|  |  |
| --- | --- |
| 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: \_\_\_\_\_\_**Цыпышев Тимофей Александрович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

*фамилия, имя, отчество*

Группа: \_\_\_\_\_\_\_**ИУ5-11Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Тип практики: **Проектно-технологическая практика\_\_\_\_\_\_\_\_\_\_\_**

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

Студент: **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_**Цыпышев Т.А.

*подпись, дата фамилия, и.о.*

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

*подпись, дата фамилия, и.о.*

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_

*2022 г.*

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Написать программу учета поездов, принадлежащих транспортной компании.

Сведения о поезде содержат: инвентарный номер поезда, название модели поезда (из документации), название производителя поезда (из документации), дата выпуска поезда (из документации), дата введения поезда в эксплуатацию, состояние поезда (считается от 1% до 100%).

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

* добавление новых данных о поезде;
* удаление данных о поезде;
* изменение данных о поезде;
* вывод сведений о всех поездах;
* поиск по инвентарному номеру поезда и вывод в консоль;
* запись всех данных в текстовый файл.
* сортировка по инвентарному номеру поездов

ОГЛАВЛЕНИЕ

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

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  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета поездов, принадлежащих транспортной компании»**

Техническое задание

6 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-11Б Цыпышев Т.А. \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва, 2022

**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 поездов, принадлежащих транспортной компании. Шифр разработки программного продукта – IISAWACC.

# 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 поездов, принадлежащих транспортной компании. В частности, программа должна обеспечивать решение задач: добавление новых данных о поездах, удаление данных о поезде, изменение данных о конкретном поезде, показ всех данных пользователю, поиск данных о конкретном поезде, сохранение данных в долговременную память, сортировка данных по инвентарным номерам поездов. Программа должна обеспечивать удобную работу с данными, высокий уровень надежности и функциональных возможностей.

# Исполнитель

Студент группы ИУ5-11Б МГТУ им. Баумана Цыпышев Тимофей Александрович,

вариант № 22.

# 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 должен отображаться список данных о поездах компании, обновляемый в режиме реального времени.

5.1.1.4. Ниже по тексту, данные об одном поезде 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:

* добавление новых данных о поезде;
* удаление данных о поезде;
* изменение данных о поезде;
* вывод сведений о всех поездах;
* поиск по инвентарному номеру поезда и вывод в консоль;
* запись всех данных в текстовый файл;
* сортировка по инвентарному номеру поездов;
* просмотр всех команд;
* завершение программы.

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

5.1.1.7. При запуске программы должно происходить выведение меню в консоль и ожидание дальнейших команд от пользователя.

**5.1.2. Основные функции.**

Программа должна обеспечивать выполнение следующих функций:

5.1.2.1. **Добавление данных**.

При добавлении данных формат вводимых данных должен быть в следующем виде: «id name brand date\_release date\_registration wear», где id - инвентарный номер поезда, name - название модели поезда (из документации), brand - название производителя поезда (из документации), date\_release - дата выпуска поезда (из документации), date\_registration - дата введения поезда в эксплуатацию, wear - состояние поезда (считается от 1% до 100%).

При добавлении данных о поезде, инвентарный номер которого уже содержится в списке, программа сообщит, что данный поезд уже есть в списке и попросит уточнить данные.

При некорректном вводе date\_release, date\_registration, wear программа будет присваивать им значения по умолчанию (значение по умолчанию для: date\_release = 2022, date\_registration = 2022, wear = 100). Таким образом это сократит кол-во человеческих ошибок и добавит возможность быстрее вводить пользователю данные (ПРИМЕР: пользователь может ввести значение wear = 0 и программа присвоит ему 100, что будет считаться за новый поезд)

5.1.2.2. **Удаление данных**.

При удалении данных о поезде программа спрашивает у пользователя инвентарный номер поезда, который нужно удалить. При отсутствии данного инвентарного номера или некорректном его вводе, программа сообщит, что данные не соответствуют правилам ввода.

5.1.2.3. **Изменение данных**.

При изменении данных формат вводимых данных должен быть в следующем виде: «old\_id new\_id name brand date\_release date\_registration wear», где old\_id – старый инвентарный номер поезда, new\_id – новый инвентарный номер поезда, name - название модели поезда (из документации), brand - название производителя поезда (из документации), date\_release - дата выпуска поезда (из документации), date\_registration - дата введения поезда в эксплуатацию, wear - состояние поезда (считается от 1% до 100%).

При изменении данных о поезде программа спрашивает у пользователя инвентарный номер поезда, который нужно изменить. При отсутствии данного инвентарного номера или некорректном его вводе, программа сообщит, что данные не соответствуют правилам ввода.

Далее программа должна попросить у пользователя новые данные. При некорректном вводе нового инвентарного номера, программа сообщит, что данные не соответствуют правилам ввода и попросит ввести их заново. Это действие будет повторяться, пока пользователь не введёт корректные данные.

При некорректном вводе date\_release, date\_registration, wear программа будет присваивать им значения по умолчанию (значение по умолчанию для: date\_release = 2022, date\_registration = 2022, wear = 100).

* + - 1. **Просмотр данных.**

5.1.2.5.1 **Вывод всех данных в консоль.**

Вывод данных в консоль должен быть отформатирован и представлен в виде таблицы.

Каждый столбец должен быть подписан в соответствии с назначением («N ID NAME BRAND DATE1 DATE2 WEAR», где N – номер поезда в списке, ID - инвентарный номер поезда, NAME - название модели поезда, BRAND - название производителя поезда, DATE1 - дата выпуска поезда, DATE2 - дата введения поезда в эксплуатацию, WEAR - состояние поезда (считается от 1% до 100%).)

5.1.2.5.2 **Вывод данных о конкретном поезде в консоль.**

Программа должна спросить у пользователя инвентарный номер поезда, а потом вывести данные об этом поезде в консоль.

Вывод данных в консоль должен быть отформатирован и представлен в виде таблицы.

