# Оглавление

[Оглавление 1](#_Toc135670805)

[Задание 2](#_Toc135670806)

[Теория 3](#_Toc135670807)

Блок-схема……………………………………………………………………………………………………………………………………………..6

[Описание функций 7](#_Toc135670808)

Код программы……………………………………………………………………………………………………………………………………17

Тесты……………………………………………………………………………………………………………………………………………………22

[Вывод 27](#_Toc135670809)

# Задание

**Схема организации данных**

Линейные и кольцевые двусвязные списки

**Темы курсовых работ**

Система учета автобусов для автовокзала: название автовокзала, марка автобуса, количество проданных билетов, расчетное количество пассажиров, цена билета, номер рейса, время отправления, пункт назначения, ФИО водителя.

В каждой работе предусмотреть функции добавления, удаления элементов, поиска, печати, перемещение в прямом и обратном направлениях (для двусвязного списка), обеспечить уникальность значений полей и защиту от одновременного использования одного ресурса, где это необходимо. Возможно несколько узлов с одинаковыми значениями полей. Обеспечить выдачу запросов по любым полям структуры.

Создать списочную структуру, заполнить достаточным количеством данных, провести тестирование. Инициализацию структур произвести данными из файла. В отчете представить разработанные структуры данных, описать алгоритмы работы с ними для каждой функции.

Проект должен состоять из трех частей: заголовочные файлы \*.h, файлы с реализациями функций \*.cpp, файл основной программы\*.cpp.

Проект должен обеспечивать простоту и удобство масштабирования поставленной задачи.

