МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Казанский национальный исследовательский технический университет

им. А.Н. Туполева-КАИ»

Институт компьютерных технологий и защиты информации

Кафедра Прикладной математики и Информатики имени Ю. В. Кожевникова

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
К КУРСОВОЙ РАБОТЕ**

**по дисциплине** «Структуры и алгоритмы обработки данных»

**Тема работы**: Реализация комбинированной структуры данных

Вариант №5.15

Выполнил:   
 студент группы 4211  
Абдуллаев Л.Х.

Проверил:   
 Доцент каф. ПМИ к.т.н.

Сотников С. В.

Казань 2020

**Оглавление**

[**ВВЕДЕНИЕ** 4](#_Toc43287000)

[**1.** **ПОСТАНОВКА ЗАДАЧИ** 5](#_Toc43287001)

[**2.** **ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ СТАТИЧЕСКИЙ СТЕК СТАТИЧЕСКИХ УПОРЯДОЧЕННЫХ СПИСКОВ** 6](#_Toc43287002)

[**2.1.** **Структура данных типа «Статический стек»** 6](#_Toc43287003)

[**2.2.** **Структура данных типа «Статический упорядоченный список»** 8](#_Toc43287004)

[**2.3.** **Комбинированная структура данных «Статический стек упорядоченных статических списков»** 9](#_Toc43287005)

[**3.** **РУКОВОДСТВО ПРОГРАММИСТА** 11](#_Toc43287006)

[**3.1.** **Описание структуры программы** 11](#_Toc43287007)

[**3.2.** **Описание структур** 11](#_Toc43287008)

[**3.1.** **Описание всех разработанных подпрограмм для работы со структурой** 12](#_Toc43287009)

[**3.3.1.** **Класс coursework.cpp** 12](#_Toc43287010)

[**3.3.2.** **Класс Railway.cpp** 13](#_Toc43287011)

[**3.3.3.** **Класс Depot.cpp** 14](#_Toc43287012)

[**3.3.4.** **Класс Train.cpp** 14](#_Toc43287013)

[**4.** **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ** 16](#_Toc43287014)

[**4.1.** **Функции основного меню** 16](#_Toc43287015)

[**4.4.1.** **Показать структуру** 16](#_Toc43287016)

[**4.4.2.** **Поиск электровоза** 17](#_Toc43287017)

[**4.4.3.** **Добавить депо** 17](#_Toc43287018)

[**4.4.4.** **Удалить депо** 18](#_Toc43287019)

[**4.4.5.** **Добавить электровоз** 18](#_Toc43287020)

[**4.4.6.** **Удалить электровоз** 18](#_Toc43287021)

[**4.4.7.** **Записать структуру в файл** 19](#_Toc43287022)

[**4.4.8.** **Новая структура** 19](#_Toc43287023)

[**4.4.9.** **Завершение работы** 19](#_Toc43287024)

[**4.2.** **Загрузка данных из файла** 20](#_Toc43287025)

[**4.3.** **Сохранение структуры во внешнем файле** 21](#_Toc43287026)

[**5.** **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО КОМПЛЕКСА** 22](#_Toc43287027)

[**5.1.** **Организация тестирования ПО** 22](#_Toc43287028)

[**5.2.** **Входные данные** 22](#_Toc43287029)

[**ЗАКЛЮЧЕНИЕ** 25](#_Toc43287030)

[**ЛИТЕРАТУРА** 26](#_Toc43287031)

[**ПРИЛОЖЕНИЯ** 27](#_Toc43287032)

[**1.** **coursework.cpp** 27](#_Toc43287033)

[**2.** **Railway.cpp** 30](#_Toc43287034)

[**3.** **Depot.cpp** 33](#_Toc43287035)

[**4.** **Train.cpp** 35](#_Toc43287036)

# **ВВЕДЕНИЕ**

Типизация данных является одним из фундаментальных понятий современного программирования. Отнесение переменных к тому или иному типу позволяет установить внутренний формат хранения значений этой переменной и набор допустимых операций. Все распространенные языки программирования имеют набор базовых простейших типов данных (целочисленные, вещественные, символьные, логические) и возможность объединения их в составные наборы – массивы, записи, файлы. Понятие структуры данных определяется двумя моментами [1]:

* способом объединения отдельных компонент в единую структуру
* способами обработки как отдельных компонент структуры, так и всей структуры в целом.

Большинство дополнительных структур данных можно реализовать двумя способами:

* статически на основе массива
* динамически с помощью специальных переменных-указателей

Каждый из этих способов имеет свои преимущества и недостатки [1].

# **ПОСТАНОВКА ЗАДАЧИ**

Необходимо реализовать комбинированную структуру вида: статический стек упорядоченных статических списков, вариант (5.5), на основе объектного подхода.

Структура должна иметь следующее информационное наполнение: Железная дорога (название) – композиция депо (номера). Депо – композиция электровозов (марка, регистрационный номер)

Требования к программному комплексу:

1. Реализация всех необходимых операций (Добавление и удаление в основной и присоединенной структурах, поиск в списке).
2. Возможность сохранения всей структуры во внешнем txt файле с обратной загрузкой.
3. Реализация структуры для хранения и обработки данных конкретной информационной задачи.
4. Именования типов, структур и их полей, классов и их свойств, и методов в соответствии с конкретной информационной задачей.

**Ожидаемый результат**

Ожидаемым результатом выполнения курсовой работы является разработанная иерархия классов, которая включает в себя коллекции и классы, функции и процедуры для работы с этими структурами, реализующие весь необходимый функционал.

# **ОПИСАНИЕ СТРУКТУРЫ ДАННЫХ СТАТИЧЕСКИЙ СТЕК СТАТИЧЕСКИХ УПОРЯДОЧЕННЫХ СПИСКОВ**

## **Структура данных типа «Статический стек»**

Стек – это линейная структура данных, в которую элементы добавляются и удаляются только с одного конца, называемого вершиной стека. Стек работает по принципу “элемент, помещенный в стек последним, извлечен будет первым”. Иногда этот принцип обозначается сокращением LIFO (Last In – First Out, т.е. последним зашел – первым вышел) [1].