Каждый столбец должен быть подписан в соответствии с назначением («ID NAME BRAND DATE1 DATE2 WEAR», ID - инвентарный номер поезда, NAME - название модели поезда, BRAND - название производителя поезда, DATE1 - дата выпуска поезда, DATE2 - дата введения поезда в эксплуатацию, WEAR - состояние поезда (считается от 1% до 100%).)

* + - 1. **Сохранение данных**.

В программе должно быть реализовано сохранение данных в текстовый файл.

Файл с сохранёнными данными должен быть представлен в виде отформатированной данных.

* + - 1. **Сортировка данных.**

Список с данными должен быть отсортирован по инвентарному номеру поезда по возрастанию

* + - 1. **Просмотр всех команд.**

Должна быть реализована вспомогательная функция, которая будетвызываться при

запуске программы, при вызове её пользователем и при неправильной работе пользователя основным меню.

* + - 1. **Завершение программы.**

При вызове этой функции пользователем программа должна прекращать свою работу.

## 5.2. Требования к программному обеспечению

Данная программа предназначена для выполнения на компьютере под управлением системы Microsoft Windows версии XP и выше, linux Ubuntu 22.04.1 LTS. Для использования данной программы требуется наличие на компьютере компилятора С++ и среды программирования JetBrains Clion 2022.3.

## 5.3. Требования к условиям эксплуатации

5.3.1. Данная программа должна эксплуатироваться на компьютере под управлением системы Microsoft Windows версии XP и выше, linux Ubuntu 22.04.1 LTS установленная среда программирования.

5.3.2. Для сопровождения программы системный программист должен быть знаком с языком C++ в среде программирования JetBrains Clion 2022.3. Для работы с данной программой системный программист должен быть знаком с навыками объектно-ориентированного программирования.

## 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 .hpp.

5.4.4. Ввод и вывод в программе должен быть на русском языке.

5.4.5. Комментарии в листинге программы должны быть на русском языке.

5.4.6. 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 Intel на поколении Coffe Lake – S (8th gen) и выше;

5.6.2. Не менее 2гб свободной оперативной памяти;

5.6.3. Монитор с HDMI/ DISPLAY PORT выходом;

5.6.4. Раскладка клавиатуры – qwerty (с нанесёнными на неё русскими символами);

5.6.5. Свободное место на жестком диске не менее 1МБ.

## 5.7. Требования к программной совместимости

Для сопровождения программы должна использоваться программа JetBrains Clion 2022.3, компьютер должен быть оснащен русской таблицей символов и кодовой поддержкой.

## 5.8. Требования к маркировке и упаковке программы

Исходный код программы 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-технологической 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 исходный код (набор файлов с расширениями .cpp и .h) для сборки в JetBrains Clion 2022.3.

# 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  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета поездов, принадлежащих транспортной компании»**

Руководство пользователя

7 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-11Б Цыпышев Т.А. \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва, 2022

**СОДЕРЖАНИЕ**

1. НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

2. ПОДКЛЮЧЕНИЕ И ЗАПУСК ПРОГРАММЫ

3. РАБОТА С ПРОГРАММОЙ

3.1. Добавление новых данных о поезде

3.2. Удаление данных о поезде

3.3. Изменение данных о поезде

3.4. Вывод сведений о всех поездах

3.5. Поиск по инвентарному номеру поезда и вывод в консоль

3.6. Запись всех данных в текстовый файл

3.7. Сортировка по инвентарному номеру поездов

3.8. Просмотр всех команд

3.9. Завершение программы

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

4.1. Неправильный формат ввода

4.2. Данные введены не верно

4.2. Запись с идентичным инвентарным номером уже существует

4.3. Записи с данным инвентарным номером не существует

# НАЗНАЧЕНИЕ И УСЛОВИЯ ПРИМЕНЕНИЯ ПРОГРАММЫ

Целью работы является создание программы для автоматизации учета поездов, принадлежащих транспортной компании. В частности, программа должна обеспечивать решение задач: добавление новых данных о поездах, удаление данных о поезде, изменение данных о конкретном поезде, показ всех данных пользователю, поиск данных о конкретном поезде, сохранение данных в долговременную память, сортировка данных по инвентарным номерам поездов. Программа должна обеспечивать 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 1МБ свободного пространства.

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

2.3. Скопировать файл программы, в раздел, которым мы получили в пункте 2.1.

2.4. Запустить JetBrains Clion 2022.3, перейти в раздел создания новых проектов, ввести местоположение раздела, который мы получили в пункте 2.1 и создать проект.

2.5. Запустите проект.

# 3. РАБОТА c ПРОГРАММой

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

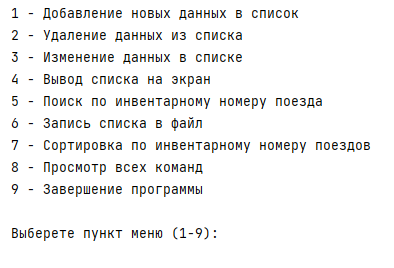


Рис.1.

## 3.1. ДОБАВЛЕНИЕ НОВЫХ ДАННЫХ В СПИСОК

После появления меню пользователя введите «1».

Далее программа попросит пользователя ввести новые данные.

На каждой новой строчке нужно ввести 6 характеристик поезда в следующем порядке: инвентарный номер поезда, название модели поезда (из документации), название производителя поезда (из документации), дата выпуска поезда (из документации), дата введения поезда в эксплуатацию, состояние поезда (считается от 1% до 100%) (рис. 2).

После программа уведомит пользователя, о том, что действие выполнено успешно и вернётся в главное меню.

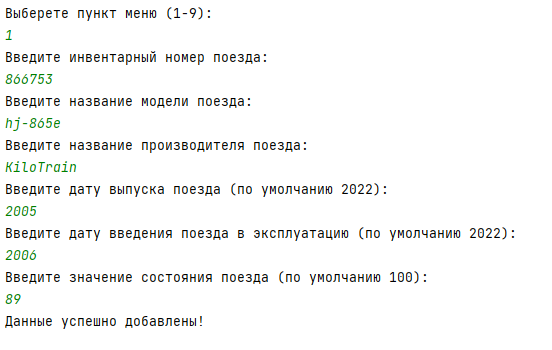


Рис. 2

## 3.2. Удаление данных о поезде из списка

После появления меню пользователя введите «2».