# Структурные схемы алгоритмов

## Функция main

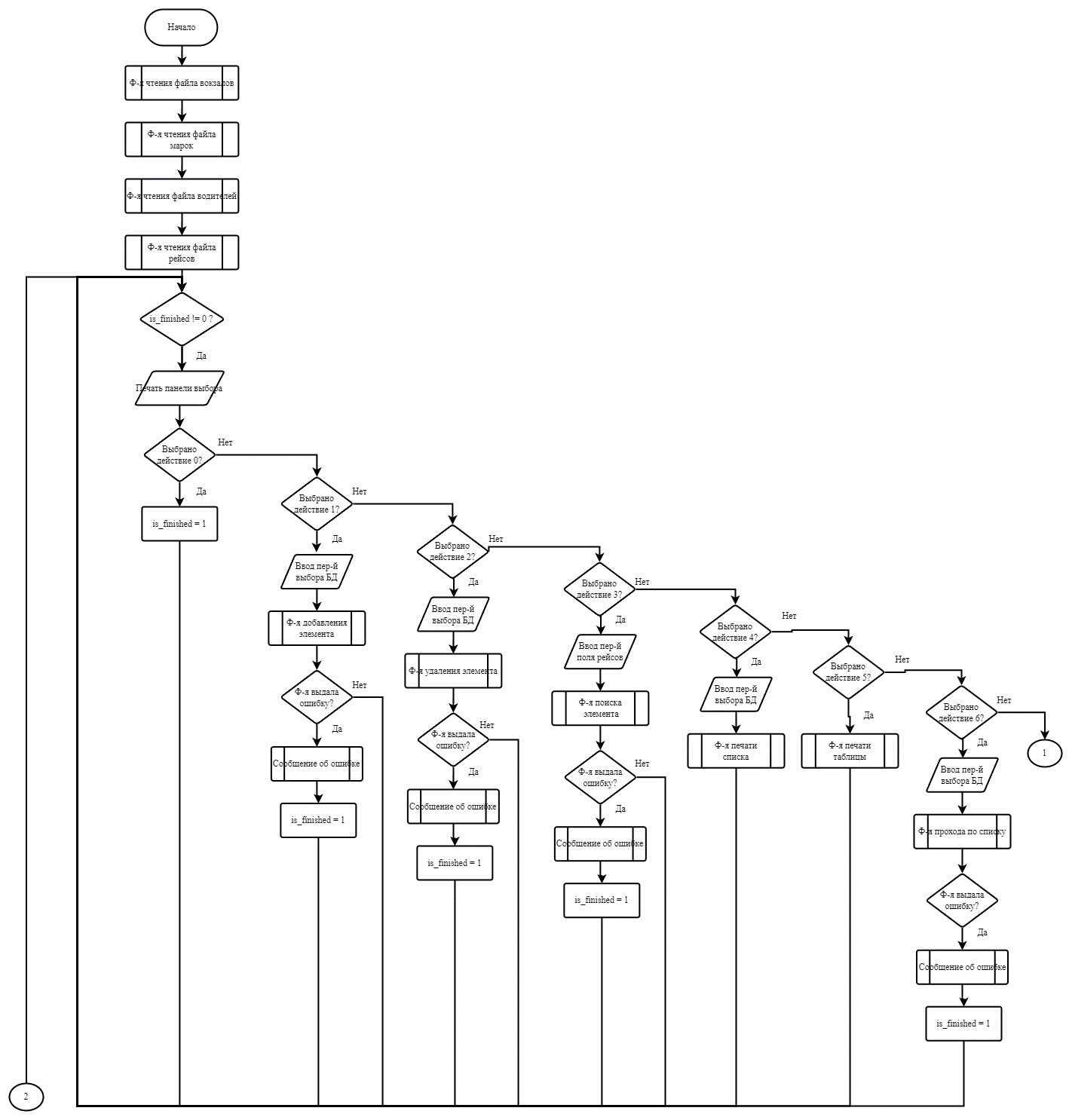


Рисунок 1. Структурная схема алгоритма функции main.

## Функция create

1. Назначение:

Создание списка из элементов, представленных в файле

2. Прототип функции:

int create(List\*& head, List\*& end, const char\* fileName)

3. Обращение к функции:

ErrCode = create(head, end, fileName)

4. Блок-схема:

Рисунок 2. Структурная схема алгоритма функции create

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | List\* | Указатель на первый элемент списка | Вход |
| end | List\* | Указатель на последний элемент списка | Вход |
| fileName | Const char\* | Имя файла | Вход |

# Тестирование программы

## Тестирование некорректных режимов

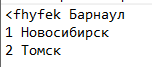
Тестирование функций записи из файлов

Тест 1

Цель: Проверить работу программы при строке в поле id

Исходные данные:

Вокзалы Марки

Водители Рейсы

Ожидаемый результат: «Несоответствие типов полей структуры с подаваемым значением!»

Полученный результат:

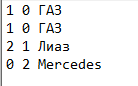


Тест 2

Цель: Проверить работу программы при наличии одинаковых данных в одном файле

Исходные данные:

Вокзалы Марки

Водители Рейсы

Ожидаемый результат:

«Все id должны быть уникальными!»

Полученный результат:



Тест 3

Цель: Проверить работу программы при повторяющихся номерах рейсов

Исходные данные:

Вокзалы Марки

Водители Рейсы

Ожидаемый результат:

«Номера рейсов не должны повторяться!»

Полученный результат:



Тест 4

Цель: Проверить работу программы при одинаковых ФИО водителей в файле

Исходные данные:

Вокзалы Марки

Водители Рейсы

Ожидаемый результат:

«Названия вокзалов, марок, ФИО водителей должны быть уникальными!»

Полученный результат:



Тестирование функций добавления элементов

Тест 5

Цель: Проверить работу программы при попытке добавить автобус в несуществующий вокзал.

Исходные данные:

Вокзалы Марки

Водители Рейсы

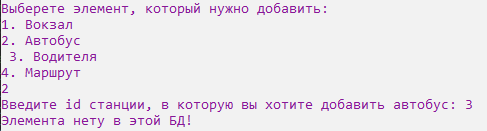
 

Id несуществующего вокзала = 3

Ожидаемый результат:

«Элемента нету в этой БД!»

Полученный результат:



Тест 6

Цель: Проверить работу программы при попытке добавить рейс с несуществующим автобусом.

