

Code Explanation and Guidance



i have viva and practical exam
if i provide you my code then can you help me by explaining the code line by line,also what is use of the code and how it works
and if possible can you give video to understand the concept in code

Yes, I can absolutely help you with that! You can share your code with me, and I will explain it line by line, detailing the logic behind it and how it works. I'll also provide an explanation of the purpose of the code and how it fits into the larger concept you're learning.

Regarding video resources, I can recommend videos or provide links to YouTube or other platforms that explain the concepts, but I can't create videos myself. However, I can guide you to the right sources for learning. Just send over your code, and we'll get started!



```
#include <cstring>
#include <fstream>
#include <iostream>
using namespace std;

typedef struct student {
    int rollNo;
    char name[50];
    char div;
    char address[100];
} student;

class studentDatabase {
    string fileName = "student_data.dat";

public:
    studentDatabase() {
        fstream fileObj(fileName);
        if (fileObj.fail()) {
            fileObj.open(fileName, ios::out);
            cout << "New file created." << endl;
        } else {
            cout << "File already exists." << endl;
        }
        fileObj.close();
    }

    void addStudent();
```

```

void searchStudent();
void deleteStudent();
void displayAll();
};

void studentDatabase::searchStudent() {
    int roll;
    student s;
    bool status = false;

    cout << "Roll number to search:\t";
    cin >> roll;

    ifstream readfile;
    readfile.open(fileName, ios::in | ios::binary);

    while (readfile.read((char*)&s, sizeof(student))) {
        if (s.rollNo == roll) {
            status = true;
            break;
        }
    }

    readfile.close();

    if (status) {
        cout << "--- RECORD FOUND ---" << endl;
        cout << "Roll number:\t" << s.rollNo << endl;
        cout << "Name:\t" << s.name << endl;
        cout << "Division:\t" << s.div << endl;
        cout << "Address:\t" << s.address << endl;
        cout << "--- END OF RECORD ---" << endl;
    } else {
        cout << "Record not found." << endl;
    }
}

void studentDatabase::deleteStudent() {
    int roll;
    student s;
    bool status = false;

    cout << "Roll number to delete:\t";
    cin >> roll;

    ifstream readfile;
    readfile.open(fileName, ios::in | ios::binary);
    ofstream writefile;
    writefile.open("~" + fileName, ios::out | ios::binary);
    writefile.clear();

```

```

while (readFile.read((char*)&s, sizeof(student))) {
    if (s.rollNo == roll) {
        status = true;
    } else {
        writeFile.write((char*)&s, sizeof(student)) << flush;
    }
}
readFile.close();
writeFile.close();

if (status) {
    readFile.open("~" + fileName, ios::in | ios::binary);
    writeFile.open(fileName, ios::out | ios::binary);
    writeFile.clear();

    writeFile << readFile.rdbuf();
    readFile.close();
    writeFile.close();

    cout << "Record deleted." << endl;
} else {
    cout << "Record not found." << endl;
}
}

void studentDatabase::addStudent() {
    student s;
    cout << "Roll number:\t";
    cin >> s.rollNo;
    cout << "Name:\t";
    cin.ignore();
    cin.getline(s.name, 50);
    cout << "Division:\t";
    cin >> s.div;
    cout << "Address:\t";
    cin.ignore();
    cin.getline(s.address, 100);
    ofstream file(fileName, ios::out | ios::binary | ios::app);
    file.write((char*)&s, sizeof(student)) << flush;
    if (file.fail()) {
        cout << "Failed to add student record." << endl;
    } else {
        cout << "Student added successfully." << endl;
    }
    file.close();
}

void studentDatabase::displayAll() {
    ifstream file;
    student s;
    int count = 0;

