|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ**

**ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_Поляков Данила Денисович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_\_\_\_ИУ5-13Б\_\_\_\_\_\_\_\_\_\_\_\_

Тип практики ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА \_\_\_

Название предприятия КАФЕДРА «СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ» ( ИУ5 ) МГТУ им. Н.Э.Баумана

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_**Поляков Д.Д.**\_\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_**Правдина А.Д.**\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2021 г.*

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Написать программу учета трамвайных маршрутов в некотором городе.

Сведения о каждом маршруте содержат: номер маршрута, протяжённость маршрута в километрах, время в пути.

Программа должна обеспечивать выбор с помощью меню и выполнение следующих функций:

• выдача сведений о всех трамвайных маршрутах города, упорядоченных по номеру;

• добавление данных о новом маршруте;

• изменение данных о маршруте;

• удаление данных об отменённых маршрутах;

• выдача сведений о всех маршрутах города, отсортированных по номеру по убыванию;

• поиск маршрута по номеру;

• сохранение (чтение) данных о маршрутах города в (из) файл с расширением .qqq .

ОГЛАВЛЕНИЕ

1. Введение
2. Техническое задание
3. Руководство пользователя
4. Исходный текст программы
5. Программа и методика испытаний
6. Список используемых источников

ВВЕДЕНИЕ

Целью и задачей данной проектно-технологической практики является освоение процедурного программирования на языке С++, углубление соответствующих навыков и дополнение теоретических знаний по дисциплине «Основы программирования», а также приобретение умения разрабатывать и оформлять техническую документацию на программный продукт.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Московский государственный технический университет имени

Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  \_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Правдина  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета трамвайных маршрутов городских транспортных служб»**

Техническое задание

5 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-13Б Поляков Д.Д.\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

Москва, 2021

**СОДЕРЖАНИЕ**

1. Наименование

2. Основание для разработки

3. Назначение разработки

4. Исполнитель

5. Технические требования

5.1. Требования к функциональным характеристикам

5.2. Требования к программному обеспечению

5.3. Требования к условиям эксплуатации

5.4. Требования к информационному обеспечению

5.5. Требования к надежности

5.6. Требования к составу и характеристикам технических средств

5.7. Требования к программной совместимости

5.8. Требования к маркировке и упаковке программы

5.9. Требования к транспортировке и хранению

6. Требования к программной документации

7. Технико-экономические показатели

8. Стадии и этапы разработки

9. Порядок контроля и приемки задания

10. Дополнительные требования

# Наименование

Информационная система для автоматизации учета трамвайных маршрутов транспортных служб города. Шифр разработки программного продукта – TRCRUDqqq

# Основание для разработки

Основанием для разработки данного программного продукта является учебный план кафедры "Системы обработки информации и управления" МГТУ им. Н. Э. Баумана на 1-м семестре (дисциплина «Основы программирования»).

# Назначение разработки

Данный программный продукт предназначен для автоматизации учета трамвайных маршрутов для транспортных организаций. В частности программа должна обеспечивать решение задач: хранение данных программы в долговременной памяти, чтение данных из долговременной памяти в оперативную, просмотр данных о маршрутах организации с возможностью просмотра, добавление данных о новых маршрутах, удаление данных о маршруте, изменение данных о маршруте. Программа должна обеспечивать удобную работу с данными, высокий уровень надежности и функциональных возможностей.

# Исполнитель

Студент группы ИУ5-13Б МГТУ им. Баумана Поляков Данила Денисович, вариант № 12.

# Технические требования

## 5.1. Требования к функциональным характеристикам

**5.1.1. Общие требования**

5.1.1.1. Интерфейс программы должен быть реализован в консоли.

5.1.1.2. Интерфейс программы должен быть полностью на русском языке, а также максимально интуитивным и понятным, каждое действие должно сопровождаться пояснением.

5.1.1.3. На протяжении всей работы программы должно отображаться меню, позволяющее пользователю выбрать необходимое ему действие.

5.1.1.4. Ниже по тексту, данные о одном трамвайном маршруте называются записью.

5.1.1.5. Интерфейс программы должен быть реализован в виде пронумерованного меню с выбором действий (меню пользователя). Меню пользователя должно содержать следующие пункты:

– Просмотр данных;

– Добавление данных;

– Изменение данных;

– Удаление данных;

– Поиск данных;

– Сохранение данных;

– Завершить работу.

5.1.1.6. Программа должна игнорировать некорректно введенные данные и сообщать об ошибке пользователю.

5.1.1.7. При запуске программы должно происходить чтение данных о трамвайных маршрутах транспортной службы в оперативную память из текстового файла, который должен находиться в текущем каталоге.

5.1.1.8. Если при запуске программы файл с данными в текущем каталоге отсутствует, то программа должна продолжать работу, и пользователь должен иметь возможность заново сформировать данные в программе.

**5.1.2. Основные функции.**

Программа должна обеспечивать выполнение следующих функций:

5.1.2.1. **Просмотр данных**.

Вывод данных в консоль должен быть отформатирован и отсортирован. При выполнении пункта меню «Просмотр данных» список должен быть отсортирован по возрастанию, по номеру маршрута. Сортировка должна производиться посимвольно с учетом регистра.

5.1.2.2. **Добавление данных**.

#### При добавлении данных формат вводимых данных должен быть в следующем виде: «Номер Длина Время», где Номер – номер добавляемого маршрута, Длина – длина нового маршрута, Время – время в пути.

5.1.2.3. **Изменение данных**. При изменении данных о маршруте пользователя попросят ввести номер маршрута для редактирования, далее будет выведена информация об изменяемом пользователем маршруте и просьба ввести новые данные.

Окончание ввода записей должно осуществляться посредствам нажатия клавиши Enter на новой строке.

5.1.2.4. **Удаление данных**. При удалении данных о маршруте программа потребует от пользователя ввести номер удаляемого маршрута и сообщит пользователю о том, какая запись будет удалена. В случае, если маршрута с таким номером не будет найдено, программа уведомит об этом пользователя и запросит повторный ввод номера.

5.1.2.5. **Поиск данных**.

В программе должен быть реализован поиск записи по номеру маршрута и вывод найденной записи на экран.

5.1.2.6. **Сохранение данных**. В программе должно быть реализовано сохранение данных в текстовый файл.

## 5.2. Требования к программному обеспечению

5.2.1. Данная программа предназначена для выполнения на компьютере под управлением системы macOS версии BigSur 11.6 и выше. Для использования данной программы требуется наличие на компьютере компилятора С++ под архитектуру процессоров ARM.

## 5.3. Требования к условиям эксплуатации

5.3.1. Данная программа должна эксплуатироваться на компьютере под управлением системы macOS версии BigSur 11.6 и выше.

