Code Explanation Help

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

Absolutely! I'd be happy to help you with that.

If you provide me with your code, I can go through it line by line, explain its purpose, how it works, and the underlying concepts. I can also clarify any concepts you might be unclear about.

As for videos, I can't generate videos directly, but I can certainly recommend resources (like YouTube tutorials or free course links) that cover the concepts in your code.

Feel free to share your code when you're ready!

```
#include <string.h>
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;

typedef struct employee {
    int empId;
    char name[50];
    char designation[50];
    int salary;
} employee;

typedef struct index_pair {
    int key;
    int position;
} index_pair;

class employeeDatabase {
    string data_file_name = "ind_employee_data.dat";
    string index_file_name = "ind_employee_index.dat";
   public:
    employeeDatabase();
    void addEmployee(int eid, char name[50], char dest[50], int sal);
    void searchEmployee(int);
    void deleteEmployee(int);
    bool isPresent(int);
    void display_all();
    employee readEmp(int pos);
};

employee employeeDatabase::readEmp(int pos) {
    fstream data_file(data_file_name, ios::binary | ios::in | ios::ate);
    employee emp;
    if (pos >= data_file.tellg() || pos == -1) {
        emp.empId = -1;
        return emp;
    }
    data_file.seekg(pos, ios::beg);
    data_file.read((char *)&emp, sizeof(emp));
    data_file.close();
    return emp;
}

employeeDatabase::employeeDatabase() {
    fstream data_file(data_file_name);
    fstream index_file(index_file_name);
    if (data_file.fail() | index_file.fail()) {
        cout << "Created a new file." << endl;
        data_file.open(data_file_name, ios::binary | ios::out | ios::app);
        index_file.open(index_file_name, ios::binary | ios::out | ios::app);
    } else {
        cout << "File already exists." << endl;
    }
    data_file.close();
    index_file.close();
}

void employeeDatabase::addEmployee(int eid, char name[50], char dest[50], int sal) {
    fstream data_file(data_file_name, ios::binary | ios::out | ios::app);
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current, temp;
    char *all_data;
    employee emp;
    int pos, ipos;
    bool update = false;

    while (index_file.read((char *)&current, sizeof(index_pair))) {

        if (current.key == eid) {
            if (current.position == -1) {
                ipos = (int)index_file.tellg() - sizeof(current);
                update = true;
                break;
            } else {
                cout << "Cannot add employee, already exists."
                     << endl;
                goto exit_addEmployee;
            }
        }
    }
    index_file.close();

    emp.empId = eid;
    strcpy(emp.name, name);
    strcpy(emp.designation, dest);
    emp.salary = sal;

    data_file.seekp(0, ios::end);
    pos = data_file.tellp();
    data_file.write((char *)&emp, sizeof(emp));
    current.key = eid;
    current.position = pos;
    if (update) {
        index_file.open(index_file_name, ios::binary | ios::out);
        index_file.seekp(ipos, ios::beg);
        index_file.write((char *)&current, sizeof(current));
    } else {
        bool written = false;
        vector<index_pair> index_pairs;
        index_file.open(index_file_name, ios::binary | ios::in);
        while (index_file.read((char *)&temp, sizeof(index_pair))) {
            if (!written and temp.key > eid) {
                written = true;
                index_pairs.push_back(current);
            }
            index_pairs.push_back(temp);
        }
        if (!written) {
            index_pairs.push_back(current);
        }

        index_file.clear();
        index_file.close();

        index_file.open(index_file_name, ios::binary | ios::out);

        for (int i = 0; i < index_pairs.size(); i++) {
            current = index_pairs[i];
            if (current.position != -1) {
                index_file.write((char *)&current, sizeof(index_pair));
            }
        }
    }
    cout << "Employee added successfully." << endl;
exit_addEmployee:
    data_file.close();
    index_file.close();
}
```

```
void display_emp(employee emp) {
    cout << "ID:\t" << emp.empId << endl;
    cout << "Name:\t" << emp.name << endl;
    cout << "Designation:\t" << emp.designation << endl;
    cout << "Salary:\tRs. " << emp.salary << endl;
}

void employeeDatabase::searchEmployee(int eid) {
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current;
    int pos = -1;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            pos = current.position;
            break;
        }
    }

    employee emp = readEmp(pos);
    if (emp.empId == -1) {
        cout << "Employee does not exist." << endl;
    } else {
        cout << "--- EMPLOYEE FOUND ---" << endl;
        display_emp(emp);
        cout << "--- END OF RECORD ---" << endl;
    }

    index_file.close();
}

bool employeeDatabase::isPresent(int eid) {
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            if (current.position == -1) {
                index_file.close();
                return false;
            } else {
                index_file.close();
                return true;
            }
        }
    }
    index_file.close();
    return false;
}
```

```
void employeeDatabase::deleteEmployee(int eid) {
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current;
    bool deleted = false;
    vector<index_pair> pairs;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            deleted = true;
        } else {
            pairs.push_back(current);
        }
    }
    index_file.close();
    if (deleted) {
        index_file.open(index_file_name, ios::binary | ios::out);
        index_file.clear();
        index_file.seekp(0, ios::beg);

        for (int i = 0; i < pairs.size(); i++) {
            current = pairs[i];
            index_file.write((char *)&current, sizeof(index_pair));
        }
        index_file.close();
        cout << "Employee removed from record." << endl;
    } else {
        cout << "Employee does not exist in the record." << endl;
    }
}

void employeeDatabase::display_all() {
    fstream index_file(index_file_name, ios::binary | ios::in);
    fstream data_file(data_file_name, ios::binary | ios::in);
    index_pair current;
    employee emp;
    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.position != -1) {
            data_file.seekg(current.position, ios::beg);
            data_file.read((char *)&emp, sizeof(emp));
            cout << emp.empId << " | " << emp.name << " | " << emp.designation
                 << " | " << emp.salary << endl;
        }
    }
    index_file.close();
    data_file.close();
}

int main() {
    employeeDatabase db;
    int eid, sal;
    char name[50], dest[50];
```

```
int ch;
do {
    cout << endl << "--- MAIN MENU ---" << endl;
    cout << "1 -> Add employee" << endl;
    cout << "2 -> Search employee" << endl;
    cout << "3 -> Delete employee" << endl;
    cout << "4 -> Display all" << endl;
    cout << "5 -> Exit" << endl;
    cout << "Choose an option (1-5):\t";
    cin >> ch;
    switch (ch) {
        case 1:
        reenter_eid:
            cout << "Employee ID:\t";
            cin >> eid;
            if (db.isPresent(eid)) {
                cout << "Employee already exists in record. Please try again."
                     << endl;
                goto reenter_eid;
            }
            cout << "Name:\t";
            cin >> name;
            cout << "Designation:\t";
            cin >> dest;
            cout << "Salary:\tRs. ";
            cin >> sal;

            db.addEmployee(eid, name, dest, sal);
            break;
        case 2:
            cout << "Employee ID:\t";
            cin >> eid;
            db.searchEmployee(eid);
            break;
        case 3:
            cout << "Employee ID:\t";
            cin >> eid;
            db.deleteEmployee(eid);
            break;
        case 4:
            db.display_all();
            break;
        case 5:
            cout << "\n\n// END OF CODE\n\n" << endl;
            break;
        default:
            cout << "Please choose a valid option (1-5)." << endl;
            break;
    }
} while (ch != 5);
```

```
    return 0;
}
```

