

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](#_Toc179505364)

[2. ЗАДАНИЕ №1 5](#_Toc179505365)

[2.1 Постановка задания 5](#_Toc179505366)

[2.2 Содерживое текстового файла 6](#_Toc179505367)

[2.3 Разработка функций для работы с текстовым файлом 6](#_Toc179505368)

[2.3.1 Функции открытия и файловых потоков 6](#_Toc179505369)

[2.3.2 Вывод содержимого файлового текста 7](#_Toc179505370)

[2.3.3 Добавление записи в конец файла 7](#_Toc179505371)

[2.3.4 Получение числа по его порядковому номеру 8](#_Toc179505372)

[2.3.5 Получение количества чисел в файле 9](#_Toc179505373)

[2.3.6 Решение задания индивидуального варианта 10](#_Toc179505374)

[2.4 Разработка основной программы 12](#_Toc179505375)

[2.5 Тестирование 15](#_Toc179505376)

[3 ЗАДАНИЕ 2 20](#_Toc179505377)

[3.1 Постановка задачи 20](#_Toc179505378)

[3.2 Содержимое текстового файла 21](#_Toc179505379)

[3.3 Структура записи 21](#_Toc179505380)

[3.4 Реализация функций программы 23](#_Toc179505381)

[3.4.1 Преобразование текстовых данных в двоичный вид 23](#_Toc179505382)

[3.4.2 Сохранение данных двоичного файла в текстовом 25](#_Toc179505383)

[3.4.3 Вывод всех записей двоичного файла 26](#_Toc179505384)

[3.4.4 Доступ к записи по ее порядковому номеру 26](#_Toc179505385)

[3.4.5 Удаление записи с заданным значением ключа 27](#_Toc179505386)

[3.4.6 Список нарушений по автомобилю заданного номера 28](#_Toc179505387)

[3.4.7 Увеличение суммы штрафа 28](#_Toc179505388)

[3.5 Разработка основной программы 28](#_Toc179505389)

[3.6 Тестирование программы 31](#_Toc179505390)

[ВЫВОД 38](#_Toc179505391)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 39](#_Toc179505392)

# 1 ЦЕЛЬ РАБОТЫ

Получить навыки применения файловых потоков языка С++ (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: номер автомобиля, 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 С++: 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) прочитать 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 Содерживое текстового файла

Исходное содержимое текстового файла z1\_text, требуемого для задания, представлено на рис. 1

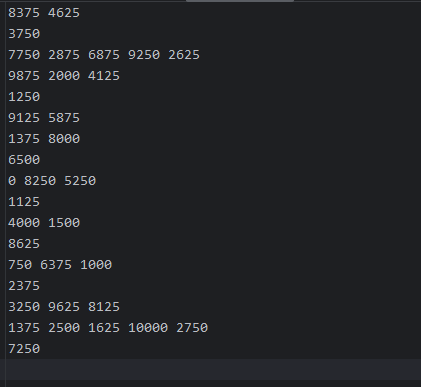


Рисунок 1 – Содержимое текстового файла z1\_text

## 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).

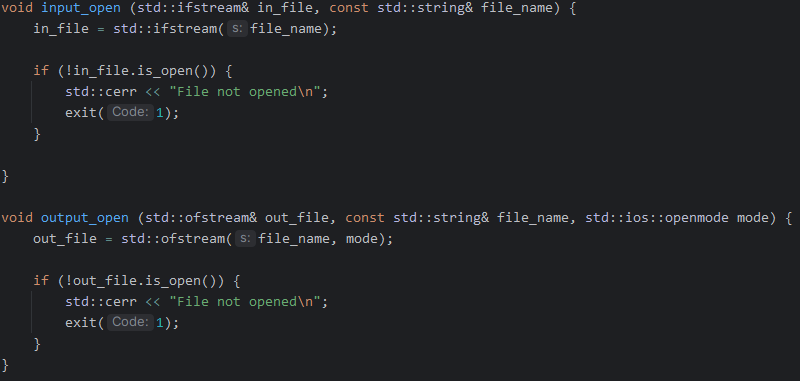


Рисунок 2 – функции открытия файловых потоков для ввода и вывода

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.

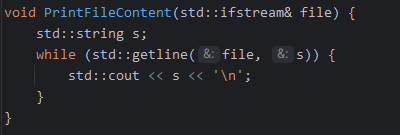


Рисунок 3 – Функция вывод содержимого текстового фала на экран

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.

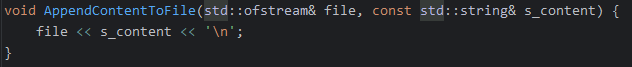


Рисунок 4 – Функция добавления записи в конец файла

Для использования этой функции необходимо передавать в нее ссылку на файловой 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

Учитывая, что в одной строке может находиться сразу несколько чисел, разделенный пробелов, необходимо реализовать код, который будет разделять эту строку на составляющие ее числа. Реализация функцииSplit представлена на рис. 5.

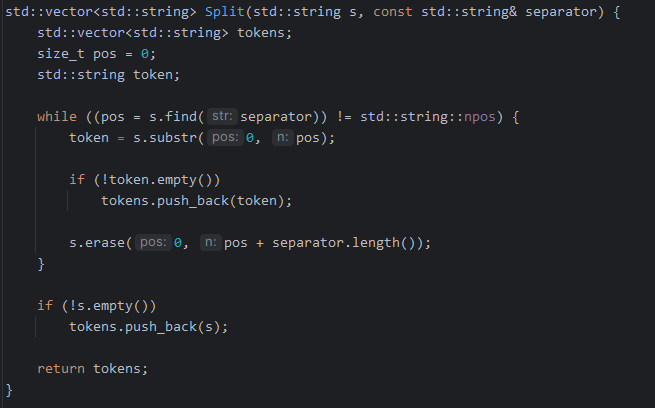


Рисунок 5 – Функция разбиения строки по разделителю

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

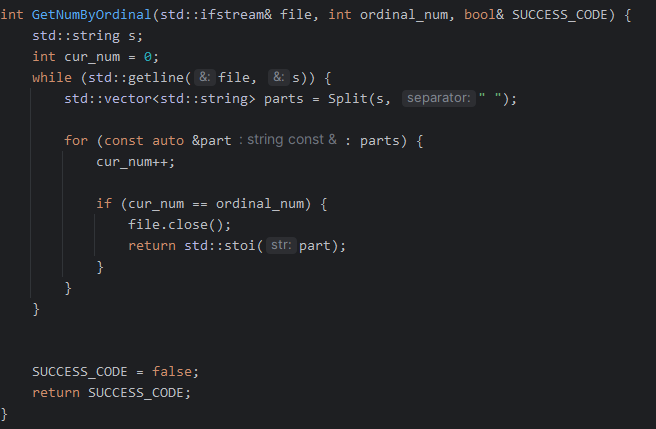


Рисунок 6 – Функция получения числа по порядковому номеру

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

Как и при получении числа по порядковому номеру, необходимо разбивать каждую строку на ее составляющие числа с помощью функции Split. На рис. 7 показана реализация функции подсчета чисел в файле.

