



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

---

---

Институт информационных технологий (ИИТ)  
Кафедра МОиСИТ

**ОТЧЕТ  
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5.2**

**«Алгоритмы поиска в таблице при работе с данными из файла»  
по дисциплине  
«Структуры и алгоритмы обработки данных»**

Выполнил студент группы ИКБО-10-24

Таганов А.А.

Практическую работу выполнил

«\_\_»\_\_\_\_ 2025 г.

«Зачтено»

«\_\_»\_\_\_\_ 2025 г.

Москва 2025

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

### Задание 1

Задача: создать двоичный файл из записей. Заполнить файл данными. Ключи записей в файле уникальны. Структура записи, заданная вариантом (№11), представляет собой железнодорожную справку, состоящую из номера поезда, пункта отправления, пункта назначения, времени отправления.

Листинг 1 – Структура TrainRecord

```
struct TrainRecord {
    uint32_t trainNum;
    char departurePoint[35];
    char destinationPoint[35];
    uint32_t departureTime;

    bool operator<(const TrainRecord& other) const {
        return trainNum < other.trainNum;
    }
};
```

Также в структуре переопределили оператор ‘<’ для корректной работы функции sort() для TrainRecord.

Для начала создается текстовый файл из записей, поля записи разделены символом ‘|’. Для записи неповторяющихся ключей в случайном порядке создаем вектор ключей, затем при помощи функции shuffle() перемешиваем их. Выбираем случайный город отправления из массива. Выбираем случайный город назначения. Цикл while гарантирует, что города не будут повторяться. Генерируем случайное время в минутах от начала суток. Затем записываем все в текстовый файл с разделителями.

Листинг 2 – Запись данных в текстовый файл

```
void createTextFile() {
    ofstream out("trains.txt");
    random_device rd;
    mt19937 gen(rd());

    vector<uint32_t> trainNumbers;
    for (uint32_t i = 1; i <= N; i++) {
        trainNumbers.push_back(i);
    }
    shuffle(trainNumbers.begin(), trainNumbers.end(), gen);
```

```

for (uint32_t i = 0; i < N; i++) {
    string depCity = CITIES[gen() % NUM_CITIES];
    string destCity = CITIES[gen() % NUM_CITIES];

    while (destCity == depCity) {
        destCity = CITIES[gen() % NUM_CITIES];
    }

    uint32_t time = gen() % 1440;

    out << trainNumbers[i] << " | " << depCity << " | " << destCity << " | " <<
time << endl;
}

out.close();
cout << "txt created with " << N << " records" << endl;
}

```

Доступные города для полей departurePoint и destinationPoint хранятся в константе.

### Листинг 3 - Константа

```

const string CITIES[] = {
    "Москва", "СПетербург", "Новосибирск", "Екатеринбург", "Казань",
    "ННовгород", "Челябинск", "Самара", "Омск", "РостовНадону",
    "Уфа", "Красноярск", "Воронеж", "Пермь", "Волгоград"
};

```

Для конвертации текстового файла в бинарный построчно читаем текстовый файл. В каждой строчке находим позиции разделителей. Берем подстроки и записываем их в соответствующие поля заранее созданной временной структуры. Затем добавляем готовую запись в заранее созданный вектор структур. Проходим по всем записям в векторе, преобразуем указатели на структуры в указатели на байты и записываем в бинарный файл. Также перед записью содержится закомментированная функция sort(). Она необходима для бинарного однородного поиска без использования дополнительной таблицы.

### Листинг 4 – Конвертация и запись в бинарный файл

```

void convertToBinary() {
    ifstream in("trains.txt");
    vector<TrainRecord> records;

    string line;
    while (getline(in, line)) {
        TrainRecord record;
        size_t pos1 = line.find('|');
        size_t pos2 = line.find('|', pos1 + 1);
        size_t pos3 = line.find('|', pos2 + 1);

```

```

    record.trainNum = stoul(line.substr(0, pos1));

    string depCity = line.substr(pos1 + 1, pos2 - pos1 - 1);
    string destCity = line.substr(pos2 + 1, pos3 - pos2 - 1);

    strcpy(record.departurePoint, depCity.c_str());
    strcpy(record.destinationPoint, destCity.c_str());

    record.departureTime = stoul(line.substr(pos3 + 1));

    records.push_back(record);
}
in.close();
// ПАКОММЕНТИРУЙ
sort(records.begin(), records.end());

ofstream out("trains.bin", ios::binary);
for (const auto& record : records) {
    out.write(reinterpret_cast<const
char*>(&record),
sizeof(TrainRecord));
}
out.close();

cout << "bin created" << endl;
}