Далее программа попросит пользователя ввести инвентарный номер поезда.

Как только данные будут удалены – программа оповестит пользователя об этом (рис. 3).

После программа вернёт пользователя в главное меню.

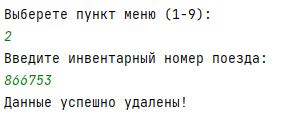


Рис. 3

**3.3. Изменение данных о поезде**

После появления меню пользователя введите «3».

Далее программа попросит пользователя ввести старый инвентарный номер поезда и новые данные о нём. (рис. 4).

После программа уведомит пользователя, о том, что действие выполнено успешно и вернётся в главное меню.

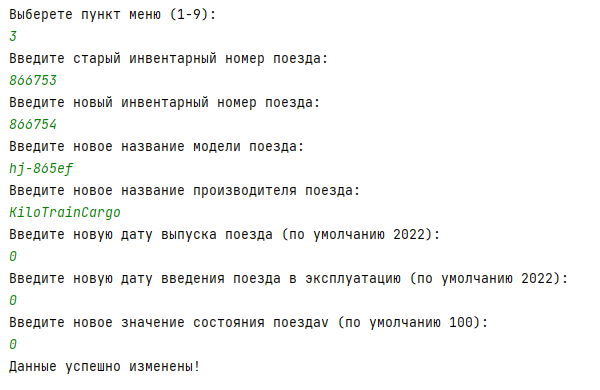


Рис. 4

**3.4.** **Вывод сведений о всех поездах из списка**

После появления меню пользователя введите «4».

Далее программа выведет таблицу со всеми поездами из списка. (рис. 5).

После программа вернёт пользователя в главное меню.

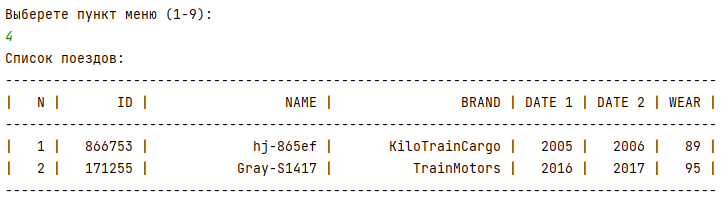


Рис. 5

**3.5.** **Вывод сведений о конкретном поезде**

После появления меню пользователя введите «5».

Далее программа попросит пользователя ввести инвентарный номер поезда и выведет таблицу с данными о поезде (рис. 6).

После программа вернёт пользователя в главное меню.

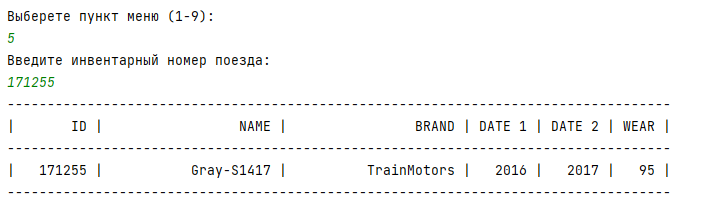


Рис. 6

**3.6.** **Запись всех данных в текстовый файл**

После появления меню пользователя введите «6».

Далее программа попросит пользователя ввести название файла (или его местоположение в заранее заготовленном месте на диске) (рис. 7).

Как только данные будут сохранены – программа оповестит пользователя об этом (рис. 7).

В файл сохраняться отформатированные данные (рис. 8).

После программа вернёт пользователя в главное меню.

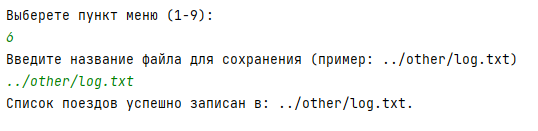


Рис. 7

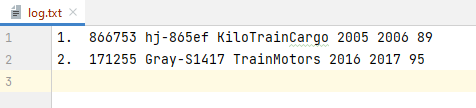


Рис. 8

**3.7.** **Запись всех данных в текстовый файл**

После появления меню пользователя введите «7».

Далее программа отсортирует список по инвентарному номеру поездов (по возрастанию) и оповестит об этом пользователя (рис. 9)

P. s. при необходимости пользователь может просмотреть список выполнив действия из пункта 3.4. (рис. 10).

После программа вернёт пользователя в главное меню.

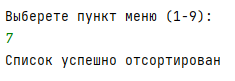


Рис. 9

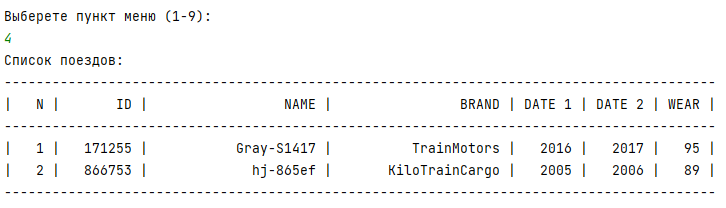


Рис. 10

**3.8.** **Просмотр всех команд**

После появления меню пользователя введите «8».

Далее программа выведет в консоль все возможные для пользователя действия (рис. 11).

После программа вернёт пользователя в главное меню.

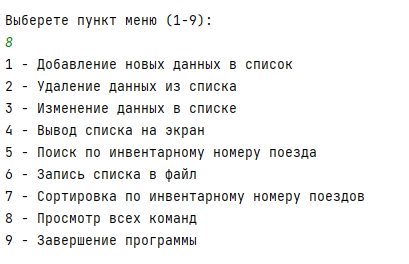


Рис. 11

**3.8.** **Просмотр всех команд**

После появления меню пользователя введите «9».

Далее программа завершит все процессы и выключиться (рис. 12).

