



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение высшего  
образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

---

Институт информационных технологий (ИТ)  
Кафедра математического обеспечения и стандартизации информационных  
технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ 5\_2**  
**по дисциплине**  
**«Структуры и алгоритмы обработки данных»**

Тема. Внешние структуры данных: текстовый и двоичный файлы.

Выполнил студент группы ИКБО-60-23

Шеенко В.А

Принял старший преподаватель

Скворцова Л.А.

Москва 2024

## СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ .....	4
2. ЗАДАНИЕ №1 .....	5
2.1 Постановка задания.....	5
2.2 Содержимое текстового файла.....	6
2.3 Разработка функций для работы с текстовым файлом .....	6
2.3.1 Функции открытия и файловых потоков.....	6
2.3.2 Вывод содержимого файлового текста.....	7
2.3.3 Добавление записи в конец файла.....	7
2.3.4 Получение числа по его порядковому номеру.....	8
2.3.5 Получение количества чисел в файле .....	8
2.3.6 Решение задания индивидуального варианта .....	9
2.4 Разработка основной программы .....	11
2.5 Тестирование .....	14
3 ЗАДАНИЕ 2 .....	19
3.1 Постановка задачи.....	19
3.2 Содержимое текстового файла .....	20
3.3 Структура записи .....	20
3.4 Реализация функций программы.....	22
3.4.1 Преобразование текстовых данных в двоичный вид .....	22
3.4.2 Сохранение данных двоичного файла в текстовом .....	24
3.4.3 Вывод всех записей двоичного файла .....	25
3.4.4 Доступ к записи по ее порядковому номеру .....	25
3.4.5 Удаление записи с заданным значением ключа .....	26

3.4.6 Список нарушений по автомобилю заданного номера .....	27
3.4.7 Увеличение суммы штрафа.....	28
3.5 Разработка основной программы .....	28
3.6 Тестирование программы.....	31
ВЫВОД.....	38
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ .....	39

## 1 ЦЕЛЬ РАБОТЫ

Получить навыки применения файловых потоков языка C++ (или файлов языка Си) по управлению текстовым и двоичным файлами.

**Задание 1.** Разработать программу, управления текстовым файлом.

Дополнительные операции в соответствии с индивидуальным вариантом (28):

Создать два новых файла из значений исходного, переписав в один из них первую половину чисел исходного, а в другой, оставшуюся часть. В исходный файл слить данные их двух новых файлов упорядоченными по возрастанию парами, т.е. прочитать первые числа двух файлов, сначала в исходный файл записать меньшее из них, а за ним большее.

**Задание 2.** Разработать программу управление двоичными файлами с записями фиксированной длины.

Структура записи и дополнительные операции в соответствии с индивидуальным вариантом 28, таблица 1.

Таблица 1 – Задания индивидуального варианта

Структура записи	Учет нарушений ПДД. Структура записи о нарушении ПДД: <u>номер автомобиля</u> , фамилия и инициалы владельца, модель, дата нарушения, место нарушения (текстом), статья (КоАП), наказание (сумма штрафа).
Доп. операция	1. Сформировать список нарушений по автомобилю заданного номера. Результат сохранить в новом двоичном файле с той же структурой записи, что и исходный файл. 2. Увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье

## **2. ЗАДАНИЕ №1**

### **2.1 Постановка задания**

Разработать программу, управления текстовым файлом.

Требования по выполнению

1. Создать текстовый файл средствами текстового редактора кодировки ASCII, содержащего десятичные числа по несколько чисел на строке. Количество чисел на разных строках можно отличаться.

2. Реализация ввода-вывода на основе файловых потоков C++: ofstream, ifstream.

3. Имя физического файла вводится пользователем и передается в функции обработки через параметр.

4. При открытии файла выполнять контроль его существования и открытия.

5. Разработать функции для выполнения операций над текстовым файлом.

1) вывод содержимого текстового файла на экран;

2) добавление новой записи в конец файла;

3) прочесть значение числа, указав его порядковый номер в файле, и вернуть его значение при успешном выполнении и код завершения если номер превышает количество чисел в файле;

4) определить количество чисел в файле.

6. Разработать программу и выполнить тестирование всех функций. Программа должно содержать диалоговый интерфейс на основе текстового меню.

7. Контроль открытия и существования файла выполнить в основной программе перед вызовом функции. Перед закрытием файла, проверить отсутствие ошибок ввода и вывода (метод good).

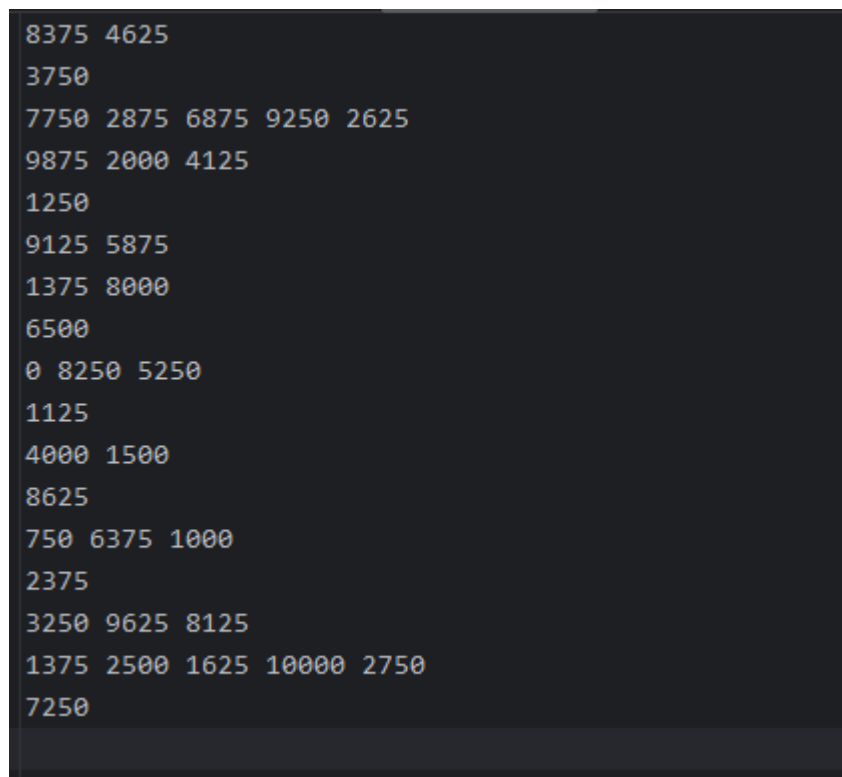
8. Создать файл заголовка и перенести в него все отлаженные функции. Исключить функции из основной программы. Отладить приложение, подключив к нему модуль с функциями.