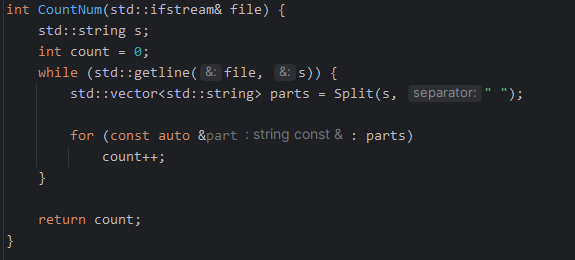


Рисунок 7 – Функция подсчета чисел в файле

### 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.

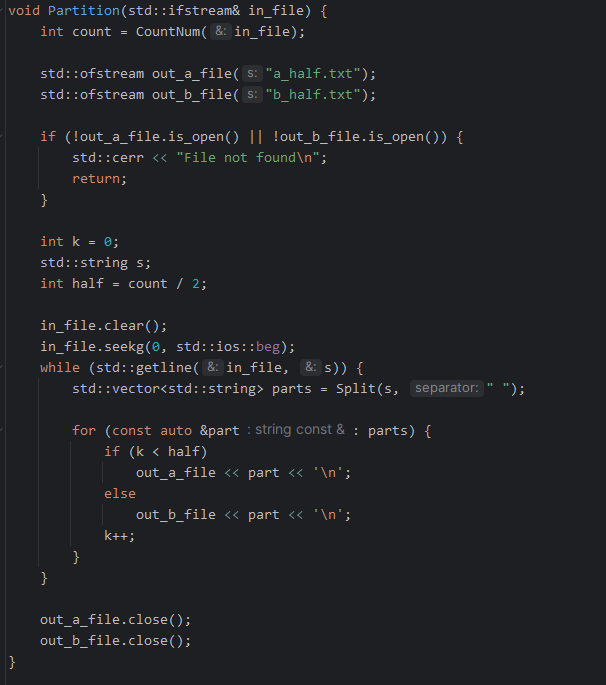


Рисунок 8 – Функция разбиения текстового файла

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

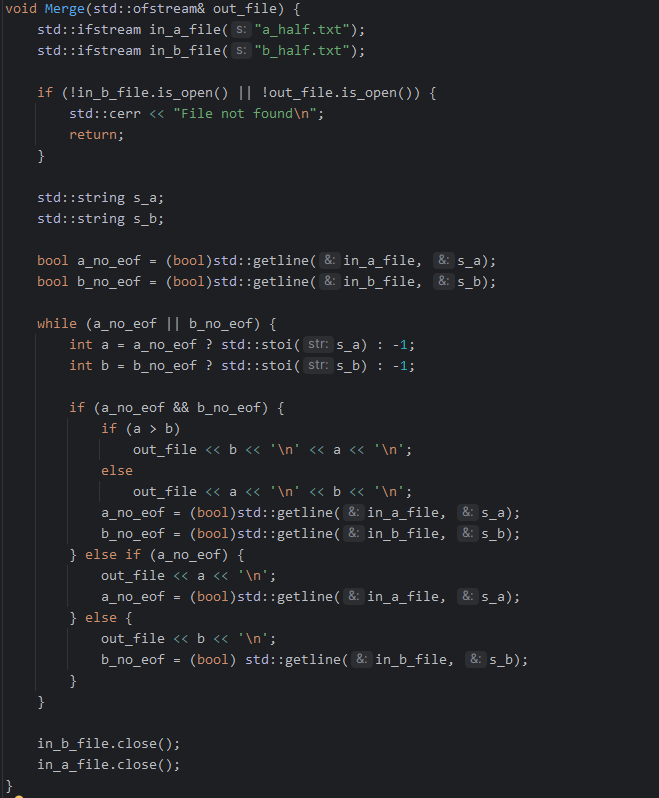


Рисунок 9 – Функция слияния

## 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).

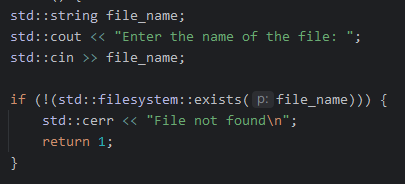


Рисунок 10 – Получение имени и проверка существования файла

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.