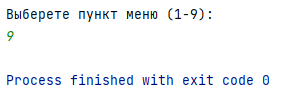


Рис. 12

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

## 4.1. Неправильный формат ввода

Если вы столкнулись с данной ошибкой – проверьте, удовлетворяют ли введенные данные допустимым типу значения (рис. 13).

**Инвентарный номер (id):** допускаются натуральные числа (без 0), состоящие из цифр от 0 до 9.

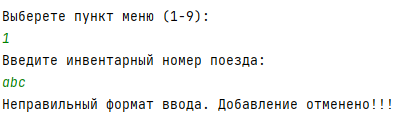


Рис. 13

**4.2. Данные введены не верно**

Если вы столкнулись с данной ошибкой – проверьте, удовлетворяют ли введенные данные допустимым значениям (рис. 14).

**Выбор пункта меню:** допускаются натуральные числа (от 1 – до 9)

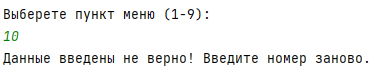


Рис. 14

## 4.3. Запись с идентичным инвентарным номером уже существует

Если вы встретили данное сообщение при работе с программой, то инвентарным номер поезда в только что введенной записи совпадает с инвентарным номером в уже имеющихся записях (рис. 15).

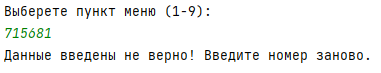


Рис. 15

## 4.4. Данный инвентарный номер не найден в списке

Если вы увидели данное сообщение – значит инвентарный номер поезда в только что введенной записи отсутствует в списке (рис. 16).

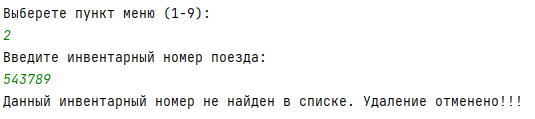


Рис.16

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  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета поездов, принадлежащих транспортной компании»**

Исходный текст программы

8 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-11Б Цыпышев Т.А. \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва, 2022

**Содержимое**

1. Файл Header.h (../include/Header.h)

2. Файл Main.cpp (../sources/Main.cpp)

3. Файл Functions.cpp (../sources/Functions.cpp)

Header.h

*/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>*#ifndef **PRACTICE\_01\_HEADER\_H**#define **PRACTICE\_01\_HEADER\_H***// main functions***void** AddItem();  
**void** RemoveItem();  
**void** EditItem();  
**void** ViewingDictionary();  
**void** FindItem();  
**void** LoggingDictionary();  
**void** SortDictionary();  
**void** Help();  
  
#endif *//PRACTICE\_01\_HEADER\_H*

Main.cpp

*/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>*#include **<iostream>**#include **"../include/Header.h"  
  
using namespace** std;  
  
  
**int** main() {  
 *// сначала вводим в виде строки, потом переводим в int  
 // необходимо, что бы избавиться от ошибки закцикливания при вводе некоректного значения пользователем* **int** item = 8;  
 **char** str\_item[1];  
 **while** (**true**) {  
 *// вызываем конкретные ф-ии в зависимости от выбранного пункта меню* **if** (item == 1) AddItem();  
 **else if** (item == 2) RemoveItem();  
 **else if** (item == 3) EditItem();  
 **else if** (item == 4) ViewingDictionary();  
 **else if** (item == 5) FindItem();  
 **else if** (item == 6) LoggingDictionary();  
 **else if** (item == 7) SortDictionary();  
 **else if** (item == 8) Help();  
 **else if** (item == 9) **return** 0;  
 **else** { cout << **"Данные введены не верно! Введите номер заново.\n"**; }  
 cout << **"\nВыберете пункт меню (1-9):\n"**;  
 cin >> str\_item;  
 item = atoi(str\_item);  
 }  
}

Functions.cpp

*/// Copyright 2022 ttsypyshev <ttsypyshev01@gmail.com>*#include **<iostream>**#include **<iomanip>**#include **<fstream>**#include **"../include/Header.h"  
  
using namespace** std;  
  
#define **size\_dictionary** 100  
#define **size\_name** 20  
#define **size\_brand** 20  
  
  
**struct** train {  
 **int** id; *// инвентарный номер поезда* **char** name[**size\_name**]; *// название модели поезда (из документации)* **char** brand[**size\_brand**]; *// название производителя поезда (из документации)* **int** date\_release = 2022; *// дата выпуска поезда (из документации) (по умолчанию 2022)* **int** date\_registration = 2022; *// дата введения поезда в эксплуатацию (по умолчанию 2022)* **int** wear = 100; *// состояние поезда (считается от 1% до 100%) (по умолчанию 100)*};  
**struct** train dictionary[**size\_dictionary**]; *// список поездов***int** last\_position = 0; *// номер первого пустого поля  
  
  
/// other functions***int** FindItem(**int** id) { *// поиск id в dictionary* **for** (**int** i = 0; i < last\_position; ++i) {  
 **if** (dictionary[i].id == id) **return** i;  
 }  
 **return** -1; *// если не нашли, то ф-ия возвращает -1*}  
  
*/// main functions***void** AddItem() { *// добавление новых train в dictionary  
 // сначала вводим в виде строки, потом переводим в int  
 // необходимо, что бы избавиться от ошибки закцикливания при вводе некоректного значения пользователем* **char** str\_id[8];  
 cout << **"Введите инвентарный номер поезда:\n"**;  
 cin >> str\_id;  
 **int** id = atoi(str\_id); *// при некоректных значения выдаёт 0* **if** (id != 0 **and** FindItem(id) == -1) { *// если значения коректны и элемент отсутствует в dictionary  
 // добавляем id (уже известно)* dictionary[last\_position].id = id;  
 **char** name[**size\_name**] = {0}, brand[**size\_brand**] = {0}, tmp[4] = {0};  
  
 *// спрашиваем и добавляем name* cout << **"Введите название модели поезда:\n"**;  
 cin >> name;  
 **for** (**int** i = 0; i < **size\_name**; ++i) dictionary[last\_position].name[i] = name[i];  
  