9. Разработать функции для реализации дополнительных операций, определенных вариантом и сохранить их в модуле с остальными функциями.

10. Выполнить тестирование приложения в полном объеме.

## **2.2 Содержимое текстового файла**

Исходное содержимое текстового файла `z1_text`, требуемого для задания, представлено на рис. 1



```
8375 4625
3750
7750 2875 6875 9250 2625
9875 2000 4125
1250
9125 5875
1375 8000
6500
0 8250 5250
1125
4000 1500
8625
750 6375 1000
2375
3250 9625 8125
1375 2500 1625 10000 2750
7250
```

Рисунок 1 – Содержимое текстового файла `z1_text`

## **2.3 Разработка функций для работы с текстовым файлом**

### **2.3.1 Функции открытия и файловых потоков**

Для работы с файлами необходимо использовать файловые потоки `ifstream` и `ofstream`, ввода и вывода соответственно. Для удобного использования этих потоков реализуем функции их открытия на ввод и вывод (рис. 2).

```

void input_open (std::ifstream& in_file, const std::string& file_name) {
    in_file = std::ifstream(s: file_name);

    if (!in_file.is_open()) {
        std::cerr << "File not opened\n";
        exit( Code: 1);
    }
}

void output_open (std::ofstream& out_file, const std::string& file_name, std::ios::openmode mode) {
    out_file = std::ofstream(s: file_name, mode);

    if (!out_file.is_open()) {
        std::cerr << "File not opened\n";
        exit( Code: 1);
    }
}

```

Рисунок 2 – функции открытия файловых потоков для ввода и вывода

В случае открытия файлового потока для вывода передаются в качестве аргументов не только поток и имя файла, но и режим открытия (mode).

### 2.3.2 Вывод содержимого файлового текста

Реализация вывода текстового файла на экран представлена на рис. 3.

```

void PrintFileContent(std::ifstream& file) {
    std::string s;
    while (std::getline( &: file, &: s)) {
        std::cout << s << '\n';
    }
}

```

Рисунок 3 – Функция вывод содержимого текстового файла на экран

В качестве аргумента передается ссылка на файловый поток ввода (file).

### 2.3.3 Добавление записи в конец файла

Добавление записи в конец файла можно реализовать различными способами, один из которых предоставлен на рис. 4.

```

void AppendContentToFile(std::ofstream& file, const std::string& s_content) {
    file << s_content << '\n';
}

```

Рисунок 4 – Функция добавления записи в конец файла

Для использования этой функции необходимо передавать в нее ссылку на файловый поток вывода, открытый с режимом `std::ios::app`, который открывает файл на «дозапись», то есть файл открывается, не удаляя все содержимое и записывая все в конец файла.

#### 2.3.4 Получение числа по его порядковому номеру

Функция, отвечающая за получение числа по порядковому номеру, показана на рис. 6.

```
int GetNumByOrdinal(std::ifstream& file, int ordinal_num, bool& SUCCESS_CODE) {
    int num;
    int cur_num = 0;
    while (file >> num) {
        cur_num++;

        if (cur_num == ordinal_num) {
            file.close();
            return num;
        }
    }

    SUCCESS_CODE = false;
    return SUCCESS_CODE;
}
```

Рисунок 6 – Функция получения числа по порядковому номеру

Кроме ссылки на файловый поток и целочисленного значения порядкового номера передается так же ссылка на булево значение, отвечающее за успешное выполнение функции.

#### 2.3.5 Получение количества чисел в файле

На рис. 7 показана реализация функции подсчета чисел в файле.



```
int CountNum(std::ifstream& file) {  
    int num;  
    int count = 0;  
    while (file >> num) {  
        count++;  
    }  
  
    return count;  
}
```

Рисунок 7 – Функция подсчета чисел в файле

### 2.3.6 Решение задания индивидуального варианта

**Задание.** Создать два новых файла из значений исходного, переписав в один из них первую половину чисел исходного, а в другой, оставшуюся часть. В исходный файл слить данные их двух новых файлов упорядоченными по возрастанию парами, т.е. прочитать первые числа двух файлов, сначала в исходный файл записать меньшее из них, а за ним большее.

Для решения этой задачи можно реализовать две функции, одна из которых разбивает исходный файл на два, а второй, наоборот, склеивает их в исходный.

Функция разбиения файла представлена на рис. 8.

```

void Partition(std::ifstream& in_file) {
    int count = CountNum( &: in_file);

    std::ofstream out_a_file( s: "a_half.txt");
    std::ofstream out_b_file( s: "b_half.txt");

    if (!out_a_file.is_open() || !out_b_file.is_open()) {
        std::cerr << "File not found\n";
        return;
    }

    int k = 0;
    int num;
    int half = count / 2;

    in_file.clear();
    in_file.seekg(0, std::ios::beg);
    while (in_file >> num) {
        if (k < half)
            out_a_file << num << '\n';
        else
            out_b_file << num << '\n';
        k++;
    }

    out_a_file.close();
    out_b_file.close();
}

