database. CSV stands for **Comma Separated Values**. The data fields in a CSV file are separated/delimited by a comma (',') and the individual rows are separated by a newline ('\n'). CSV File management in C++ is similar to text-type file management, except for a few modifications.

This article discusses about how to **create, update and delete records** in a CSV file:

Note: Here, a reportcard.csv file has been created to store the student's roll number, name and marks in math, physics, chemistry and biology.

1. Create operation:

The create operation is similar to creating a text file, i.e. input data from the user and write it to the csv file using the file pointer and appropriate delimiters(',') between different columns and '\n' after the end of each row.

AD

CREATE

```
void create()
{
    // file pointer
```



```

fstream fout;

// opens an existing csv file or creates a new file.
fout.open("reportcard.csv", ios::out | ios::app);

cout << "Enter the details of 5 students:"
      << " roll name maths phy chem bio";
<< endl;

int i, roll, phy, chem, math, bio;
string name;

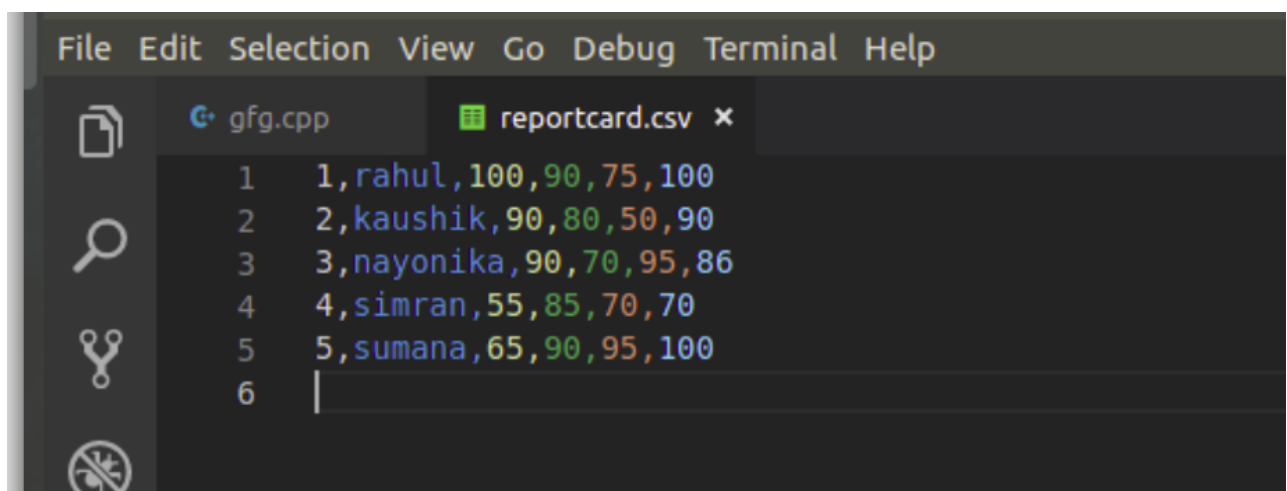
// Read the input
for (i = 0; i < 5; i++) {

    cin >> roll
        >> name
        >> math
        >> phy
        >> chem
        >> bio;

    // Insert the data to file
    fout << roll << ", "
          << name << ", "
          << math << ", "
          << phy << ", "
          << chem << ", "
          << bio
          << "\n";
}
}

```

Output:



2. Read a particular record:

In reading a CSV file, the following approach is implemented:-

1. Using `getline()`, file pointer and `'\n'` as the delimiter, read an entire row and store it in a string variable.

2. Using stringstream, separate the row into words.
3. Now using getline(), the stringstream pointer and ',' as the delimiter, read every word in the row, store it in a string variable and push that variable to a string vector.
4. Retrieve a required column data through row[index]. Here, row[0] always stores the roll number of a student, so compare row[0] with the roll number input by the user, and if it matches, display the details of the student and break from the loop.

Note: Here, since whatever data reading from the file, is stored in string format, so always convert string to the required datatype before comparing or calculating, etc.