 *// спрашиваем и добавляем brand* cout << **"Введите название производителя поезда:\n"**;  
 cin >> brand;  
 **for** (**int** i = 0; i < **size\_brand**; ++i) dictionary[last\_position].brand[i] = brand[i];  
  
 *// спрашиваем и добавляем date\_release* cout << **"Введите дату выпуска поезда (по умолчанию 2022):\n"**;  
 cin >> tmp;  
 **int** date\_release = atoi(tmp);  
 **if** (date\_release > 1600) dictionary[last\_position].date\_release = date\_release;  
  
 *// спрашиваем и добавляем date\_registration* cout << **"Введите дату введения поезда в эксплуатацию (по умолчанию 2022):\n"**;  
 cin >> tmp;  
 **int** date\_registration = atoi(tmp);  
 **if** (date\_registration > 1600) dictionary[last\_position].date\_registration = date\_registration;  
  
 *// спрашиваем и добавляем wear* cout << **"Введите значение состояния поезда (по умолчанию 100):\n"**;  
 cin >> tmp;  
 **int** wear = atoi(tmp);  
 **if** (wear >= 1 **and** wear <= 100) dictionary[last\_position].wear = wear;  
 *//если 1<=wear<=100 (т.к. проценты, то значение <= 100)* cout << **"Данные успешно добавлены!\n"**;  
 last\_position++;  
 } **else if** (id != 0) {  
 cout << **"Данный инвентарный номер уже присутствует в списке. Добавление отменено!!!\n"**;  
 } **else** cout << **"Неправильный формат ввода. Добавление отменено!!!\n"**;  
}  
  
**void** RemoveItem() { *// удаление старых train из dictionary* **char** str\_id[8]; *// по анологии с void AddItem()* cout << **"Введите инвентарный номер поезда:\n"**;  
 cin >> str\_id;  
 **int** id = atoi(str\_id);  
 **int** index = FindItem(id);  
 **if** (id != 0 **and** index != -1) {  
 last\_position--;  
 *// мы заменяем удаленный элемент последним, чтобы не тратить ресурсы на сдвиг* dictionary[index].id = dictionary[last\_position].id;  
 **for** (**int** i = 0; i < **size\_name**; ++i) dictionary[index].name[i] = dictionary[last\_position].name[i];  
 **for** (**int** i = 0; i < **size\_brand**; ++i) dictionary[index].brand[i] = dictionary[last\_position].brand[i];  
 dictionary[index].date\_release = dictionary[last\_position].date\_release;  
 dictionary[index].date\_registration = dictionary[last\_position].date\_registration;  
 dictionary[index].wear = dictionary[last\_position].wear;  
 cout << **"Данные успешно удалены!\n"**;  
 } **else if** (id != 0) {  
 cout << **"Данный инвентарный номер не найден в списке. Удаление отменено!!!\n"**;  
 } **else** cout << **"Неправильный формат ввода. Удаление отменено!!!\n"**;  
}  
  
**void** EditItem() { *// изменение старых train в dictionary* **char** old\_str\_id[8]; *// по анологии с void AddItem()* cout << **"Введите старый инвентарный номер поезда:\n"**;  
 cin >> old\_str\_id;  
 **int** old\_id = atoi(old\_str\_id);  
 **int** index = FindItem(old\_id);  
 **if** (old\_id != 0 **and** index != -1) {  
 **char** new\_str\_id[8] = {0}, name[**size\_name**] = {0}, brand[**size\_brand**] = {0}, tmp[4] = {0};  
 **int** new\_id = 0;  
  
 *// нам необходим коректный id, т.к. на нём завязана функция поиска* **while** (new\_id == 0) {  
 cout << **"Введите новый инвентарный номер поезда:\n"**;  
 cin >> new\_str\_id;  
 new\_id = atoi(new\_str\_id);  
 }  
 dictionary[index].id = new\_id;  
  
 *// спрашиваем новые значения -> заменяем старые значения* cout << **"Введите новое название модели поезда:\n"**;  
 cin >> name;  
 **for** (**int** i = 0; i < **size\_name**; ++i) dictionary[index].name[i] = name[i];  
  
 cout << **"Введите новое название производителя поезда:\n"**;  
 cin >> brand;  
 **for** (**int** i = 0; i < **size\_brand**; ++i) dictionary[index].brand[i] = brand[i];  
  
 cout << **"Введите новую дату выпуска поезда (по умолчанию 2022):\n"**;  
 cin >> tmp;  
 **int** date\_release = atoi(tmp);  
 **if** (date\_release > 1600) dictionary[index].date\_release = date\_release;  
  
 cout << **"Введите новую дату введения поезда в эксплуатацию (по умолчанию 2022):\n"**;  
 cin >> tmp;  
 **int** date\_registration = atoi(tmp);  
 **if** (date\_registration > 1600) dictionary[index].date\_registration = date\_registration;  
  
 cout << **"Введите новое значение состояния поездаv (по умолчанию 100):\n"**;  
 cin >> tmp;  
 **int** wear = atoi(tmp);  
 **if** (wear >= 1 **and** wear <= 100) dictionary[index].wear = wear;  
  
 cout << **"Данные успешно изменены!\n"**;  
 } **else if** (old\_id != 0) {  
 cout << **"Данный инвентарный номер не найден в списке. Изменение отменено!!!\n"**;  
 } **else** cout << **"Неправильный формат ввода. Изменение отменено!!!\n"**;  
}  
  
**void** ViewingDictionary() { *// вывод всех train из dictionary в консоль* cout << **"Список поездов: \n"**;  
 cout << setw(50 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
 cout << **"| "** << setw(3) << **"N"** << **" | "** << setw(8) << **"ID"** << **" | "** << setw(**size\_name**) << **"NAME"** << **" | "** << setw(**size\_brand**) << **"BRAND"** << **" | "** << setw(6) << **"DATE 1"** << **" | "** << setw(6) << **"DATE 2"** << **" | "** << setw(4) << **"WEAR"** << **" |\n"**;  
 cout << setw(50 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
  
