



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

Лещенко Вячеслав Романович

Проверил:

ассистент Муравьёва Е.А.

МОСКВА 2023 г.

Практическая работа № 1

Цель работы

Получить практический опыт по применению алгоритмов поиска в таблицах данных. Разработать программу поиска записей с заданным ключом в двоичном файле с применением различных алгоритмов.

Ход работы

Вариант 20

20	Фибоначчи поиск	Владельцев автомобилей. <u>номер машины</u> , марка, сведения о владельце.
----	-----------------	----------------------------------------------------------------------------

Задание 1

Формулировка задачи:

Создать двоичный файл из записей. 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.

Решение:

- 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.

```

int bin_search_one()
{
    FIO fio("data", "data");
    FIO bin_fio("", "data.bin", "bin");

    size_t entries = io.input<size_t>("Enter amount of entries: ");
    std::set<std::string> unique_entries;

    srand(time(0));

    for (size_t i = 0; i < entries; i++)
    {
        carDriver driver;
        strcpy_s(driver.name, ("Driver_" + std::to_string(i)).c_str());
        strcpy_s(driver.car_brand, ("Car_" + std::to_string(i)).c_str());
        std::string license_number;

        do
        {
            for (size_t j = 0; j < 6; j++)
            {
                int num = rand() % 36;
                license_number += (char)(num < 10 ? num + '0' : num - 10 +
'A');
            }
            while (unique_entries.count(license_number) > 0);

            unique_entries.insert(license_number);

            strcpy_s(driver.license_number, license_number.c_str());

            fio.output(driver.license_number);
            fio.output(driver.car_brand);
            fio.output(driver.name);
        }

        io.output("File generated!");

        for (size_t i = 0; i < entries; i++)
        {
            carDriver driver;
            std::string data;

            data = fio.input<std::string>();
            strcpy_s(driver.license_number, data.c_str());

            data = fio.input<std::string>();
            strcpy_s(driver.car_brand, data.c_str());

            data = fio.input<std::string>();
            strcpy_s(driver.name, data.c_str());

            bin_fio.write(reinterpret_cast<const char*>(&driver), sizeof(driver));
        }

        io.output("Binary file generated!");
        io.output("Entry size: ", "");
        io.output(sizeof(carDriver), " bytes\n");

        fio.close();
        bin_fio.close();

        return 0;
    }
}

```

Листинг 1 – код задачи 1

```
1 | YAE22D|Car_0|Driver_0|TCCFUS|Car_1|Driver_1|
```

```
1 | YAE22D|
2 | Car_0|
3 | Driver_0|
4 | TCCFUS|
5 | Car_1|
6 | Driver_1|
7 | 69E7FV|
8 | Car_2|
9 | Driver_2|
10 | L0WIK|
11 | Car_3|
12 | Driver_3|
13 | 986NSS|
14 | Car_4|
15 | Driver_4|
16 | UH8ZB|
17 | Car_5|
18 | Driver_5|
19 | QD8ULE|
20 | Car_6|
21 | Driver_6|
22 | U9KP79|
23 | Car_7|
24 | Driver_7|
25 | HNL1UX|
```

Рисунок 1-2 – результат тестирования 1

Задание 2

Формулировка задачи:

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.

Псевдокод:

Функция **ЛинейныйПоиск(массив, цель):**

Для каждого элемента в массиве с индексом *i* от 0 до длины массива - 1:

Если элемент[*i*] равен ключу:

Вернуть *i* (или сам элемент)

Вернуть -1 (или другое значение, чтобы указать, что элемент не найден)