5.3.2. Для сопровождения программы системный программист должен быть знаком с языком C++ в любой среде программирования. Для работы с данной программой системный программист должен быть знаком с навыками объектно-ориентированного программирования, знать структуры данных STL С++ и уметь работать с файловыми потоками ввода и вывода.

## 5.4. Требования к информационному обеспечению

5.4.1. Программа должна быть реализована на языке С++.

5.4.2. Информация о структурах данных программы должна быть представлена в техническом описании.

5.4.3. Программа должна быть представлена набором файлом с расширениями .cpp и .h.

5.4.4. Допускается использование английского языка для работы с программой.

5.4.5. Программа должна работать в консольном режиме.

## 5.5. Требования к надежности

5.5.1. Программа должна безотказно работать при отсутствии сбоев операционной системы и технического обеспечения.

5.5.2. В программе не допускается выделение динамической памяти без последующего её освобождения.

## 5.6. Требования к составу и характеристикам технических средств

Данная программа должна выполняться на компьютерах следующей конфигурации:

5.6.1. Компьютер с процессором Apple M1 Silicon и выше;

5.6.2. Не менее 2гб свободной оперативной памяти;

5.6.3. Монитор с VGA выходом;

5.6.4. Стандартная клавиатура;

5.6.5. Свободное место на жестком диске не менее 450 Кбайт.

## 5.7. Требования к программной совместимости

Для сопровождения программы рекомендуется к использованию программа JetBrains CLion, компьютер должен быть оснащен русской таблицей символов и кодовой поддержкой.

## 5.8. Требования к маркировке и упаковке программы

Программа предоставляется на флэш-носителе или другом сменном носителе информации.

## 5.9. Требования к транспортировке и хранению

Программа транспортируется и хранится на флэш-носителе или другом сменном носителе информации.

# Требования к программной документации

6.1. По окончанию работы должны быть предъявлены следующие документы:

6.1.1. Техническое задание;

6.1.2. Программа и методика испытаний;

6.1.3. Руководство пользователя;

6.1.4. Исходный текст программы.

# 7. Технико-экономические показатели

7.1. Требования по данным показателям при выполнении задания проектно-технологичеcкой практики не предъявляются.

# 8. Стадии и этапы разработки

8.1. Сроки выполнения отдельных этапов работ

8.1.1. Получение и уточнение задания – 12 неделя семестра.

8.1.2. Подписание ТЗ – 13 неделя семестра.

8.1.3. Разработка программы – 14 неделя семестра.

8.1.4. Кодирование и отладка программы – 15 неделя семестра.

8.1.5. Разработка документации – 16 неделя семестра.

8.1.6. Защита и проведение испытаний – 17 неделя семестра.

# 9. Порядок контроля и приемки задания

9.1. Тестирование программного продукта будет осуществляться в соответствии с документом «Программа и методика испытаний» (ПМИ) на компьютере, который удовлетворяет пункту 5.6.1.

9.2. Контроль и приемка программы и программной документации производится дистанционно в программе Discord.

9.3. Для проверки и тестирования программы предоставляется исполняемый модуль для UNIX-like операционных систем и среда разработки JetBrains CLion с установленным компилятором MinGW

# 10. Дополнительные требования

Данное техническое задание может уточняться в процессе разработки.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Московский государственный технический университет имени

Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  \_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Правдина  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета трамвайных маршрутов городских транспортных служб»**

Руководство пользователя

7 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-13Б Поляков Д.Д.\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

Москва, 2021

**СОДЕРЖАНИЕ**

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

2. ПОДКЛЮЧЕНИЕ И ЗАПУСК ПРОГРАММЫ

3. РАБОТА С ПРОГРАММОЙ

3.1. Чтение данных из файла и вывод их на экран

3.2. Добавление новых данных в список

3.3. Изменение данных в списке

3.4. Удаление записей из списка

3.5. Поиск по имени компьютера

3.6. Запись списка в файл

3.7. Выход из программы

4. ОПИСАНИЕ СООБЩЕНИЙ ОБ ОШИБКАХ И ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦИЯХ ПРИ РАБОТЕ С ПРОГРАММОЙ

4.1. Неправильный формат ввода

4.2. Запись не найдена

4.3 Файл с данными неисправен или не найден

# НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

Целью работы является создание программы для автоматизации учета трамвайных маршрутов служб общественного транспорта. В частности программа обеспечивает решение следующих задач: сохранение данных программы в долговременной памяти; чтение данных из долговременной памяти в оперативную; просмотр отсортированных по номеру маршрута данных о маршрутах; добавление данных о новых маршрутах; удаление данных о маршрутах; изменение данных о маршруте. Программа обеспечивает удобную работу с данными, высокий уровень надежности и функциональных возможностей.

# ПОДКЛЮЧЕНИЕ и запуск ПРОГРАММЫ

Для установки данной программы на компьютер следует выполнить следующие действия:

2.1. Освободить на жестком диске как минимум 450 килобайт свободного пространства.

2.2. Вставить флэш-носитель (или другой сменный носитель информации) с исходным модулем программы в USB-порт.

2.3. Скопировать файл программы, который указан в разделе 4 этого документа, с носителя в нужный каталог.

2.4. Для начала работы с программой необходимо выполнить файл TRCRUDqqq.

# 3. РАБОТА c ПРОГРАММой

После первого запуска программы в консоли появляется меню пользователя (рис.1).

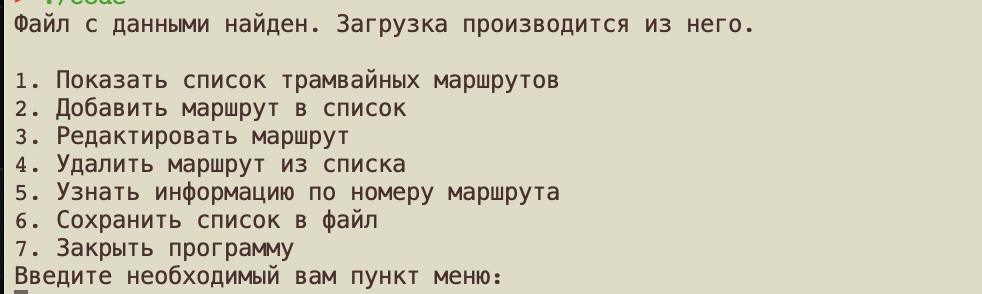


Рис.1.

## 3.1. ПРОСМОТР СУЩЕСТВУЮЩИХ ДАННЫХ

После появления меню пользователя (рис.3) введите «1». В консоль будет выведен список трамвайных маршрутов, загруженных из файла базы данных. (рис. 2).