 **for** (**int** i = 0; i < last\_position; ++i) {  
 cout << **"| "** << setw(3) << i + 1 << **" | "** << setw(8) << dictionary[i].id << **" | "** << setw(**size\_name**) << dictionary[i].name << **" | "** << setw(**size\_brand**) << dictionary[i].brand << **" | "** << setw(6) << dictionary[i].date\_release << **" | "** << setw(6) << dictionary[i].date\_registration << **" | "** << setw(4) << dictionary[i].wear << **" |\n"**;  
 }  
  
 cout << setw(50 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
}  
  
**void** FindItem() { *// вывод конкретного train из dictionary в консоль* **char** str\_id[8]; *// по анологии с void AddItem()* cout << **"Введите инвентарный номер поезда:\n"**;  
 cin >> str\_id;  
 **int** id = atoi(str\_id);  
 **int** index = FindItem(id);  
  
 **if** (id != 0 **and** index != -1) { *// проходимся по dictionary* cout << setw(44 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
 cout << **"| "** << setw(8) << **"ID"** << **" | "** << setw(**size\_name**) << **"NAME"** << **" | "** << setw(**size\_brand**) << **"BRAND"** << **" | "** << setw(6) << **"DATE 1"** << **" | "** << setw(6) << **"DATE 2"** << **" | "** << setw(4) << **"WEAR"** << **" |\n"**;  
 cout << setw(44 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
  
 cout << **"| "** << setw(8) << dictionary[index].id << **" | "** << setw(**size\_name**) << dictionary[index].name << **" | "** << setw(**size\_brand**) << dictionary[index].brand << **" | "** << setw(6) << dictionary[index].date\_release << **" | "** << setw(6) << dictionary[index].date\_registration << **" | "** << setw(4) << dictionary[index].wear << **" |\n"**;  
  
 cout << setw(44 + **size\_name** + **size\_brand**) << setfill(**'-'**) << **"\n"** << setfill(**' '**);  
 } **else if** (id != 0) {  
 cout << **"Данный инвентарный номер не найден в списке. Поиск отменён!!!\n"**;  
 } **else** cout << **"Неправильный формат ввода. Поиск отменён!!!\n"**;  
}  
  
**void** LoggingDictionary() { *// запись всех train из dictionary в файл* **char** filename[100] = {0}; *// узнаём название файла* cout << **"Введите название файла для сохранения (пример: ../other/log.txt)\n"**;  
 cin >> filename;  
*// char filename[] = "../other/log.txt";  
  
 // открываем или создаём файл* ofstream log\_file(filename, ios\_base::trunc | ios\_base::out);  
 **if** (!(log\_file)) { log\_file.open(filename); }  
  
 **for** (**int** i = 0; i < last\_position; ++i)  
 log\_file << i + 1 << **".\t"** << dictionary[i].id << **" "** << dictionary[i].name << **" "** << dictionary[i].brand << **" "** << dictionary[i].date\_release << **" "** << dictionary[i].date\_registration << **" "** << dictionary[i].wear << **"\n"**;  
 cout << **"Список поездов успешно записан в: "** << filename << **".\n"**;  
}  
  
**void** SortDictionary() { *// сортируем dictionary по id из train  
 // метод пузырьковой сортировки* **for** (**int** i = 0; i < last\_position; i++)  
 **for** (**int** j = 0; j < last\_position - 1; j++)  
 **if** (dictionary[j].id > dictionary[j + 1].id)  
 swap(dictionary[j], dictionary[j + 1]);  
 cout << **"Список успешно отсортирован\n"**;  
}  
  
**void** Help() { *// вывод в консоль всех доступных действий* cout << **"1 - Добавление новых данных в список \n"** << **"2 - Удаление данных из списка \n"** << **"3 - Изменение данных в списке \n"** << **"4 - Вывод списка на экран \n"** << **"5 - Поиск по инвентарному номеру поезда \n"** << **"6 - Запись списка в файл \n"** << **"7 - Сортировка по инвентарному номеру поездов \n"** << **"8 - Просмотр всех команд \n"** << **"9 - Завершение программы \n"**;  
}

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  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

**ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ПРАКТИКА**

**«Программа автоматизации учета поездов, принадлежащих транспортной компании»**

Программа и методика испытаний

10 листов

|  |
| --- |
| ИСПОЛНИТЕЛЬ:  Студент группы ИУ5-11Б Цыпышев Т.А. \_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |

Москва, 2022

**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 поездов, принадлежащих транспортной компании.

# 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, установлена среда разработки JetBrains Clion 2022.3.

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. Сборка проекта программы должна быть выполнена без ошибок и предупреждений.

## 4.2. Требования к техническим средствам

4.2.1 Компьютер с процессором Intel на поколении Coffe Lake - S (8th gen) и выше;

4.2.2. Не менее 2гб свободной оперативной памяти;

4.2.3. Монитор с HDMI/ DISPLAY PORT выходом;

4.2.4. Раскладка клавиатуры – qwerty (с нанесёнными на неё русскими символами);

4.2.5. Свободное место на жестком диске не менее 1МБ.

# 5. ПОРЯДОК ПРОВЕДЕНИЯ ИСПЫТАНИЯ

## 5.1. Состав и структура технических и программных средств

5.1.1. Запуск программы выполняется в режиме командной строки в среде разработки JetBrains Clion 2022.3.

5.1.2. Переключить (reload) кодировку файла main.cpp с UTF-8 на IBM866

5.1.3. Запуск режима командной строки производиться из главного меню JetBrains Clion 2022.3: кнопка Run (или комбинацией клавиш Shift + F10) (рис. 1).





