|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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**»**  **Тема: «Внешние структуры данных: текстовый и двоичный файлы»** | |
|  | |
| Выполнил студент группы ИКБО-15-22 | Ератин Н. В. |
| Принял преподаватель | Скворцова Л. А. |

|  |  |  |
| --- | --- | --- |
| Самостоятельная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

Содержание

**1.** [**Цель работы** 3](#_Toc147084314)

[**2.** **Задача №1** 3](#_Toc147084315)

[**2.1 Постановка задачи** 3](#_Toc147084316)

[**2.2** **Код программы** 4](#_Toc147084317)

[**2.3** **Выполнение индивидуального задания** 6](#_Toc147084318)

[**2.4** **Тестирование программы** 9](#_Toc147084319)

[**3.** **Задача №2** 12](#_Toc147084320)

[**3.1. Код программы.** 13](#_Toc147084321)

[**3.2. Тестирование программы.** 19](#_Toc147084322)

[**3.3. Входные данные программы.** 21](#_Toc147084323)

# **Цель работы**

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

# **Задача №1**

## **2.1 Постановка задачи**

Разработать программу, управления текстовым файлом. Требования:

* + - 1. Реализация ввода-вывода на основе файловых потоков С++: ofstream, ifstream.
      2. Имя физического файла вводится пользователем и передается в функции обработки через параметр.
      3. При открытии файла выполнять контроль его существования и открытия.

Примечание. При отладке программы можете имя физического файла определить через константу.

* + - 1. Разработать функции для выполнения операций над текстовым файлом.
         1. создание текстового файла средствами текстового редактора кодировки ASCII, содержащего десятичные числа по несколько чисел на строке;
         2. вывод содержимого текстового файла;
         3. добавление новой записи в конец файла;
         4. прочитать значение числа, указав его порядковый номер в файле, и вернуть его значение;
         5. определить количество чисел в файле.
      2. Разработать приложение и выполнить тестирование всех функций. Приложение должно содержать диалоговый интерфейс на основе текстового меню.
      3. Контроль открытия и существования файла выполнить в основной программе перед вызовом функции. Перед закрытием файла, проверить отсутствие ошибок ввода и вывода (метод good).
      4. Создать модуль и перенести в него все отлаженные функции. Исключить функции из приложения. Отладить приложение, подключив к нему модуль с функциями.
      5. Разработать функции для реализации дополнительных операций, определенных вариантом и сохранить их в модуле с остальными функциями.
      6. Выполнить тестирование приложения в полном объеме.

## **Код программы**

Функция inFile выполняет создание текстового файла.

void inFile(std::ofstream& fout, const char\* inputFileName) {

std::cout << "Открытие и заполнение первого файла\n";

fout.open(inputFileName, std::ios::out | std::ios::trunc);

for (int i = 1; i <= 10; i++) {

fout << i << std::endl;

}

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

fout.close();

}

*Функция inFile*

Функция fileOutput выводит содержимое файла на экран.

void fileOutput(const char\* inputFileName) {

int x;

std::ifstream fout;

fout.open(inputFileName, std::ios::in);

while (fout >> x) {

std::cout << x << " ";

}

std::cout << std::endl;

fout.close();

}

*Функция fileOutput*

Функция addNewLine добавляет новую запись в конец файл.

void addNewLine(std::ofstream& fout, const char\* inputFileName) {

int x = 0;

std::cout << "Введите элемент, который хотите добавить: ";

std::cin >> x;

fout.open(inputFileName, std::ios::out | std::ios::app);

fout << x;

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

fout.close();

}

*Функция addNewLine*

Функция findNumbers прочитывает значение числа, указав его порядковый̆ номер в файле, и возвращает его значение.

void findNumbers(const char\* inputFileName) {

int nums=0;

std::cout << "Введите порядковый номер в файле: ";

std::cin >> nums;

std::ifstream fout;

fout.open(inputFileName);

int x;

int i;

std::cout << "> ";

for (i = 1; (i < nums && (!fout.eof())); i++) {

fout >> x;

fout.get();