Пусть в стеке требуется сохранять целые числа, причем заранее известно их максимальное количество. Тогда для реализации стека надо объявить массив и одну переменную – **указатель вершины стека** (SP – Stack Pointer). Будем считать, что стек-массив заполняется (растет) от первых элементов массива к последним. Тогда указатель SP может определять либо последнюю занятую ячейку массива, либо первую свободную ячейку. Выберем второй способ. Тогда для пустого стека переменная SP = 1 (если индексация элементов массива начинается с 1), и при каждом добавлении нового элемента переменная SP увеличивается на 1, а при удалении – уменьшается на 1. Последовательность операций для массива из пяти элементов показана на следующей схеме

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

1. Пустой стек: sp=1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 |  |  |  |  |

2. Добавлено первое число 15, sp=2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | 33 |  |  |  |

3. Добавлено второе число 33, sp=3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | 33 | 07 |  |  |

4. Добавлено третье число 07, sp=4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 | 33 |  |  |  |

5. Удалено с вершины число 07, sp=3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 15 |  |  |  |  |

6. Удалено с вершины число 33, sp=2

Со стеком связываются две основные операции: добавление (вталкивание, PUSH) элемента в стек и удаление (выталкивание, POP) элемента из стека.

**Добавление включает в себя** [1]**:**

* проверку возможности добавления (стек-массив заполнен полностью?)
* размещение нового элемента в ячейке, определяемой значением переменной SP
* увеличение SP на 1

**Удаление включает в себя** [1]**:**

* проверку возможности удаления (стек пустой?)
* уменьшение SP на 1
* извлечение элемента из ячейки, определяемой значением переменной SP

## **Структура данных типа «Статический упорядоченный список»**

Статическая реализация списка на базе массива может быть выполнена двумя способами. Был выбран первый способ реализации статического упорядоченного списка.

Он основан на том, что логический номер элемента списка полностью соответствует номеру ячейки массива, где этот элемент хранится: первый элемент списка – в первой ячейке массива, второй – во второй, третий – в третьей и т.д.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Элемент 1 | Элемент 2 | Элемент 3 | . . . . . . . | Элемент N |  |

Проверка наличия в массиве хотя бы одного элемента и, хотя бы одной свободной ячейки выполняется элементарно с помощью счетчика числа элементов списка. Проход по списку – также элементарен. Поиск заданного элемента сводится к обычному поиску в массиве.

**Добавление** элемента в список включает [1]:

* Проверка возможности добавления;
* Определение индекса первого элемента, большего чем заданный (элемент i);
* Сдвига всех элементов от N до i на одну ячейку вправо;
* Добавление в ячейку i нового элемента.

**Удаление** элемента из списка [1]:

* Освобождается ячейка i;
* Все последующие элементы сдвигаются влево;

## **Комбинированная структура данных «Статический стек упорядоченных статических списков»**

Более сложным случаем является использование стека списков. Здесь каждый элемент основного стека хранит вспомогательный список. Логическая схема реализации комбинированной структуры данных представлена на рис.1.

