|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ | | | | | | | | | | | | |
| Федеральное государственное автономное образовательное учреждение  высшего образования | | | | | | | | | | | | |
| **Дальневосточный федеральный университет** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **Департамент программной инженерии и искусственного интеллекта** | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| **О Т Ч Е Т** | | | | | | | | | | | | |
| по лабораторной работе №2.2  дисциплина «Фундаментальные структуры данных и алгоритмы» | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Студент гр. Б9122-09.03.04прогин | | | |
|  |  |  | | А. В. Поляков | |
|  | | | | | | |  |  | (подпись) | |  | |
|  | | | | | | | | | | | | |
|  | | | | | | |  |  | Руководитель | | |  |
|  | | | | | | |  |  | ст. преподаватель | | | |
|  | | | |  |  | |  |  |  |  | О.А. Крестникова | |
|  | | | |  |  | |  |  | (подпись) |  | (И.О. Фамилия) | |
|  |  |  |  | | |  |  |  |  | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
|  | | | | | | | | | | | | |
| г. Владивосток | | | | | | | | | | | | |
| 2024 | | | | | | | | | | | | |

**1 Неформальная постановка задачи**

**Входные данные:**

Текстовый файл, каждая из строк которого содержит 4 поля в табличном виде (первое поле – код направления, второе поле – ФИО, третье поле – название направления и четвертое поле – номер заявки (число)). Все данные во входном файле корректные (проверять на корректность не нужно). Количество строк входного файла – 1000000.

Целое число n – количество строк входного файла для обработки (10 <= n <= 1000000).

Ключ поиска.

**Выходные данные:**

Список номеров строк входного файла, содержащих заданный ключ поиска.

Количество шагов поиска Линейного с барьером и Бинарного с таблицей шагов алгоритмов (без учета сортировки).

**2 Описание алгоритмов поиска**

// Линейный поиск

Function linearSearchWithBarrier (students, key)

// Добавляем барьерный элемент в конец вектора

barrierStudent.applicationNum = key

students.push\_back(barrierStudent)

i = 0

while (students[i].applicationNum != key) do

i = i + 1

// Удаляем барьерный элемент из вектора

students.pop\_back()

return i  
  
// Бинарный поиск с таблицей шагов

Function binarySearchWithSteps (students, key)

n = students.size()

counter = 1

// Определение таблицы шагов

steps = []

step = 1

while (step <= n) do

steps.push\_back(step)

step = step \* 2

// Поиск

low = 0

high = n - 1

for i = steps.size() - 1 to 0 do

mid = low + steps[i] - 1

if (mid >= n) then

mid = n - 1

if (students[mid].applicationNum == key) then

print counter + " - итераций, " + students[mid].lineNumber + " - лайн намбер"

return

else if (students[mid].applicationNum < key) then

low = mid + 1 // Поиск в правой половине

else

high = mid - 1 // Поиск в левой половине

counter = counter + 1

print "Элемент не найден"

**3 Текст программы**

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <ctime>

#include <vector>

#include <chrono>

using namespace std;

// Структура для хранения кода направления студента

struct Code {

    char letter;

    unsigned int numbers;

};

// Структура для хранения ФИО студента

struct FullName {

    string surname;

    string name;

    string patronymic;

    string FIO;

};

// Структура для хранения названия направления студента

struct CourseName {

    string name;

};

// Структура для хранения данных студента

struct Student {

    Code code;

    FullName fullName;

    CourseName name;

    int applicationNum;

    int lineNumber;

};

// Функция для генерации случайного кода направления

Code generateCode() {

    Code code;

    code.letter = 'B';

    code.numbers = 1000 + rand() % 9000;

    return code;

}

// Перегрузки оператора сравнения для Student

bool operator<(const Student& a, const Student& b) {

    return (a.applicationNum < b.applicationNum);

}

bool operator>(const Student& a, const Student& b) {

    return (a.applicationNum > b.applicationNum);

}

// Перегрузка оператора вывода для Code

ostream& operator<<(ostream& os, const Code& code) {

    os << code.letter << code.numbers;

    return os;

}

// Перегрузка оператора вывода для CourseName

ostream& operator<<(ostream& os, const CourseName& name) {

    os << name.name;

    return os;

}

// Перегрузка оператора вывода для Student

ostream& operator<<(ostream& os, const Student& student) {

    os << student.code.letter << student.code.numbers << " " << student.fullName.FIO << " " << student.name.name << " " << student.applicationNum;

    return os;

}

// Генерация рандомного имени студента

FullName generateFullName() {

    string names[] = {"Artem", "Ekaterina", "Maxim", "Alexander", "Dmitry", "Nikita", "Egor", "Mikhail", "Roman", "Vladimir", "Pavel", "Daniel", "Anton", "Alexey", "Sergei"};

    string surnames[] = {"Polyakov", "Krushinina", "Johnson", "Brown", "Taylor", "Wilson", "Harris", "Clark", "White", "Anderson", "Allen", "Davis", "Martin", "Hall", "Lee"};

    string patronymics[] = {"Victorovich", "Vladislavovna", "Ivanovich", "Petrovich", "Sidorovich", "Mikhailovich", "Fedorovich", "Egorovich", "Romanovich", "Vladimirovich", "Pavlovich", "Daniilovich", "Antonovich", "Alexandrovich", "Sergeevich"};

    FullName fullName;

    fullName.FIO = surnames[rand() % 15] + " " + names[rand() % 15] + " " + patronymics[rand() % 15];

    return fullName;