```

Результаты работы функций для создания 100 записей представлены на Рис.1 и Рис.2

```

my.go          trains.txt
Файл   Изменить   Просмотр

|20|Москва|Омск|435
4|Самара|Уфа|216
9|РостовНадону|Екатеринбург|484
3|РостовНадону|Новосибирск|43
6|Челябинск|СанктПетербург|1136
63|Омск|Екатеринбург|364
21|Челябинск|Красноярск|674
91|Новгород|Омск|625
59|Челябинск|Волгоград|721
24|Новосибирск|Омск|1352
29|РостовНадону|Новосибирск|178
36|Новосибирск|Пермь|606
2|Волгоград|Челябинск|812
92|Омск|Самара|1122
38|Казань|Екатеринбург|1274
17|Челябинск|Пермь|490
86|РостовНадону|Екатеринбург|1305
61|Воронеж|РостовНадону|966
97|Воронеж|СанктПетербург|213
95|Самара|Волгоград|1416
34|Новосибирск|Челябинск|341
99|Москва|Воронеж|242
25|Новгород|Екатеринбург|768
37|Омск|Пермь|1325
83|Казань|Челябинск|1071
8|Новосибирск|Казань|524

```

*Rис.1 – Содержание текстового файла*

```

    Москва   Кэ У     Омск   s r+эс °хi(в i r+эс   и   Самара   Кэ У     Уфа   s r+эс °
    хi(в i r+эс   и   РостовНадону   Кэ У     Екатеринбург   эс °хi(в i r+эс   и   РостовНадону   Кэ У
    Новосибирск   эс °хi(в i r+эс   и   Челябинск ну   Кэ У     Следующий   эс °хi(в i r+эс   и   РостовНадону   Кэ У
    Кэ У     Екатеринбург   эс °хi(в i r+эс   и   Челябинск ну   Кэ У     Красноярск   эс °хi(в i r+эс   и   Уфы
    [ ННовгород ну   Кэ У     Омск оярск   эс °хi(в i r+эс   и   Челябинск ну   Кэ У     Волгоград   эс °
    хi(в i r+эс   и   Сибирь   Новосибирск   Кэ У     Омск град   эс °хi(в i r+эс   и   РостовНадону   Кэ У
    Новосибирск   эс °хi(в i r+эс   и   Новосибирск   Кэ У     Пермь бирск   эс °хi(в i r+эс   и   Волгоград к
    Кэ У     Челябинск   эс °хi(в i r+эс   и   Омск град к   Кэ У     Самара ск к   эс °хi(в i r+эс   и   Волгоград
    Казань ад к   Кэ У     Екатеринбург   эс °хi(в i r+эс   и   Челябинск к   Кэ У     Пермь инбург   эс °хi(в
    i r+эс   и   РостовНадону   Кэ У     Екатеринбург   эс °хi(в i r+эс   и   Воронеж Дону   Кэ У
    РостовНадону   эс °хi(в i r+эс   и   Воронеж Дону   Кэ У     Следующий   эс °хi(в i r+эс   и   Самара Дону
    Кэ У     Волгоград у   эс °хi(в i r+эс   и   Новосибирск   Кэ У     Челябинск у   эс °хi(в i r+эс   и   Уфа с
    Москва ирск   Кэ У     Воронеж к   у   эс °хi(в i r+эс   и   ННовгород к   Кэ У     Екатеринбург   эс °хi(в
    i r+эс   и   Омск ород к   Кэ У     Пермь инбург   эс °хi(в i r+эс   и   Казань од к   Кэ У
    Челябинск   эс °хi(в i r+эс   и   Новосибирск   Кэ У     Казань ск   эс °хi(в i r+эс   и
    Г Челябинск   Кэ У     РостовНадону   эс °хi(в i r+эс   и   Ф
    Омск инск   Кэ У     Следующий   эс °хi(в i r+эс   и   Новосибирск   Кэ У     Воронеж   эс °
    хi(в i r+эс   и   Омск ирск   Кэ У     Челябинск у   эс °хi(в i r+эс   и   О   РостовНадону   Кэ У
    Москва ск   у   эс °хi(в i r+эс   и
    - Пермь Надону   Кэ У     Екатеринбург   эс °хi(в i r+эс   и   Воронеж   эс °хi(в i r+эс   и   Казань нбург
    °хi(в i r+эс   и   Воронеж Дону   Кэ У     Волгоград   эс °хi(в i r+эс   и   Следующий   эс °хi(в
    Казань ад рг   эс °хi(в i r+эс   и   Воронеж   эс °хi(в i r+эс   и   Следующий   эс °хi(в
    Кэ У     Омск а ад рг   эс °хi(в i r+эс   и   Воронеж   эс °хi(в i r+эс   и   Следующий   эс °хi(в
    Новосибирск   Кэ У     РостовНадону   эс °хi(в i r+эс   и   Воронеж   эс °хi(в i r+эс   и   Следующий   эс °хi(в
    i r+эс   и   Красноярск   Кэ У     Волгоград   эс °хi(в i r+эс   и   Уфа сибирск   Кэ У     Волгоград   эс °хi(в
    Новосибирск   эс °хi(в i r+эс   и   Следующий   эс °хi(в i r+эс   и   Омск оярск   Кэ У     Волгоград   эс °хi(в
    Казань ск   эс °хi(в i r+эс   и   Воронеж   эс °хi(в i r+эс   и   ННовгород
  
```

*Rus.2 – Содержание бинарного файла*

## Задание 2

Задача: реализовать поиск в файле с применением линейного поиска.