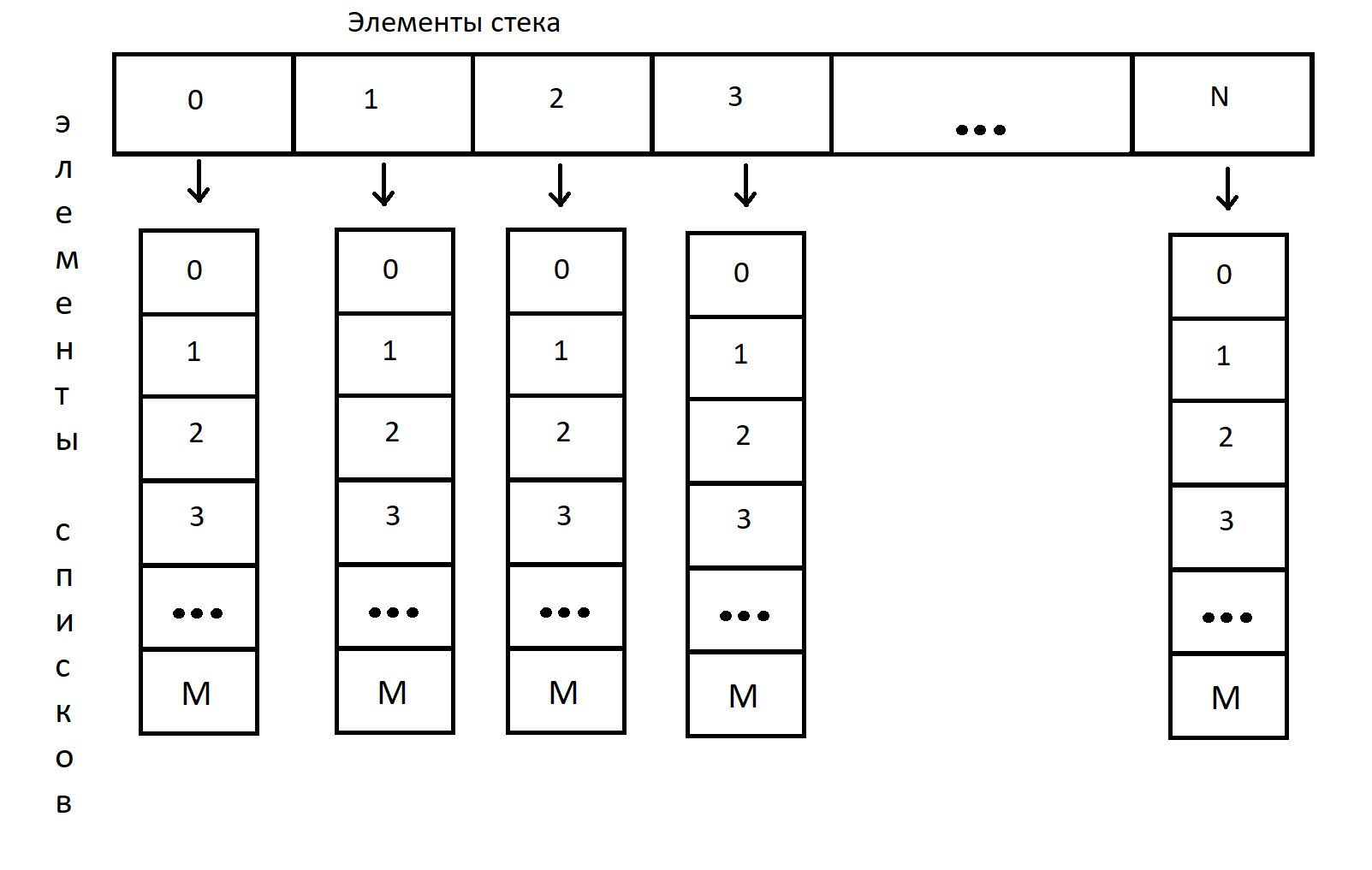


Рисунок 1 Схема комбинированной структуры данных – статического стека статических списков, 0…N –ячейки массива стека, 0…M – ячейки массивов списков

В первую очередь вводится тип данных, описывающий структуру элементов вспомогательного списка. После этого описывается структура элементов основного стека. Обработка таких структур включает больше операций, поскольку практически любая типовая операция (поиск, просмотр, добавление, удаление) может выполняться как с основным стеком, так и с любым вспомогательным списком. Например, полный **поиск** или **полный проход** реализуется двойным циклом: внешний цикл проходит по элементам основного стека, а внутренний обрабатывает отдельно каждый вспомогательный список.

**Добавление** и **удаление** элементов во вспомогательных списках выполняется обычным образом. Небольшие особенности имеет **удаление** элемента из основного стека, поскольку в этом случае как правило надо удалить и весь вспомогательный список. Поэтому прежде всего организуется проход по вспомогательному списку с удалением каждого элемента, а потом уже производится удаление элемента основного списка [1].

# **РУКОВОДСТВО ПРОГРАММИСТА**

## **Описание структуры программы**

Для разработки проекта была использована среда разработки программного обеспечения Visual Studio 2017.

Весь программный код, включая основную программу, хранится в файлах .cpp. Разработанные отношения агрегации классов, представленная на рисунке 3, включают в себя коллекции и классы, отвечающие за информационное заполнение, а также функции и процедуры, предназначенные для работы с созданными структурами данных.

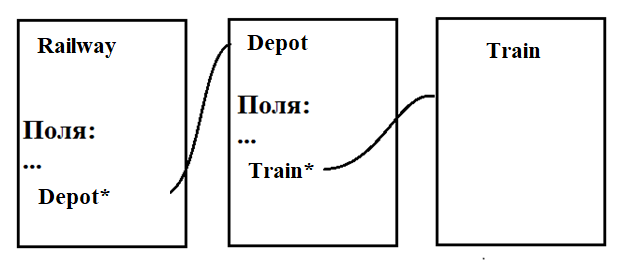


Рисунок 2 Отношения агрегации классов

## **Описание структур**

Контейнер представляет собой комбинированную структуру. Структура железная дорога – статический стек. Структура депо – упорядоченный статический список.

Разработаны две основные структуры, представленные в следующих контейнерах.

## **Описание всех разработанных подпрограмм для работы со структурой**

Контейнер представляет собой комбинированную структуру. Главная структура – статический стек. Дочерняя структура – упорядоченный статический список.

Разработаны две основные структуры, представленные в следующих контейнерах:

1. Класс Railway – представляет собой железную дорогу – сеть депо, реализованную через статический стек.

class Railway {

private:

Depot \*d\_arr; //массив депо

int count; //текущее количество депо

int len; //длина массива депо

string name;//название депо

}

1. Класс Depot – статический список электровозов.

class Depot {

private:

Train \*t\_arr; //массив электровозов

int len; //длина массива электровозов

int count; //текущее количество электровозов в депо

int d\_num; //номер депо

}

### **Класс coursework.cpp**

Описание класса, который реализует взаимодействие с пользователем через консоль.

Таблица 1 Описание методов класса coursework.cpp

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| int main() | Метод реализует взаимодействие с пользователем посредством вывода на экран меню и отлова нажатий. Затем вызывает соответствующую функцию. |
| bool Read(Railway\*& rway) | Метод, который производит чтение файла в корневой папке и сохраняет структуру в памяти в указателе, который принят в качестве аргумента. |

### **Класс Railway.cpp**

Таблица 2 Описание методов класса Railway.cpp

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| public string GetName() | Свойство доступа к переменной name. |
| public int GetCount() | Свойство доступа к переменной count. |
| public Depot\* GetDarr() | Свойство доступа к переменной d\_arr. |
| public void Depot\* GetDarr(int i) | Свойство доступа к i-той ячейке d\_arr. |
| public Railway() | Конструктор класса. |
| public Railway(string name) | Конструктор класса с заданным параметром – именем железной дороги. |
| public void push\_depot() | Метод добавляет новое депо в структуру, запрашивая у пользователя номер. |
| public void push\_depot(int num) | Метод добавляет новое депо в структуру, номер берет из аргумента. |
| public void pop\_depot() | Метод удаления депо из железной дороги. |
| public void push\_train() | Метод добавления электровоза в депо. Номер депо и данные о новом электровозе метод запрашивает у пользователя. |
| public void push\_train(int depot, int num, string mark) | Метод добавления электровоза в депо. Номер депо и данные о новом электровозе метод берет из аргументов. |
| public void pop\_train() | Метод удаляет электровоз. Номер депо и данные об удаляемом электровозе метод запрашивает у пользователя. |
| public void search\_train() | Метод ищет электровоз по всей структуре. Он запрашивает регистрационный номер и, если электровоз с таким номером найден, выводит информацию о нем, иначе выводит сообщение о неудачном поиске. |
| public void show() | Метод вывода всей структуры в консоль. |
| public void Write() | Метод выгрузки всей структуры в текстовый файл. |
| ~Railway() | Деструктор класса. | |