READ

```
void read_record()
{
    // File pointer
    fstream fin;

    // Open an existing file
    fin.open("reportcard.csv", ios::in);

    // Get the roll number
    // of which the data is required
    int rollnum, roll2, count = 0;
    cout << "Enter the roll number "
         << "of the student to display details: ";
    cin >> rollnum;

    // Read the Data from the file
    // as String Vector
    vector<string> row;
    string line, word, temp;

    while (fin >> temp) {

        row.clear();

        // read an entire row and
        // store it in a string variable 'line'
        getline(fin, line);

        // used for breaking words
        stringstream s(line);

        // read every column data of a row and
        // store it in a string variable, 'word'
        while (getline(s, word, ',')) {

            // add all the column data
            // of a row to a vector
            row.push_back(word);
        }
    }
}
```

```

// convert string to integer for comparison
roll2 = stoi(row[0]);

// Compare the roll number
if (roll2 == rollnum) {

    // Print the found data
    count = 1;
    cout << "Details of Roll " << row[0] << " : \n";
    cout << "Name: " << row[1] << "\n";
    cout << "Maths: " << row[2] << "\n";
    cout << "Physics: " << row[3] << "\n";
    cout << "Chemistry: " << row[4] << "\n";
    cout << "Biology: " << row[5] << "\n";
    break;
}
}
if (count == 0)
    cout << "Record not found\n";
}

```

Output:

```

OUTPUT  PROBLEMS  DEBUG CONSOLE  TERMINAL

nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$ ./a.out
Enter the roll number of the student to display details: 2
Details of Roll 2 :
Name: kaushik
Maths: 90
Physics: 80
Chemistry: 50
Biology: 90
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$ ./a.out
Enter the roll number of the student to display details: 8
Record not found
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$

```

3. Update a record:

The following approach is implemented while updating a record:-

1. Read data from a file and compare it with the user input, as explained under read operation.
2. Ask the user to enter new values for the record to be updated.
3. update row[index] with the new data. Here, index refers to the required column field that is to be updated.
4. Write the updated record and all other records into a new file('reportcardnew.csv').
5. At the end of operation, remove the old file and rename the new file, with the old file name, i.e. remove 'reportcard.csv' and rename 'reportcardnew.csv' with 'reportcard.csv'

UPDATE

```
void update_recode()
{
    // File pointer
    fstream fin, fout;

    // Open an existing record
    fin.open("reportcard.csv", ios::in);

    // Create a new file to store updated data
    fout.open("reportcardnew.csv", ios::out);

    int rollnum, roll1, marks, count = 0, i;
    char sub;
    int index, new_marks;
    string line, word;
    vector<string> row;

    // Get the roll number from the user
    cout << "Enter the roll number "
         << "of the record to be updated: ";
    cin >> rollnum;

    // Get the data to be updated
    cout << "Enter the subject "
         << "to be updated(M/P/C/B): ";
    cin >> sub;

    // Determine the index of the subject
    // where Maths has index 2,
    // Physics has index 3, and so on
    if (sub == 'm' || sub == 'M')
        index = 2;
    else if (sub == 'p' || sub == 'P')
        index = 3;
    else if (sub == 'c' || sub == 'C')
        index = 4;
    else if (sub == 'b' || sub == 'B')
        index = 5;
    else {
        cout << "Wrong choice.Enter again\n";
        update_record();
    }

    // Get the new marks
    cout << "Enter new marks: ";
    cin >> new_marks;

    // Traverse the file
    while (!fin.eof()) {

        row.clear();
```

```
getline(fin, line);
stringstream s(line);

while (getline(s, word, ',')) {
    row.push_back(word);
}

roll1 = stoi(row[0]);
int row_size = row.size();

if (roll1 == rollnum) {
    count = 1;
    stringstream convert;

    // sending a number as a stream into output string
    convert << new_marks;

    // the str() converts number into string
    row[index] = convert.str();

    if (!fin.eof()) {
        for (i = 0; i < row_size - 1; i++) {

            // write the updated data
            // into a new file 'reportcardnew.csv'
            // using fout
            fout << row[i] << ", ";
        }

        fout << row[row_size - 1] << "\n";
    }
}
else {
    if (!fin.eof()) {
        for (i = 0; i < row_size - 1; i++) {

            // writing other existing records
            // into the new file using fout.
            fout << row[i] << ", ";
        }

        // the last column data ends with a '\n'
        fout << row[row_size - 1] << "\n";
    }
}
if (fin.eof())
    break;
}

if (count == 0)
    cout << "Record not found\n";

fin.close();
fout.close();

// removing the existing file
remove("reportcard.csv");
```

```

// renaming the updated file with the existing file name
rename("reportcardnew.csv", "reportcard.csv");
}