Рис.2.

**Выход в главное меню:** после того, как выбранный вами пункт меню закончит работу, снова будет выведено главное меню с просьбой выбрать необходимый вам пункт (рис. 3).

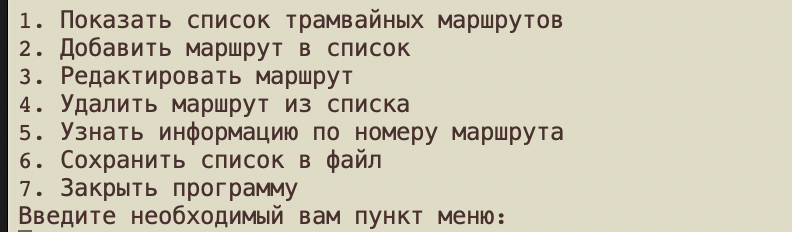


Рис.3.

## 3.2. ДОБАВЛЕНИЕ ДАННЫХ В СПИСОК

После появления меню пользователя (рис.3) введите «2». В консоль будет выведены по очереди сообщения о необходимости ввода определённых данных. (рис. 4).

Обращаем ваше внимание, что для ввода букв в номере маршрута рекомендуется использовать английские символы.

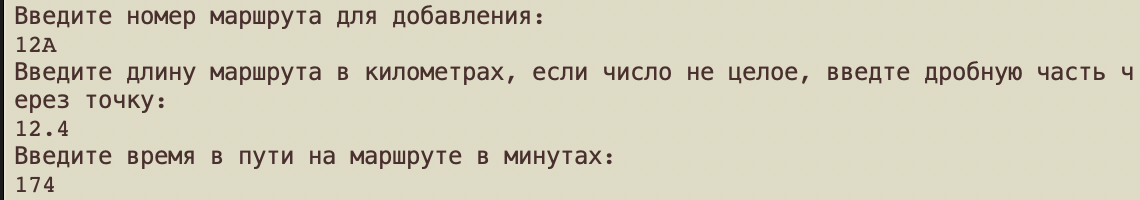


Рис. 4.

После успешного добавления маршрута будет выведена информация о нём (рис. 5) и выведено меню (рис. 3). Сам маршрут запишется в список, который можно посмотреть, выбрав первый пункт меню.

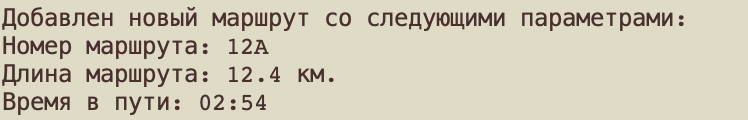


Рис. 5.

## 3.3. ИЗМЕНЕНИЕ ДАННЫХ В СПИСКЕ

После появления меню пользователя (рис.3) введите «3». Введите номер маршрута, который вы желаете редактировать, после чего в консоль будет выведена информация о редактируемом маршруте. (рис. 6).

Далее будут выведены предложения ввести новые данные для выбранного маршрута. После ввода новых данных в консоль выведется информация об изменениях (рис. 7) и меню (рис. 3).

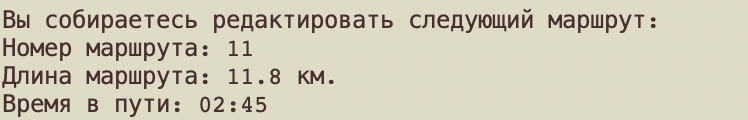


Рис. 6.

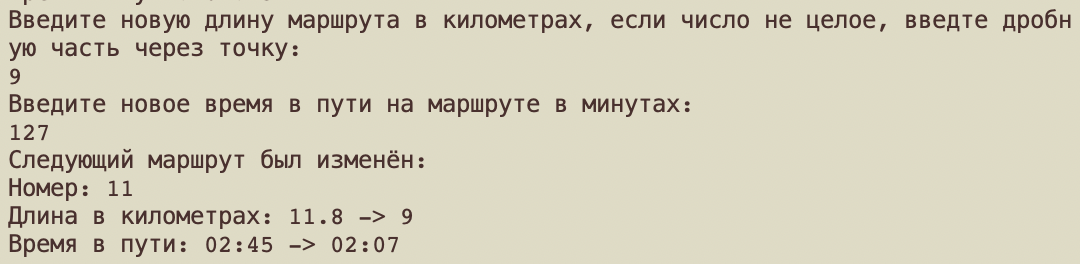


Рис 7.

# 3.4. УДАЛЕНИЕ ДАННЫХ ИЗ СПИСКА

После появления меню пользователя (рис.3) введите «4». Далее последует просьба ввести номер маршрута, который вы желаете удалить. После чего будет выведено сообщение об его успешном удалении (рис. 8) и в списке он значится не будет.

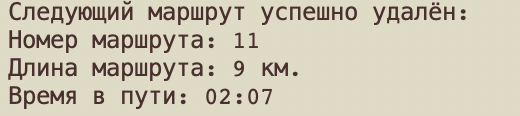


Рис. 8.

## 3.5. ПОИСК МАРШРУТА ПО НОМЕРУ В СПИСКЕ

После появления меню пользователя (рис.3) введите «5». Далее последует просьба ввести номер маршрута, информацию о котором вы желаете получить. После ввода номера маршрута информация о нём будет выведена (рис. 9).

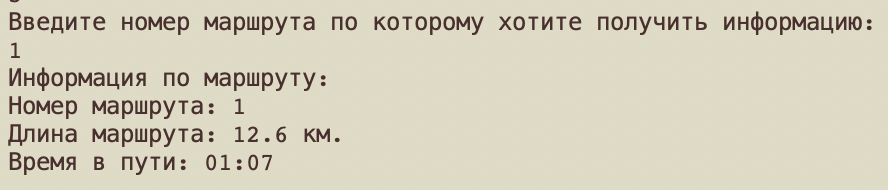


Рис. 9.

## ЗАПИСЬ СПИСКА В ФАЙЛ

После появления меню пользователя (рис.3) введите «5». Список сохранится в файл и будет выведено сообщение об успехе операции (рис. 10).



Рис. 10.

## ВЫХОД ИЗ ПРОГРАММЫ

Для выхода из программы выберите пункт меню (рис.3).

# ОПИСАНИЕ СООБЩЕНИЙ ОБ ОШИБКАХ И ИСКЛЮЧИТЕЛЬНЫХ СИТУАЦиях ПРИ РАБОТЕ С ПРОГРАММОЙ

## 4.1. Неправильный формат ввода (для разделов добавления и изменения данных в списке)

Если вы столкнулись с данной ошибкой – проверьте, удовлетворяют ли введенные данные допустимым значениям. Сообщение об ошибке будет выведено на экран, после чего программа предложит ввести данные повторно (рис. 11).