```

```

file.open(fileName, ios::in | ios::binary);
while (file.read((char*)&s, sizeof(student))) {
    count++;
    cout << count << " ";
    cout << s.rollNo << " | ";
    cout << s.name << " | ";
    cout << s.div << " | ";
    cout << s.address << endl;
}
if (count == 0) {
    cout << "No records found." << endl;
}
file.close();
}

int main() {
    int ch;
    studentDatabase db;s

    do {
        cout << endl;
        cout << "--- MAIN MENU ---" << endl;
        cout << "1 -> Add record" << endl;
        cout << "2 -> Delete record" << endl;
        cout << "3 -> Search record" << endl;
        cout << "4 -> Display all records" << endl;
        cout << "0 -> Exit" << endl << endl;
        cout << "Choose an option (0-4):\t";
        cin >> ch;
        switch (ch) {
            case 0:
                cout << "\n\n// END OF CODE\n\n" << endl;
                break;
            case 1:
                db.addStudent();
                break;
            case 2:
                db.deleteStudent();
                break;
            case 3:
                db.searchStudent();
                break;
            case 4:
                cout << "All records are:\t" << endl;
                db.displayAll();
                break;

            default:
                cout << "Please choose a valid option (0-4):\t" << endl;
                break;
        }
    } while (ch != 0);
}

```

```
return 0;
}
```

Let's go through your code step-by-step.

Header Files

```
cpp

#include <cstring>
#include <fstream>
#include <iostream>
using namespace std;
```

- `#include <cstring>`: This header allows the use of C-string functions like `strcpy` and `strlen`.
- `#include <fstream>`: This is for file handling (reading/writing files).
- `#include <iostream>`: This is for standard input/output functions.

Structure Definition (`student`)

```
cpp

typedef struct student {
    int rollNo;
    char name[50];
    char div;
    char address[100];
} student;
```

This defines a `student` structure, which will store the details of a student:

- `rollNo`: An integer for the roll number.
- `name`: A character array (C-string) to store the student's name.
- `div`: A character to store the division.
- `address`: A character array (C-string) to store the student's address.

Class Definition (`studentDatabase`)

```
cpp

class studentDatabase {
    string fileName = "student_data.dat";
```

This is a class named `studentDatabase`, and it has a private member `fileName`, which stores the filename where student data will be saved. By default, the file is `"student_data.dat"`.

Constructor: `studentDatabase()`

cpp

```
studentDatabase() {
    fstream fileObj(fileName);
    if (fileObj.fail()) {
        fileObj.open(fileName, ios::out);
        cout << "New file created." << endl;
    } else {
        cout << "File already exists." << endl;
    }
    fileObj.close();
}
```

- This constructor opens the file `student_data.dat`.
- If the file doesn't exist, it creates a new file and prints `"New file created."`.
- If the file exists, it prints `"File already exists."`.
- Finally, it closes the file.

Member Functions

`addStudent()`

cpp

```
void studentDatabase::addStudent() {
    student s;
    cout << "Roll number:\t";
    cin >> s.rollNo;
    cout << "Name:\t";
    cin.ignore();
    cin.getline(s.name, 50);
    cout << "Division:\t";
    cin >> s.div;
    cout << "Address:\t";
    cin.ignore();
    cin.getline(s.address, 100);
    ofstream file(fileName, ios::out | ios::binary | ios::app);
    file.write((char*)&s, sizeof(student)) << flush;
    if (file.fail()) {
        cout << "Failed to add student record." << endl;
    } else {
        cout << "Student added successfully." << endl;
    }
    file.close();
}
```

- This function allows adding a new student to the database.
- The function prompts the user for the student's details (roll number, name, division, and address).
- It then opens the file in append binary mode (`ios::app`), and writes the student record into the file.
- If writing fails, an error message is shown. Otherwise, a success message is shown.