### **Класс Depot.cpp**

Таблица 3 Описание методов класса Depot.cpp

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| public int GetDnum() | Свойство доступа к переменной d\_num для чтения. |
| public void SetNum(int num) | Свойство доступа к переменной d\_num для записи. |
| public Train\* GetTarr() | Свойство доступа к переменной t\_arr. |
| public Train\* GetTarr(int i) | Свойство доступа к i-той ячейке массива t\_arr. |
| public int GetCount() | Свойство доступа к переменной count. |
| public Depot() | Конструктор класса. |
| public void push(int reg\_num, string mark) | Метод добавляет новый электровоз в список, если список не полон. Список сортируется по регистрационному номеру электровоза в возрастающем порядке. |
| public int search(int reg\_num) | Метод принимает в качестве аргумента регистрационный номер электровоза для поиска и возвращает индекс, если найден. Если не найден – возвращает -1. |
| public void pop() | Метод удаляет электровоз, запрашивая его регистрационный номер у пользователя. |
| public void show() | Метод поиска вывода депо в консоль. |
| ~Depot() | Деструктор класса. |

### **Класс Train.cpp**

Таблица 4 Описание методов класса Train.cpp

|  |  |
| --- | --- |
| **Название метода** | **Описание** |
| public int GetNum() | Метод доступа к переменной reg\_num для чтения. |
| public SetNum(int num) | Метод доступа к переменной reg\_num для записи. |
| public string GetMark() | Метод доступа к переменной mark для чтения. |
| public void SetMark(string m) | Метод доступа к переменной mark для записи. |
| public Train() | Конструктор класса. |
| ~Train() | Деструктор класса. |

# **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Взаимодействие с пользователем осуществляется через консольное меню. Запуск осуществляется посредством открытия файла coursework.exe. После запуска программа выведет основное меню:

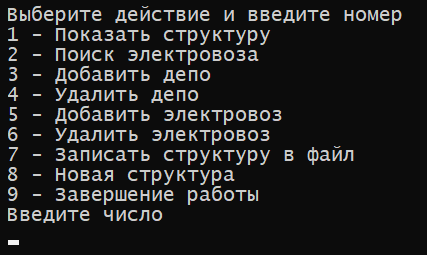


Рисунок 3 Основное меню

## **Функции основного меню**

### **Показать структуру**

Подпрограмма проверяет, создана ли структура. Если нет, то выводит об этом сообщение.

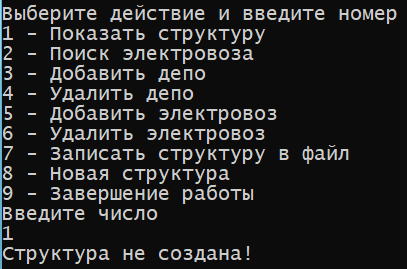


Рисунок 4 Вывод не созданной структуры

Иначе подпрограмма выводит структуру.

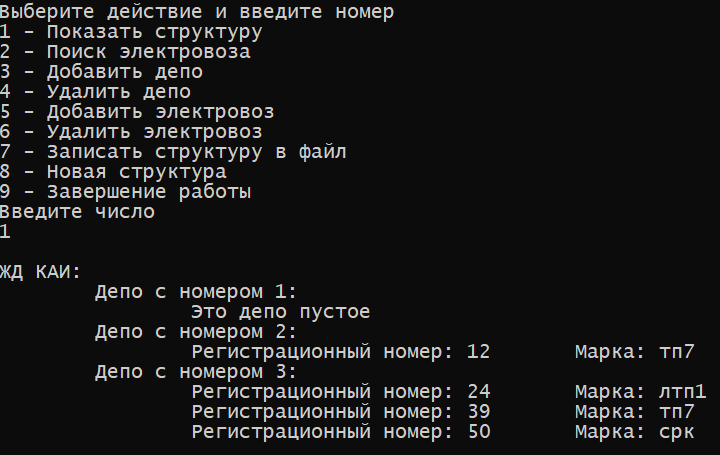
**

Рисунок 5 Вывод структуры в консоль

### **Поиск электровоза**

Если железная дорога создана, подпрограмма запросит ввод регистрационного номера электровоза, а после выведет информацию о нем, если он найден или сообщение о неудачном поиске.

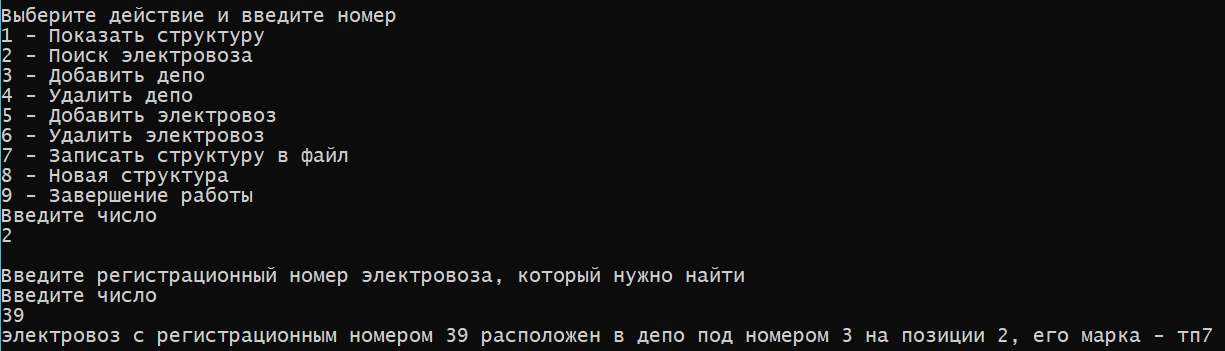


Рисунок 6 Поиск электровоза

### **Добавить депо**

Если железная дорога создана и есть место для добавления нового депо, подпрограмма запросит номер депо.



Рисунок 7 Добавление депо

### **Удалить депо**

Подпрограмма, если железная дорога создана и в ней есть хоть одно депо, удалит последнее добавленное депо.

### **Добавить электровоз**

Если железная дорога создана и список депо не пустой, подпрограмма запросит номер депо, в который нужно добавить электровоз. Если такое депо найдено, и оно имеет свободное место, то запрашиваются регистрационный номер и марка электровоза.