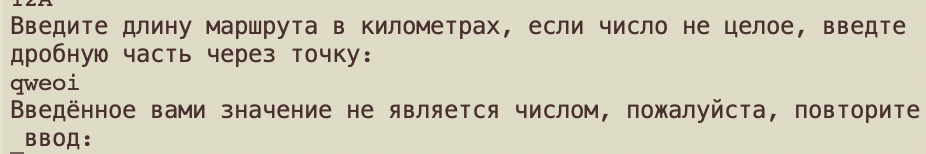
****

Рис. 11.

**Номер маршрута:** допускаются буквы русского и латинского алфавита, а так же цифры 0..9, но для букв рекомендуется использовать латиницу.

**Длина маршрута:** допускаются числа с точкой и дробной частью, либо целые.

**Время в пути:** допускаются целые числа.

## 4.2. Запись не найдена (для разделов редактирования и удаления данных в списке)

Если вы встретили данное сообщение при работе с программой, то введённый вами номер маршрута отсутствует в списке. Об этом будет сообщено и предложено повторно ввести номер (рис. 12)

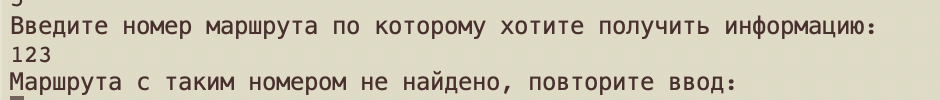
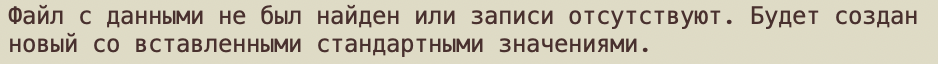


Рис. 12.

## 4.3. Файл с данными неисправен или не найден (ошибка возможна при запуске программы)

Если вы встретили данное сообщение при работе с программой, то файл базы данных либо пуст, либо не найден в директории программы, либо повреждён. При данной ошибке программа сама создаст новый файл и заполнит его стандартными данными, предупредив об этом пользователя (рис. 13)

 Рис. 13.

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Московский государственный технический университет имени

Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  \_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Правдина  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета трамвайных маршрутов городских транспортных служб»**

Исходный текст программы

11 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-13Б Поляков Д.Д.\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

Москва, 2021

**Содержимое**

1. Файл imports.h

2. Файл functions.h

3. Файл functions.cpp

4. Файл main.cpp

**Файл imports.h:**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>*#ifndef **CODE\_IMPORTS\_H**#define **CODE\_IMPORTS\_H**#define **FILE\_NAME "database.qqq"**#include **<string>**#include **<fstream>**#include **<iomanip>**#include **<iostream>**#include **<algorithm>**#include **<set>**#include **<cmath>  
  
using** std::cin, std::cout, std::ifstream, std::ofstream, std::string;  
  
**struct** TramRoute{  
 string Number;  
 **double** Length;  
 **int** RoadTime; *//in minutes*};  
  
**enum** MenuItem {  
 ***ShowRoutes*** = 1,  
 ***AddRoute***,  
 ***EditRoute***,  
 ***DeleteRoute***,  
 ***GetInfo***,  
 ***SaveToFile***,  
 ***Exit***};  
  
#endif *//CODE\_IMPORTS\_H*

**Файл functions.h:**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>*#ifndef **CODE\_FUNCTIONS\_H**#define **CODE\_FUNCTIONS\_H**#include **"imports.h"  
  
bool operator**<(**const** TramRoute &route1, **const** TramRoute &route2);  
  
**void** SeedData(std::set<TramRoute> &);  
  
**void** FillDataFromFile(std::set<TramRoute> &, ifstream &);  
  
**void** PrintMenu();  
  
**void** Menu(std::set<TramRoute> &);  
  
**void** ShowRoutesList(**const** std::set<TramRoute> &);  
  
**void** InsertRoute(std::set<TramRoute> &);  
  
**void** EditRouteByName(std::set<TramRoute> &);  
  
**void** EraseRoute(std::set<TramRoute> &);  
  
**void** GetRoute(**const** std::set<TramRoute> &);  
  
**void** SaveRoutes(**const** std::set<TramRoute>&);  
  
#endif *//CODE\_FUNCTIONS\_H*

**Файл functions.cpp**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>*#include **"functions.h"  
  
bool operator**<(**const** TramRoute &route1, **const** TramRoute &route2) {  
 **return** route1.Number < route2.Number;  
}  
  
**void** SeedData(std::set<TramRoute> &arrayToFill) {  
 **const** TramRoute defaultRoutes[] = {{**"17A"**, 13.2, 184},  
 {**"47"**, 8, 140},  
 {**"11"**, 11.8, 165},  
 {**"167"**, 3.8, 38},  
 {**"15"**, 27.1, 273},  
 {**"174K"**, 19.2, 204},  
 {**"812"**, 10.6, 120},  
 {**"83A"**, 10.1, 117},  
 {**"12A"**, 17, 189},  
 {**"74K"**, 9, 98},  
 {**"87A"**, 1.2, 12},  
 {**"87"**, 13, 132},  
 {**"1"**, 14.5, 154},  
 };  
 ofstream ofFile(**FILE\_NAME**);  
 **for** (**const auto** &route: defaultRoutes) {  
 arrayToFill.insert(route);  
 ofFile.write((**char** \*) &route, **sizeof**(TramRoute));  
 }  
 ofFile.close();  
}  
  
**void** FillDataFromFile(std::set<TramRoute> &routes, ifstream &file) {  
 TramRoute tempRoute;  
 **while** (!file.eof()) {  
 file.read((**char** \*) &tempRoute, **sizeof**(TramRoute));  
 routes.insert(tempRoute);}  
}  
  
**void** PrintMenu() {  
 **const char** \*menu[7] = {**"Показать список трамвайных маршрутов"**,  
 **"Добавить маршрут в список"**,  
 **"Редактировать маршрут"**,  
 **"Удалить маршрут из списка"**,  
 **"Узнать информацию по номеру маршрута"**,  
 **"Сохранить список в файл"**,  
 **"Закрыть программу"**};  
 cout << **'\n'**;  
 **for** (size\_t i = 0; i < 7; ++i) {  
 cout << i + 1 << **". "** << menu[i] << **'\n'**;  
 }  
 cout << **"Введите необходимый вам пункт меню:\n"**;  
}  
  