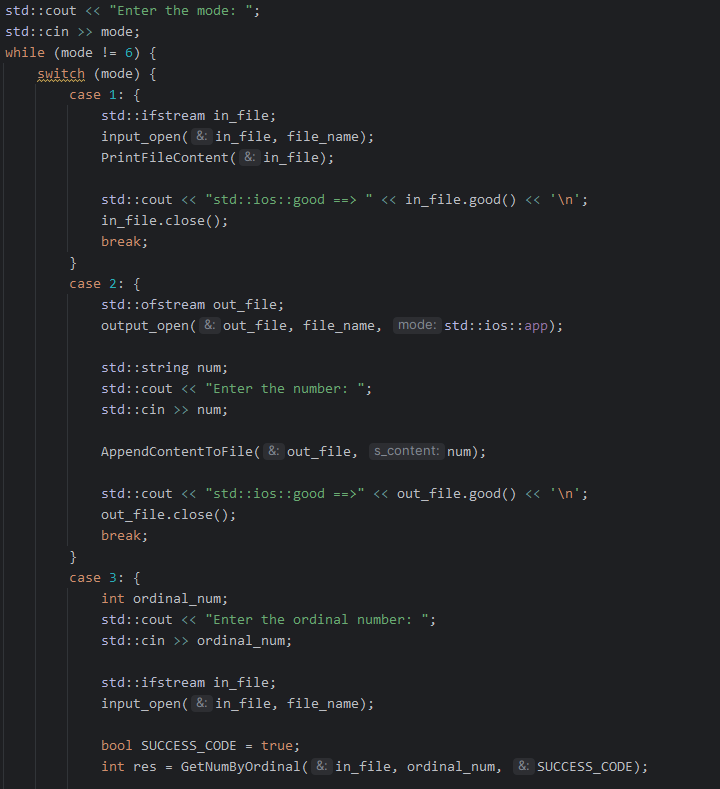


Рисунок 11 – реализация диалогового интерфейса, часть 1

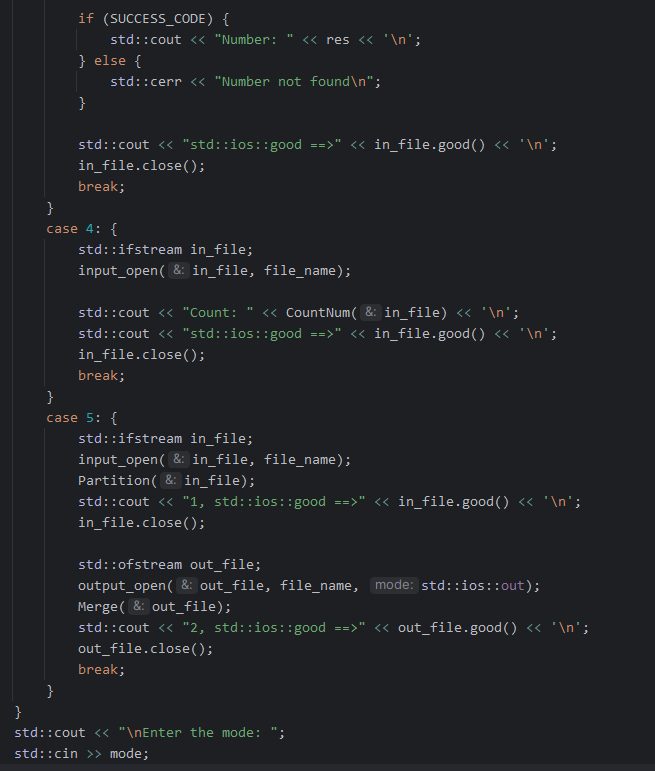


Рисунок 12 – реализация диалогового интерфейса, часть 2

## 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)

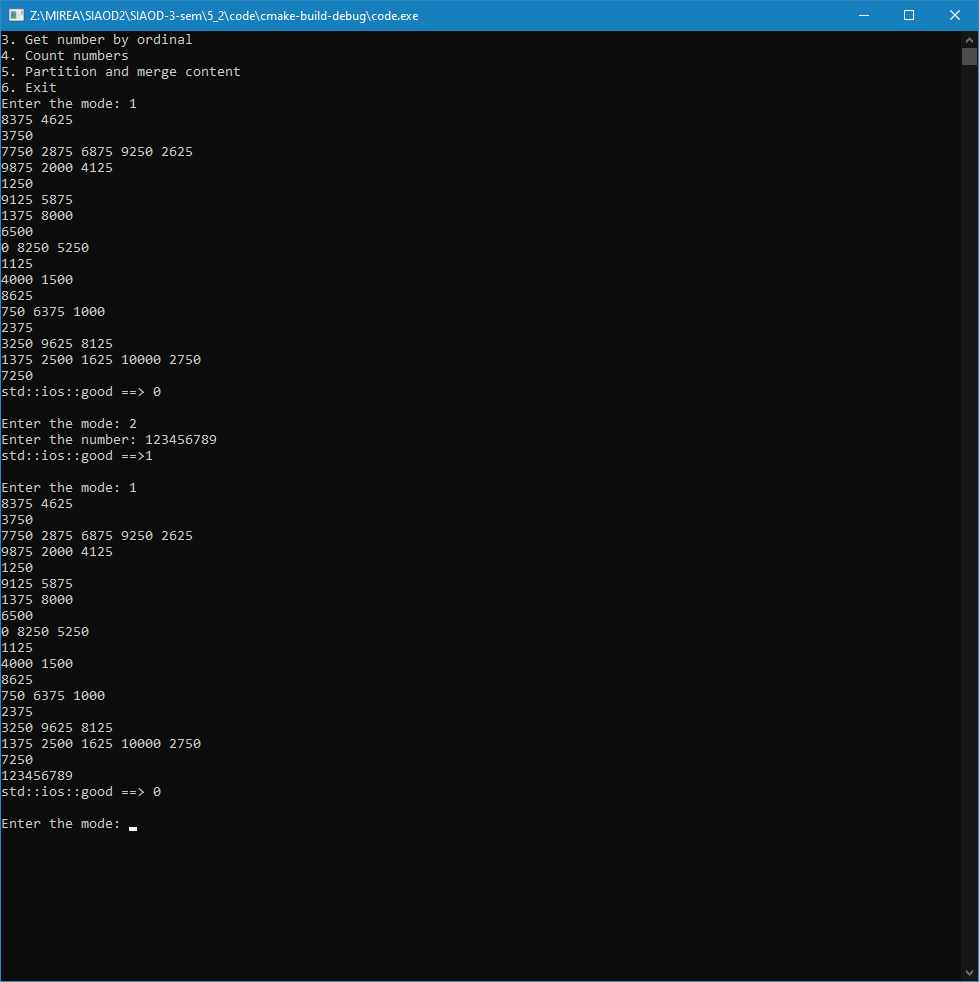


Рисунок 13 – Тестирование функций вывода и добавления в конец

Тестирование функции получения числа по порядковому номеру:

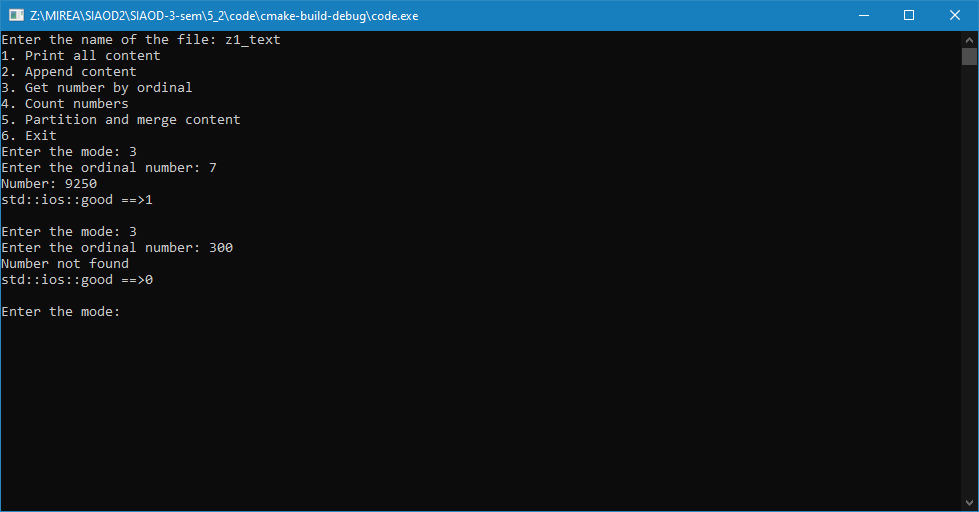


Рисунок 14 – Получение числа по порядковому номеру

Тестирование подсчета чисел в файле:

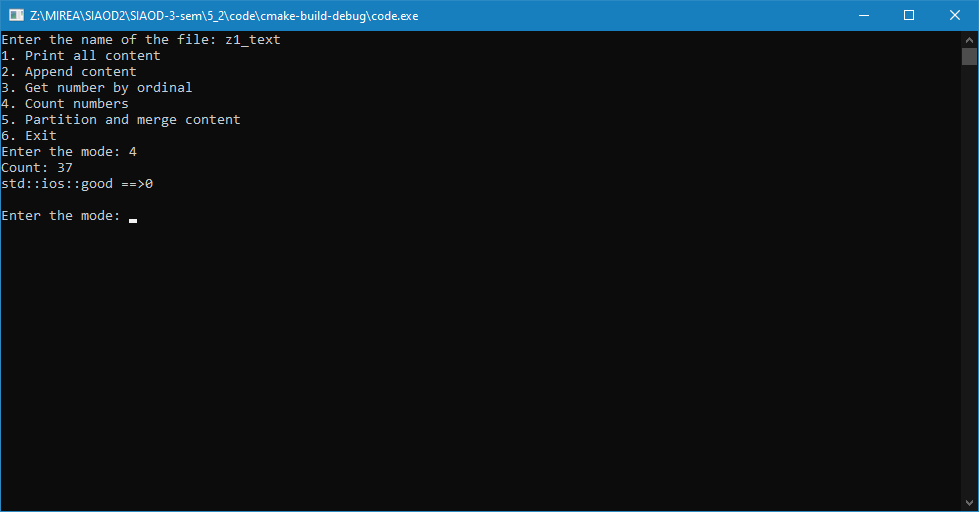


Рисунок 15 – Получение количества чисел

Тестирование функций для индивидуального задания:

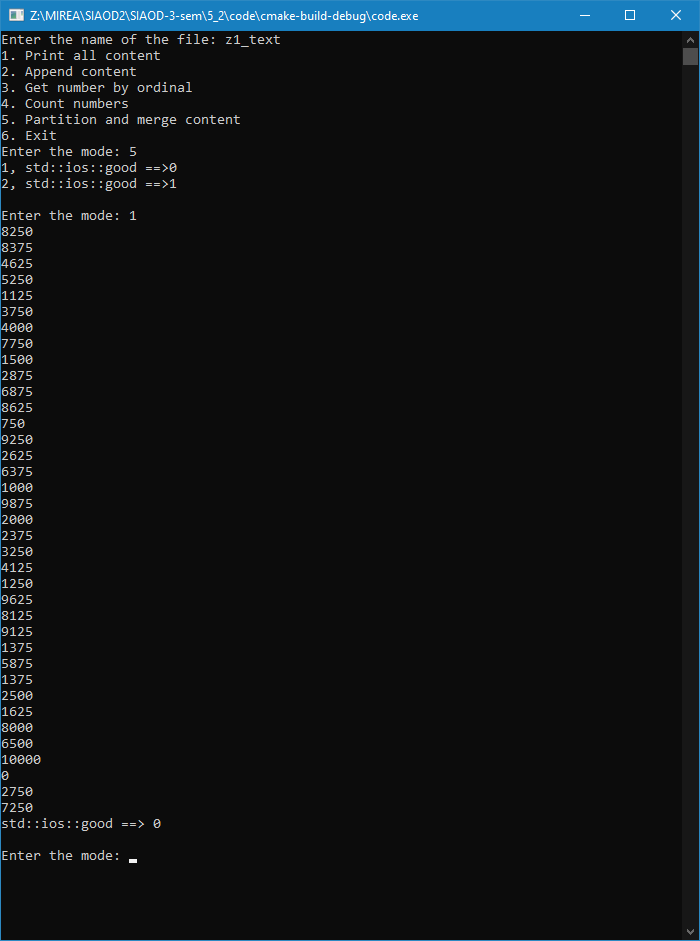


Рисунок 16 – Тестирование задания индивидуального варианта

Тестирование поведение программы при неправильном название файла:

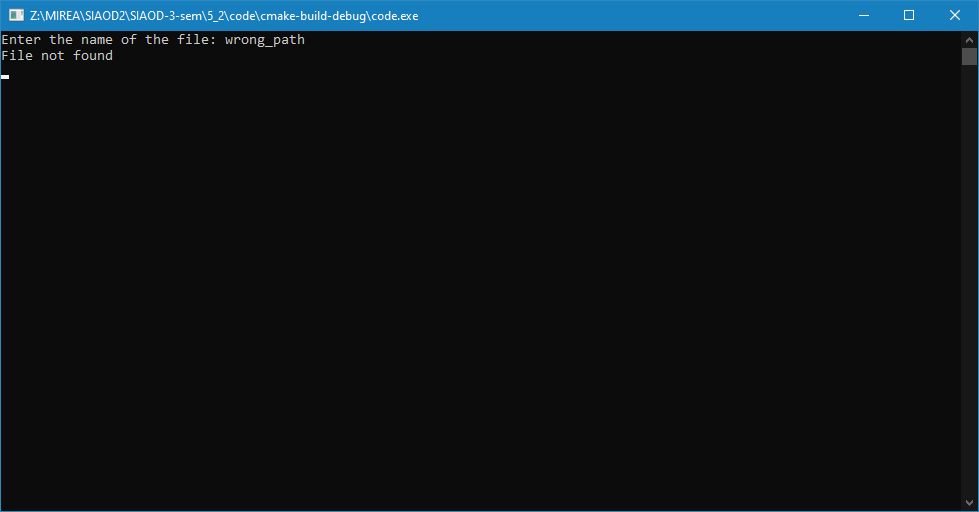


Рисунок 17 – Неправильное название файла

# 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: номер автомобиля, 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