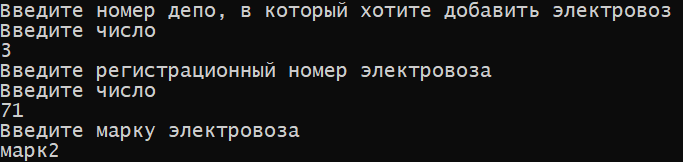


Рисунок 8 Ввод нового электровоза

### **Удалить электровоз**

Если железная дорога создана, и стек депо не пустой, подпрограмма запросит номер депо, в котором нужно произвести удаление и, если депо существует и не пустое, подпрограмма запросит регистрационный номер и марку электровоза для удаления.

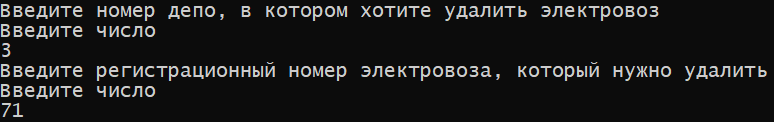
**

Рисунок 9 Удаление электровоза

### **Записать структуру в файл**

Если железная дорога создана, подпрограмма запишет структуру в файл с названием “file.txt”.

### **Новая структура**

Если структура создана, спрашивает подтверждение ее очищения. Затем выводит подменю – вопрос о том, следует ли создать структуру в консоли, или загрузить ее с файла.

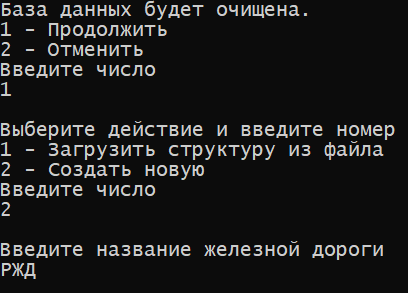


Рисунок 10 Новая структура

### **Завершение работы**

Подпрограмма очистит структуру и завершит работу программы.

## **Загрузка данных из файла**

В файле “file.txt” должна быть записана структура с соответствующими строками-тегами. Пример корректного заполнения файла показан на рис. 11.

-Открывается тег Железная дорога. В следующей строке пишется название железной дороги.

-Записывается нужное количество депо, но не более 10:  
Открываем тег депо строкой Депо

Записываем не более 10 строк для каждого электровоза – пишем через тире марку электровоза и регистрационный номер как показано на рис. 11.

Закрываем тег депо строкой /Депо

-Закрываем тег /Железная дорога (но можно и без него, если после последнего закрывающего тега /Депо строк нет)

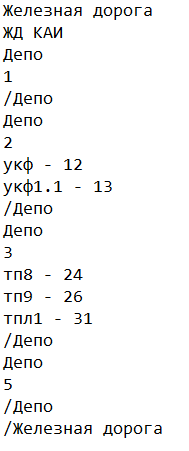


Рисунок 11 “file.txt”

## **Сохранение структуры во внешнем файле**

Подпрограмма выгрузки создаст файл “file.txt” и запишет в нее структуру. Пример выгрузки показан на рис. 12.

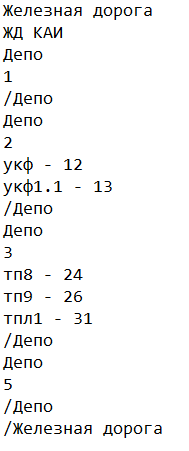


Рисунок 12 file.txt

# **РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ ПРОГРАММНОГО КОМПЛЕКСА**



## **Организация тестирования ПО**

Проводилось функциональное тестирование с целью установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям, то есть способности ПО в определённых условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

## **Входные данные**

Название железной дороги – любые символы.  
Номер депо – любые числа типа int ( макс. 2147483647);  
Марка электровоза – любые символы;  
Регистрационный номер электровоза – любые числа типа int ( макс. 2147483647);

При тестировании программы рассмотрены следующие ситуации (ошибки), возникающие во время использования программного комплекса, в других случаях программа выдает об успешности выполненной операции:

1. Попытка добавить депо, добавить электровоз, удалить депо, удалить электровоз, вывести структуру, поиск, когда структура не создана – вывод предупреждения, представленного на риc. 13.



Рисунок 13 Обработка исключения

1. Ввод символа вместо числа – повторный запрос на ввод числа, представленный на рис. 14.

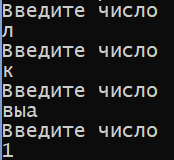


Рисунок 14 Обработка исключения

1. Ввод пустой строки – вывод предупреждения, представленного на рис. 15 и повторение запроса.



Рисунок 15 Обработка исключения

1. Если при добавлении электровоза стек пуст – вывод предупреждения, представленного на рис. 16.



Рисунок 16 Обработка исключения

1. Если при добавлении электровоза в стеке нет указанного депо – вывод предупреждения, представленного на рис. 17.

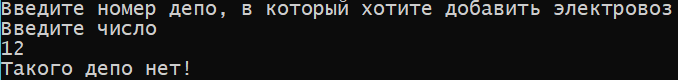


Рисунок 17 Обработка исключения

1. Поиск или удаление депо, электровоза при пустом стеке депо – вывод предупреждения, представленного на рис. 18.



Рисунок 18 Обработка исключения

1. Отсутствует файл “file.txt” – вывод предупреждения, представленного на рис. 19.

**

Рисунок 19 Обработка исключения

1. В файле для чтения неправильно заданы теги – вывод предупреждения, представленного на рис. 20.



Рисунок 20 Обработка исключения

1. В файле для чтения не задан атрибут – номер депо - вывод предупреждения, представленного на рис. 21.



Рисунок 21 Обработка исключения

1. В файле для чтения не задан атрибут – марка электровоза - вывод предупреждения, представленного на рис. 22.



Рисунок 22 Обработка исключения

1. В файле для чтения не задан атрибут – регистрационный номер электровоза - вывод предупреждения, представленного на рис. 23.