**void** Menu(std::set<TramRoute> &routes) {  
 **unsigned** menuPoint;  
 **while** (**true**) {  
 PrintMenu();  
 cin >> menuPoint;  
 **switch** (menuPoint) {  
 **case *ShowRoutes***:  
 ShowRoutesList(routes);  
 **break**;  
 **case *AddRoute***:  
 InsertRoute(routes);  
 **break**;  
 **case *EditRoute***:  
 EditRouteByName(routes);  
 **break**;  
 **case *DeleteRoute***:  
 EraseRoute(routes);  
 **break**;  
 **case *GetInfo***:  
 GetRoute(routes);  
 **break**;  
 **case *SaveToFile***:  
 SaveRoutes(routes);  
 **break**;  
 **case *Exit***:  
 **return**;  
 **default**:  
 **break**;  
 }  
 }  
}  
  
**void** ShowRoutesList(**const** std::set<TramRoute> &routes) {cout << **' '**;  
 **for** (size\_t i = 0; i < 54; ++i) {  
 cout << **"\_"**;  
 }  
 cout << **'\n'**;cout << std::left << std::setw(3) << **'|'** << **"Номер маршрута"** << std::setw(3) << std::right << **'|'** << std::setw(strlen(**"Протяжённость"**) + 3) << **"Протяжённость"** << std::setw(3) << **'|'** << std::setw(  
 strlen(**"Время в пути"**) + 2) << **"Время в пути"** << std::setw(3) << **'|'** << **'\n'**;  
 cout << **' '**;  
 **for** (size\_t i = 0; i < 54; ++i) {  
 cout << **"‾"**;  
 }  
 cout << **'\n'**;  
 **for** (**const auto** &route: routes) {  
 cout << **'|'** << std::setw(7) << **' '** << std::left << route.Number  
 << std::right << std::setw(12 - route.Number.length()) << **'|'** << std::left << std::setw(5) << **' '** <<  
 std::setw(5) << route.Length << **" км."** << std::right << std::setw(5) << **'|'**;  
 **if** (route.RoadTime / 60 < 10) {  
 cout << std::right  
 << std::setw(7) << **'0'** << route.RoadTime / 60;  
 } **else** {  
 cout << std::right  
 << std::setw(7) << route.RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** (route.RoadTime % 60 < 10) {  
 cout << **'0'** << route.RoadTime % 60;  
 } **else** {  
 cout << route.RoadTime % 60;  
 }  
 cout << std::setw(6) << **'|'**;  
 cout << **'\n'**;  
 }  
 cout << **' '**;  
 **for** (size\_t i = 0; i < 54; ++i) {  
 cout << **"‾"**;  
 }  
}  
  
**void** InsertRoute(std::set<TramRoute> &routes) {  
 TramRoute tempRoute;  
 cout << **"Введите номер маршрута для добавления:"** << std::endl;  
 cin >> tempRoute.Number;  
 cout << **"Введите длину маршрута в километрах, если число не целое, введте дробную часть через точку:"** << **'\n'**;  
 **bool** aux = **true**;  
 cin.exceptions(std::istream::failbit);  
 **do** {  
 **try** {  
 cin >> tempRoute.Length;  
 aux = **true**;  
 }  
 **catch** (std::ios\_base::failure &fail) {  
 aux = **false**;  
 cout << **"Введённое вами значение не является числом, пожалуйста, повторите ввод:"** << **'\n'**;  
 cin.clear();  
 std::string tmp;  
 getline(cin, tmp);  
 }  
 } **while** (!aux);  
  
 cout << **"Введите время в пути на маршруте в минутах:"** << **'\n'**;  
 aux = **true**;  
 cin.exceptions(std::istream::failbit);  
 **do** {  
 **try** {  
 cin >> tempRoute.RoadTime;  
 **if** (tempRoute.RoadTime <= 0) {  
 aux = **false**;  
 cout << **"Введённое вами значение не превосходит ноль, пожалуйста, повторите ввод:"** << **'\n'**;  
 } **else** aux = **true**;  
  
 }  
 **catch** (std::ios\_base::failure &fail) {  
 aux = **false**;  
 cout  
 << **"Введённое вами значение не является числом, пожалуйста, повторите ввод:"** << **'\n'**;  
 cin.clear();  
 std::string tmp;  
 getline(cin, tmp);  
 }  
 } **while** (!aux);  
  
 routes.insert(tempRoute);  
 cout << **"Добавлен новый маршрут со следующими параметрами:"** << **'\n'**;  
 cout << **"Номер маршрута: "** << tempRoute.Number << **'\n'**;  
 cout << **"Длина маршрута: "** << tempRoute.Length << **" км."** << **'\n'**;  
 cout << **"Время в пути: "**;  
 **if** (tempRoute.RoadTime / 60 < 10) {  
 cout << **'0'** << tempRoute.RoadTime / 60;  
 } **else** {  
 cout << tempRoute.RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** (tempRoute.RoadTime % 60 < 10) {  
 cout << **'0'** << tempRoute.RoadTime % 60;  
 } **else** {  
 cout << tempRoute.RoadTime % 60;  
 }  
 cout << **'\n'**;  
 cout << **'\n'**;  
}  
  
  
**void** EditRouteByName(std::set<TramRoute> &routes) {  
 string numberToEdit;  
 cout << **"Введите номер маршрута для редактирования:"** << **'\n'**;  
 cin >> numberToEdit;  
 **auto** c = std::find\_if(routes.begin(), routes.end(),  
 [numberToEdit](**const** TramRoute &route) {  
 **return** route.Number == numberToEdit;  
 });  
 **while** (c == routes.end()) {  
 cout << **"Маршрута с таким номером не найдено, повторите ввод:"** << **'\n'**;  
 cin >> numberToEdit;  
 c = std::find\_if(routes.begin(), routes.end(),  
 [numberToEdit](**const** TramRoute &route) {  
 **return** route.Number == numberToEdit;  
 });  
 }  
  
 cout << **"Вы собираетесь редактировать следующий маршрут:"** << **'\n'**;  
 cout << **"Номер маршрута: "** << (\*c).Number << **'\n'**;  
 cout << **"Длина маршрута: "** << (\*c).Length << **" км."** << **'\n'**;  
 cout << **"Время в пути: "**;  
 **if** ((\*c).RoadTime / 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime / 60;  
 } **else** {  
 cout << (\*c).RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** ((\*c).RoadTime % 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime % 60;  
 } **else** {  
 cout << (\*c).RoadTime % 60;  
 }  
 cout << **'\n'**;  
  