}

while (!fout.eof() && (i == nums)) {

fout >> x;

fout.get();

std::cout << x;

i++;

}

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

fout.close();

}

*Функция findNumbers*

Функция length\_file - определяет количество чисел в файле.

void length\_file(const char\* inputFileName) {

std::ifstream fout;

fout.open(inputFileName);

int n = 0;

int x;

std::cout << std::endl;

while (!fout.eof()) {

fout >> x;

fout.get();

n++;

}

std::cout << "Количество чисел: " << n << std::endl;

fout.close();

} std::cout << " Ошибок не возникло\n";

}

fout.close();

}

*Функция length\_file*

## **Выполнение индивидуального задания**

Индивидуальный вариант №11

Задание: Создать новый файл из значений исходного, разделив все нечетные по абсолютной величине числа на среднее арифметическое чисел в файле.

Функция divideOddNumbersByAverage создает новый файл из значений исходного, разделив все нечетные по абсолютной величине числа на среднее арифметическое чисел в файле.

void divideOddNumbersByAverage(const char\* inputFileName, const char\* outputFileName) {

std::cout << " Открытие и заполнение второго файла: \n";

std::ifstream inputFile(inputFileName);

if (!inputFile.is\_open()) {

std::cout << "Ошибка открытия исходного файла\n";

return;

}

int number;

int count = 0;

double sum = 0.0;

// Вычисляем среднее арифметическое

while (inputFile >> number) {

sum += number;

count++;

}

double average = sum / count;

// Переходим к началу файла

inputFile.clear();

inputFile.seekg(0, std::ios::beg);

// Открываем новый файл для записи

std::ofstream outputFile(outputFileName);

if (!outputFile.is\_open()) {

std::cout << "Ошибка открытия файла для записи\n";

inputFile.close();

return;

}

while (inputFile >> number) {

// Если число нечетное по абсолютной величине, делим его на среднее арифметическое и записываем в новый файл

if (std::abs(number) % 2 == 1) {

double dividedValue = number / average;

outputFile << dividedValue << " ";

std::cout << dividedValue << " ";

}

else {

outputFile << number << " ";

}

}

if (outputFile.good()) {

std::cout << "\n Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

inputFile.close();

outputFile.close();

std::cout << "\n///Завершение программы///\n";

}

*Функция divideOddNumbersByAverage*

Файл Main.cpp

Функция main — главная функция программы.

#include <iostream>

#include <fstream>

#include "Header.h"

int main() {

setlocale(LC\_ALL, "Russian");

const char\* inputFileName = "input.txt";

const char\* outputFileName = "output.txt";

std::ofstream fout;

inFile(fout, inputFileName);

fileOutput(inputFileName);

addNewLine(fout, inputFileName);

findNumbers(inputFileName);

length\_file(inputFileName);

divideOddNumbersByAverage(inputFileName, outputFileName);

return 0;

}

*Функция main*

Файл Header.h

Заголовочный файл программы, содержит объявления функций

#ifndef FUNCTIONS\_H

#define FUNCTIONS\_H

#include <iostream>

#include <fstream>

void inFile(std::ofstream& fout, const char\* inputFileName);

void fileOutput(const char\* inputFileName);

void addNewLine(std::ofstream& fout, const char\* inputFileName);

void findNumbers(const char\* inputFileName);

void length\_file(const char\* inputFileName);

void divideOddNumbersByAverage(const char\* inputFileName, const char\* outputFileName);

#endif

Файл Functions.cpp

Содержит реализацию функций программы

#include "Header.h"

void inFile(std::ofstream& fout, const char\* inputFileName) {

std::cout << "Открытие и заполнение первого файла\n";

fout.open(inputFileName, std::ios::out | std::ios::trunc);

for (int i = 1; i <= 10; i++) {

fout << i << std::endl;

}

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

fout.close();