```

Output:

```

dit Selection View Go Debug Terminal Help
gfg.cpp reportcard.csv x
1 1,rahul,100,90,75,100
2 2,kaushik,90,80,50,90
3 3,nayonika,90,78,95,86
4 4,simran,55,85,70,70
5 5,sumana,65,90,95,100
6 6,shiv,78,75,84,85
7
OUTPUT PROBLEMS DEBUG CONSOLE TERMINAL
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$ ./a.out
Enter the roll number of the record to be updated: 3
Enter the subject to be updated(M/P/C/B): p
Enter new marks: 78
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$

```

4. Delete a record:

The following approach is implemented while deleting a record

1. Read data from a file and compare it with the user input, as explained under read and update operation.
2. Write all the updated records, except the data to be deleted, onto a new file(reportcardnew.csv).
3. Remove the old file, and rename the new file, with the old file's name.

DELETE

```

void delete_record()
{
    // Open File pointers
    fstream fin, fout;

    // Open the existing file
    fin.open("reportcard.csv", ios::in);

    // Create a new file to store the non-deleted data
    fout.open("reportcardnew.csv", ios::out);

    int rollnum, roll1, marks, count = 0, i;
    char sub;
    int index, new_marks;
    string line, word;
    vector<string> row;

    // Get the roll number
    // to decide the data to be deleted

```

```

cout << "Enter the roll number "
      << "of the record to be deleted: ";
cin >> rollnum;

// Check if this record exists
// If exists, leave it and
// add all other data to the new file
while (!fin.eof()) {

    row.clear();
    getline(fin, line);
    stringstream s(line);

    while (getline(s, word, ',')) {
        row.push_back(word);
    }

    int row_size = row.size();
    roll1 = stoi(row[0]);

    // writing all records,
    // except the record to be deleted,
    // into the new file 'reportcardnew.csv'
    // using fout pointer
    if (roll1 != rollnum) {
        if (!fin.eof()) {
            for (i = 0; i < row_size - 1; i++) {
                fout << row[i] << ", ";
            }
            fout << row[row_size - 1] << "\n";
        }
    }
    else {
        count = 1;
    }
    if (fin.eof())
        break;
}
if (count == 1)
    cout << "Record deleted\n";
else
    cout << "Record not found\n";

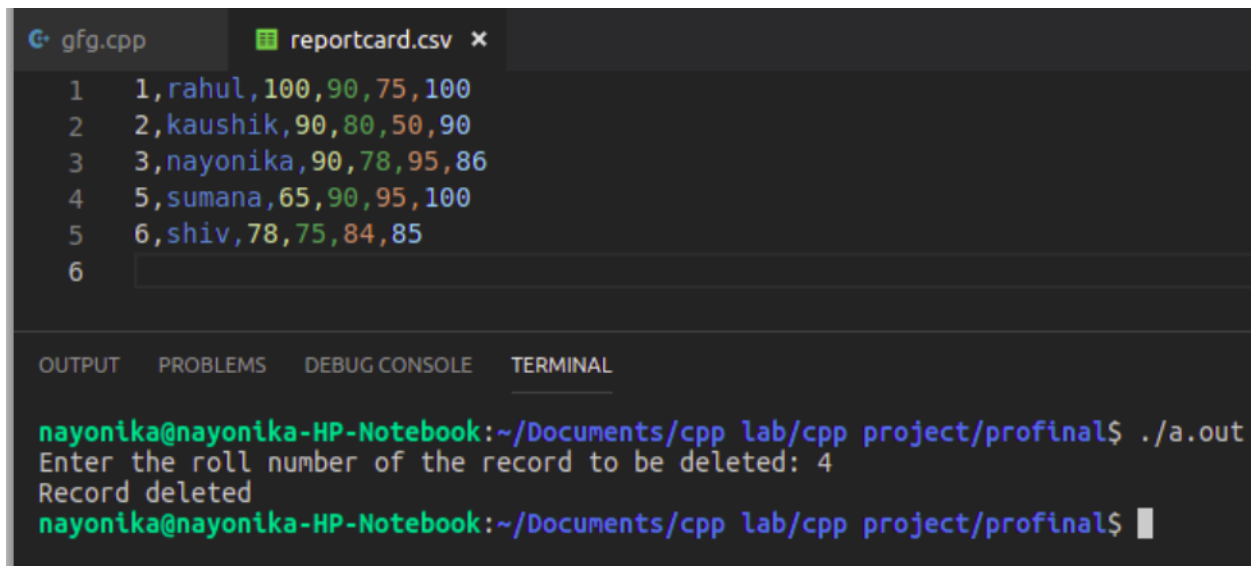
// Close the pointers
fin.close();
fout.close();

// removing the existing file
remove("reportcard.csv");

// renaming the new file with the existing file name
rename("reportcardnew.csv", "reportcard.csv");
}