Let's go through this C++ code, which implements an employee database system using file handling and indexing. The program allows you to add, search, delete, and display employee records from binary files. Here's the breakdown:

1. Includes and Type Definitions

```cpp
#include <string.h>
#include <fstream>
#include <iostream>
#include <vector>
using namespace std;

typedef struct employee {
    int empId;
    char name[50];
    char designation[50];
    int salary;
} employee;

typedef struct index_pair {
    int key;
    int position;
} index_pair;
```

- **`#include` statements**: These include necessary libraries:
  - `string.h`: For string manipulation.
  - `fstream`: For file operations.
  - `iostream`: For input-output.
  - `vector`: For using dynamic arrays.
- `typedef struct`: Defines two structures:
  - `employee`: Holds the data of an employee (ID, name, designation, salary).
  - `index_pair`: Stores an index pair for each employee (ID and the position where the employee's data is located in the data file).

2. Class Definition

```cpp
class employeeDatabase {
    string data_file_name = "ind_employee_data.dat";
    string index_file_name = "ind_employee_index.dat";
public:
    employeeDatabase();
    void addEmployee(int eid, char name[50], char dest[50], int sal);
    void searchEmployee(int);
    void deleteEmployee(int);
    bool isPresent(int);
    void display_all();
    employee readEmp(int pos);
};
```

- `` `employeeDatabase` `` **class**: This encapsulates the functions to manage employee data.
    - **Data and index file names**: `` `data_file_name` `` holds the employee data, and `` `index_file_name` `` holds the indexing information.
    - **Public methods**:
        - `` `addEmployee()` ``: Adds a new employee to the data file and updates the index.
        - `` `searchEmployee()` ``: Searches for an employee by ID.
        - `` `deleteEmployee()` ``: Deletes an employee from both the data and index files.
        - `` `isPresent()` ``: Checks if an employee exists in the index.
        - `` `display_all()` ``: Displays all employees.
        - `` `readEmp()` ``: Reads an employee's data from the data file based on position.