Для выполнения этого задания была реализована функция линейного поиска. В ней мы открываем файл для чтения в бинарном режиме и создаем времененную структуру для хранения текущей записи. Мы преобразуем структуру в указатель на байты и читаем размер одной структуры. Сравниваем поле ключа с искомым ключом.

*Листинг 5 – Функция для линейного поиска*

```

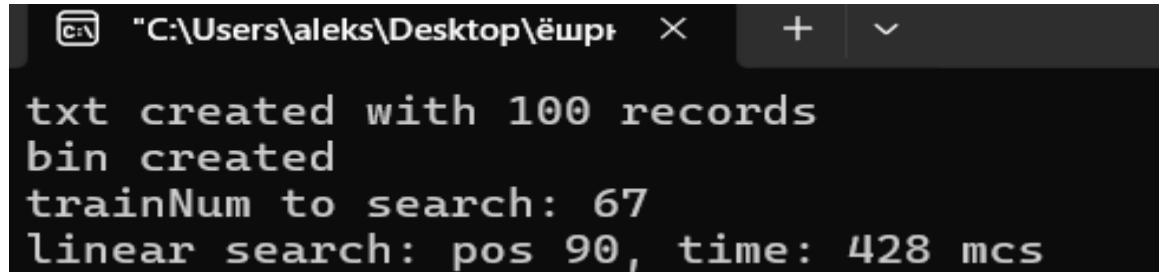
int linearSearch(uint32_t trainNumber) {
    ifstream file("trains.bin", ios::binary);
    if (!file.is_open()) return -1;

    TrainRecord record;
    int position = 0;

    while (file.read(reinterpret_cast<char*>(&record), sizeof(TrainRecord)))
    {
        if (record.trainNum == trainNumber) {
            file.close();
            return position;
        }
        position++;
    }
}
  
```

```
    file.close();
    return -1;
}
```

Результат тестирования программы для 100 записей представлен на Рис.3



```
C:\Users\aleks\Desktop\ёшри > + | v
txt created with 100 records
bin created
trainNum to search: 67
linear search: pos 90, time: 428 mcs
```

*Рис.3 – Результат тестирования для 100 записей*

Результаты тестирования для 100, 1000 и 10000 записей представлены в таблице.

Количество записей	Время поиска(мкс)
100	369
1000	371
10000	958

### Задание 3

Задача: поиск записи в файле с применением бинарного однородного поиска без использования дополнительной таблицы.

В алгоритме бинарного поиска мы делим последовательность пополам и сравниваем искомый элемент с серединой. Если он больше, то продолжаем в правой части. Если меньше, то в левой. При реализации бинарного поиска без дополнительной таблицы, мы работаем напрямую с файлом. Поэтому приходиться использовать функцию для перемещения на определенную позицию для чтения файла. Реализация алгоритма представлена в листинге 6.

#### Листинг 6 – Бинарный поиск без дополнительной таблицы

```
int binarySearch(uint32_t trainNumber) {
    ifstream file("trains.bin", ios::binary);
```

```

if (!file.is_open()) return -1;

int left = 0;
int right = N - 1;

while (left <= right) {
    int mid = left + (right - left) / 2;

    file.seekg(mid * sizeof(TrainRecord));
    TrainRecord record;
    file.read(reinterpret_cast<char*>(&record), sizeof(TrainRecord));

    if (record.trainNum == trainNumber) {
        file.close();
        return mid;
    } else if (record.trainNum < trainNumber) {
        left = mid + 1;
    } else {
        right = mid - 1;
    }
}

file.close();
return -1;
}

```

Результат тестирования программы для 100 записей представлен на Рис.4

```

txt created with 100 records
bin created
trainNum to search: 67
binary serach: pos 66, time: 352 mcs

```

*Рис.4 – Результаты тестирования для 100 записей*

Результаты тестирования для 100, 1000 и 10000 записей представлены в таблице.

Количество записей	Время поиска(мкс)
100	352
1000	330
10000	306

### **Выход**

Поставленные задачи выполнены: реализованы функции поиска в массиве данных по ключу, кодирования данных в текстовый или бинарный файл и чтение из них.

## **Литература.**

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
2. Документация по языку C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ruru/cpp/cpp/> (дата обращения 31.09.2021).
3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=4020> (дата обращения 31.09.2021).