```

Output:



The screenshot shows a C++ IDE with two tabs: `gfg.cpp` and `reportcard.csv`. The `reportcard.csv` file contains the following data:

Roll Number	Name	Maths	Science	English	History
1	rahul	100	90	75	100
2	kaushik	90	80	50	90
3	nayonika	90	78	95	86
4	sumana	65	90	95	100
5	shiv	78	75	84	85

The terminal output shows the execution of the program:

```
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$ ./a.out
Enter the roll number of the record to be deleted: 4
Record deleted
nayonika@nayonika-HP-Notebook:~/Documents/cpp lab/cpp project/profinal$
```

References: <https://www.geeksforgeeks.org/file-handling-c-classes/>,
<https://www.geeksforgeeks.org/stringstream-c-applications/>

30

Related Articles

1. Bookshop management system using file handling
2. C program to copy contents of one file to another file
3. How to create Binary File from the existing Text File?
4. C++ Program to Copy One File into Another File
5. C++ Program to Read Content From One File and Write it Into Another File
6. C++ Program to Copy the Contents of One File Into Another File
7. Difference Between C++ Text File and Binary File
8. ATM Management System using C++
9. Student record management system using linked list
10. Department Store Management System(DSMS) using C++

[Previous](#)

[Next](#)

SALE!

✕

GeeksforGeeks Courses Upto 25% Off Enroll Now!

[Save 25% on Courses](#) [DSA](#) [Data Structures](#) [Algorithms](#) [Interview Preparation](#) [Data Science](#) [T](#)

Four File Handling Hacks which every C/C++ Programmer should know

Difficulty Level : Medium • Last Updated : 15 Mar, 2023

[Read](#) [Discuss](#) [Courses](#) [Practice](#) [Video](#)

We will discuss about four file hacks listed as below-

1. Rename – Rename a file using C/C++
2. Remove – Remove a file using C/C++
3. File Size – Get the file size using C/C++
4. Check existence – Check whether a file exists or not in C/C++

C++

```
// C++ Program to demonstrate the
// four file hacks every C/C++ must know

// Note that we are assuming that the files
// are present in the same file as the program
// before doing the below four hacks

#include <iostream>
#include <stdlib.h>
using namespace std;

// A Function to get the file size
unsigned long long int fileSize(const char *filename)
{
    // Open the file
    FILE *fh = fopen(filename, "rb");
    fseek(fh, 0, SEEK_END);
    unsigned long long int size = ftell(fh);
    fclose(fh);

    return size;
}
```

```
// A Function to check if the file exists or not
bool fileExists(const char * fname)
{
    FILE *file;
    if (file = fopen(fname, "r"))
    {
        fclose(file);
        return true;
    }
    return false;
}

// Driver Program to test above functions
int main()
{
    cout << fileSize("Passwords.txt") << " Bytes\n";
    cout << fileSize("Notes.docx") << " Bytes\n";

    if (fileExists("OldData.txt") == true )
        cout << "The File exists\n";
    else
        cout << "The File doesn't exist\n";

    rename("Videos", "English_Videos");
    rename("Songs", "English_Songs");

    remove("OldData.txt");
    remove("Notes.docx");

    if (fileExists("OldData.txt") == true )
        cout << "The File exists\n";
    else
        cout << "The File doesn't exist\n";

    return 0;
}

// This code is contributed by sarajadhav12052009
```

