|  |  |  |
| --- | --- | --- |
| Image result for latest marwadi university logo | **Marwadi University**  **Faculty of Technology**  **Department of Information and Communication Technology** | |
| **Subject: DSC  (01CT0308)** | Aim: Implementations of searching methods (Index Sequential, Interpolation Search) menu-driven program. | |
| **Experiment No: 9** | **Date: 26- 10 - 2023** | **Enrolment No:-** 92200133030 |

**Experiment – 9**

**Objective:** Implementations of searching methods (Index Sequential, Interpolation Search) menu-driven program.

**Code :-**

#include <iostream>

#include <vector>

#include <algorithm>

#include <cmath>

using namespace std;

class Record {

public:

int key;

string data;

Record(int k, const string& d) : key(k), data(d) {}

};

// Function to compare records based on their keys for sorting

bool compareRecords(const Record& a, const Record& b) {

return a.key < b.key;

}

int indexSequentialSearch(vector<Record>& records, int key) {

int block\_size = sqrt(records.size());

int block\_number = key / block\_size;

for (int i = block\_number \* block\_size; i < min((block\_number + 1) \* block\_size, static\_cast<int>(records.size())); i++) {

if (records[i].key == key) {

return i;

}

}

return -1;

}

int interpolationSearch(vector<Record>& records, int key) {

int left = 0;

int right = records.size() - 1;

while (left <= right && key >= records[left].key && key <= records[right].key) {

if (left == right) {

if (records[left].key == key) {

return left;

}

return -1;

}

int pos = left + ((key - records[left].key) \* (right - left)) / (records[right].key - records[left].key);

if (records[pos].key == key) {

return pos;

} else if (records[pos].key < key) {

left = pos + 1;

} else {

right = pos - 1;

}

}

return -1;

}

int main() {

vector<Record> records = {

Record(10, "Record 1"),

Record(20, "Record 2"),

Record(30, "Record 3"),

Record(40, "Record 4"),

Record(50, "Record 5"),

Record(60, "Record 6"),

Record(70, "Record 7"),

Record(80, "Record 8"),

Record(90, "Record 9"),

Record(100, "Record 10")

};

// Sort the records by key before searching

sort(records.begin(), records.end(), compareRecords);

int choice;

int key;

do {

cout << "Menu:" << endl;

cout << "1. Index Sequential Search" << endl;

cout << "2. Interpolation Search" << endl;

cout << "3. Exit" << endl;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1: {

cout << "Enter key to search: ";

cin >> key;

int index = indexSequentialSearch(records, key);

if (index != -1) {

cout << "Key found at index " << index << ": " << records[index].data << endl;

} else {

cout << "Key not found." << endl;

}

break;

}

case 2: {

cout << "Enter key to search: ";

cin >> key;

int interpolationIndex = interpolationSearch(records, key);

if (interpolationIndex != -1) {

cout << "Key found at index " << interpolationIndex << ": " << records[interpolationIndex].data << endl;

} else {

cout << "Key not found." << endl;

}

break;

}

case 3: {

cout << "Exiting the program." << endl;

break;

}

default: {

cout << "Invalid choice. Please try again." << endl;

break;

}

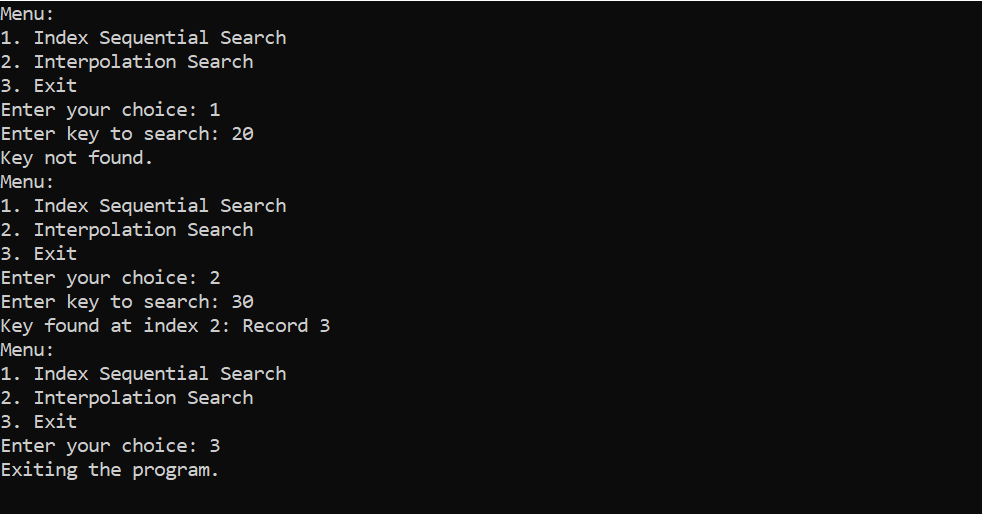
}

} while(choice != 3);

return 0;

}

**Output:**

****