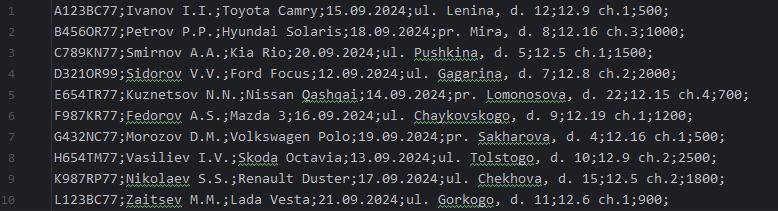


Рисунок 18 – Содержимое текстового файла z2\_text

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

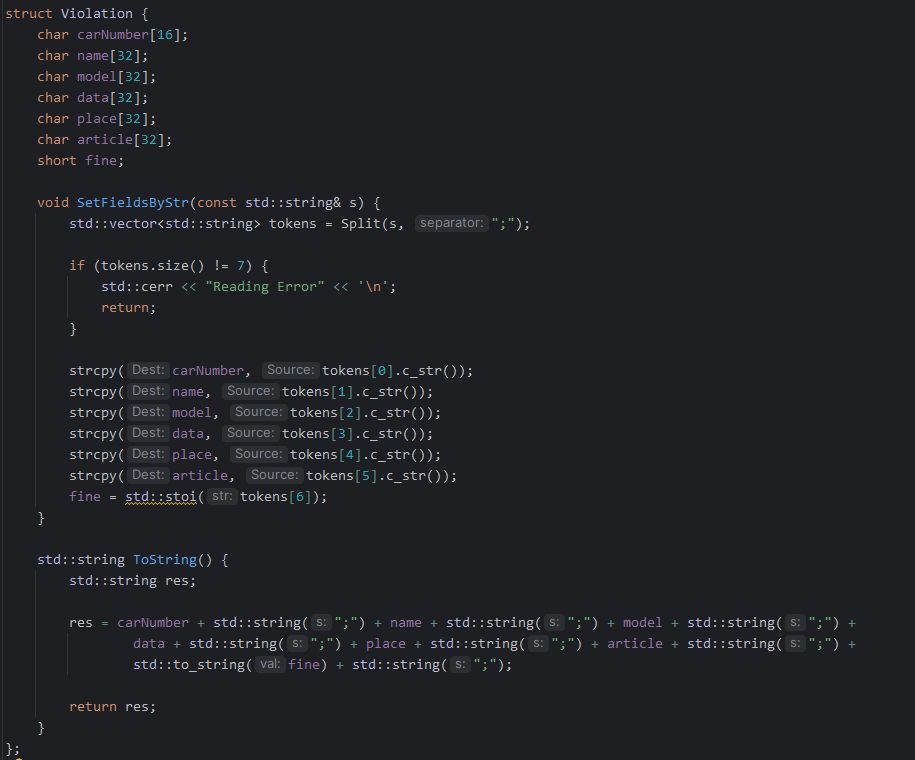


Рисунок 19 – Определение структуры и реализация ее методов

## 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.

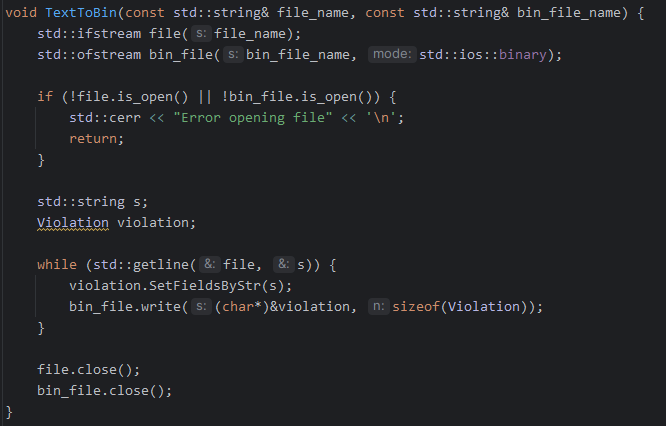


Рисунок 20 – Функция преобразования текстового файла

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

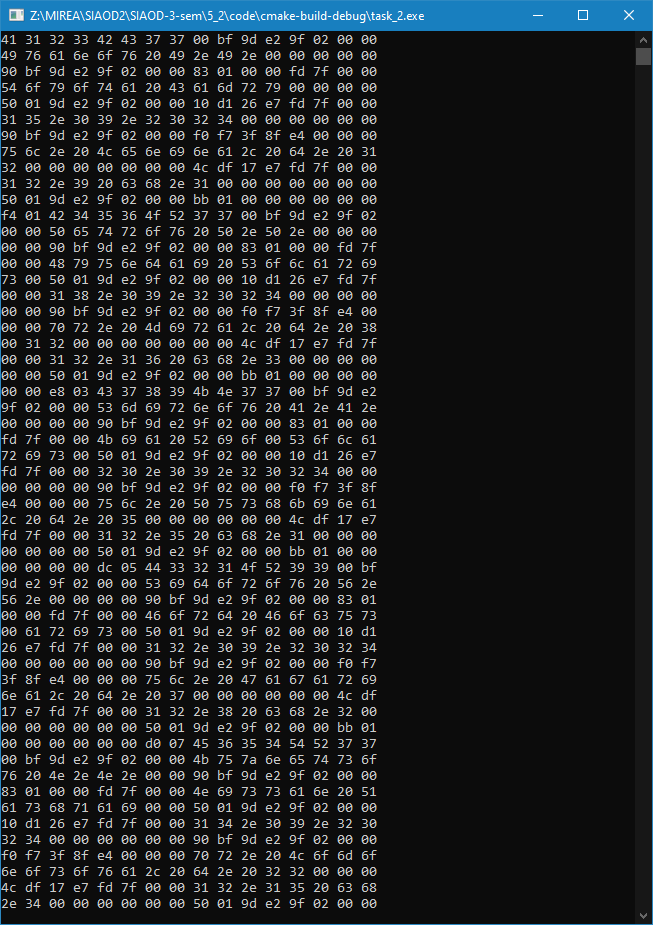


Рисунок 21 – Представление текстового файла z2\_text в шестнадцатеричном формате

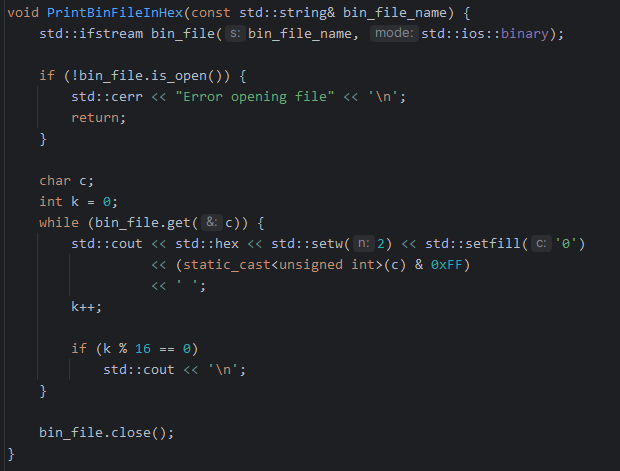


Рисунок 22 – Функция вывода содержания бинарного файла

### 3.4.2 Сохранение данных двоичного файла в текстовом

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

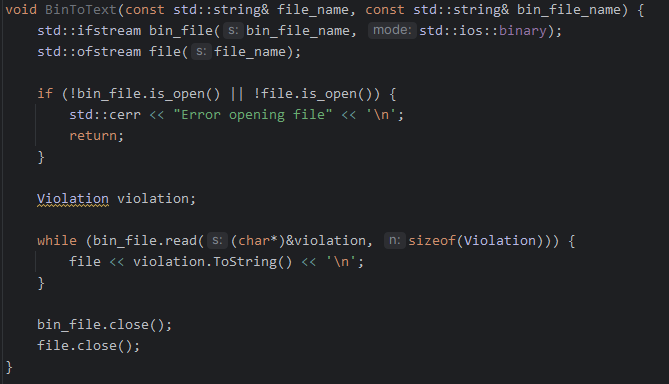


Рисунок 23 – Функция преобразования бинарного файла

### 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).

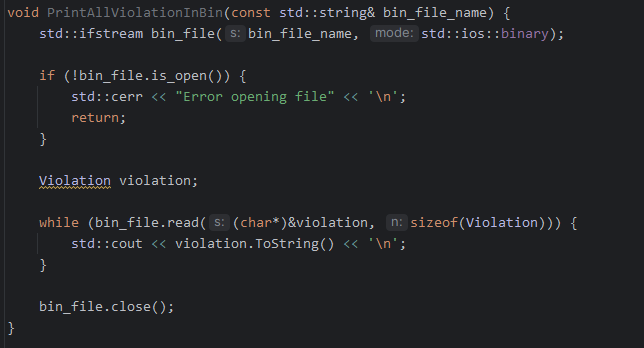


Рисунок 24 – Функция вывода всех записей из бинарного файла

### 3.4.4 Доступ к записи по ее порядковому номеру

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

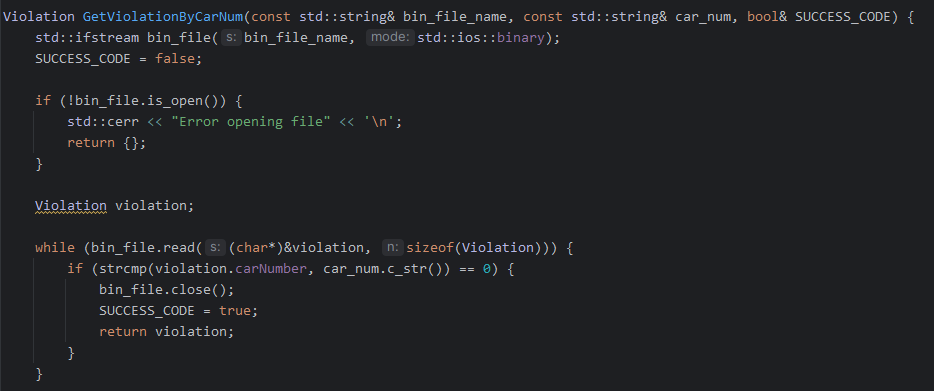


Рисунок 25 – Функция доступа к записи по ее порядковому номеру

### 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.

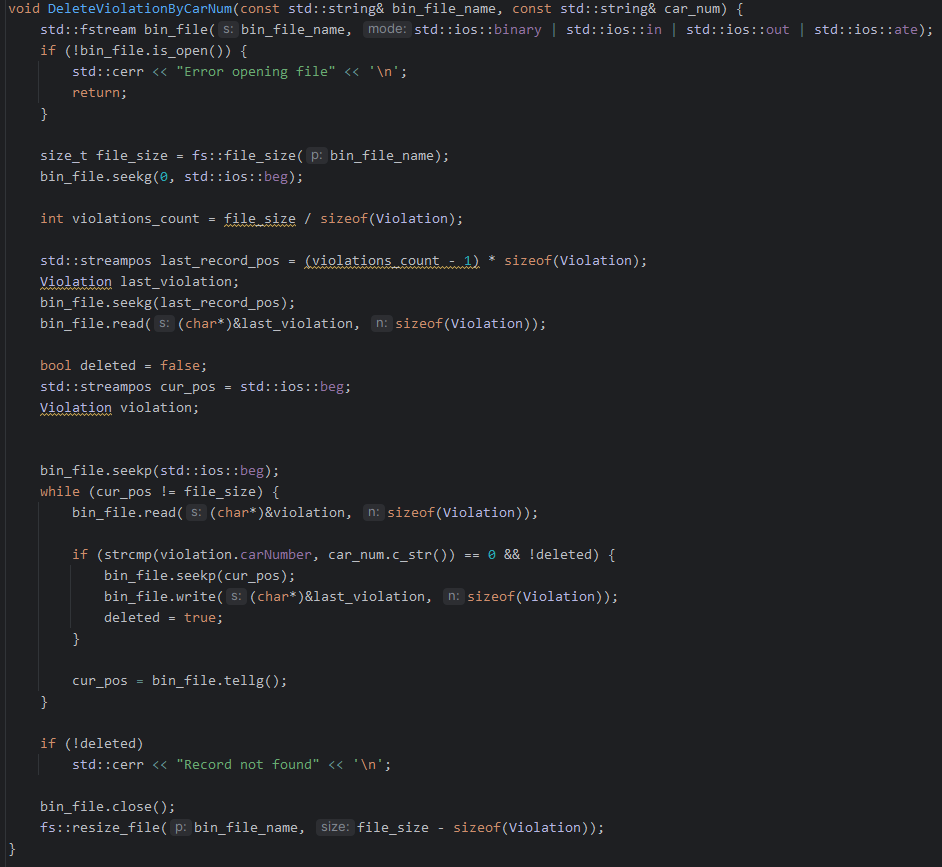


Рисунок 27 – Функция удаления записи по ключу

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. Дальше 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.