**

Рисунок 23 Обработка исключения

# **ЗАКЛЮЧЕНИЕ**

Полученная структура данных обладает всем требуемым функционалом, который был определен в постановке задачи. Консольное приложение позволяет наглядно продемонстрировать функциональность разработанных структур. Результаты работы предоставлены в виде исходного кода программного комплекса. Была написана пояснительная записка по курсовой работе. Сделав вывод, можно сказать, что программа соответствует всем требованиям.

# **ЛИТЕРАТУРА**

1. Козин А. Н. Учебно-методическое пособие «Структуры и алгоритмы обработки данных». – Казань.: КГТУ им. А.Н. Туполева, 2007.

# **ПРИЛОЖЕНИЯ**

Листинг программы:

## **coursework.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <string>

#include <iostream>

#include <fstream>

#include <Windows.h>

#include "Railway.cpp"

using namespace std;

bool Read(Railway\*& rway) {

string path = "file.txt";

ifstream fin;

try { fin.open(path, ios::in); }

catch (const ofstream::failure) {

cerr << "Не удалось открыть файл." << endl;

return false;

}

if (!fin) {

cout << "Файл не найден." << endl;

return false;

}

string InStr;

string SDNum;

string Name;

string Line;

string Mark;

int DNum;

int RegNum;

int d\_count = 0;

int t\_count = 0;

getline(fin, InStr);

while (fin && InStr != "/Железная дорога") {

if (InStr == "Железная дорога") {

getline(fin, Name);

if (!issp(Name[0])) {

rway = new Railway(Name);

}

else {

cout << "Ошибка при чтении файла. Неверно задано название железной дороги (первый символ пробельный).";

delete rway;

rway = nullptr;

return false;

}

}

else if (InStr == "Депо") {

if (rway == nullptr) {

cout << "Ошибка при чтении файла. Не задано название железной дороги" << endl;

return false;

}

getline(fin, SDNum);

if (SDNum == "") {

cout << "Ошибка при чтении файла. Не задан номер депо.";

delete rway;

rway = nullptr;

return false;

}

for (int i = 0; i < SDNum.length(); i++) {

if (!isdigit(SDNum[i])) {

cout << "Ошибка при чтении файла. Неверно задан номер депо.";

delete rway;

rway = nullptr;

return false;

}

}

DNum = stoi(SDNum);

d\_count++;

if (d\_count > railway\_len) {

cout << "Ошибка при чтении файла. Неверное количество депо." << endl;

delete rway;

rway = nullptr;

return false;

}

t\_count = 0;

rway->push\_depot(DNum);

getline(fin, Line);

while (Line == "") getline(fin, Line);

while (Line != "/Депо" && fin)

{

if (Line == "Депо") {

cout << "Ошибка при чтении файла. Неверно заданы теги." << endl;

delete rway;

rway = nullptr;

return false;

}

string buffer;

for (unsigned int i = 0; i < Line.length(); i++) {

if (Line[i] != '-') {

buffer += Line[i];

}

else {

Mark = buffer;

buffer = "";

for (unsigned int j = i + 1; j < Line.length(); j++) {

buffer += Line[j];

}

SDNum = buffer;

}

}

while (Mark[0] == ' ') {

Mark.erase(0, 1);

}

while (SDNum[0] == ' ')

{

SDNum.erase(0, 1);

}

if (Mark == "") {

cout << "Ошибка при чтении марки электровоза." << endl;

delete rway;

rway = nullptr;

return false;

}

if (SDNum == "") {

cout << "Ошибка при чтении регистрационного номера электровоза." << endl;

delete rway;

rway = nullptr;

return false;

}

try {

RegNum = stoi(SDNum);

}

catch (...) {

cout << "Ошибка при чтении файла. Неверно задан регистрационный номер электровоза." << endl;

delete rway;

rway = nullptr;

return false;

}

t\_count++;

if (t\_count > depot\_len) {

cout << "Ошибка при чтении файла. Неверное количество электровозов в депо с номером " << DNum << endl;

delete rway;

rway = nullptr;

return false;

}

rway->push\_train(DNum, RegNum, Mark);

getline(fin, Line);

}

}

else {

cout << "Ошибка при чтении файла. Неправильно объявлены теги." << endl;

if (rway != nullptr) {

delete rway;

rway = nullptr;

}

return false;

}

getline(fin, InStr);

}

fin.close();

cout << "Структура успешно загружена." << endl;

return true;

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "RUS");

Railway \*rway = nullptr;

bool flag = true;

int n;

while (true) {

flag = 1;

while (flag) {

cout << "\nВыберите действие и введите номер" << endl;

cout << "1 - Показать структуру" << endl;

cout << "2 - Поиск электровоза" << endl;

cout << "3 - Добавить депо" << endl;

cout << "4 - Удалить депо" << endl;

cout << "5 - Добавить электровоз" << endl;

cout << "6 - Удалить электровоз" << endl;

cout << "7 - Записать структуру в файл" << endl;

cout << "8 - Новая структура" << endl;

cout << "9 - Завершение работы" << endl;

int\_input(&n);

if (n >= 1 && n <= 9) {

flag = 0;

}

}

switch (n) {

case 1:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->show();

break;

case 2:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->search\_train();

break;

case 3:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->push\_depot();

break;

case 4:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->pop\_depot();

break;

case 5:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->push\_train();

break;

case 6:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->pop\_train();

break;

case 7:

if (rway == nullptr) {

cout << "Структура не создана!" << endl;

}

else rway->Write();

break;

case 8:

if (rway != nullptr) {

int action = 0;

while (action != 1 && action != 2) {

cout << "\nБаза данных будет очищена.\n1 - Продолжить\n2 - Отменить\n";

int\_input(&action);

}

if (action == 1) {

delete rway;

rway = nullptr;

}

else {

break;

}

}

while (1) { //просим выбрать метод ввода структуры, пока не введет 1 или 2

flag = true;

while (flag) {

cout << "\nВыберите действие и введите номер" << endl;

cout << "1 - Загрузить структуру из файла" << endl;

cout << "2 - Создать новую" << endl;

int\_input(&n); //описание этой функции есть в Railway.cpp

if (n >= 1 && n <= 2) {

flag = 0;

}

}

if (n == 1) {

if (Read(rway)) break;

else {

cout << endl;

break;