```

Рисунок 8 – Функция разбиения текстового файла

Функция, которая упорядочивает пары чисел, используя два новых файла, полученных с помощью функции выше, показана на рис. 9.

```

void Merge(std::ofstream& out_file) {
    std::ifstream in_a_file( s: "a_half.txt");
    std::ifstream in_b_file( s: "b_half.txt");

    if (!in_b_file.is_open() || !out_file.is_open()) {
        std::cerr << "File not found\n";
        return;
    }

    std::string s_a;
    std::string s_b;

    bool a_no_eof = (bool)std::getline( &: in_a_file, &: s_a);
    bool b_no_eof = (bool)std::getline( &: in_b_file, &: s_b);

    while (a_no_eof || b_no_eof) {
        int a = a_no_eof ? std::stoi( str: s_a) : -1;
        int b = b_no_eof ? std::stoi( str: s_b) : -1;

        if (a_no_eof && b_no_eof) {
            if (a > b)
                out_file << b << '\n' << a << '\n';
            else
                out_file << a << '\n' << b << '\n';
            a_no_eof = (bool)std::getline( &: in_a_file, &: s_a);
            b_no_eof = (bool)std::getline( &: in_b_file, &: s_b);
        } else if (a_no_eof) {
            out_file << a << '\n';
            a_no_eof = (bool)std::getline( &: in_a_file, &: s_a);
        } else {
            out_file << b << '\n';
            b_no_eof = (bool)std::getline( &: in_b_file, &: s_b);
        }
    }

    in_b_file.close();
    in_a_file.close();
}

```

Рисунок 9 – Функция слияния

## 2.4 Разработка основной программы

Для работы с файлом необходимо знать имя файла, которое передается в программу с помощью ввода пользователя, кроме того, необходимо убедиться в существовании этого файла (рис. 10).

```
std::string file_name;  
std::cout << "Enter the name of the file: ";  
std::cin >> file_name;  
  
if (!(std::filesystem::exists(p: file_name))) {  
    std::cerr << "File not found\n";  
    return 1;  
}
```

Рисунок 10 – Получение имени и проверка существования файла

Перед закрытием каждого файлового потока в основной программе выводится его состояние ошибок с помощью `std::ios::good`. Реализация диалогового интерфейса представлена на рисунках 11-12.

```

std::cout << "Enter the mode: ";
std::cin >> mode;
while (mode != 6) {
    switch (mode) {
        case 1: {
            std::ifstream in_file;
            input_open( & in_file, file_name);
            PrintFileContent( & in_file);

            std::cout << "std::ios::good ==> " << in_file.good() << '\n';
            in_file.close();
            break;
        }
        case 2: {
            std::ofstream out_file;
            output_open( & out_file, file_name, mode: std::ios::app);

            std::string num;
            std::cout << "Enter the number: ";
            std::cin >> num;

            AppendContentToFile( & out_file, s_content: num);

            std::cout << "std::ios::good ==>" << out_file.good() << '\n';
            out_file.close();
            break;
        }
        case 3: {
            int ordinal_num;
            std::cout << "Enter the ordinal number: ";
            std::cin >> ordinal_num;

            std::ifstream in_file;
            input_open( & in_file, file_name);

            bool SUCCESS_CODE = true;
            int res = GetNumByOrdinal( & in_file, ordinal_num, & SUCCESS_CODE);

```

Рисунок 11 – реализация диалогового интерфейса, часть 1

```

        if (SUCCESS_CODE) {
            std::cout << "Number: " << res << '\n';
        } else {
            std::cerr << "Number not found\n";
        }

        std::cout << "std::ios::good ==>" << in_file.good() << '\n';
        in_file.close();
        break;
    }
    case 4: {
        std::ifstream in_file;
        input_open( & in_file, file_name);

        std::cout << "Count: " << CountNum( & in_file) << '\n';
        std::cout << "std::ios::good ==>" << in_file.good() << '\n';
        in_file.close();
        break;
    }
    case 5: {
        std::ifstream in_file;
        input_open( & in_file, file_name);
        Partition( & in_file);
        std::cout << "1, std::ios::good ==>" << in_file.good() << '\n';
        in_file.close();

        std::ofstream out_file;
        output_open( & out_file, file_name, mode: std::ios::out);
        Merge( & out_file);
        std::cout << "2, std::ios::good ==>" << out_file.good() << '\n';
        out_file.close();
        break;
    }
}
std::cout << "\nEnter the mode: ";
std::cin >> mode;

```

Рисунок 12 – реализация диалогового интерфейса, часть 2

## 2.5 Тестирование

Перед каждым тестом содержимое исходно файла соответствует рис. 1.

Тестирование функций вставки записи в конец файла и вывода содержимого файла на экран (рис. 13)

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\code.exe
3. Get number by ordinal
4. Count numbers
5. Partition and merge content
6. Exit
Enter the mode: 1
8375 4625
3750
7750 2875 6875 9250 2625
9875 2000 4125
1250
9125 5875
1375 8000
6500
0 8250 5250
1125
4000 1500
8625
750 6375 1000
2375
3250 9625 8125
1375 2500 1625 10000 2750
7250
std::ios::good ==> 0

Enter the mode: 2
Enter the number: 123456789
std::ios::good ==>1

Enter the mode: 1
8375 4625
3750
7750 2875 6875 9250 2625
9875 2000 4125
1250
9125 5875
1375 8000
6500
0 8250 5250
1125
4000 1500
8625
750 6375 1000
2375
3250 9625 8125
1375 2500 1625 10000 2750
7250
123456789
std::ios::good ==> 0

Enter the mode: _
```

Рисунок 13 – Тестирование функций вывода и добавления в конец  
Тестирование функции получения числа по порядковому номеру:

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\code.exe
Enter the name of the file: z1_text
1. Print all content
2. Append content
3. Get number by ordinal
4. Count numbers
5. Partition and merge content
6. Exit
Enter the mode: 3
Enter the ordinal number: 7
Number: 9250
std::ios::good ==>1

Enter the mode: 3
Enter the ordinal number: 300
Number not found
std::ios::good ==>0

Enter the mode:
```

Рисунок 14 – Получение числа по порядковому номеру

Тестирование подсчета чисел в файле:

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\code.exe
Enter the name of the file: z1_text
1. Print all content
2. Append content
3. Get number by ordinal
4. Count numbers
5. Partition and merge content
6. Exit
Enter the mode: 4
Count: 37
std::ios::good ==>0

Enter the mode: 4
```

Рисунок 15 – Получение количества чисел

Тестирование функций для индивидуального задания:



```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\code.exe
Enter the name of the file: z1_text
1. Print all content
2. Append content
3. Get number by ordinal
4. Count numbers
5. Partition and merge content
6. Exit
Enter the mode: 5
1, std::ios::good ==>0
2, std::ios::good ==>1