`searchStudent()`

cpp

```
void studentDatabase::searchStudent() {
    int roll;
    student s;
    bool status = false;

    cout << "Roll number to search:\t";
    cin >> roll;

    ifstream readFile;
    readFile.open(fileName, ios::in | ios::binary);

    while (readFile.read((char*)&s, sizeof(student))) {
        if (s.rollNo == roll) {
            status = true;
            break;
        }
    }

    readFile.close();

    if (status) {
        cout << "--- RECORD FOUND ---" << endl;
        cout << "Roll number:\t" << s.rollNo << endl;
        cout << "Name:\t" << s.name << endl;
        cout << "Division:\t" << s.div << endl;
        cout << "Address:\t" << s.address << endl;
        cout << "--- END OF RECORD ---" << endl;
    } else {
        cout << "Record not found." << endl;
    }
}
```

- This function searches for a student record based on the roll number.
- It opens the file in binary read mode and reads the records one by one.
- If a matching roll number is found, it prints the student's details.
- If no record is found, it notifies the user.

`deleteStudent()``

cpp

```
void studentDatabase::deleteStudent(){
    int roll;
    student s;
    bool status = false;

    cout << "Roll number to delete:\t";
    cin >> roll;

    ifstream readFile;
    readFile.open(fileName, ios::in | ios::binary);
    ofstream writeFile;
    writeFile.open("~" + fileName, ios::out | ios::binary);
    writeFile.clear();

    while (readFile.read((char*)&s, sizeof(student))) {
        if (s.rollNo == roll) {
            status = true;
        } else {
            writeFile.write((char*)&s, sizeof(student)) << flush;
        }
    }
    readFile.close();
    writeFile.close();
}
```

```

if (status) {
    readFile.open("~" + fileName, ios::in | ios::binary);
    writeFile.open(fileName, ios::out | ios::binary);
    writeFile.clear();

    writeFile << readFile.rdbuf();
    readFile.close();
    writeFile.close();

    cout << "Record deleted." << endl;
} else {
    cout << "Record not found." << endl;
}
}

```

- This function deletes a student record based on the roll number.
- It reads the file and writes all records except the one to be deleted into a new temporary file (~student_data.dat).
- After deleting, it copies the data from the temporary file back to the original file, thus removing the student record.

`displayAll()`

```

cpp

void studentDatabase::displayAll() {
    ifstream file;
    student s;
    int count = 0;
    file.open(fileName, ios::in | ios::binary);
    while (file.read((char*)&s, sizeof(student))) {
        count++;
        cout << count << " ";
        cout << s.rollNo << "|";
        cout << s.name << "|";
        cout << s.div << "|";
        cout << s.address << endl;
    }
    if (count == 0) {
        cout << "No records found." << endl;
    }
    file.close();
}

```

- This function displays all the student records stored in the file.
- It reads each record and prints the student's roll number, name, division, and address.
- If no records are found, it displays a message saying so.

Main Function

```

cpp

int main() {
    int ch;
    studentDatabase db;

    do {
        cout << endl;
        cout << "--- MAIN MENU ---" << endl;
        cout << "1 -> Add record" << endl;
    }
}

```



```

cout << "2 -> Delete record" << endl;
cout << "3 -> Search record" << endl;
cout << "4 -> Display all records" << endl;
cout << "0 -> Exit" << endl << endl;
cout << "Choose an option (0-4):\t";
cin >> ch;
switch (ch) {
    case 0:
        cout << "\n\n// END OF CODE\n\n" << endl;
        break;
    case 1:
        db.addStudent();
        break;
    case 2:
        db.deleteStudent();
        break;
    case 3:
        db.searchStudent();
        break;
    case 4:
        cout << "All records are:\t" << endl;
        db.displayAll();
        break;

    default:
        cout << "Please choose a valid option (0-4):\t" << endl;
        break;
}
} while (ch != 0);

return 0;
}

```

- The `main()` function displays a menu for the user to choose actions like adding, deleting, searching, or displaying student records.
- It continuously shows the menu until the user chooses to exit (option `0`).

Summary of Functionality:

- The code is a simple **student database management system** that allows you to add, delete, search, and display student records using a binary file.
- It uses file I/O operations (reading, writing, appending) and structures to manage student data.

For videos, you can find good tutorials on file handling and C++ structures on platforms like YouTube. Look for videos specifically on "C++ file handling" or "C++ structure-based data management."

Let me know if you need anything else!

Do you like this personality?