}

// Генерация рандомного названия курса

CourseName generateCourseName() {

    string names[] = {"progin", "economic", "menegment", "urfuck", "med", "arhitecture", "himbio", "fizvoz", "politech", "ped"};

    CourseName name;

    name.name = names[rand() % 10];

    return name;

}

// Генератор ключей для входного файла

void generateKyesFile() {

    ofstream inFile("C:/Users/User/Desktop/FEFU/Second\_course/FDSA/Num\_Search/inputFile.txt");

    for (int i = 0; i < 1000000; ++i) {

        Student student;

        student.code = generateCode();

        student.fullName = generateFullName();

        student.name = generateCourseName();

        student.applicationNum = rand() % 10000;

        inFile << student.code << " " << student.fullName.FIO << " " << student.name << " " << student.applicationNum << endl;

    }

    inFile.close();

    cout << "Ключи сгенерированы\n" << endl;

}

// Функция для чтения данных студентов из файла

vector<Student> readKeyFromFile(int numOfLines) {

    ifstream inFile("C:/Users/User/Desktop/FEFU/Second\_course/FDSA/Num\_Search/inputFile.txt");

    vector<Student> students;

    int lineNumber = 1;

    int i = 1;

    while (i <= numOfLines) {

        Student student;

        inFile >> student.code.letter >> student.code.numbers;

        inFile >> student.fullName.surname >> student.fullName.name >> student.fullName.patronymic;

        student.fullName.FIO = student.fullName.surname + " " + student.fullName.name + " " + student.fullName.patronymic;

        inFile >> student.name.name;

        inFile >> student.applicationNum;

        i++;

        if (!inFile.fail()) {

            student.lineNumber = lineNumber++;

            students.push\_back(student);

        }

    }

    inFile.close();

    return students;

}

void printStudentVector(vector<Student> students) {

    for (const auto& student : students) {

        cout << student.code << " " << student.fullName.FIO << " " << student.name.name << " " << student.applicationNum << endl;

    }

}

// Функция сортировки вставками

void insertionSort(vector<Student>& students) {

    for (size\_t i = 1; i < students.size(); i++) {

        Student key = students[i];

        int j = i - 1;

        while (j >= 0 && students[j] > key) {

            students[j + 1] = students[j];

            j = j - 1;

        }

        students[j + 1] = key;

    }

}

// Линейный поиск с барьером

void linearSearchWithBarrier(vector<Student>& students, int key) {

    // Добавляем барьерный элемент в конец вектора

    Student barrierStudent;

    barrierStudent.applicationNum = key;

    students.push\_back(barrierStudent);

    int i = 0;

    while (students[i].applicationNum != key)

        i++;

    // Удаляем барьерный элемент из вектора

    students.pop\_back();

    if (i == students.size()) {

        cout << "Элемент не найден" << endl;

    } else {

        cout << i + 1 << " - итерраций," << " " << students[i].lineNumber << " - лайн намбер" << endl;

    }

}

// Однородный бинарный поиск с таблицей шагов

void binarySearchWithSteps(const vector<Student>& students, int key) {

    int n = students.size();

    int counter = 1;

    // Определение таблицы шагов

    vector<int> steps;

    int step = 1;

    while (step <= n) {

        steps.push\_back(step);

        step \*= 2;

    }

    // Поиск

    int low = 0, high = n - 1;

    for (int i = steps.size() - 1; i >= 0; --i) {

        int mid = low + steps[i] - 1;

        if (mid >= n) {

            mid = n - 1;

        }

        if (students[mid].applicationNum == key) {

            cout << counter << " - итерраций," << " " << students[mid].lineNumber << " - лайн намбер" << endl; // Элемент найден

            return;

        } else if (students[mid].applicationNum < key) {

            low = mid + 1; // Поиск в правой половине

        } else {

            high = mid - 1; // Поиск в левой половине

        }

        counter++;

    }

    cout << "Элемент не найден" << endl;

}

int main() {

    //generateKyesFile();

    vector<Student> students = readKeyFromFile(13);

    //insertionSort(students);

    //printStudentVector(students);

    // Линейный с барьером

    int key = 5724;

    linearSearchWithBarrier(students, key);

    insertionSort(students);

    // Однородный бинарный с таблицей шагов

    binarySearchWithSteps(students, key);

    //printStudentVector(students);

    return 0;

}

**4 Тесты**

|  |  |  |
| --- | --- | --- |
| **Описание тестовой ситуации** | **Линейный поиск c барьером** | **Бинарный поиск с таблицей шагов** |
| Поиск ключа в пустом файле | Входные данные: inputFile.txt, key = 5977, n = 10;  Выходные данные:  Сообщение: «Элемент не найден» | Входные данные: inputFile.txt, key = 5977, n = 10;  Выходные данные:  Сообщение: «Элемент не найден» |
| Поиск ключа, которые находятся в начале файла | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  1 - итерраций, 1 – номер строки | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  14 - итеррации, 1 – номер строки |
| Поиск ключа, которые находятся в середине файла | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  5000 - итерраций, 5000 – номер строки | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  14 - итеррации, 5000 – номер строки |
| Поиск ключа, которые находятся в конце файла | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  10000 - итерраций, 10000 – номер строки | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  14 - итеррации, 10000 – номер строки |
| Поиск ключа, который находится в случайной строке | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  6491 - итерраций, 6491 – номер строки | Входные данные: inputFile.txt, key = 5977 – уникальный ключ, n = 10000;  Выходные данные:  14 - итерраций, 6491 – номер строки |