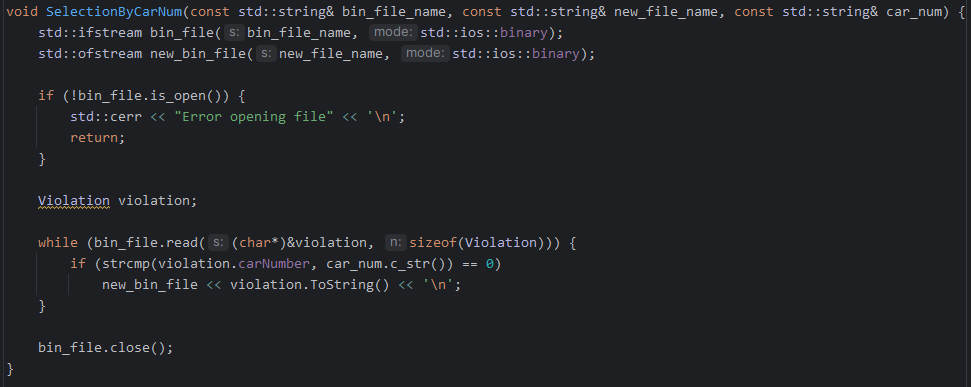


Рисунок 28 – Функция создания выборки по ключу

### 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.

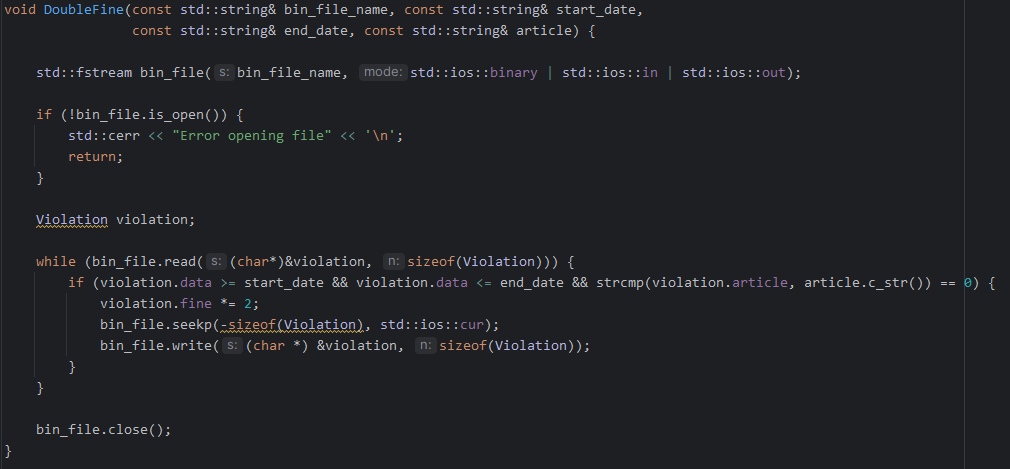


Рисунок 29 – Функция увеличения суммы штрафа

## 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.

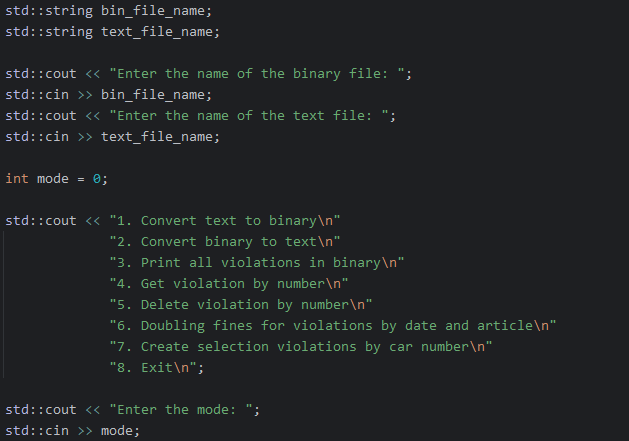


Рисунок 30 – Код основной программы, часть 1

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

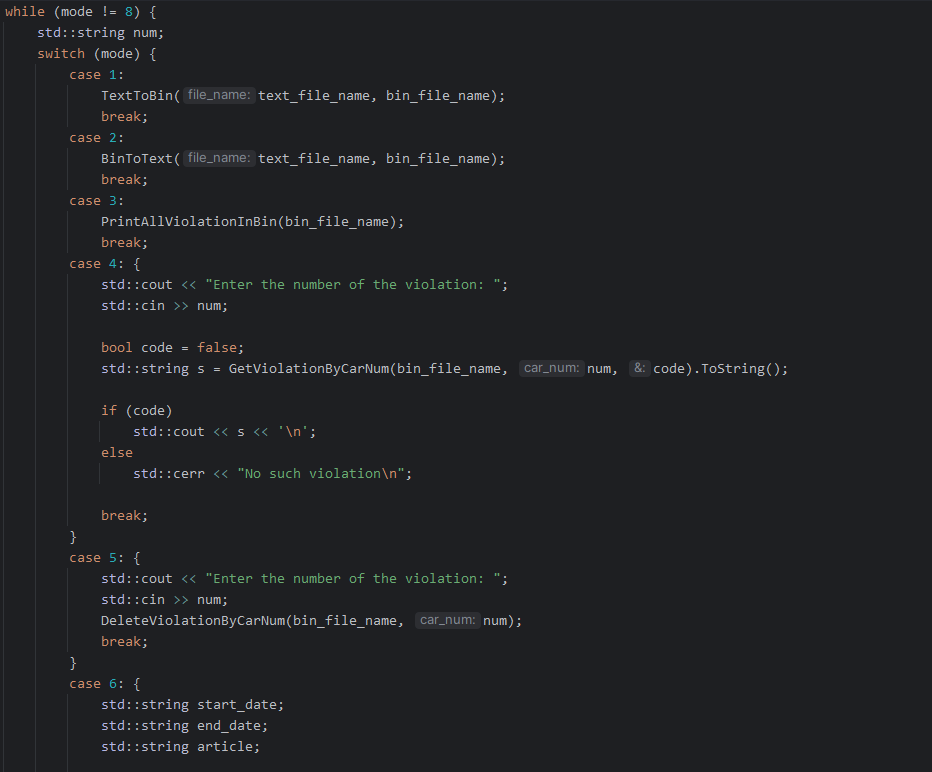


Рисунок 31 – Код основной программы, часть 2



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).