Рис. 1

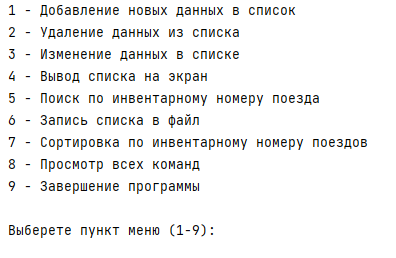
5.1.4. Программа для испытаний запускается с заранее подготовленного раздела на диске (в долговременной памяти).

## 5.2. Последовательность испытаний программы

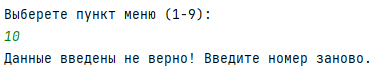
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№ п/п** | **№ пункта Т.З.** | **Выполняемые действия** | **Ожидаемый результат** | **Примечание** |
|  | Запуск программы | В главном меню нажать кнопка Run | Выведется главное меню программы | После программа попросит выбрать пункт меню |
|  | Проверка на корректность вводимых данных | Ввести «10», нажать Enter | Выведется предупреждение «Данные введены не верно» |  |
|  | Проверка на формат вводимых данных | Ввести «abc», нажать Enter | Выведется предупреждение «Данные введены не верно» |  |
|  | Проверка на добавление данных | Ввести «1», нажать Enter,  ввести «816798», нажать Enter,  ввести «HGe-755l», нажать Enter,  ввести «TrainMotors», нажать Enter,  ввести «2005», нажать Enter,  ввести «2006», нажать Enter,  ввести «67», нажать Enter | Программа попросит ввести все данные (по типам), сохранит их, оповестит пользователя об успехе | Необходимо проверить правильно программа сохранила данные (ввести «4», нажать Enter, сравнить) |
|  | Проверка на корректность вводимых данных при добавлении | Выполнить пункт 4,  вести «1», нажать Enter,  ввести «816798», нажать Enter | Выведется предупреждение «Данный инвентарный номер уже присутствует в списке» |  |
|  | Проверка на формат вводимых данных при добавлении | Ввести «abc», нажать Enter | Выведется предупреждение «Неправильный формат ввода» |  |
|  | Проверка на значение по умолчанию | Ввести «1», нажать Enter,  ввести «213987», нажать Enter,  ввести «fGe-755l», нажать Enter,  ввести «TrainMotors», нажать Enter,  ввести «0», нажать Enter,  ввести «0», нажать Enter,  ввести «0», нажать Enter | Программа попросит ввести все данные (по типам), сохранит их, и оповестит пользователя об успехе | Необходимо проверить правильно программа сохранила данные (ввести «4», нажать Enter, сравнить) |
|  | Проверка на удаление данных | Ввести «2», нажать Enter,  ввести «816798», нажать Enter | Программа попросит инвентарный номер, удалит данные, оповестит пользователя об успехе |  |
|  | Проверка на корректность вводимых данных при добавлении | Ввести «2», нажать Enter,  ввести «123», нажать Enter | Выведется предупреждение «Данный инвентарный номер не найден в списке» |  |
|  | Проверка на изменение данных | Ввести «3», нажать Enter,  ввести «213987», нажать Enter,  ввести «213414», нажать Enter,  ввести «fGe-757l», нажать Enter,  ввести «TrainMotors», нажать Enter,  ввести «2016», нажать Enter,  ввести «2016», нажать Enter,  ввести «88», нажать Enter | Программа попросит инвентарный номер,  попросит все данные (по типам), изменит данные, оповестит пользователя об успехе | Необходимо проверить правильно программа изменила  данные (ввести «4», нажать Enter, сравнить) |
|  | Проверка на вывод всех данных | Выполнить пункт 4 и 7,  ввести «4», нажать Enter | Программа выведет отформатированную таблицу со всеми данными |  |
|  | Проверить поиск по инвентарному номеру поезда и вывод в консоль | Выполнить пункт 10,  ввести «5», нажать Enter,  ввести «213414», нажать Enter | Программа выведет отформатированную таблицу с данными поезда с инвентарным номером «213414» |  |
|  | Проверить запись всех данных в текстовый файл | Ввести «6», нажать Enter,  ввести «../other/log.txt», нажать Enter | Программа попросит название файла, откроет (или создаст) файл с таким названием, сохранит данные, оповестит пользователя об успехе | Необходимо проверить правильно программа сохранила данные |
|  | Проверка сортировки списка | Ввести «7», нажать Enter | Программа отсортирует список, оповестит пользователя об успехе | Необходимо проверить правильно программа отсортировала список (ввести «4», нажать Enter, сравнить) |
|  | Проверка на просмотр всех команд | Ввести «8», нажать Enter | Программа выведет все команды |  |
|  | Проверка на завершение программы | Ввести «9», нажать Enter | Программа завершиться |  |