Enter the mode: 1
8250
8375
4625
5250
1125
3750
4000
7750
1500
2875
6875
8625
750
9250
2625
6375
1000
9875
2000
2375
3250
4125
1250
9625
8125
9125
1375
5875
1375
2500
1625
8000
6500
10000
0
2750
7250
std::ios::good ==> 0

Enter the mode: _
```

Рисунок 16 – Тестирование задания индивидуального варианта

Тестирование поведение программы при неправильном названии файла:

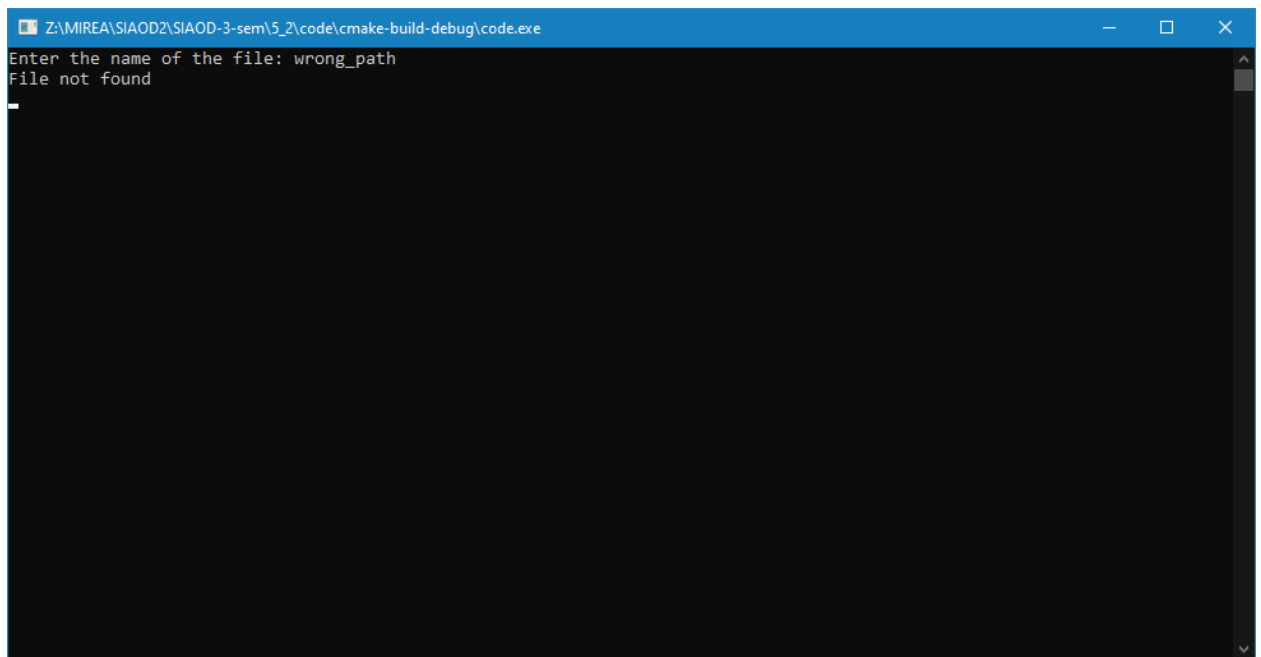


Рисунок 17 – Неправильное название файла

## **3 ЗАДАНИЕ 2**

### **3.1 Постановка задачи**

Разработать программу управление двоичными файлами с записями фиксированной длины. Файл состоит из записей определенной структуры, согласно варианту. Записи имеют ключ, уникальный в пределах файла.

Требования к подготовке и выполнению задания

1. Разработать структуру записи двоичного файла согласно варианту задания.

2. Подготовить тестовые данные в текстовом файле с кодировкой ASCII, в соответствии со структурой записи варианта. При открытии файла выполнить контроль его существования и открытия.

3. Имя файла вводит пользователь.

4. При открытии файла обеспечить контроль существования и открытия файла.

5. При применении механизма прямого доступа к записи файла выполнить контроль присутствия записи с заданным номером в файле.

6. Разработать функции для выполнения операций:

- преобразование тестовых данных из текстового файла в двоичный файл;

- сохранение данных двоичного файла в текстовом, так, чтобы используя их можно было восстановить двоичный файл;

- вывод всех записей двоичного файла;

- доступ к записи по ее порядковому номеру в файле, используя механизм прямого доступа к записи в двоичном файле;

- удаление записи с заданным значением ключа, выполнить путем замены на последнюю запись.

- манипулирование записями в двоичном файле согласно дополнительным операциям, определенным в варианте;

7. Сохраните функции в новом модуле.

8. Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.

9. Выполнить тестирование приложения, продемонстрировав выполнение всех операций.

Задание индивидуального варианта:

Структура записи	Учет нарушений ПДД. Структура записи о нарушении ПДД: <u>номер автомобиля</u> , фамилия и инициалы владельца, модель, дата нарушения, место нарушения (текстом), статья (КоАП), наказание (сумма штрафа).
Доп. операция	1. Сформировать список нарушений по автомобилю заданного номера. Результат сохранить в новом двоичном файле с той же структурой записи, что и исходный файл. 2. Увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье

### 3.2 Содержимое текстового файла

Исходное содержимое текстового файла z2\_text, требуемого для задания, представлено на рис. 18

```
1 A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.1;500;
2 B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
3 C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
4 D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
5 E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
6 F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Чайковского, d. 9;12.19 ch.1;1200;
7 G432NC77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
8 H654TM77;Vasiliev I.V.;Skoda Octavia;13.09.2024;ul. Tolstogo, d. 10;12.9 ch.2;2500;
9 K987RP77;Nikolaev S.S.;Renault Duster;17.09.2024;ul. Chekhova, d. 15;12.5 ch.2;1800;
10 L123BC77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;
```

Рисунок 18 – Содержимое текстового файла z2\_text

### 3.3 Структура записи

Запись представляет собой данные о нарушении ПДД, а именно:

1. Номер автомобиля - char carNumber[16]
2. Имя владельца автомобиля – char name[32]

3. Модель автомобиля – char model[32]
4. Дата нарушения – char date[32]
5. Место нарушения – char place[32]
6. Статья нарушения – char article[32]
7. Сумма штрафа – int fine

Кроме полей структуры так же присутствуют вспомогательные методы, такие как ToString (преобразует все поля структуры в строку определенного формата) и SetFieldByStr (извлекает из строки определенного формата значения полей структуры).