}

void fileOutput(const char\* inputFileName) {

int x;

std::ifstream fout;

fout.open(inputFileName, std::ios::in);

while (fout >> x) {

std::cout << x << " ";

}

std::cout << std::endl;

fout.close();

}

void addNewLine(std::ofstream& fout, const char\* inputFileName) {

int x = 0;

std::cout << "Введите элемент, который хотите добавить: ";

std::cin >> x;

fout.open(inputFileName, std::ios::out | std::ios::app);

fout << x;

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

fout.close();

}

void findNumbers(const char\* inputFileName) {

int nums=0;

std::cout << "Введите порядковый номер в файле: ";

std::cin >> nums;

std::ifstream fout;

fout.open(inputFileName);

int x;

int i;

std::cout << "> ";

for (i = 1; (i < nums && (!fout.eof())); i++) {

fout >> x;

fout.get();

}

while (!fout.eof() && (i == nums)) {

fout >> x;

fout.get();

std::cout << x;

i++;

}

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

fout.close();

}

void length\_file(const char\* inputFileName) {

std::ifstream fout;

fout.open(inputFileName);

int n = 0;

int x;

std::cout << std::endl;

while (!fout.eof()) {

fout >> x;

fout.get();

n++;

}

std::cout << "Количество чисел: " << n << std::endl;

fout.close();

}

void divideOddNumbersByAverage(const char\* inputFileName, const char\* outputFileName) {

std::cout << " Открытие и заполнение второго файла: \n";

std::ifstream inputFile(inputFileName);

if (!inputFile.is\_open()) {

std::cout << "Ошибка открытия исходного файла\n";

return;

}

int number;

int count = 0;

double sum = 0.0;

// Вычисляем среднее арифметическое

while (inputFile >> number) {

sum += number;

count++;

}

double average = sum / count;

// Переходим к началу файла

inputFile.clear();

inputFile.seekg(0, std::ios::beg);

// Открываем новый файл для записи

std::ofstream outputFile(outputFileName);

if (!outputFile.is\_open()) {

std::cout << "Ошибка открытия файла для записи\n";

inputFile.close();

return;

}

while (inputFile >> number) {

// Если число нечетное по абсолютной величине, делим его на среднее арифметическое и записываем в новый файл

if (std::abs(number) % 2 == 1) {

double dividedValue = number / average;

outputFile << dividedValue << " ";

std::cout << dividedValue << " ";

}

else {

outputFile << number << " ";

}

}

if (outputFile.good()) {

std::cout << "\n Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

inputFile.close();

outputFile.close();

std::cout << "\n///Завершение программы///\n";

}

## **Тестирование программы**

while (!fout.eof() && (i == nums)) {

fout >> x;

fout.get();

std::cout << x;

i++;

}

if (fout.good()) {

std::cout << " Ошибок не возникло\n";

}

fout.close();

}

void length\_file(const char\* inputFileName) {

std::ifstream fout;

fout.open(inputFileName);

int n = 0;

int x;

std::cout << std::endl;

while (!fout.eof()) {

fout >> x;

fout.get();

n++;

}

std::cout << "Количество чисел: " << n << std::endl;

fout.close();

}

void divideOddNumbersByAverage(const char\* inputFileName, const char\* outputFileName) {

std::cout << " Открытие и заполнение второго файла: \n";

std::ifstream inputFile(inputFileName);

if (!inputFile.is\_open()) {

std::cout << "Ошибка открытия исходного файла\n";

return;

}

int number;

int count = 0;

double sum = 0.0;

// Вычисляем среднее арифметическое

while (inputFile >> number) {

sum += number;

count++;

}

double average = sum / count;

// Переходим к началу файла

inputFile.clear();

inputFile.seekg(0, std::ios::beg);

// Открываем новый файл для записи

std::ofstream outputFile(outputFileName);

if (!outputFile.is\_open()) {

std::cout << "Ошибка открытия файла для записи\n";

inputFile.close();

return;

