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 группы ИКБО-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.

29	Бинарный однородный с использование таблицы смещений	Студент: номер зачетной книжки, номер группы, ФИО

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.

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.

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;


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.

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.

Алгоритм линейного поиска записи с ключом в файле(листинг 2).

Листинг 2.

```
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

Псевдокод задания 1,2.

```
Функция linear_search(file, key):
    Открыть файл file для чтения
    Прочитать первую запись из файла и сохранить её в переменной record

    Пока не достигнут конец файла:
        Если значение ключа в записи record равно key:
            Вернуть запись record
        Прочитать следующую запись из файла и сохранить её в переменной record

    Вернуть None, так как запись с ключом не была найдена

Конец функции
```

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.

Бинарный однородный с использование таблицы смещений (Листинг 4).

В листинге представлены фрагменты кода добавленные для короткой работы 3 задания.

Листинг 4.

```
struct OffsetTableEntry
{
    long long credit_card_number = 0;
    streampos offset; //streampos хранит позицию(смещение) в бинарном файле
};

//функция для сортировки таблицы смещений по ключу
bool CompareBycredit_card_number(const OffsetTableEntry& a, const
OffsetTableEntry& b)
{
    return a.credit_card_number < b.credit_card_number;
}

streampos ModifiedBinarySearch(const vector<OffsetTableEntry>& offsetTable, long
long key) //однородный бинарный поиск с применением таблицы смещений
{
    size_t N = offsetTable.size();
    vector<size_t> delta(N);

    // Вычисляем значения delta заранее
    for (size_t j = 1; j <= N; j++)
    {
        delta[j - 1] = (N + (1ULL << (j - 1))) / (1ULL << j); //вычисляем j-е
значение по формуле (N+2^(j-1))/2^j
    }

    size_t i = delta[0];
    size_t j = 2; //начиная с первого элемента [j-1]

    while (delta[j - 1] > 0)
    {
        if (offsetTable[i].credit_card_number == key)
        {
            // Запись найдена, возвращаем смещение
            return offsetTable[i].offset;
        }
        else if (offsetTable[i].credit_card_number < key)
        {
            // Сдвигаемся вправо, увеличиваем i и j
            i += delta[j - 1];
            j++;
        }
        else
        {
            // Сдвигаемся влево, уменьшаем i и увеличиваем j
            i -= delta[j - 1];
```

```
            j++;
        }
    }

    // Запись не найдена
    return -1;
}
```

```
vector<OffsetTableEntry> offsetTable; // Таблица смещений

ifstream fileBinaryIN("Bintext.bin", ios::binary);
if (!fileBinaryIN)
{
    cout << "Бинарный файл не удалось открыть для чтения." << endl;
    return 1;
}


streampos currentOffset = 0; //изначальная позиция смещения
while (fileBinaryIN.read(reinterpret_cast<char*>(&data), sizeof(data)))
{
    OffsetTableEntry entry; //создание записи в таблице смещений
    entry.credit_card_number = data.credit_card_number;
    entry.offset = currentOffset;
    offsetTable.push_back(entry);

    currentOffset = fileBinaryIN.tellg(); //Обновление позиции чтения для
следующей записи
}

//сортировка массива по ключу(для бинарного поиска)
sort(offsetTable.begin(), offsetTable.end(), CompareBycredit_card_number);

fileBinaryIN.close();
```

Таблица тестирований.

N	T,сек
100	0.0019311s
1000	0.0019503s
10000	0.0019745s

Псевдокод задания 3.

1. Инициализация переменных:
   M = положение на таблице смещения
   Delta = шаг на таблице смешения
2. Создания массива значений delta
   Через цикл for(I;i<размер масива)
   По формуле N+2^(j-1))/2^2
3. Поиск нужного элемента
   Через цикл while(dekta[j-1] > 0)
   Если элемент на котором мы находимся равен искомому
       Вернем положение элемента
   Если элемент меньше искомого
       Делаем шаг на половину массива в лево
       J++
   Если элемент больше искомого
       Делаем шаг на половину массива в право
       J++

Полный листинг кода заданий 1, 2(листинг 5).

Листинг 5.

```
#include <iostream>
#include <fstream>
#include <string>
#include <chrono>
#include <set>
#include <vector>
#include <algorithm>

using namespace std;


struct Students
{
    int credit_card_number = 0;
    int group_number = 0;
    string name;
};

int main()
{

    setlocale(LC_ALL, "ru");


    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 data;


    for (int i = 0; i < 10000; ++i)
    {
        do {
            data.credit_card_number = rand() % 99999;

        } while (unique.count(data.credit_card_number) > 0);
        unique.insert(data.credit_card_number);

        do {
            data.group_number = rand() % 99999;
        } while (unique.count(data.group_number) > 0);
        unique.insert(data.group_number);

        data.name = "Вася Пупкин Далерович";

        fileText << data.credit_card_number << " " << data.group_number << " " <<
data.name << "\n";
        fileBinaryOUT.write(reinterpret_cast<const char*>(&data), sizeof(data));
    }

    fileText.close();
    fileBinaryOUT.close();