```
struct Violation {
    char carNumber[16];
    char name[32];
    char model[32];
    char data[32];
    char place[32];
    char article[32];
    short fine;

    void SetFieldsByStr(const std::string& s) {
        std::vector<std::string> tokens = Split(s, separator: ";");

        if (tokens.size() != 7) {
            std::cerr << "Reading Error" << '\n';
            return;
        }

        strcpy( Dest: carNumber, Source: tokens[0].c_str());
        strcpy( Dest: name, Source: tokens[1].c_str());
        strcpy( Dest: model, Source: tokens[2].c_str());
        strcpy( Dest: data, Source: tokens[3].c_str());
        strcpy( Dest: place, Source: tokens[4].c_str());
        strcpy( Dest: article, Source: tokens[5].c_str());
        fine = std::stoi( str: tokens[6]);
    }

    std::string ToString() {
        std::string res;

        res = carNumber + std::string( S: ";" ) + name + std::string( S: ";" ) + model + std::string( S: ";" ) +
            data + std::string( S: ";" ) + place + std::string( S: ";" ) + article + std::string( S: ";" ) +
            std::to_string( val: fine ) + std::string( S: ";" );

        return res;
    }
};
```

Рисунок 19 – Определение структуры и реализация ее методов

### 3.4 Реализация функций программы

#### 3.4.1 Преобразование текстовых данных в двоичный вид

В функцию передаются имена исходного текстового файла и нового бинарного файла (если он не существует, то будет создан). Реализация функции показана на рис. 20.

```
void TextToBin(const std::string& file_name, const std::string& bin_file_name) {
    std::ifstream file(s: file_name);
    std::ofstream bin_file(s: bin_file_name, mode: std::ios::binary);

    if (!file.is_open() || !bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    std::string s;
    Violation violation;

    while (std::getline(& file, & s)) {
        violation.SetFieldsByStr(s);
        bin_file.write(s: (char*)&violation, n: sizeof(Violation));
    }

    file.close();
    bin_file.close();
}
```

Рисунок 20 – Функция преобразования текстового файла

После выполнения данной функции будет создан бинарный файл, содержание (рис. 21) которого можно вывести на экран с помощью вспомогательной функции (рис. 22).

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
41 31 32 33 42 43 37 37 00 bf 9d e2 9f 02 00 00
49 76 61 6e 6f 76 20 49 2e 49 2e 00 00 00 00 00
90 bf 9d e2 9f 02 00 00 83 01 00 00 fd 7f 00 00
54 6f 79 6f 74 61 20 43 61 6d 72 79 00 00 00 00
50 01 9d e2 9f 02 00 00 10 d1 26 e7 fd 7f 00 00
31 35 2e 30 39 2e 32 30 32 34 00 00 00 00 00 00
90 bf 9d e2 9f 02 00 00 f0 f7 3f 8f e4 00 00 00
75 6c 2e 20 4c 65 6e 69 6e 61 2c 20 64 2e 20 31
32 00 00 00 00 00 00 00 4c df 17 e7 fd 7f 00 00
31 32 2e 39 20 63 68 2e 31 00 00 00 00 00 00
50 01 9d e2 9f 02 00 00 bb 01 00 00 00 00 00
f4 01 42 34 35 36 4f 52 37 37 00 bf 9d e2 9f 02
00 00 50 65 74 72 6f 76 20 50 2e 50 2e 00 00 00
00 00 90 bf 9d e2 9f 02 00 00 83 01 00 00 fd 7f
00 00 48 79 75 6e 64 61 69 20 53 6f 6c 61 72 69
73 00 50 01 9d e2 9f 02 00 00 10 d1 26 e7 fd 7f
00 00 31 38 2e 30 39 2e 32 30 32 34 00 00 00 00
00 00 90 bf 9d e2 9f 02 00 00 f0 f7 3f 8f e4 00
00 00 70 72 2e 20 4d 69 72 61 2c 20 64 2e 20 38
00 31 32 00 00 00 00 00 4c df 17 e7 fd 7f
00 00 31 32 2e 31 36 20 63 68 2e 33 00 00 00 00
00 00 50 01 9d e2 9f 02 00 00 bb 01 00 00 00 00
00 00 e8 03 43 37 38 39 4b 4e 37 37 00 bf 9d e2
9f 02 00 00 53 6d 69 72 6e 6f 76 20 41 2e 41 2e
00 00 00 00 90 bf 9d e2 9f 02 00 00 83 01 00 00
fd 7f 00 00 4b 69 61 20 52 69 6f 00 53 6f 6c 61
72 69 73 00 50 01 9d e2 9f 02 00 00 10 d1 26 e7
fd 7f 00 00 32 30 2e 30 39 2e 32 30 32 34 00 00
00 00 00 00 90 bf 9d e2 9f 02 00 00 f0 f7 3f 8f
e4 00 00 00 75 6c 2e 20 50 75 73 68 6b 69 6e 61
2c 20 64 2e 20 35 00 00 00 00 00 00 4c df 17 e7
fd 7f 00 00 31 32 2e 35 20 63 68 2e 31 00 00 00
00 00 00 00 50 01 9d e2 9f 02 00 00 bb 01 00 00
00 00 00 00 dc 05 44 33 32 31 4f 52 39 39 00 bf
9d e2 9f 02 00 00 53 69 64 6f 72 6f 76 20 56 2e
56 2e 00 00 00 00 90 bf 9d e2 9f 02 00 00 83 01
00 00 fd 7f 00 00 46 6f 72 64 20 46 6f 63 75 73
00 61 72 69 73 00 50 01 9d e2 9f 02 00 00 10 d1
26 e7 fd 7f 00 00 31 32 2e 30 39 2e 32 30 32 34
00 00 00 00 00 00 90 bf 9d e2 9f 02 00 00 f0 f7
3f 8f e4 00 00 00 75 6c 2e 20 47 61 67 61 72 69
6e 61 2c 20 64 2e 20 37 00 00 00 00 00 00 4c df
17 e7 fd 7f 00 00 31 32 2e 38 20 63 68 2e 32 00
00 00 00 00 00 00 50 01 9d e2 9f 02 00 00 bb 01
00 00 00 00 00 00 d0 07 45 36 35 34 54 52 37 37
00 bf 9d e2 9f 02 00 00 4b 75 7a 6e 65 74 73 6f
76 20 4e 2e 4e 2e 00 00 90 bf 9d e2 9f 02 00 00
83 01 68 00 fd 7f 00 00 4e 69 73 73 61 6e 20 51
61 73 68 71 61 69 00 00 50 01 9d e2 9f 02 00 00
10 d1 26 e7 fd 7f 00 00 31 34 2e 30 39 2e 32 30
32 34 00 00 00 00 00 00 90 bf 9d e2 9f 02 00 00
f0 f7 3f 8f e4 00 00 00 70 72 2e 20 4c 6f 6d 6f
6e 6f 73 6f 76 61 2c 20 64 2e 20 32 32 00 00 00
4c df 17 e7 fd 7f 00 00 31 32 2e 31 35 20 63 68
2e 34 00 00 00 00 00 00 50 01 9d e2 9f 02 00 00
```