}

}

else if (n == 2) { //ввод из консоли

string name;

cout << "\nВведите название железной дороги\n";

//getline(cin, name);

string\_input(&name);

rway = new Railway(name);

break;

}

}

break;

case 9:

delete rway;

rway = nullptr;

exit(1);

break;

}

}

}

## **Railway.cpp**

#include <string>

#include <iostream>

#include "Depot.cpp"

#include <fstream>

using namespace std;

const int railway\_len = 10; //длина ЖД (массива депо)

inline bool issp(char sym) {

if (sym == ' ' || sym == '\t' || sym == '\n' || sym == '\v' || sym == '\f' || sym == '\r') {

return true;

}

else return false;

}

inline void string\_input(string \*in) {

getline(cin, \*in);

if (\*in == "") {

cout << "Ошибка! Введена пустая строка, попробуйте еще раз.\n";

string\_input(in);

}

else if (issp((\*in)[0])) {

cout << "Ошибка! Строка начинается с пробельного символа, попробуйте еще раз.";

cout << endl;

string\_input(in);

}

}

//статический стек - железная дорога

class Railway {

private:

Depot \*d\_arr; //массив депо

int count; //текущее количество депо

int len; //длина массива депо

string name;//название депо

public:

string GetName() {

return name;

}

int GetCount() {

return count;

}

Depot\* GetDarr() {

return d\_arr;

}

Depot\* GetDarr(int i) {

return &d\_arr[i];

}

Railway() {

len = railway\_len;

d\_arr = new Depot[railway\_len];

count = 0;

name = "";

}

Railway(string name) {

len = railway\_len;

d\_arr = new Depot[railway\_len];

count = 0;

this->name = name;

}

~Railway() {

delete[] d\_arr;

d\_arr = nullptr;

}

//добавление депо

void push\_depot() {

if (count == len) { //проверка заполненности массива

cout << "\nЖелезная дорога заполнена!\n";

return;

}

int d\_num;

cout << "\nВведите номер депо\n";

int\_input(&d\_num); //принимаем от пользователя номер нового депо

d\_arr[count].SetNum(d\_num); //вводим новое депо в первую свободную ячейку

++count;

}

//добавление депо с уже данным аргументом - номером депо, планировал использовать при вводе с файла

void push\_depot(int num) {

d\_arr[count].SetNum(num);

++count;

}

//удаление депо

void pop\_depot() {

if (count == 0) { //проверка на пустоту

cout << "\nЖелезная дорога пустая!\n";

return;

}

--count; //просто уменьшаем количество депо, так у нас не будет обрабатываться в других функциях последняя ячейка

}

//добавление электровоза

void push\_train() {

if (count == 0) { //проверка на пустоту (есть ли хоть одно депо, куда можно засунуть электровоз)

cout << "\nЖелезная дорога пустая!\n";

return;

}

int d\_num;

cout << "\nВведите номер депо, в который хотите добавить электровоз\n";

int\_input(&d\_num); //принимаем номер депо, в который нужно засунуть электровоз

for (int i = 0; i < count; i++) { //проходим по массиву депо

if (d\_arr[i].GetDnum() == d\_num) { //если нашли нужное депо по номеру

int reg\_num;

string mark;

cout << "Введите регистрационный номер электровоза\n";

int\_input(&reg\_num); //принимаем регистрационный номер электровоза

cout << "Введите марку электровоза\n";

//cin >> mark; //принимаем марку электровоза

//getline(cin, mark);

string\_input(&mark);

d\_arr[i].push(reg\_num, mark); //добавляем электровоз в нужную ячейку массива депо

return; //выходим из функции

}

}

cout << "Такого депо нет!\n"; //если дошли сюда, то запрошенное депо не было найдено, пишем об этом

}

//добавление электровоза с уже данными аргументами - номером депо, рег номером и маркой, планировал использовать при вводе с файла

void push\_train(int depot, int num, string mark) {

for (int i = 0; i < count; i++) {

if (d\_arr[i].GetDnum() == depot) {

d\_arr[i].push(num, mark);

return;

}

}

}

//удаление электровоза; проверка на пустоту и поиск депо такие же как в добавлении

void pop\_train() {

if (count == 0) {

cout << "\nЖелезная дорога пустая!\n";

return;

}

int d\_num;

cout << "\nВведите номер депо, в котором хотите удалить электровоз\n";

int\_input(&d\_num);

for (int i = 0; i < count; i++) {

if (d\_arr[i].GetDnum() == d\_num) {

/\*int reg\_num;\*/

d\_arr[i].pop();

return;

}

}

cout << "Такого депо нет!\n";

}

//поиск электровоза, идет по всей структуре, не спрашивая про депо

void search\_train() {

if (count == 0) { //проверка на пустоту

cout << "\nЖелезная дорога пустая!\n";

return;

}

int reg\_num;

cout << "\nВведите регистрационный номер электровоза, который нужно найти\n";

int\_input(&reg\_num); //принимаем рег номер электровоза, который нужно найти

int cur = 0;

for (int i = 0;i < count; i++) { //идем по всем депо

cur = d\_arr[i].search(reg\_num); //записываем в cur результат поиска (индекс в массиве электровозов, либо 0 если не найден)

if (cur != 0) { //если нашелся в текущем (i-ом) депо, пишем информацию

cout << "электровоз с регистрационным номером " << reg\_num;

cout << " расположен в депо под номером " << d\_arr[i].GetDnum();

cout << " на позиции " << cur;

cout << ", его марка - " << d\_arr[i].GetTarr()[cur - 1].GetMark() << endl;

return; //и выходим из функции

}

}

cout << "Такого электровоза нет!\n"; //если не нашелся ни в одном депо, пишем об этом

}

//вывод структуры в консоль

void show() {

cout << endl << name << ":\n"; //пишем название ЖД

if (count == 0) { //проверяем на пустоту

cout << "\tЖелезная дорога пустая!\n";

return;