# 6. РЕЗУЛЬТАТЫ ИСПЫТАНИЙ

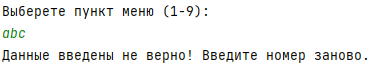
**Тест №1**



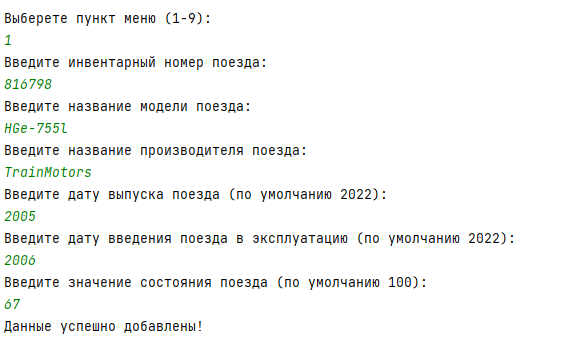
**Тест №2**



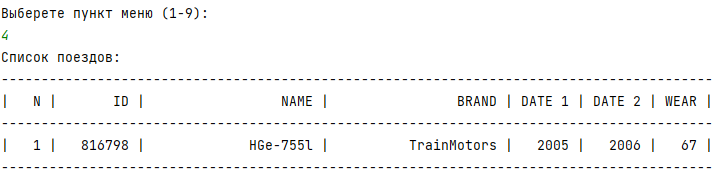
**Тест №3**



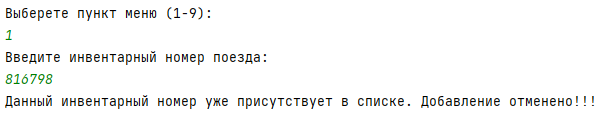
**Тест №4**



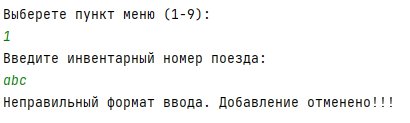
**Примечание к тесту №4**



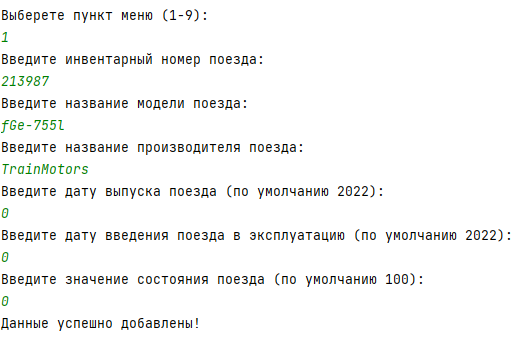
**Тест №5**

****

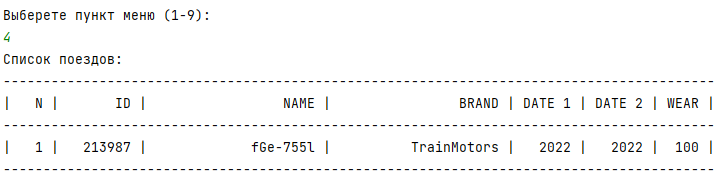
**Тест №6**

****

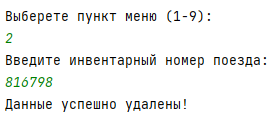
**Тест №7**

****

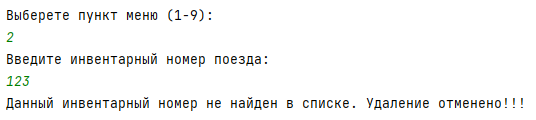
**Примечание к тесту №7**

****

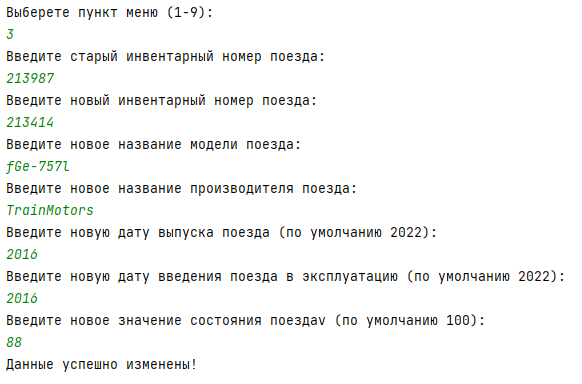
**Тест №8**

****

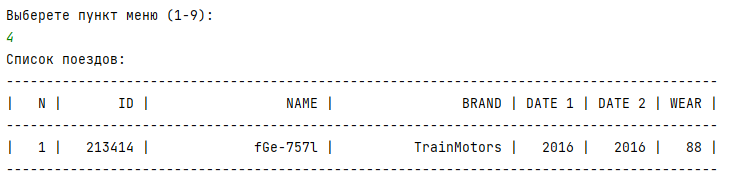
**Тест №9**

****

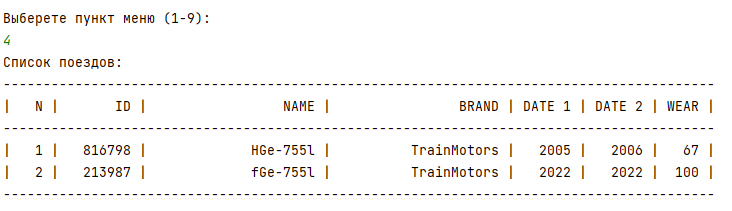
**Тест №10**

****

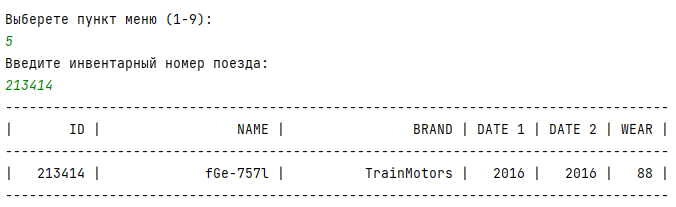
**Примечание к тесту №10**

****

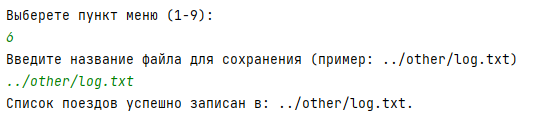
**Тест №11**

****

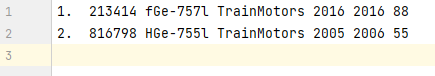
**Тест №12**

****

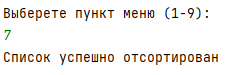
**Тест №13**

****

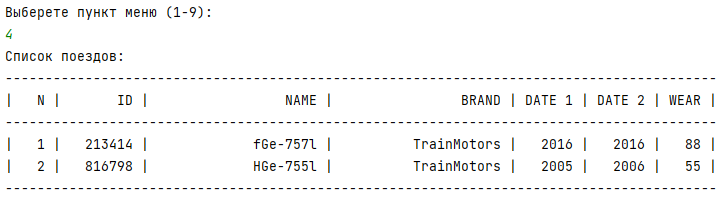
**Примечание к тесту №13**

****

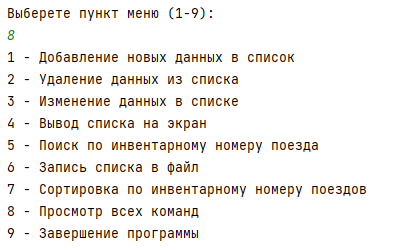
**Тест №14**



**Примечание к тесту №14**



**Тест №15**



**Тест №16**