Рисунок 21 – Представление текстового файла z2\_text в шестнадцатеричном формате

```

void PrintBinFileInHex(const std::string& bin_file_name) {
    std::ifstream bin_file( s: bin_file_name, mode: std::ios::binary);

    if (!bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    char c;
    int k = 0;
    while (bin_file.get( &c)) {
        std::cout << std::hex << std::setw( n: 2) << std::setfill( c: '0')
                    << (static_cast<unsigned int>(c) & 0xFF)
                    << ' ';
        k++;

        if (k % 16 == 0)
            std::cout << '\n';
    }

    bin_file.close();
}

```

Рисунок 22 – Функция вывода содержания бинарного файла

### 3.4.2 Сохранение данных двоичного файла в текстовом

Для восстановления данных из бинарного файла (рис. 23).

```

void BinToText(const std::string& file_name, const std::string& bin_file_name) {
    std::ifstream bin_file( s: bin_file_name, mode: std::ios::binary);
    std::ofstream file( s: file_name);

    if (!bin_file.is_open() || !file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    Violation violation;

    while (bin_file.read( s: (char*)&violation, n: sizeof(Violation))) {
        file << violation.ToString() << '\n';
    }

    bin_file.close();
    file.close();
}

```

Рисунок 23 – Функция преобразования бинарного файла



### 3.4.3 Вывод всех записей двоичного файла

Функция вывода записей двоичного файла схожа с функцией преобразования бинарного файла в текстовый, за тем исключением, что данные выводятся в стандартный поток вывода (рис. 24).

```
void PrintAllViolationInBin(const std::string& bin_file_name) {
    std::ifstream bin_file(s: bin_file_name, mode: std::ios::binary);

    if (!bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    Violation violation;

    while (bin_file.read(s: (char*)&violation, n: sizeof(Violation))) {
        std::cout << violation.ToString() << '\n';
    }

    bin_file.close();
}
```

Рисунок 24 – Функция вывода всех записей из бинарного файла

### 3.4.4 Доступ к записи по ее порядковому номеру

В функцию дополнительно передается ссылка на булеву переменную, отвечающую за успешность завершения функции. Реализация данной функции представлена на рисунке 25.

```

Violation GetViolationByOrdinalNum(const std::string& bin_file_name, const size_t ordinal_num, bool& SUCCESS_CODE) {
    std::fstream bin_file(s: bin_file_name, mode: std::ios::binary | std::ios::in | std::ios::out | std::ios::ate);
    SUCCESS_CODE = false;

    if (!bin_file.is_open()) {
        return {};
    }

    size_t file_size = fs::file_size(p: bin_file_name);
    bin_file.seekg(0, std::ios::beg);
    int violations_count = file_size / sizeof(Violation);

    if (ordinal_num > violations_count - 1 || ordinal_num < 0) {
        return {};
    }

    std::streampos record_pos = ordinal_num * sizeof(Violation);
    bin_file.seekg(record_pos, std::ios::beg);

    Violation target_violation;
    bin_file.read(s: (char*)&target_violation, n: sizeof(Violation));

    SUCCESS_CODE = true;
    return target_violation;
}

```

Рисунок 25 – Функция доступа к записи по ее порядковому номеру

### 3.4.5 Удаление записи с заданным значением ключа

Удаление необходимо выполнить путем замены на последнюю запись, после чего нужно обрезать файл. Код функции показан на рис. 26.

```

void DeleteViolationByCarNum(const std::string& bin_file_name, const std::string& car_num) {
    std::fstream bin_file( s: bin_file_name, mode: std::ios::binary | std::ios::in | std::ios::out | std::ios::ate);
    if (!bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    size_t file_size = fs::file_size( p: bin_file_name);
    bin_file.seekg(0, std::ios::beg);

    int violations_count = file_size / sizeof(Violation);

    std::streampos last_record_pos = (violations_count - 1) * sizeof(Violation);
    Violation last_violation;
    bin_file.seekg(last_record_pos);
    bin_file.read( s: (char*)&last_violation, n: sizeof(Violation));

    bool deleted = false;
    std::streampos cur_pos = std::ios::beg;
    Violation violation;

    bin_file.seekp(std::ios::beg);
    while (cur_pos != file_size) {
        bin_file.read( s: (char*)&violation, n: sizeof(Violation));

        if (strcmp(violation.carNumber, car_num.c_str()) == 0 && !deleted) {
            bin_file.seekp(cur_pos);
            bin_file.write( s: (char*)&last_violation, n: sizeof(Violation));
            deleted = true;
        }

        cur_pos = bin_file.tellg();
    }

    if (!deleted)
        std::cerr << "Record not found" << '\n';

    bin_file.close();
    fs::resize_file( p: bin_file_name, size: file_size - sizeof(Violation));
}

```

Рисунок 27 – Функция удаления записи по ключу

Сначала определяется количество записей в файле, с помощью общего размера (в байтах) файла и размера (в байтах) одной записи, благодаря чему несложно найти позицию последней записи. Далее остается найти позицию необходимой записи и заменить ее на последнюю. В самом конце размер файла уменьшается на размер одной записи.