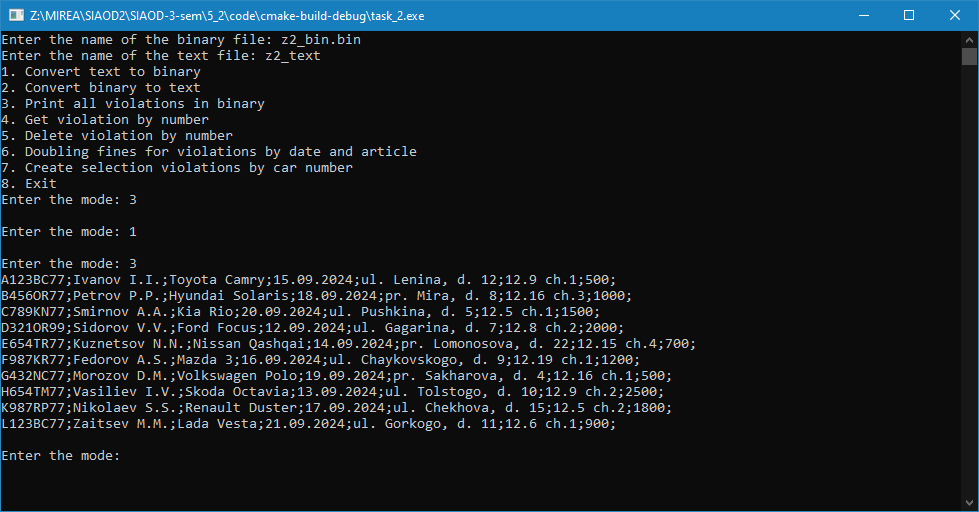


Рисунок 33 – Тестирование функций преобразование текстового файла и вывода на экран

