MIT Art Design and Technology University

MIT School of Computing, Pune

Department of Computer Science and Engineering

Second Year B. Tech

Academic Year 2022-2023. (SEM-II)

Subject: Advance Data Structures Laboratory

Assignment 9

Assignment Title: Implementation of simple index file.

Aim: Implementation of simple index file.

Prerequisite:

1. Basic understanding of C++ programming language.
2. Familiarity with file handling in C++.

Objectives:

Implement a Program in C++ for the following operations:

1. To learn how to create an index file in C++.
2. To understand the importance of index files in organizing and searching data.
3. To gain practical experience in file handling using C++.

Outcomes:

Upon Completion of the assignment the students will be able to

1. Understanding of how to create and manipulate index files in C++.
2. Ability to organize and search data efficiently using index files.
3. Improved skills in file handling with C++.

Theory:

A simple index file is a file that contains an ordered list of key-value pairs that are used to organize and retrieve data quickly.
In C++, the standard library provides file stream classes that can be used to implement index files.

To create an index file in C++, we can use the ofstream class to write data to the file. We can use the open() function to open the file in write mode and the close() function to close the file. To write data to the file, we can use the insertion operator (<<) or the put() function.

To search for data in the index file, we can use the ifstream class to read data from the file. We can use the open() function to open the file in read mode and the close() function to close the file. To read data from the file, we can use the extraction operator (>>) or the get() function.

To organize the data in the index file, we can use a simple key-value pair structure. For example, we can use the name of the item as the key and the location of the item in the file as the value. When we want to retrieve an item, we can search for the key in the index file and use the corresponding value to locate the item in the file.

In summary, a simple index file in C++ is a file that contains an ordered list of key-value pairs that are used to organize and retrieve data quickly.

We can use the ofstream class to write data to the file and the ifstream class to read data from the file.

We can organize the data using a simple key-value pair structure, and retrieve data by searching for the key in the index file and using the corresponding value to locate the data in the file.

Conclusion:

We have created index file.

Frequently Asked Questions:

1. What is an index file?
2. Why are index files important?
3. How do I create an index file in C++?
4. How do I search for data in an index file?

MIT SCHOOL OF ENGINEERING
Rajbaug, Loni-Kalbhor, Pune

MIT-ADT UNIVERSITY
PUNE, INDIA

Name: Sourav Shailesh Toshniwal
Class: SY CSE - 1
Roll no.: 2213047
Subject: ADSL

Assignment - 9.

1. What is an index file?

Ans - Index file refers to a file that contains an index or catalog of the records or data stored in database.
- It is used to speed up searches and retrieval of data from a larger data set.
- An index file typically contains information such as the search key and a pointer to the location.

2. Why are index files important?

Ans - They speed up searches and retrieval of data.
- Different indexing techniques can be used to optimize the search & retrieval of data.
- They are used in databases, file systems, etc.
- Overall, index files improves performance & efficiency of data retrieval operations.

3. How do I create an index file in C++?

Ans You can create index file using B-trees, hash table, and BST. Here are steps using B-trees:

1. Define a data structure for index file that includes the search key value and a pointer to a location of corresponding record in the data file

2. Implement B-tree data structure using class or struct.
3. Write function to insert, delete and search.
4. Use file handling to R/W data to index file.
5. Create or update the index file as necessary when creating or updating the data file.
6. Test the implementation by inserting, searching & deleting.

4. How do I search for data in an index file?

Ans 1. Choose an appropriate search key for the data you are looking for.
2. Determine the indexing technique used to create the index file.
3. Use the search key to navigate the index file to the appropriate record or records that match the criteria.
4. Use the pointers or other information in the index file to locate the matching data in data file.
5. Return the matching data to the user.

Code:

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <iomanip>
using namespace std;
struct IndexRecord
{
    char key[10];
    long offset;
};
const int RECORD_SIZE = sizeof(IndexRecord);
void addRecord(fstream& indexFile)
{
    IndexRecord record;
    cout << "Enter the key: ";
    cin >> record.key;
    cout << "Enter the offset: ";
    cin >> record.offset;
    indexFile.seekp(0, ios::end);
    indexFile.write(reinterpret_cast<char*>(&record), RECORD_SIZE);
}
void searchRecord(fstream& indexFile)
{
    char key[10];
    IndexRecord record;
    cout << "Enter the key to search for: ";
    cin >> key;
    indexFile.seekg(0, ios::beg);
    while (indexFile.read(reinterpret_cast<char*>(&record), RECORD_SIZE))
    {
        if (strcmp(record.key, key) == 0)
        {
        cout << "Record found:" << endl;
        cout << "Key: " << record.key << endl;
        cout << "Offset: " << record.offset << endl;
        return;
        }
    }
    cout << "Record not found." << endl;
}
void printRecords(fstream& indexFile)
{
    IndexRecord record;
    cout << setw(10) << "Key" << setw(10) << "Offset" << endl;
    cout << "----------------------" << endl;
    indexFile.seekg(0, ios::beg);
    while (indexFile.read(reinterpret_cast<char*>(&record), RECORD_SIZE))
    {
        cout << setw(10) << record.key << setw(10) << record.offset << endl;
    }
}
int main()
{
    fstream indexFile("index.dat", ios::in | ios::out | ios::binary);
    if (!indexFile)
    {
        indexFile.open("index.dat", ios::out | ios::binary);
        indexFile.close();
        indexFile.open("index.dat", ios::in | ios::out | ios::binary);
```

```
    }
    int choice;
    do
    {
        cout << endl;
        cout << "1. Add a new record" << endl;
        cout << "2. Search for a record" << endl;
        cout << "3. Print all records" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice)
        {
            case 1:
            addRecord(indexFile);
            break;
            case 2:
            searchRecord(indexFile);
            break;
            case 3:
            printRecords(indexFile);
            break;
            case 4:
            break;
            default:
            cout << "Invalid choice. Please try again." << endl;
            break;
        }
    } while (choice != 4);
    indexFile.close();
    return 0;
}
```

Output:

```
PS C:\SOURAV\CODE\C++ language codes\ADS assignment> cd "c:\SOURAV\CODE\C++ language codes\ADS assignment\" ; if ($?) { g++ Assignment9.cp

1. Add a new record
2. Search for a record
3. Print all records
4. Exit
Enter your choice: 1
Enter the key: 2
Enter the offset: 1

1. Add a new record
2. Search for a record
3. Print all records
4. Exit
Enter your choice: 2
Enter the key to search for: 2
Record found:
Key: 2
Offset: 1
```