}

for (int i = 0; i < count; i++) { //идем по всем депо

cout << "\tДепо с номером " << d\_arr[i].GetDnum() << ":\n"; //пишем номер текущего депо

d\_arr[i].show(); //выводим текущее депо

}

}

//вывод структуры в файл

void Write() {

ofstream fout; //обьявляется поток записи в файл

string path = "file.txt";

try { fout.open(path, ios::trunc); } //Открывается файл (или создается) по указанному пути

catch (const ofstream::failure) {

cerr << "Не удалось открыть файл" << endl;

}

fout << "Железная дорога" << endl << GetName() << endl;

if (GetCount() == 0) {

cout << "Стек пуст. Записано только название железной дороги." << endl;

return;

}

for (int i = 0; i < GetCount(); i++) {

fout << "Депо" << endl << GetDarr(i)->GetDnum() << endl;

for (int j = 0; j < GetDarr(i)->GetCount(); j++) {

fout << GetDarr(i)->GetTarr(j)->GetMark() << " - " << GetDarr(i)->GetTarr(j)->GetNum() << endl;

}

fout << "/Депо" << endl;

}

fout << "/Железная дорога";

fout.close();

}

};

## **Depot.cpp**

#include <string>

#include <iostream>

#include "Train.cpp"

using namespace std;

inline void int\_input(int \*in) { //функция защищенного ввода int значений, обновляет поток cin и просит ввести число, пока не примет такое, которое можно записать в int

int flag = 1; //принимает указатель на переменную, и записывает туда введенное значение

while (flag) {

cout << "Введите число\n";

if (cin >> \*in)

flag = 0;

cin.clear();

cin.ignore(32767, '\n');

}

}

const int depot\_len = 10; //длина депо (массива электровозов)

//статический упорядоченный список - депо

class Depot {

private:

Train \*t\_arr; //массив электровозов

int len; //длина массива электровозов

int count; //текущее количество электровозов в депо

int d\_num; //номер депо

public:

int GetDnum() {

return d\_num;

}

void SetNum(int num) {

this->d\_num = num;

}

Train\* GetTarr() {

return t\_arr;

}

//Ф-я возвращает i-й элемент массива электровозов

Train\* GetTarr(int i) {

return &t\_arr[i];

}

int GetCount() {

return count;

}

Depot() {

len = depot\_len;

t\_arr = new Train[depot\_len];

count = 0;

d\_num = -1;

}

//Depot(int d\_num) {

// len = depot\_len;

// t\_arr = new Train[depot\_len];

// count = 0;

// this->d\_num = d\_num;

//}

~Depot() {

delete[] t\_arr;

t\_arr = nullptr;

}

//добавление нового электровоза в депо

void push(int reg\_num, string mark) { //принимает регистрационный номер электровоза и марку электровоза

if (count == len) { //проверка заполненности депо

cout << "Не добавлено, т.к. это депо заполнено.\n";

return;

}

if (count == 0) { //случай, когда электровоз добавляется в пустое депо

t\_arr[0].SetNum(reg\_num);

t\_arr[0].SetMark(mark);

++count;

return;

}

//далее общий случай добавления электровоза в непустое депо

int cur = 0; //сюда запишем индекс подходящей ячейки для нового электровоза

while (t\_arr[cur].GetNum() <= reg\_num && cur < count) { //находим индекс подходящей ячейки для нового электровоза, сравнивая рег номера

++cur;

}

for (int i = count; i > cur; i--) { //сдвигаем все ячейки, после нужной нам для ввода нового электровоза, вправо

t\_arr[i].SetNum(t\_arr[i - 1].GetNum());

t\_arr[i].SetMark(t\_arr[i - 1].GetMark());

}

t\_arr[cur].SetNum(reg\_num); //добавляем новый электровоз

t\_arr[cur].SetMark(mark);

++count;

}

//поиск электровоза в депо

int search(int reg\_num) { //принимает регистрационный номер

for (int i = 0; i < count; i++) {

if (t\_arr[i].GetNum() == reg\_num) { //проходит по депо и сравнивает рег номер

return i + 1; //если находит, то возвращает индекс в массиве + 1

}

}

return 0; //если прошел по депо и не нашел электровоз с заданным рег номером, возвращает 0

}

//удаление электровоза

void pop() { //принимает рег номер электровоза, который нужно удалить

if (count == 0) { //проверка на пустоту

cout << "\nЭто депо пустое!\n";

return;

}

cout << "Введите регистрационный номер электровоза, который нужно удалить\n";

int reg\_num;

int\_input(&reg\_num); //приняли номер электровоза который нужно удалить

int ind = search(reg\_num) - 1; //ищем электровоз по рег номеру и передаем в ind индекс (-1 здесь т.к. search возвращает номер элемента считая с 1 а не с 0)

if (ind < 0) {

cout << "В этом депо нет такого электровоза\n"; //т.к в ind передается результат поиска -1, то если электровоз не был найден в депо, передается 0-1, т.е. ind = -1 < 0

return;

}

for (int i = ind;i < count - 1; i++) { //сдвигаем влево все ячейки с ind+1 до конца

t\_arr[i].SetNum(t\_arr[i + 1].GetNum());

t\_arr[i].SetMark(t\_arr[i + 1].GetMark());

}

--count; //уменьшаем количество электровозов на 1

}

//вывод электровозов этого депо в консоль

void show() {

if (count == 0) { //проверка на пустоту

cout << "\t\tЭто депо пустое\n";

return;

}

for (int i = 0; i < count; i++) { //идем по всем электровозам и выводим информацию

cout << "\t\tРегистрационный номер: " << t\_arr[i].GetNum();

cout << "\tМарка: " << t\_arr[i].GetMark() << endl;

}

}

};

## **Train.cpp**

#include <string>

using namespace std;

class Train {

private:

int \*reg\_num; //регистрационный номер электровоза

string \*mark; //марка электровоза

public:

int GetNum() {

return \*reg\_num;

}

void SetNum(int num) {

\*reg\_num = num;

}

string GetMark() {

return \*mark;

}

void SetMark(string m) {

\*mark = m;

}

Train() {

reg\_num = new int();

mark = new string();

}

~Train() {

delete reg\_num;

reg\_num = nullptr;

delete mark;

mark = nullptr;

}

};