 TramRoute tempRoute;  
 tempRoute.Number = (\*c).Number;  
 cout << **"Введите новую длину маршрута в километрах, если число не целое, введте дробную часть через точку:"** << **'\n'**;  
 **bool** aux = **true**;  
 cin.exceptions(std::istream::failbit);  
 **do** {  
 **try** {  
 cin >> tempRoute.Length;  
 aux = **true**;  
 }  
 **catch** (std::ios\_base::failure &fail) {  
 aux = **false**;  
 cout << **"Введённое вами значение не является числом, пожалуйста, повторите ввод:"** << **'\n'**;  
 cin.clear();  
 std::string tmp;  
 getline(cin, tmp);  
 }  
 } **while** (!aux);  
  
 cout << **"Введите новое время в пути на маршруте в минутах:"** << **'\n'**;  
 aux = **true**;  
 cin.exceptions(std::istream::failbit);  
 **do** {  
 **try** {  
 cin >> tempRoute.RoadTime;  
 **if** (tempRoute.RoadTime <= 0) {  
 aux = **false**;  
 cout << **"Введённое вами значение не превосходит ноль, пожалуйста, повторите ввод:"** << **'\n'**;  
 } **else** aux = **true**;  
  
 }  
 **catch** (std::ios\_base::failure &fail) {  
 aux = **false**;  
 cout  
 << **"Введённое вами значение не является числом, пожалуйста, повторите ввод:"** << **'\n'**;  
 cin.clear();  
 std::string tmp;  
 getline(cin, tmp);  
 }  
 } **while** (!aux);  
  
 cout << **"Следующий маршрут был изменён:"** << **'\n'**;  
 cout << **"Номер: "** << (\*c).Number << **'\n'**;  
 cout << **"Длина в километрах: "**;  
 **if** ((\*c).Length == tempRoute.Length) {  
 cout << (\*c).Length;  
 } **else** {  
 cout << (\*c).Length << **" -> "** << tempRoute.Length;  
 }  
 cout << **'\n'**;  
  
 cout << **"Время в пути: "**;  
 **if** ((\*c).RoadTime == tempRoute.Length) {  
 **if** ((\*c).RoadTime / 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime / 60;  
 } **else** {  
 cout << (\*c).RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** ((\*c).RoadTime % 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime % 60;  
 } **else** {  
 cout << (\*c).RoadTime % 60;  
 }  
 cout << **'\n'**;  
 } **else** {  
 **if** ((\*c).RoadTime / 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime / 60;  
 } **else** {  
 cout << (\*c).RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** ((\*c).RoadTime % 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime % 60;  
 } **else** {  
 cout << (\*c).RoadTime % 60;  
 }  
 cout << **" -> "**;  
 **if** (tempRoute.RoadTime / 60 < 10) {  
 cout << **'0'** << tempRoute.RoadTime / 60;  
 } **else** {  
 cout << tempRoute.RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** (tempRoute.RoadTime % 60 < 10) {  
 cout << **'0'** << tempRoute.RoadTime % 60;  
 } **else** {  
 cout << tempRoute.RoadTime % 60;  
 }  
 }  
 cout << **'\n'**;  
  
 routes.erase(c);  
 routes.insert(tempRoute);}  
  
**void** EraseRoute(std::set<TramRoute> &routes) {  
 string numberToDelete;  
 cout << **"Введите номер маршрута для удаления:"** << **'\n'**;  
 cin >> numberToDelete;  
 **auto** c = std::find\_if(routes.begin(), routes.end(),  
 [numberToDelete](**const** TramRoute &route) {  
 **return** route.Number == numberToDelete;  
 });  
 **while** (c == routes.end()) {  
 cout << **"Маршрута с таким номером не найдено, повторите ввод:"** << **'\n'**;  
 cin >> numberToDelete;  
 c = std::find\_if(routes.begin(), routes.end(),  
 [numberToDelete](**const** TramRoute &route) {  
 **return** route.Number == numberToDelete;  
 });  
 }  
 cout << **"Следующий маршрут успешно удалён:"** << **'\n'**;  
 cout << **"Номер маршрута: "** << (\*c).Number << **'\n'**;  
 cout << **"Длина маршрута: "** << (\*c).Length << **" км."** << **'\n'**;  
 cout << **"Время в пути: "**;  
 **if** ((\*c).RoadTime / 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime / 60;  
 } **else** {  
 cout << (\*c).RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** ((\*c).RoadTime % 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime % 60;  
 } **else** {  
 cout << (\*c).RoadTime % 60;  
 }  
 cout << **'\n'**;  
 routes.erase(c);  
}  
  
**void** GetRoute(**const** std::set<TramRoute> &routes) {  
 string numberToFind;  
 cout << **"Введите номер маршрута по которому хотите получить информацию:"** << **'\n'**;  
 cin >> numberToFind;  
 **auto** c = std::find\_if(routes.begin(), routes.end(),  
 [numberToFind](**const** TramRoute &route) {  
 **return** route.Number == numberToFind;  
 });  
 **while** (c == routes.end()) {  
 cout << **"Маршрута с таким номером не найдено, повторите ввод:"** << **'\n'**;  
 cin >> numberToFind;  
 c = std::find\_if(routes.begin(), routes.end(),  
 [numberToFind](**const** TramRoute &route) {  
 **return** route.Number == numberToFind;  
 });  
 }  
 cout << **"Информация по маршруту:"** << **'\n'**;  
 cout << **"Номер маршрута: "** << (\*c).Number << **'\n'**;  
 cout << **"Длина маршрута: "** << (\*c).Length << **" км."** << **'\n'**;  
 cout << **"Время в пути: "**;  
 **if** ((\*c).RoadTime / 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime / 60;  
 } **else** {  
 cout << (\*c).RoadTime / 60;  
 }  
 cout << **':'**;  
 **if** ((\*c).RoadTime % 60 < 10) {  
 cout << **'0'** << (\*c).RoadTime % 60;  
 } **else** {  
 cout << (\*c).RoadTime % 60;  
 }  
 cout << **'\n'**;  
}  
  
**void** SaveRoutes(**const** std::set<TramRoute> &routes) {  
 ofstream ofFile(**FILE\_NAME**);  
 ofFile.clear();  
 **for** (**const auto** &route: routes) {  
 ofFile.write((**char** \*) &route, **sizeof**(TramRoute));  
 }  
 ofFile.close();  
 cout << **"Данные успешно сохранены в файл \""**<<**FILE\_NAME**<<**'\"'** << **'\n'**;  
}