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

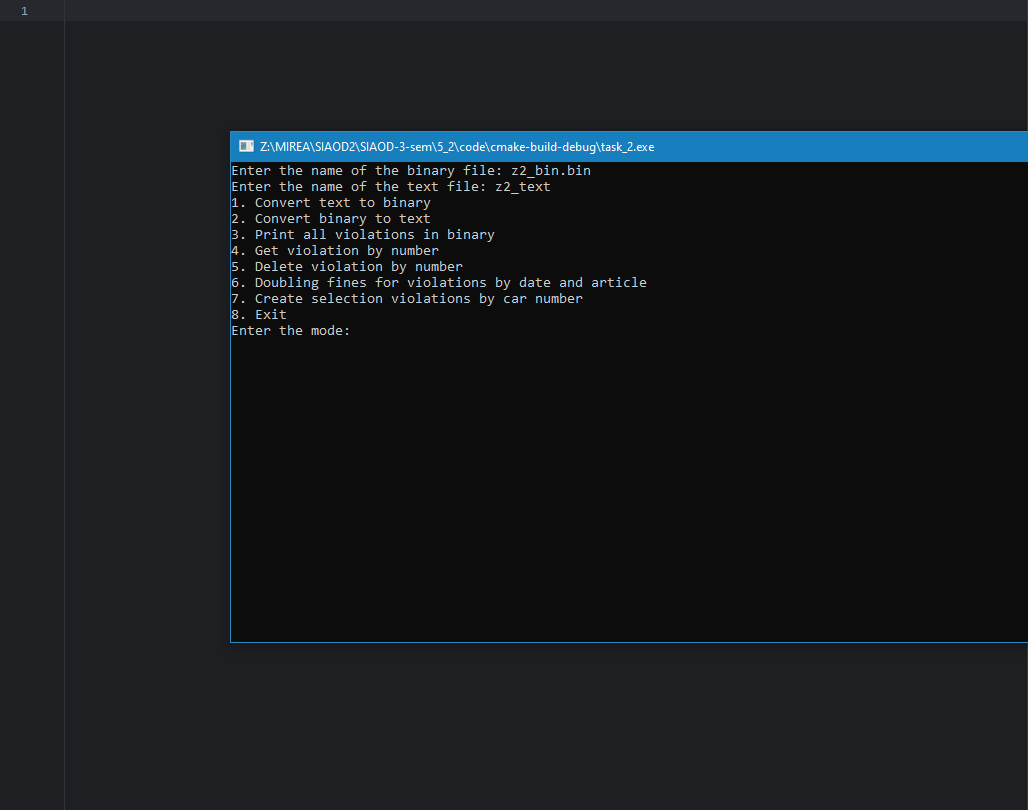


Рисунок 34 – Тестирование преобразования бинарного файла, часть 1

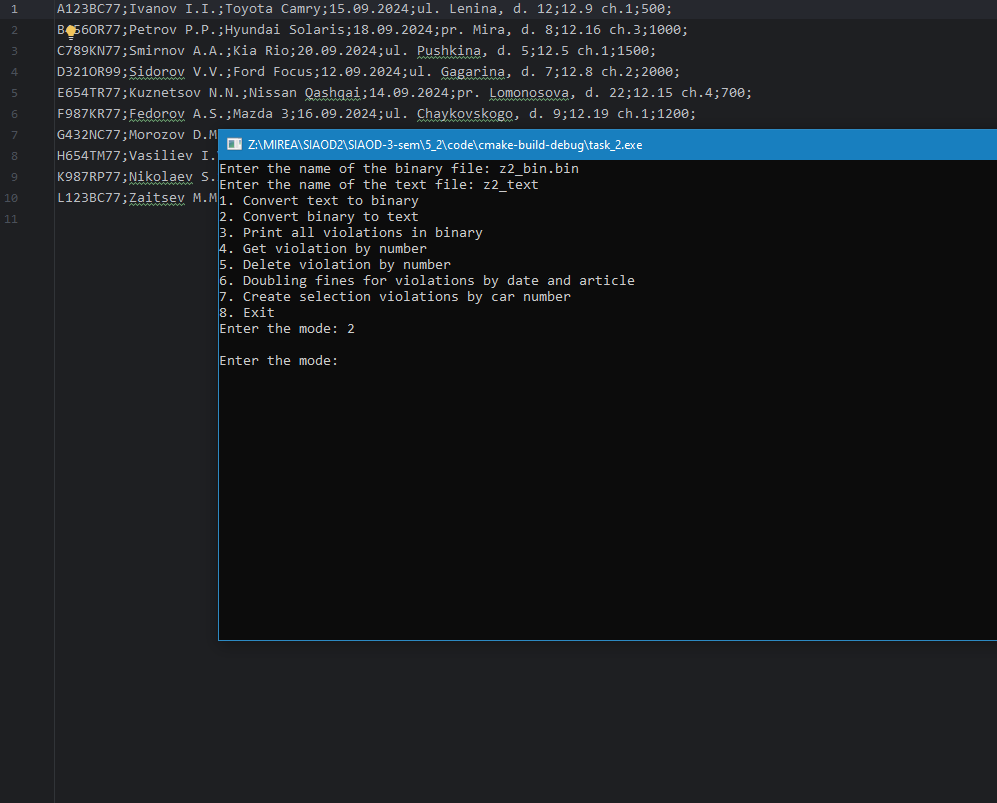


Рисунок 35 - Тестирование преобразования бинарного файла, часть 2

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

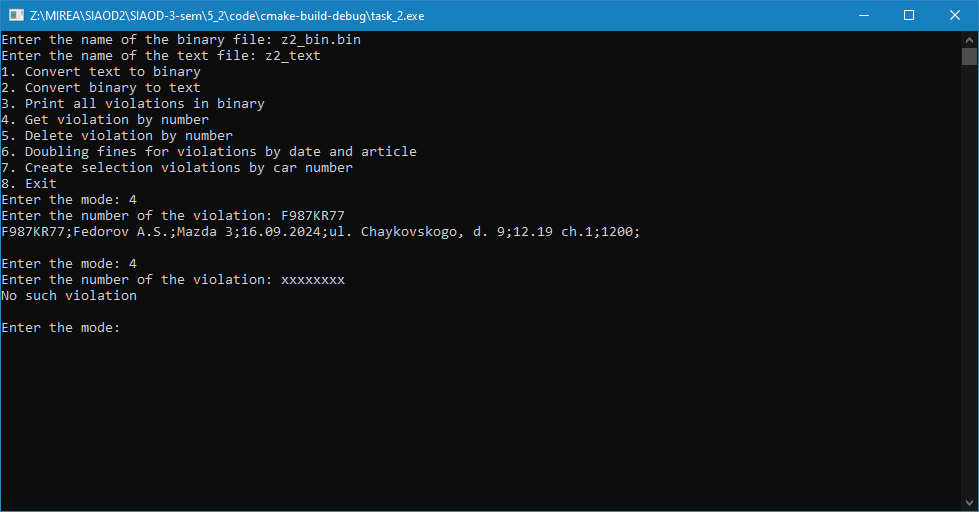


Рисунок 36 – Тестирование функции получения записи по ключу

Тестирование функции удаления записи путем замены на последнюю запись (рис. 37).

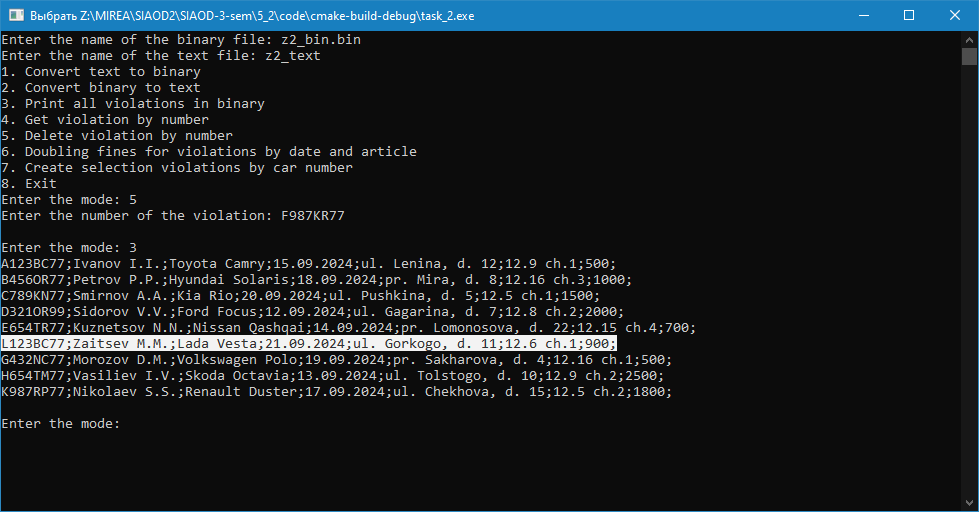


Рисунок 37 – Тестирование функции удаления записи

Тестирование функции индивидуального задания увеличения штрафа все автомобилей за определенную дату и за определенную статью (рис. 38).

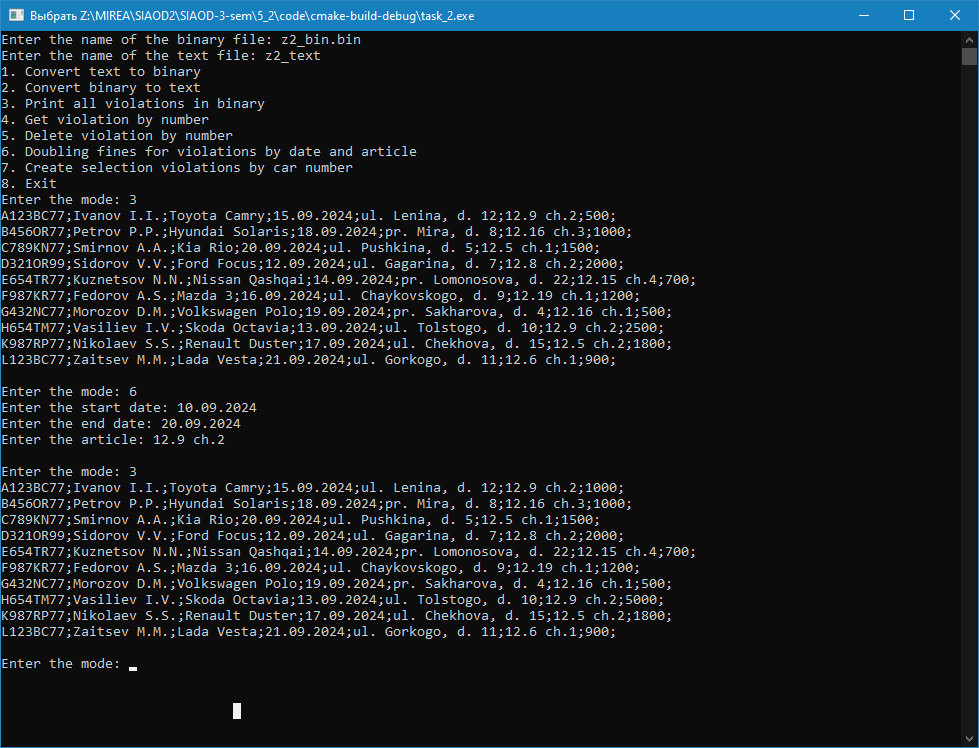


Рисунок 38 – Тестирование функции увеличения штрафа

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

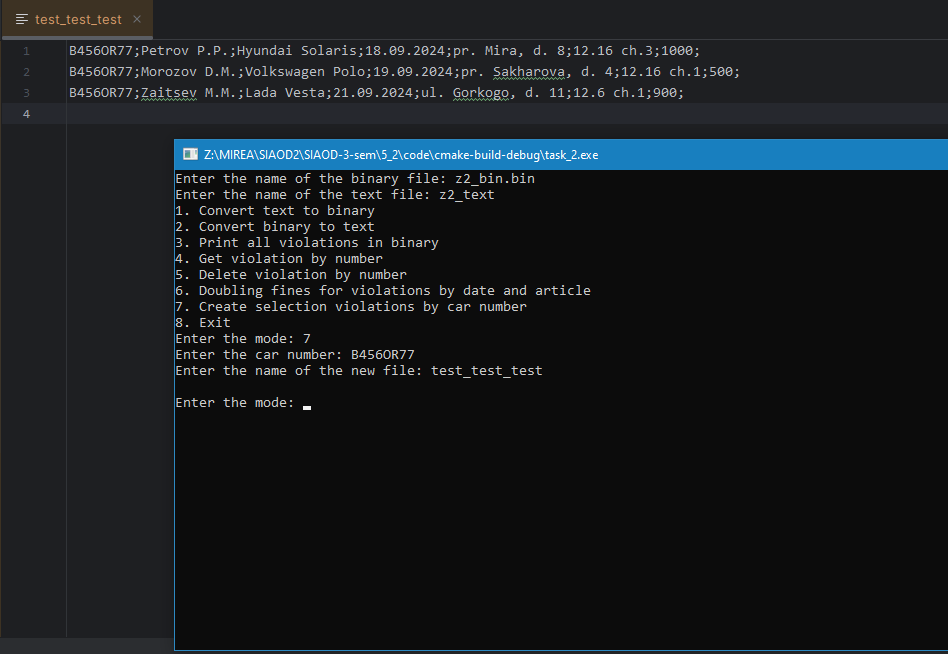


Рисунок 39 – Тестирование функции создания выборки

# 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