3. Constructor

```cpp
employeeDatabase::employeeDatabase() {
    fstream data_file(data_file_name);
    fstream index_file(index_file_name);
    if (data_file.fail() | index_file.fail()) {
        cout << "Created a new file." << endl;
        data_file.open(data_file_name, ios::binary | ios::out | ios::app);
        index_file.open(index_file_name, ios::binary | ios::out | ios::app);
    } else {
        cout << "File already exists." << endl;
    }
    data_file.close();
    index_file.close();
}
```

- The constructor checks if the data and index files already exist.
    - If they do not exist, it creates new files for employee data and the index.
    - If the files exist, it outputs that the files are already present.

4. Adding an Employee

```cpp
void employeeDatabase::addEmployee(int eid, char name[50], char dest[50], int sal) {
    fstream data_file(data_file_name, ios::binary | ios::out | ios::app);
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current, temp;
    char *all_data;
    employee emp;
    int pos, ipos;
    bool update = false;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            if (current.position == -1) {
                ipos = (int)index_file.tellg() - sizeof(current);
                update = true;
                break;
            } else {
                cout << "Cannot add employee, already exists."
                     << endl;
                goto exit_addEmployee;
            }
        }
    }
    index_file.close();

    emp.empId = eid;
    strcpy(emp.name, name);
    strcpy(emp.designation, dest);
    emp.salary = sal;

    data_file.seekp(0, ios::end);
    pos = data_file.tellp();
    data_file.write((char *)&emp, sizeof(emp));
    current.key = eid;
    current.position = pos;
    // Additional code for updating or inserting into index...
}
```

- This function adds a new employee:
    - **Checks for duplicates**: It reads the index file to see if the employee already exists.
    - **Write to data file**: If the employee doesn't exist, it writes the employee to the end of the data file.
    - **Update index**: It updates the index file with the employee's ID and the position of the data in the file.

5. Searching for an Employee

```cpp
void employeeDatabase::searchEmployee(int eid) {
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current;
    int pos = -1;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            pos = current.position;
            break;
        }
    }

    employee emp = readEmp(pos);
    if (emp.empId == -1) {
        cout << "Employee does not exist." << endl;
    } else {
        cout << "--- EMPLOYEE FOUND ---" << endl;
        display_emp(emp);
        cout << "--- END OF RECORD ---" << endl;
    }

    index_file.close();
}
```

- **Search**: It reads through the index file, and if the employee is found, it fetches the employee data using `` `readEmp()` ``.

6. Deleting an Employee

```cpp
void employeeDatabase::deleteEmployee(int eid) {
    fstream index_file(index_file_name, ios::binary | ios::in);
    index_pair current;
    bool deleted = false;
    vector<index_pair> pairs;

    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.key == eid) {
            deleted = true;
        } else {
            pairs.push_back(current);
        }
    }
    index_file.close();
    if (deleted) {
        index_file.open(index_file_name, ios::binary | ios::out);
        index_file.clear();
        index_file.seekp(0, ios::beg);

        for (int i = 0; i < pairs.size(); i++) {
            current = pairs[i];
            index_file.write((char *)&current, sizeof(index_pair));
        }
        index_file.close();
        cout << "Employee removed from record." << endl;
    } else {
        cout << "Employee does not exist in the record." << endl;
    }
}
```

- **Delete**: It searches for an employee in the index file, removes it if found, and rewrites the index file without the deleted record.

7. Displaying All Employees

```cpp
void employeeDatabase::display_all() {
    fstream index_file(index_file_name, ios::binary | ios::in);
    fstream data_file(data_file_name, ios::binary | ios::in);
    index_pair current;
    employee emp;
    while (index_file.read((char *)&current, sizeof(index_pair))) {
        if (current.position != -1) {
            data_file.seekg(current.position, ios::beg);
            data_file.read((char *)&emp, sizeof(emp));
            cout << emp.empId << " | " << emp.name << " | " << emp.designation
                 << " | " << emp.salary << endl;
        }
    }
    index_file.close();
    data_file.close();
}
```

- **Display All**: It reads the index file, retrieves employee data from the data file based on the index, and displays it.

8. Main Program

```cpp
int main() {
    employeeDatabase db;
    int eid, sal;
    char name[50], dest[50];
    int ch;
    do {
        // Main menu for user interaction
        cin >> ch;
        switch (ch) {
            case 1:
                // Add employee code
                break;
            case 2:
                // Search employee code
                break;
            case 3:
                // Delete employee code
                break;
            case 4:
                db.display_all();
                break;
            case 5:
                cout << "\n\n// END OF CODE\n\n" << endl;
                break;
            default:
                cout << "Please choose a valid option (1-5)." << endl;
                break;
        }
    } while (ch != 5);

    return 0;
}
```

- The main function provides a menu-driven interface to interact with the employee database system. It allows you to add, search, delete, or display all employees.

Video Resource Recommendation:

For understanding file handling in C++ and the concepts of indexing, here are a couple of useful video resources:

- **C++ File Handling (YouTube)**: [Link to Video

A network error occurred. Please check your connection and try again. If this issue persists please contact us through our help center at help.openai.com.

Retry