**Файл main.cpp:**

#include **"functions.h"  
  
int** main() {  
 std::set<TramRoute> routesList;  
  
 ifstream dataFile(**FILE\_NAME**);  
 **if** (dataFile.bad() || dataFile.peek() == std::ifstream::traits\_type::eof()) {  
 cout  
 << **"Файл с данными не был найден или записи отсутствуют. Будет создан новый со вставленными стандартными значениями."** << **'\n'**;  
 SeedData(routesList);  
 } **else** {  
 cout << **"Файл с данными найден. Загрузка производится из него."** << **'\n'**;  
 FillDataFromFile(routesList, dataFile);  
 }  
 dataFile.close();  
  
 Menu(routesList);  
  
  
 **return** 0;  
}

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего

образования «Московский государственный технический университет имени

Н.Э. Баумана (национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

|  |  |
| --- | --- |
|  | УТВЕРЖДАЮ  \_\_\_\_\_\_\_\_\_\_\_\_ А.Д. Правдина  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета компьютеров организации»**

Программа и методика испытаний

7 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-13Б Поляков Д.Д.\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г. |

Москва, 2021

**СОДЕРЖАНИЕ**

1. ОБЪЕКТ ИСПЫТАНИЙ

2. ЦЕЛЬ ИСПЫТАНИЙ

3. СОСТАВ ПРЕДЪЯВЛЯЕМОЙ ДОКУМЕНТАЦИИ

4. ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ИСПЫТАНИЯМ И УСЛОВИЯ

ПРОВЕДЕНИЯ ИСПЫТАНИЯ

4.1. Требования к условиям проведения испытаний

4.2. Требования к техническим средствам

5. ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЯ

5.1. Состав и структура технических и программных средств

5.2. Последовательность испытаний программы

6. РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

# ОБЪЕКТ ИСПЫТАНИЙ

Объектом испытаний является программа для автоматизации учета трамвайных маршрутов служб общественного транспорта.

# ЦЕЛЬ ИСПЫТАНИЙ

Целью испытаний программного продукта состоит в проверке работоспособности программы и проверке соответствия основных функциональных возможностей программы требованиям, заложенным в техническом задании на программный продукт.

# СОСТАВ ПРЕДЪЯВЛЯЕМОЙ ДОКУМЕНТАЦИИ

При проведении испытаний предъявляются следующие документы:

* 1. Техническое задание (ТЗ)
  2. Руководство пользователя (РП)
  3. Исходный текст программы
  4. Программа и методика испытаний (ПМИ)

# ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ К ИСПЫТАНИЯМ И УСЛОВИЯ ПРОВЕДЕНИЯ ИСПЫТАНИЯ

## 4.1. Требования к условиям проведения испытаний

4.1.1. Программа должна работать в операционной среде, где не инсталлирована система программирования.

4.1.2. Программа должна работать в режиме командной строки, при этом вывод и ввод информации должен быть обеспечен на русском языке.

4.1.3. При необходимости и наличии системы программирования (JetBrains CLion), сборка проекта программы должна быть выполнена без ошибок и предупреждений.

## 4.2. Требования к техническим средствам

4.2.1. Компьютер с процессором Apple M1 Silicon и выше

4.2.2. 2 гигабайта оперативной памяти

4.2.3. Монитор с VGA выходом

4.2.4. Стандартная клавиатура

4.2.5. 229 килобайт свободного места на жестком диске для программы

4.2.6. 400 килобайт свободного места на жестком диске для базы данных.

# 5. ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЯ

## 5.1. Состав и структура технических и программных средств

5.1.1. Запуск программы выполняется в командной строке или из любого файл-менеджера.

5.1.2. Запуск режима командной строки производиться из главного меню стандартного приложения командной строки macOS.

5.1.3. Программа для испытаний запускается с флэш-носителя или другого сменного носителя информации. Для запуска программы из текущей директории вводится:

./TRCRUDqqq.

**5.1.1. Общие требования**

5.1.1.1. Интерфейс программы должен быть реализован в консоли.

5.1.1.2. Интерфейс программы должен быть полностью на русском языке, а также максимально интуитивным и понятным, каждое действие должно сопровождаться пояснением.

5.1.1.3. На протяжении всей работы программы должно отображаться меню, позволяющее пользователю выбрать необходимое ему действие.

5.1.1.4. Ниже по тексту, данные о одном трамвайном маршруте называются записью.

5.1.1.5. Интерфейс программы должен быть реализован в виде пронумерованного меню с выбором действий (меню пользователя). Меню пользователя должно содержать следующие пункты:

– Просмотр данных;

– Добавление данных;

– Изменение данных;

– Удаление данных;

– Поиск данных;

– Сохранение данных;

– Завершить работу.

5.1.1.6. Программа должна игнорировать некорректно введенные данные и сообщать об ошибке пользователю.

5.1.1.7. При запуске программы должно происходить чтение данных о трамвайных маршрутах транспортной службы в оперативную память из текстового файла, который должен находиться в текущем каталоге.

5.1.1.8. Если при запуске программы файл с данными в текущем каталоге отсутствует, то программа должна продолжать работу, и пользователь должен иметь возможность заново сформировать данные в программе.

**5.1.2. Основные функции.**

Программа должна обеспечивать выполнение следующих функций:

5.1.2.1. **Просмотр данных**.

Вывод данных в консоль должен быть отформатирован и отсортирован. При выполнении пункта меню «Просмотр данных» список должен быть отсортирован по возрастанию, по имени компьютера. Сортировка должна производиться посимвольно с учетом регистра.

5.1.2.2. **Добавление данных**.

#### При добавлении данных формат вводимых данных должен быть в следующем виде: «Номер Длина Время», где Номер – номер добавляемого маршрута, Длина – длина нового маршрута, Время – время в пути.

5.1.2.3. **Изменение данных**. При изменении данных о маршруте пользователя попросят ввести номер маршрута для редактирования, далее будет выведена информация об изменяемом пользователем маршруте и просьба ввести новые данные.

Окончание ввода записей должно осуществляться посредствам нажатия клавиши Enter на новой строке.

5.1.2.4. **Удаление данных**. При удалении данных о маршруте программа потребует от пользователя ввести номер удаляемого маршрута и сообщит пользователю о том, какая запись будет удалена. В случае, если маршрута с таким номером не будет найдено, программа уведомит об этом пользователя и запросит повторный ввод номера.

5.1.2.5. **Поиск данных**.

В программе должен быть реализован поиск записи по номеру маршрута и вывод найденной записи на экран.

5.1.2.6. **Сохранение данных**. В программе должно быть реализовано сохранение данных в текстовый файл.