Листинг 2 – псевдокод задачи 2

```

int bin_search_two()
{
    FIO bin_fio("data.bin", "", "bin");

    carDriver driver;

    bool was_found = false;
    std::string license_number = io.input<std::string>("Enter license: ");

    measure(
        while (bin_fio.read(reinterpret_cast<char*>(&driver), sizeof(driver)))
        {
            if (driver.license_number == license_number)
            {
                was_found = true;
                break;
            }
        }, "Elapsed time: ");

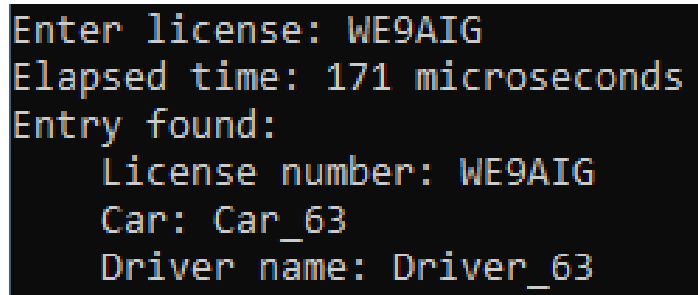
    if (was_found)
    {
        io.output("Entry found: ");
        io.output("    License number: ", "");
        io.output(driver.license_number);
        io.output("    Car: ", "");
        io.output(driver.car_brand);
        io.output("    Driver name: ", "");
        io.output(driver.name);
    }
    else
        io.output("Entry wasn't found.");

    bin_fio.close();

    return 0;
}

```

Листинг 3 – Код задачи 2



```

Enter license: WE9AIG
Elapsed time: 171 microseconds
Entry found:
    License number: WE9AIG
    Car: Car_63
    Driver name: Driver_63

```

Рисунок 3 – результат тестирования для N=100

```
Enter license: 33M90Z
Elapsed time: 344 microseconds
Entry found:
    License number: 33M90Z
    Car: Car_930
    Driver name: Driver_930
```

Рисунок 4 – результат тестирования для N=1000

```
Enter license: Z5T88Z
Elapsed time: 1803 microseconds
Entry found:
    License number: Z5T88Z
    Car: Car_8828
    Driver name: Driver_8828
```

Рисунок 5 – результат тестирования для N=10000

Таблица тестирований (таблица 1)

N	Время (мкс)
100	171
1000	344
10 000	1803

Задание 3

Формулировка задачи:

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.

```
Функция ФибоначчиПоиск(массив, цель):
    n = ДлинаМассива(массив)
    FnMinus2 = 0
    FnMinus1 = 1
    Fn = FnMinus1 + FnMinus2

    Пока Fn < n:
        FnMinus2 = FnMinus1
        FnMinus1 = Fn
        Fn = FnMinus1 + FnMinus2

    offset = -1

    Пока Fn > 1:
        i = Минимум(offset + FnMinus2, n - 1)

        Если массив[i] < цель:
            Fn = FnMinus1
            FnMinus1 = FnMinus2
            FnMinus2 = Fn - FnMinus1
            offset = i
        Иначе Если массив[i] > цель:
            Fn = FnMinus2
            FnMinus1 = FnMinus1 - FnMinus2
            FnMinus2 = Fn - FnMinus1
        Иначе:
            Вернуть i // Элемент найден
    Если FnMinus1 == 1 И массив[offset + 1] == цель:
        Вернуть offset + 1
    Вернуть -1 // Элемент не найден
```