}

while (inputFile >> number) {

// Если число нечетное по абсолютной величине, делим его на среднее арифметическое и записываем в новый файл

if (std::abs(number) % 2 == 1) {

double dividedValue = number / average;

outputFile << dividedValue << " ";

std::cout << dividedValue << " ";

}

else {

outputFile << number << " ";

}

}

if (outputFile.good()) {

std::cout << "\n Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

}

inputFile.close();

outputFile.close();

std::cout << "\n///Завершение программы///\n";

}

if (std::abs(number) % 2 == 1) {

double dividedValue = number / average;

outputFile << dividedValue << " ";

std::cout << dividedValue << " ";

}

else {

outputFile << number << " ";

}

}

if (outputFile.good()) {

std::cout << "\n Ошибок не возникло\n";

}

else {

std::cout << " Ошибки возникли\n";

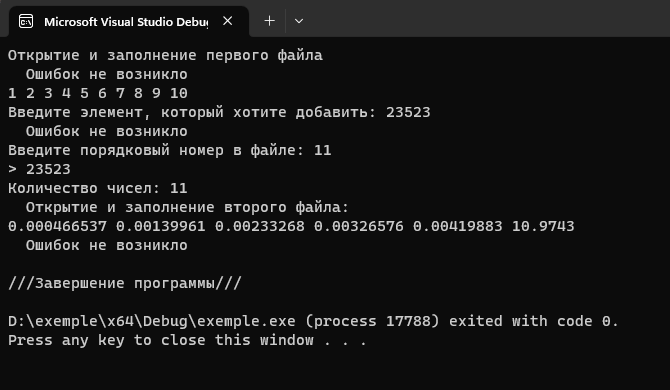
}

inputFile.close();

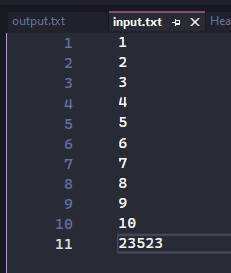
outputFile.close();

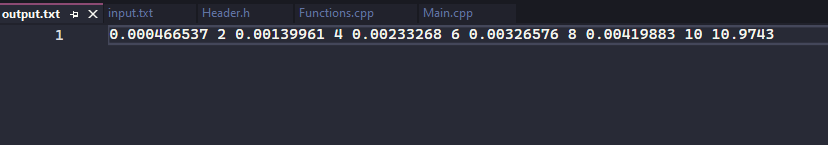
std::cout << "\n///Завершение программы///\n";

}



*Вывод программы*





*Содержание файлов*

# **Задача №2**

Разработать программу управление двоичными файлами с записями фиксированной длины

Общие требования: 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. Примечание. Реализация операций по чтению данных из файла будет проще, если значение для каждого поля записи размещать на отдельной строке текстового редактора.
      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;

* + - 1. Сохраните функции в новом модуле.
      2. Разработать приложение, демонстрирующее выполнение всех операций, подключив к нему модуль с функциями.
      3. Выполнить тестирование приложения, продемонстрировав выполнение всех операций.

## **3.1. Код программы.**

Индивидуальный вариант №11.

|  |  |
| --- | --- |
| Структура записи | Железнодорожная справка: номер поезда, пункт  отправления, пункт назначения, время отправления. |
| Доп. операция | 1. Сформировать справку по поездам,  отправляющимся в указанный пункт назначения.  Результаты записать в новый двоичный файл  такой же структуры, как и исходный.  2. Удалить сведения по указанному поезду |

Показаны структура Railway, имеющая 4 поля (number типа char, out типа char, in типа char и time типа char) и функция ConvertToBinary, которая переносит данные из текстового варианта в бинарный.