## 

## 5.2. Последовательность испытаний программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **№ пункта Т.З.** | **Выполняемые действия** | **Ожидаемый результат** | **Примечание** |
|  | Запуск программы | В командной строке в текущем каталоге, где находится программа ввести: «./TRCRUDqqq» и нажать Enter | Выведется статус файла базы данных и главное меню программы (рис. 1) | Включает пункты 5.1.1.1, 5.1.1.2, 5.1.1.3 |
|  | 5.1.2.1  Вывод без базы данных | Удалить в директории программы файл с базой данных. Ввести 1, нажать Enter. | Файл будет снова создан и заполнен стандартными значениями, которые выведутся на экран (рис. 2) |  |
|  | 5.1.2.2  Неверно введённая длина маршрута | Ввести текст после просьбы ввода длины маршрута(“qqq”) нажать Enter | Программа сообщит о неправильном формате ввода (рис. 3) | Для продолжения ввести правильные данные и нажать Enter |
|  | 5.1.2.2  Неверно введённое время | Ввести текст после просьбы ввода времени в пути маршрута(“qqq”) нажать Enter | Программа сообщит о неправильном формате ввода (рис. 4) | Для продолжения ввести правильные данные и нажать Enter |
|  | 5.1.2.3  Неверный ввод даты и маршрута при редактировании | Ввести текст в поля длины и времени в пути после просьбы ввода новых данных | Аналогично пунктам 3 и 4 |  |
|  | 5.1.2.5  Попытка изменить несуществующий маршрут | Выбрав пункт 5 меню, ввести номер маршрута, отсутствующего в списке | Программа предупредит, что данного маршрута не найдено (рис. 5) | Для продолжения ввести правильные данные и нажать Enter |
|  | 5.1.2.4  Удаление маршрута по номеру, отсутствующему в списке | Выбрав пункт 4 меню, ввести номер маршрута, отсутствующего в списке | Программа предупредит, что данного маршрута не найдено (рис. 6) | Для продолжения ввести правильные данные и нажать Enter |

# 6. РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

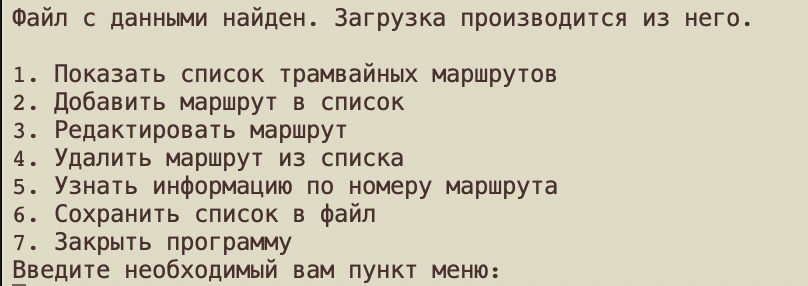
****

Рис. 1

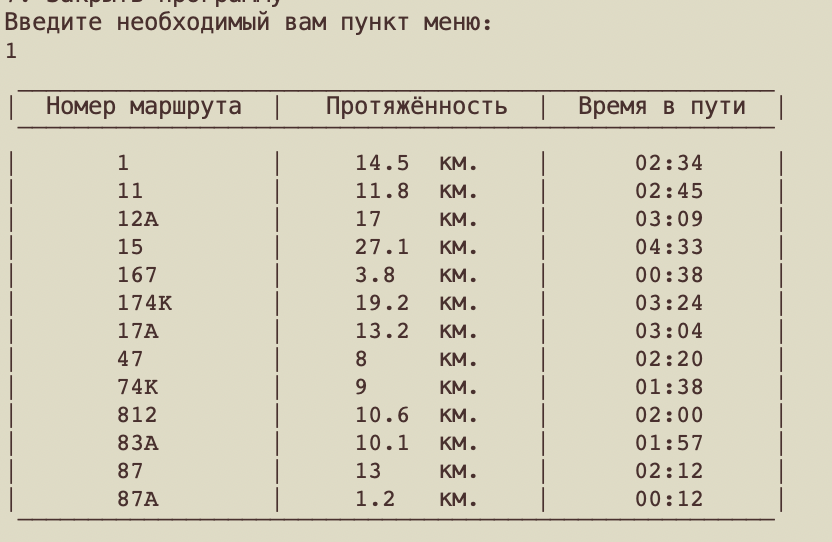


Рис. 2

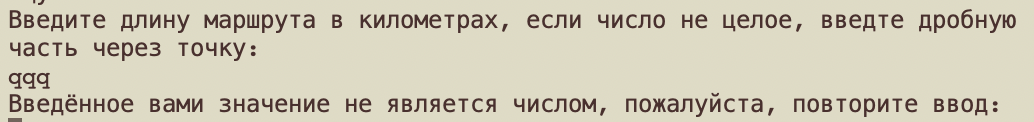
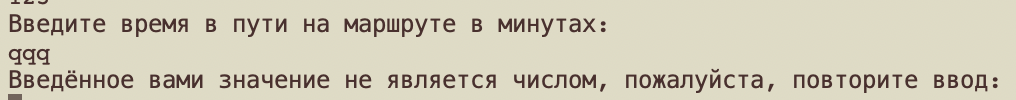


Рис. 3

 Рис. 4

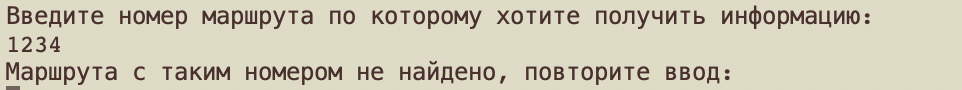


Рис. 5

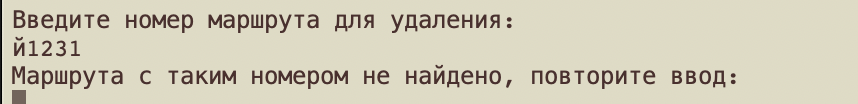


Рис. 6

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации (ЕСПД)
2. Эккель Б. Философия C++. Введение в стандартный C++. 2-е изд. — СПб.: Питер, 2004. — 572 с.: ил.Г. Шилдт “С++ Базовый курс”: Пер. с англ.- М., Издательский дом “Вильямс”, 2011 г. – 672с
3. Шилдт Г., “С++ Руководство для начинающих”: Пер. с англ. - М., Издательский дом “Вильямс”, 2005 г. – 672с
4. Шилдт Г., “Полный справочник по С++”: Пер. с англ.- М., Издательский дом “Вильямс”, 2006 г. – 800с
5. Страуструп Б., "Язык программирования С++"- М., Бином, 2010 г.
6. Керниган Б., Ритчи Д. К36 Язык программирования Си.\Пер. с англ., 3-е изд., испр. - СПб.: "Невский Диалект", 2001. - 352 с.: ил.
7. Страуструп Б. "Дизайн и эволюция С++. Классика CS" – СПб.: Питер, 2007. – 445с.