### 3.4.6 Список нарушений по автомобилю заданного номера

Реализация функции выборки записей по ключу продемонстрирована на рисунке 28.

```

void SelectionByCarNum(const std::string& bin_file_name, const std::string& new_file_name, const std::string& car_num) {
    std::ifstream bin_file( S: bin_file_name, mode: std::ios::binary);
    std::ofstream new_bin_file( S: new_file_name, mode: std::ios::binary);

    if (!bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    Violation violation;

    while (bin_file.read( S: (char*)&violation, n: sizeof(Violation))) {
        if (strcmp(violation.carNumber, car_num.c_str()) == 0)
            new_bin_file << violation.ToString() << '\n';
    }

    bin_file.close();
}

```

Рисунок 28 – Функция создания выборки по ключу

### 3.4.7 Увеличение суммы штрафа

Необходимо увеличить сумму штрафа вдвое по всем авто за указанную дату и по заданной статье. Код данной функции показан на рис. 29.

```

void DoubleFine(const std::string& bin_file_name, const std::string& start_date,
               const std::string& end_date, const std::string& article) {

    std::fstream bin_file( S: bin_file_name, mode: std::ios::binary | std::ios::in | std::ios::out);

    if (!bin_file.is_open()) {
        std::cerr << "Error opening file" << '\n';
        return;
    }

    Violation violation;

    while (bin_file.read( S: (char*)&violation, n: sizeof(Violation))) {
        if (violation.data >= start_date && violation.data <= end_date && strcmp(violation.article, article.c_str()) == 0) {
            violation.fine *= 2;
            bin_file.seekp(-sizeof(Violation), std::ios::cur);
            bin_file.write( S: (char *) &violation, n: sizeof(Violation));
        }
    }

    bin_file.close();
}

```

Рисунок 29 – Функция увеличения суммы штрафа

## 3.5 Разработка основной программы

Для работы с файлом необходимо знать имя файла, которое передается в программу с помощью ввода пользователя. Кроме того, должен быть реализован диалоговый интерфейс. Отрывок кода, отвечающий за данные аспекты, представлен на рисунке 30.

```
std::string bin_file_name;
std::string text_file_name;

std::cout << "Enter the name of the binary file: ";
std::cin >> bin_file_name;
std::cout << "Enter the name of the text file: ";
std::cin >> text_file_name;

int mode = 0;

std::cout << "1. Convert text to binary\n"
            "2. Convert binary to text\n"
            "3. Print all violations in binary\n"
            "4. Get violation by number\n"
            "5. Delete violation by number\n"
            "6. Doubling fines for violations by date and article\n"
            "7. Create selection violations by car number\n"
            "8. Exit\n";

std::cout << "Enter the mode: ";
std::cin >> mode;
```

Рисунок 30 – Код основной программы, часть 1

Участок кода, предназначенный для выбора и выполнения пунктов диалогового интерфейса показан на рис. 31 и рис 32.

```

while (mode != 8) {
    std::string num;
    switch (mode) {
        case 1:
            TextToBin( file_name: text_file_name, bin_file_name);
            break;
        case 2:
            BinToText( file_name: text_file_name, bin_file_name);
            break;
        case 3:
            PrintAllViolationInBin(bin_file_name);
            break;
        case 4: {
            std::cout << "Enter the number of the violation: ";
            std::cin >> num;

            bool code = false;
            std::string s = GetViolationByCarNum(bin_file_name, car_num: num, & code).ToString();

            if (code)
                std::cout << s << '\n';
            else
                std::cerr << "No such violation\n";

            break;
        }
        case 5: {
            std::cout << "Enter the number of the violation: ";
            std::cin >> num;
            DeleteViolationByCarNum(bin_file_name, car_num: num);
            break;
        }
        case 6: {
            std::string start_date;
            std::string end_date;
            std::string article;

```

Рисунок 31 – Код основной программы, часть 2

```

    case 6: {
        std::string start_date;
        std::string end_date;
        std::string article;

        std::cout << "Enter the start date: ";
        std::cin >> start_date;
        std::cout << "Enter the end date: ";
        std::cin >> end_date;
        std::cout << "Enter the article: ";

        std::cin >> std::ws;
        std::getline(&std::cin, &article);

        DoubleFine(bin_file_name, start_date, end_date, article);
        break;
    }
    case 7: {
        std::string file_name;

        std::cout << "Enter the car number: ";
        std::cin >> num;

        std::cout << "Enter the name of the new file: ";
        std::cin >> file_name;

        SelectionByCarNum(bin_file_name, new_file_name: file_name, car_num: num);
        break;
    }
    default:
        std::cout << "Invalid mode\n";
        break;
}
std::cout << "\nEnter the mode: ";
std::cin >> mode;
}

```

Рисунок 32 – Код основной программы, часть 3

### 3.6 Тестирование программы

Для начала необходимо проверить работоспособность функций преобразования текстового файла в бинарный и вывода записей бинарного файла (рис. 33).

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode: 3

Enter the mode: 1

Enter the mode: 3
A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.1;500;
B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Chaykovskogo, d. 9;12.19 ch.1;1200;
G432NC77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
H654TM77;Vasiliev I.V.;Skoda Octavia;13.09.2024;ul. Tolstogo, d. 10;12.9 ch.2;2500;
K987RP77;Nikolaev S.S.;Renault Duster;17.09.2024;ul. Chekhova, d. 15;12.5 ch.2;1800;
L123BC77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;

Enter the mode:
```

Рисунок 33 – Тестирование функций преобразование текстового файла и вывода на экран

Тестирование функции восстановления текстового файла из бинарного продемонстрировано на рис. 34 и рис. 35.