#include <iostream>

#include <fstream>

#include <cstring>

#include <vector>

using namespace std;

struct Railway {

char number[11]; // номер поезда

char out[43]; // пункт отправления

char in[23]; // пункт назначения

char time[6]; // время отправления

};

void ConvertToBinary(const char\* textFileName, const char\* binaryFileName) {

ifstream inputFile(textFileName);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия текстового файла" << endl;

return;

}

ofstream outputFile(binaryFileName, ios::binary);

if (!outputFile.is\_open()) {

cerr << "Ошибка создания двоичного файла" << endl;

return;

}

Railway record;

while (inputFile >> record.number >> record.out >> record.in >> record.time) {

outputFile.write(reinterpret\_cast<const char\*>(&record), sizeof(Railway));

}

inputFile.close();

outputFile.close();

cout << "Преобразование завершено" << endl;

}

*структура Railwayи функция ConvertToBinary.*

Функция SaveToText переносит данные из текстового варианта в бинарный. Функция PrintAllRecords выводит данные на экран.

void SaveToText(const char\* binaryFileName, const char\* textFileName) {

ifstream inputFile(binaryFileName, ios::binary);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия двоичного файла" << endl;

return;

}

ofstream outputFile(textFileName);

if (!outputFile.is\_open()) {

cerr << "Ошибка создания текстового файла" << endl;

return;

}

Railway record;

while (inputFile.read(reinterpret\_cast<char\*>(&record), sizeof(Railway))) {

outputFile << record.number << " " << record.out << " " << record.in << " " << record.time << endl;

}

inputFile.close();

outputFile.close();

cout << "Сохранение в текстовый файл завершено" << endl;

}

void PrintAllRecords(const char\* binaryFileName) {

ifstream inputFile(binaryFileName, ios::binary);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия двоичного файла" << endl;

return;

}

Railway record;

while (inputFile.read(reinterpret\_cast<char\*>(&record), sizeof(Railway))) {

cout << "Номер поезда: " << record.number << ", Пункт отправления: " << record.out << ", Пункт назначения: " << record.in << ", Время: " << record.time << endl;

}

inputFile.close();

}

*функции SaveToText и PrintAllRecords.*

Функция AccessRecordByIndex позволяет получить доступ к записи по индексу (рис. 12).

void AccessRecordByIndex(const char\* binaryFileName, int index) {

ifstream inputFile(binaryFileName, ios::binary);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия двоичного файла" << endl;

return;

}

Railway record;

inputFile.seekg(index \* sizeof(Railway));

if (inputFile.read(reinterpret\_cast<char\*>(&record), sizeof(Railway))) {

cout << "Номер поезда: " << record.number << ", Пункт отправления: " << record.out << ", Пункт назначения: " << record.in << ", Время: " << record.time << endl;

}

else {

cout << "Запись с указанным номером не найдена" << endl;

}

inputFile.close();

}

*функция AccessRecordByIndex.*

Функция DeleteRecordByName позволяет удалить запись по ключу.

void DeleteRecordByName(const char\* binaryFileName, const char\* outToDelete) {

ifstream inputFile(binaryFileName, ios::binary);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия двоичного файла" << endl;

return;

}

ofstream tempFile("temp.dat", ios::binary);

if (!tempFile.is\_open()) {

cerr << "Ошибка создания временного файла" << endl;

return;

}

Railway record;

bool recordFound = false;

while (inputFile.read(reinterpret\_cast<char\*>(&record), sizeof(Railway))) {

if (strcmp(record.number, outToDelete) == 0) {

recordFound = true;

}

else {

tempFile.write(reinterpret\_cast<const char\*>(&record), sizeof(Railway));

}

}

inputFile.close();

tempFile.close();

if (recordFound) {

remove(binaryFileName);

rename("temp.dat", binaryFileName);

cout << "Запись с именем '" << outToDelete << "' удалена" << endl;

}

else {

remove("temp.dat");

cout << "Запись с именем '" << outToDelete << "' не найдена" << endl;

}

}

*функция DeleteRecordByName.*

Функция ListRailsByIn формирует список поездов, прибывающих в указанное место.