Листинг 4 – псевдокод поиска Фибоначчи

```
struct fileKey
{
    std::string key;
    size_t offset;
};

int fib(const std::vector<fileKey>& arr, std::string key)
{
    int size = arr.size();

    int n_prev_prev = 0;
    int n_prev = 1;
    int n = n_prev + n_prev_prev;
    while (n < size)
    {
        n_prev_prev = n_prev;
        n_prev = n;
        n = n_prev + n_prev_prev;
    }
    int offset = -1;
    while (n > 1)
    {
```

```

        int i = std::min(offset + n_prev_prev, size - 1);
        if (arr[i].key < key)
        {
            n = n_prev;
            n_prev = n_prev_prev;
            n_prev_prev = n - n_prev;
            offset = i;
        }
        else if (arr[i].key > key)
        {
            n = n_prev_prev;
            n_prev = n_prev - n_prev_prev;
            n_prev_prev = n - n_prev;
        }
        else
        {
            return arr[i].offset;
        }
    }
    if (n_prev == 1 && arr[offset + 1].key == key)
    {
        return arr[offset + 1].offset;
    }
    return -1;
}

int bin_search_three()
{
    FIO bin_fio("data.bin", "", "bin");

    carDriver driver;
    std::vector<fileKey> keyTable;
    size_t offset = 0;

    bool was_found = false;

    std::string license_number = io.input<std::string>("Enter license: ");

    while (bin_fio.read(reinterpret_cast<char*>(&driver), sizeof(driver)))
    {
        fileKey key;
        key.key = driver.license_number;
        key.offset = offset;
        keyTable.push_back(key);
        offset++;
    }

    std::sort(keyTable.begin(), keyTable.end(), [](const fileKey& a, const
fileKey& b) {return a.key < b.key; });

    io.output("Table is ready");

    bin_fio.close();

    measure(
        offset = fib(keyTable, license_number);

        if (offset != -1)
            was_found = true;

        if (was_found)
        {
            bin_fio.open();
            bin_fio.move(offset * sizeof(carDriver));
            bin_fio.read(reinterpret_cast<char*>(&driver), sizeof(driver));

```



```

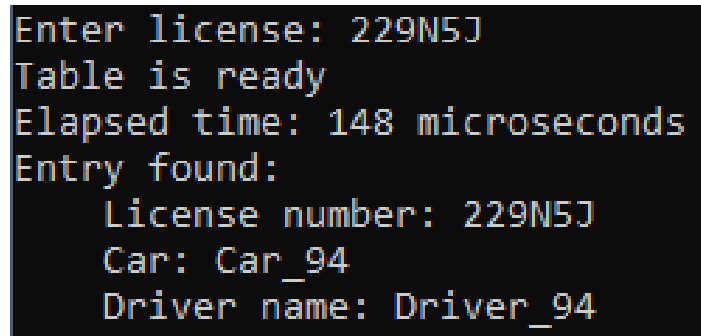
    },
    "Elapsed time: ");
if (was_found)
{
    io.output("Entry found: ");
    io.output("    License number: ", "");
    io.output(driver.license_number);
    io.output("    Car: ", "");
    io.output(driver.car_brand);
    io.output("    Driver name: ", "");
    io.output(driver.name);
}
else
    io.output("Entry wasn't found.");

bin_fio.close();

return 0;
}

```

Листинг 5 – код задачи 3

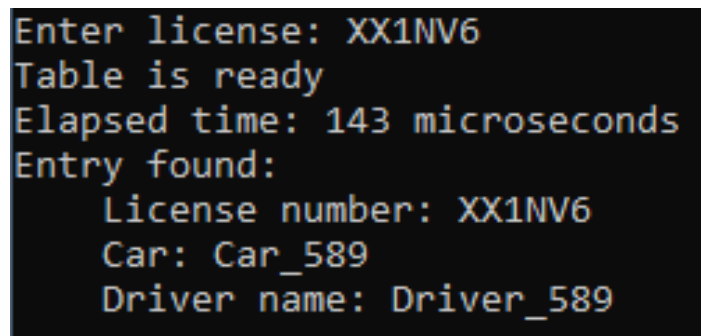


```

Enter license: 229N5J
Table is ready
Elapsed time: 148 microseconds
Entry found:
    License number: 229N5J
    Car: Car_94
    Driver name: Driver_94

```

Рисунок 6 – результат тестирования для N=100



```

Enter license: XX1NV6
Table is ready
Elapsed time: 143 microseconds
Entry found:
    License number: XX1NV6
    Car: Car_589
    Driver name: Driver_589

```

Рисунок 7 – результат тестирования для N=1000

```
Enter license: 0QBRY
Table is ready
Elapsed time: 153 microseconds
Entry found:
  License number: 0QBRY
  Car: Car_9761
  Driver name: Driver_9761
```

Рисунок 8 – результат тестирования для N=10 000

Таблица тестирований (таблица 2)

N	Время (мкс)
100	148
1000	143
10 000	153

Вывод

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