|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №2**

по дисциплине «Структуры и алгоритмы обработки данных»

по теме «Алгоритмы поиска в таблице (массиве). Применение алгоритмов поиска к поиску по ключу записей в файле»

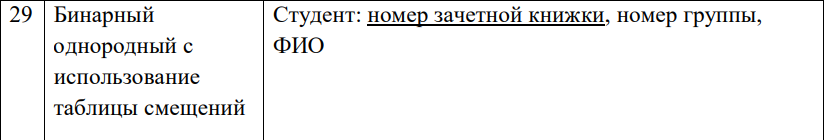
|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-13-22 | Тринеев Павел Сергеевич |
| **Проверил:** | ассистент Муравьёва Е.А. |

МОСКВА 2023 г.

**Цель:** получить практический опыт по применению алгоритмов поиска в таблицах данных.

**Задание:** разработать программу поиска записей с заданным ключом в двоичном файле с применением различных алгоритмов.

**Вариант 29.**

****

**Задание 1.**

Создать двоичный файл из записей (структура записи определена вариантом – смотрите в конце файла). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны.

Рекомендация: создайте сначала текстовый файл, а затем преобразуйте его в двоичный.

При открытии файла обеспечить контроль существования и открытия файла.

**Задание 2**

Поиск в файле с применением линейного поиска:

1. Разработать программу поиска записи по ключу в бинарном файле, созданном в первом задании, с применением алгоритма линейного поиска.

2. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.

3. Составить таблицу с указанием результатов замера времени.

**Задание 3**

Поиск записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.

1. Для оптимизации поиска в файле создать в оперативной памяти структур данных – таблицу, содержащую ключ и ссылку (смещение) на запись в файле.

2. Разработать функцию, которая принимает на вход ключ и ищет в таблице элемент, содержащий ключ поиска, а возвращает ссылку на запись в файле. Алгоритм поиска определен в варианте.

3. Разработать функцию, которая принимает ссылку на запись в файле, считывает ее, применяя механизм прямого доступа к записям файла. Возвращает прочитанную запись как результат.

4. Провести практическую оценку времени выполнения поиска на файле объемом 100, 1000, 10 000 записей.

5. Составить таблицу с указанием результатов замера времени.

**Задание 1, Решение.**

Нахождение размера текстового и бинарного файла.(листинг 1)

Листинг 1.

size\_t recordSize = sizeof(Students);

cout << "Размер одной записи (Students) в байтах: " << recordSize << " байт" << endl;

Размер одной записи (Students) в байтах: 48 байт

Выполнение создания файлов(листинг 2).

Листинг 2.

for (int i = 0; i < 100; ++i)

{

do {

policy.credit\_card\_number = rand() % 9999;

} while (unique.count(policy.credit\_card\_number) > 0);

unique.insert(policy.credit\_card\_number);

do {

policy.group\_number = rand() % 99;

} while (unique.count(policy.group\_number) > 0);

unique.insert(policy.group\_number);

policy.name = "Вася Пупкин Далерович";

fileText << policy.credit\_card\_number << " " << policy.group\_number << " " << policy.name << "\n";

fileBinaryOUT.write(reinterpret\_cast<const char\*>(&policy), sizeof(policy));

}

**Задание 2.**

Алгоритм линейного поиска записи с ключом в файле(листинг 3).

Листинг 3:

int key;

cout << "введите номер зачетной книжки";

cin >> key;

auto start = chrono::high\_resolution\_clock::now();

while (fileBinaryIN.read(reinterpret\_cast<char\*>(&policy), sizeof(policy)))

{

if (policy.credit\_card\_number == key)

{

cout << endl << "Данные студента:" << endl;

cout << "номер зачетной книжки: " << policy.credit\_card\_number << endl;

cout << "номер группы: " << policy.group\_number << endl;

cout << "ФИО: " << policy.name << endl;

fileBinaryIN.close();

return 0;

}

}

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "время поиска: " << duration.count() << "s" << endl;

cout << "Студента с таким номером нет" << endl;

fileBinaryIN.close();

return 0;

Таблица тестирований.

|  |  |
| --- | --- |
| N | T,сек |
| 100 | 0.0024711s |
| 1000 | 0.0025203s |
| 10000 | 0.0027485s |

Полный листинг кода(листинг 4).

Листинг 4.

#include <iostream>

#include <fstream>

#include <string>

#include <chrono>

#include <set>

using namespace std;

struct Students

{

int credit\_card\_number = 0;

int group\_number;

string name;

};

int main()

{

setlocale(LC\_ALL, "ru");

size\_t recordSize = sizeof(Students);

cout << "Размер одной записи (Students) в байтах: " << recordSize << " байт" << endl;

ofstream fileText("Text.txt");

if (!fileText)

{

cout << "file is not open" << endl;

return 1;

}

ofstream fileBinaryOUT("Binary.bin", ios::binary);

if (!fileBinaryOUT) {

cout << "bin\_file is not open" << endl;

return 1;

}

set<int> unique;

Students policy;

for (int i = 0; i < 1000; ++i)

{

do {

policy.credit\_card\_number = 1000 + rand() % (9999 - 1000 + 1);

} while (unique.count(policy.credit\_card\_number) > 0);

unique.insert(policy.credit\_card\_number);

do {

policy.group\_number = 100 + rand() % (999 - 100 + 1);

} while (unique.count(policy.group\_number) > 0);

unique.insert(policy.group\_number);

policy.name = "Вася Пупкин Далерович";

fileText << policy.credit\_card\_number << " " << policy.group\_number << " " << policy.name << "\n";

fileBinaryOUT.write(reinterpret\_cast<const char\*>(&policy), sizeof(policy));

}

fileText.close();

fileBinaryOUT.close();

ifstream fileBinaryIN("Binary.bin", ios::binary);

if (!fileBinaryIN)

{

cout << "bin\_file is not open" << endl;

return 1;

}

int key;

cout << "введите номер зачетной книжки";

cin >> key;

auto start = chrono::high\_resolution\_clock::now();

while (fileBinaryIN.read(reinterpret\_cast<char\*>(&policy), sizeof(policy)))

{

if (policy.credit\_card\_number == key)

{

cout << endl << "Данные студента:" << endl;

cout << "номер зачетной книжки: " << policy.credit\_card\_number << endl;

cout << "номер группы: " << policy.group\_number << endl;

cout << "ФИО: " << policy.name << endl;

fileBinaryIN.close();

auto end = chrono::high\_resolution\_clock::now();

chrono::duration<float> duration = end - start;

cout << "время поиска: " << duration.count() << "s" << endl;

return 0;

}

}

cout << "Студента с таким номером нет" << endl;

fileBinaryIN.close();

return 0;

}

**Вывод.**

Был получен практический опыт по применению алгоритмов поиска в таблицах данных.