void ListRailsByIn(const char\* binaryFileName, const char\* in) {

ifstream inputFile(binaryFileName, ios::binary);

if (!inputFile.is\_open()) {

cerr << "Ошибка открытия двоичного файла" << endl;

return;

}

Railway record;

bool found = false;

while (inputFile.read(reinterpret\_cast<char\*>(&record), sizeof(Railway))) {

if (strcmp(record.in, in) == 0) {

cout << "Номер поезда: " << record.number << ", Пункт отправления: " << record.out << ", Пункт назначения: " << record.in << ", Время: " << record.time << endl;

found = true;

}

}

inputFile.close();

if (!found) {

cout << "Поездов, отправляющихся в указанный пункт, не найдено" << endl;

}

}<< "' не найдена" << endl;

}

}

*функция ListRailsByIn.*

int main() {

setlocale(LC\_ALL, "RUS");

const char\* binaryFileName = "Railway.dat";

char textFileName[100]; // Буфер для хранения имени текстового файла

cout << "Введите имя текстового файла: ";

cin >> textFileName;

int choice;

while (true) {

cout << "Выберите операцию:" << endl;

cout << "1. Преобразовать текстовый файл в двоичный" << endl;

cout << "2. Сохранить двоичный файл в текстовый" << endl;

cout << "3. Вывести все записи из двоичного файла" << endl;

cout << "4. Доступ к записи по номеру" << endl;

cout << "5. Удалить сведения по указанному поезду" << endl;

cout << "6. Сформировать справку по поездам, отправляющимся в указанный пункт назначения" << endl;

cout << "7. Выход" << endl;

cout << "Введите номер операции: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1:

ConvertToBinary(textFileName, binaryFileName);

break;

case 2:

SaveToText(binaryFileName, textFileName);

break;

case 3:

PrintAllRecords(binaryFileName);

break;

case 4:

int index;

cout << "Введите номер записи: ";

cin >> index;

AccessRecordByIndex(binaryFileName, index);

break;

case 5:

char outToDelete[50];

cout << "Введите номер поезда для удаления: ";

cin.ignore(); // Очистка буфера ввода перед чтением строки

cin.getline(outToDelete, sizeof(outToDelete));

DeleteRecordByName(binaryFileName, outToDelete);

break;

case 6:

char in[3];

cout << "Введите пункт (5 букв): ";

cin >> in;

ListRailsByIn(binaryFileName, in);

break;

case 7:

return 0;

default:

cout << "Неверный выбор. Пожалуйста, выберите действие из списка." << endl;

break;

}

cout << endl;

}

return 0;

}

*функция main .*

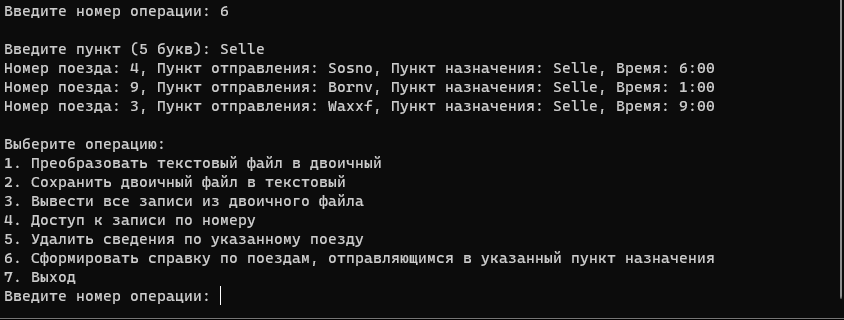
## **3.2. Тестирование программы.**

**

*вывод программы.*

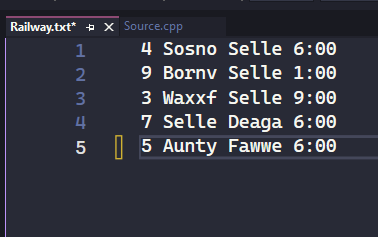


*вывод программы.*

**

*вывод программы.*

## **3.3. Входные данные программы.**



*входные данные программы.*