```
1
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode:
```

Рисунок 34 – Тестирование преобразования бинарного файла, часть 1

```
1 A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.1;500;
2 B560R77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
3 C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
4 D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
5 E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
6 F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Chaykovskogo, d. 9;12.19 ch.1;1200;
7 G432NC77;Morozov D.M.
8 H654TM77;Vasiliev I.
9 K987RP77;Nikolaev S.
10 L123BC77;Zaitsev M.M.
11
```

Z:\MIREA\SIAOD2\SIAOD-3-sem\5\_2\code\cmake-build-debug\task\_2.exe

Enter the name of the binary file: z2\_bin.bin  
Enter the name of the text file: z2\_text

1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit

Enter the mode: 2

Enter the mode:

Рисунок 35 - Тестирование преобразования бинарного файла, часть 2

Тестирование функции получения записи по номеру автомобиля  
(рис. 36)

```
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode: 4
Enter the number of the violation: F987KR77
F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Chaykovskogo, d. 9;12.19 ch.1;1200;
Enter the mode: 4
Enter the number of the violation: xxxxxxxx
No such violation
Enter the mode:
```

Рисунок 36 – Тестирование функции получения записи по ключу  
Тестирование функции удаления записи путем замены на последнюю запись (рис. 37).

```
Выбрать Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode: 5
Enter the number of the violation: F987KR77

Enter the mode: 3
A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.1;500;
B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
L123BC77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;
G432NC77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
H654TM77;Vasiliev I.V.;Skoda Octavia;13.09.2024;ul. Tolstogo, d. 10;12.9 ch.2;2500;
K987RP77;Nikolaev S.S.;Renault Duster;17.09.2024;ul. Chekhova, d. 15;12.5 ch.2;1800;

Enter the mode:
```

Рисунок 37 – Тестирование функции удаления записи  
Тестирование функции индивидуального задания увеличения штрафа все автомобилей за определенную дату и за определенную статью (рис. 38).

```
Выбрать Z:\MIREA\SAOD2\SAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode: 3
A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.2;500;
B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Chaykovskogo, d. 9;12.19 ch.1;1200;
G432NC77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
H654TM77;Vasiliev I.V.;Skoda Octavia;13.09.2024;ul. Tolstogo, d. 10;12.9 ch.2;2500;
K987RP77;Nikolaev S.S.;Renault Duster;17.09.2024;ul. Chekhova, d. 15;12.5 ch.2;1800;
L123BC77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;

Enter the mode: 6
Enter the start date: 10.09.2024
Enter the end date: 20.09.2024
Enter the article: 12.9 ch.2

Enter the mode: 3
A123BC77;Ivanov I.I.;Toyota Camry;15.09.2024;ul. Lenina, d. 12;12.9 ch.2;1000;
B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
C789KN77;Smirnov A.A.;Kia Rio;20.09.2024;ul. Pushkina, d. 5;12.5 ch.1;1500;
D321OR99;Sidorov V.V.;Ford Focus;12.09.2024;ul. Gagarina, d. 7;12.8 ch.2;2000;
E654TR77;Kuznetsov N.N.;Nissan Qashqai;14.09.2024;pr. Lomonosova, d. 22;12.15 ch.4;700;
F987KR77;Fedorov A.S.;Mazda 3;16.09.2024;ul. Chaykovskogo, d. 9;12.19 ch.1;1200;
G432NC77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
H654TM77;Vasiliev I.V.;Skoda Octavia;13.09.2024;ul. Tolstogo, d. 10;12.9 ch.2;5000;
K987RP77;Nikolaev S.S.;Renault Duster;17.09.2024;ul. Chekhova, d. 15;12.5 ch.2;1800;
L123BC77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;

Enter the mode: _
```

Рисунок 38 – Тестирование функции увеличения штрафа

Тестирование функции создания выборки по номеру автомобиля (рис. 39).

The image shows a code editor with a file named `test_test_test`. The editor contains four lines of text, each representing a record with a car number, driver name, car model, date, location, driver details, and fines. A terminal window is open in the foreground, showing the execution of a program. The program prompts for a binary file name (`z2_bin.bin`) and a text file name (`z2_text`). It then displays a menu of options: 1. Convert text to binary, 2. Convert binary to text, 3. Print all violations in binary, 4. Get violation by number, 5. Delete violation by number, 6. Doubling fines for violations by date and article, 7. Create selection violations by car number, and 8. Exit. The user selects option 7, enters the car number `B456OR77`, and the program creates a new file named `test_test_test`. The terminal then prompts for a mode, with a cursor visible.

```
test_test_test x
1 B456OR77;Petrov P.P.;Hyundai Solaris;18.09.2024;pr. Mira, d. 8;12.16 ch.3;1000;
2 B456OR77;Morozov D.M.;Volkswagen Polo;19.09.2024;pr. Sakharova, d. 4;12.16 ch.1;500;
3 B456OR77;Zaitsev M.M.;Lada Vesta;21.09.2024;ul. Gorkogo, d. 11;12.6 ch.1;900;
4
Z:\MIREA\SIAOD2\SIAOD-3-sem\5_2\code\cmake-build-debug\task_2.exe
Enter the name of the binary file: z2_bin.bin
Enter the name of the text file: z2_text
1. Convert text to binary
2. Convert binary to text
3. Print all violations in binary
4. Get violation by number
5. Delete violation by number
6. Doubling fines for violations by date and article
7. Create selection violations by car number
8. Exit
Enter the mode: 7
Enter the car number: B456OR77
Enter the name of the new file: test_test_test
Enter the mode: _
```

Рисунок 39 – Тестирование функции создания выборки

## **ВЫВОД**

В результате выполнения данной практической работы были освоены навыки работы файловыми потоками по управлению текстовыми и бинарными файлами.

## СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Рысин, М. Л. Введение в структуры и алгоритмы обработки данных : учебное пособие / М. Л. Рысин, М. В. Сартаков, М. Б. Туманова. — Москва : РТУ МИРЭА, 2022 — Часть 2 : Поиск в тексте. Нелинейные структуры данных. Кодирование информации. Алгоритмические стратегии — 2022. — 111 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/310826> (дата обращения: 10.09.2024).
2. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения : межгосударственный стандарт : дата введения 1992-01- 01 / Федеральное агентство по техническому регулированию. — Изд. официальное. — Москва : Стандартинформ, 2010. — 23 с