C

```
// A C Program to demonstrate the
// four file hacks every C/C++ must know

// Note that we are assuming that the files
// are present in the same file as the program
// before doing the below four hacks
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>

// A Function to get the file size
unsigned long long int fileSize(const char *filename)
{
```

```
// Open the file
FILE *fh = fopen(filename, "rb");
fseek(fh, 0, SEEK_END);
unsigned long long int size = ftell(fh);
fclose(fh);

return (size);
}

// A Function to check if the file exists or not
bool fileExists(const char * fname)
{
    FILE *file;
    if (file = fopen(fname, "r"))
    {
        fclose(file);
        return (true);
    }
    return (false);
}

// Driver Program to test above functions
int main()
{
    printf("%llu Bytes\n", fileSize("Passwords.txt"));
    printf("%llu Bytes\n", fileSize("Notes.docx"));

    if (fileExists("OldData.txt") == true )
        printf("The File exists\n");
    else
        printf("The File doesn't exist\n");

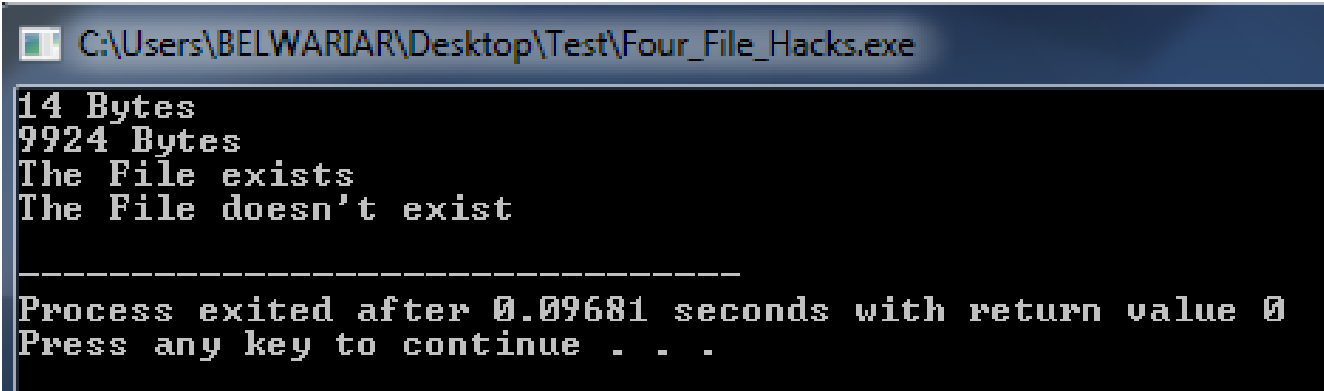
    rename("Videos", "English_Videos");
    rename("Songs", "English_Songs");

    remove("OldData.txt");
    remove("Notes.docx");

    if (fileExists("OldData.txt") == true )
        printf("The File exists\n");
    else
        printf("The File doesn't exist\n");

    return 0;
}
```

Output:








Screenshot before executing the program :

AD

Name	Date modified	Type	Size
PythonPrograms	19-06-2016 08:09	File folder	
Songs	19-06-2016 08:07	File folder	
Videos	19-06-2016 08:07	File folder	
Four_File_Hacks	19-06-2016 08:19	C++ Source File	2 KB
Notes	19-06-2016 08:19	Microsoft Office ...	10 KB
OldData	19-06-2016 08:08	Notepad++ Docu...	1 KB
Passwords	19-06-2016 08:10	Notepad++ Docu...	1 KB

Screenshot after executing the program :

Name	Date modified	Type	Size
 English_Songs	19-06-2016 08:07	File folder	
 English_Videos	19-06-2016 08:07	File folder	
 PythonPrograms	19-06-2016 08:09	File folder	
 Four_File_Hacks	19-06-2016 08:33	C++ Source File	2 KB
 Passwords	19-06-2016 08:10	Notepad++ Docu...	1 KB

This article is contributed by **Rachit Belwariar**. If you like GeeksforGeeks and would like to contribute, you can also write an article and mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

14

Related Articles

1. Bitwise Hacks for Competitive Programming
2. 10 C++ Programming Tricks That You Should Know
3. C | File Handling | Question 1
4. C | File Handling | Question 2
5. C | File Handling | Question 3
6. C | File Handling | Question 4
7. C | File Handling | Question 5
8. Set Position with seekg() in C++ File Handling