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


    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*>(&data), sizeof(data)))
    {
        if (data.credit_card_number == key)
        {
            cout << endl << "Данные студента:" << endl;
            cout << "номер зачетной книжки: " << data.credit_card_number << endl;
            cout << "номер группы: " << data.group_number << endl;
            cout << "ФИО: " << data.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;
}
```

Полный листинг кода заданий 3(листинг 6).

Листинг 6.

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>
#include <vector>
#include <set> //для набора уникальных элементов
#include <chrono> // Для измерения времени
#include <algorithm>

using namespace std;

struct Students
{
    int credit_card_number = 0;
    int group_number;
    string name;
};


struct OffsetTableEntry
{
    long long credit_card_number = 0;
    streampos offset; //streampos хранит позицию(смещение) в бинарном файле
};

//функция для сортировки таблицы смещений по ключу
bool CompareBycredit_card_number(const OffsetTableEntry& a, const
OffsetTableEntry& b)
{
    return a.credit_card_number < b.credit_card_number;
}

streampos ModifiedBinarySearch(const vector<OffsetTableEntry>& offsetTable, long
long key) //однородный бинарный поиск с применением таблицы смещений
{
    size_t N = offsetTable.size();
    vector<size_t> delta(N);

    // Вычисляем значения delta заранее
    for (size_t j = 1; j <= N; j++)
    {
        delta[j - 1] = (N + (1ULL << (j - 1))) / (1ULL << j); //вычисляем j-е
значение по формуле (N+2^(j-1))/2^j
    }

    size_t i = delta[0];
    size_t j = 2; //начиная с первого элемента [j-1]

    while (delta[j - 1] > 0)
    {
```

```
        if (offsetTable[i].credit_card_number == key)
        {
            // Запись найдена, возвращаем смещение
            return offsetTable[i].offset;
        }
        else if (offsetTable[i].credit_card_number < key)
        {
            // Сдвигаемся вправо, увеличиваем i и j
            i += delta[j - 1];
            j++;
        }
        else
        {
            // Сдвигаемся влево, уменьшаем i и увеличиваем j
            i -= delta[j - 1];
            j++;
        }
    }

    // Запись не найдена
    return -1;
}

int main()
{
    setlocale(LC_ALL, "Rus");

    srand(static_cast<unsigned int>(time(nullptr)));

    ofstream fileText("text.txt");
    if (!fileText)
    {
        cout << "Текстовый файл не удалось открыть для записи." << endl;
        return 1;
    }

    ofstream fileBinaryOUT("Bintext.bin", ios::binary);
    if (!fileBinaryOUT) {
        cout << "Бинарный файл не удалось открыть для записи." << endl;
        return 1;
    }

    int quant = 10000;
    set<int> unique;

    Students data;
    for (int i = 0; i < quant; ++i)
    {
        do {
            data.credit_card_number = rand() % 99999;

        } while (unique.count(data.credit_card_number) > 0);
        unique.insert(data.credit_card_number);

        do {
            data.group_number = rand() % 99999;
        } while (unique.count(data.group_number) > 0);
        unique.insert(data.group_number);

        data.name = "Вася Пупкин Далерович";

        fileText << data.credit_card_number << " " << data.group_number << " " <<
data.name << "\n";
        fileBinaryOUT.write(reinterpret_cast<const char*>(&data), sizeof(data));
    }
```

```
    fileText.close();
    fileBinaryOUT.close();

    vector<OffsetTableEntry> offsetTable; // Таблица смещений

    ifstream fileBinaryIN("Bintext.bin", ios::binary);
    if (!fileBinaryIN)
    {
        cout << "Бинарный файл не удалось открыть для чтения." << endl;
        return 1;
    }


    streampos currentOffset = 0; //изначальная позиция смещения
    while (fileBinaryIN.read(reinterpret_cast<char*>(&data), sizeof(data)))
    {
        OffsetTableEntry entry; //создание записи в таблице смещений
        entry.credit_card_number = data.credit_card_number;
        entry.offset = currentOffset;
        offsetTable.push_back(entry);

        currentOffset = fileBinaryIN.tellg(); //Обновление позиции чтения для
следующей записи
    }

    //сортировка массива по ключу(для бинарного поиска)
    sort(offsetTable.begin(), offsetTable.end(), CompareBycredit_card_number);

    fileBinaryIN.close();

    long long key;
    cout << "Введите введите номер зачетной книжк: ";
    cin >> key;

    auto start = chrono::high_resolution_clock::now();

    streampos offset = ModifiedBinarySearch(offsetTable, key);

    //если запись нашлась
    if (offset != -1)
    {
        ifstream fileBinaryIN("Bintext.bin", ios::binary);
        if (fileBinaryIN)
        {
            fileBinaryIN.seekg(offset); //установка позиции чтения на место
найденной записи

            if (fileBinaryIN.read(reinterpret_cast<char*>(&data), sizeof(data)))
//считывание записи
            {
                cout << endl << "Данные студента:" << endl;
                cout << "номер зачетной книжки: " << data.credit_card_number <<
endl;
                cout << "номер группы: " << data.group_number << endl;
                cout << "ФИО: " << data.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 << "Запись с номером " << key << " не была найдена." << endl;

    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;
    cout << "Время выполнения поиска: " << duration.count() << " секунд." << endl;

    fileBinaryIN.close();

    return 0;
}
```

Вывод.

Был получен практический опыт по применению алгоритмов поиска в таблицах данных, работа с бинарными файлами и бинарным однородным поиском с таблицей смешения.