Исходные данные:

Вокзалы Марки

Водители Рейсы

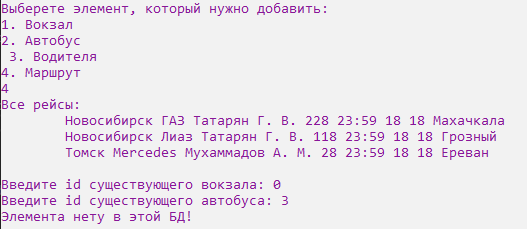
 

Id несуществующего автобуса = 3

Ожидаемый результат:

«Элемента нету в этой БД!»

Полученный результат:



Тестирование функций удаления элементов

Цель: Проверить работу программы при попытке добавить рейс с несуществующим автобусом.

Исходные данные:

Вокзалы Марки

Водители Рейсы

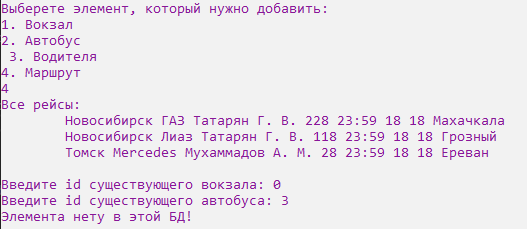
 

Id несуществующего автобуса = 3

Ожидаемый результат:

«Элемента нету в этой БД!»

Полученный результат:



## Тестирование корректных режимов

Тест 1

Цель: Проверить работу программы при корректных входных данных из вещественных чисел

Исходные данные:

Файл corr1.txt

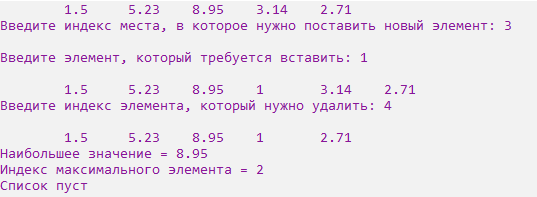
reqData = 1, reqIdx = 3, reqIdx2 = 4

Ожидаемый результат:

Список = [1.5, 5.23, 8.95, 1, 2.71]

Наибольшее значение = 8.95, индекс = 2

Полученный результат:



Тест 2

Цель: Проверить работу программы при корректных входных данных из целых чисел

Исходные данные:

Файл corr2.txt

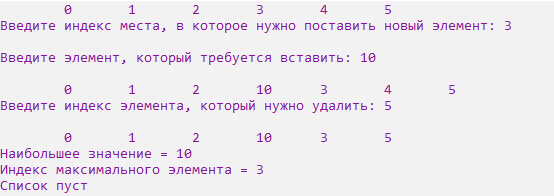
reqData = 10, reqIdx = 3, reqIdx2 = 5

Ожидаемый результат:

Список = [0, 1, 2, 10, 3, 5]

Наибольшее значение = 10, индекс = 3

Полученный результат:



Тест 3

Цель: Проверить работу программы при корректных входных данных из целых чисел

Исходные данные:

Файл corr3.txt

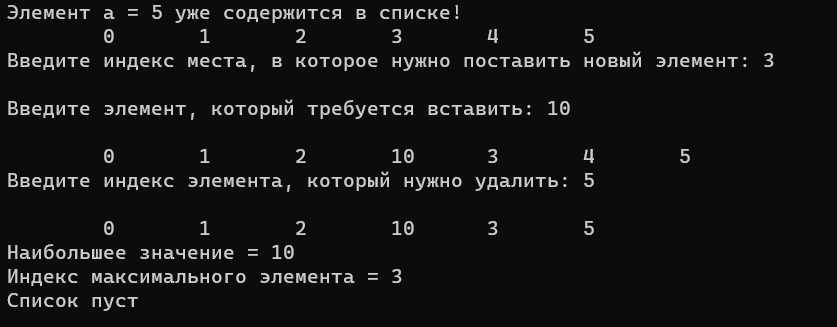
reqData = 10, reqIdx = 3, reqIdx2 = 5

Ожидаемый результат:

Список = [0, 1, 2, 10, 3, 5]

Наибольшее значение = 10, индекс = 3

Полученный результат:



Вывод по тестам: программа работает корректно.

# Вывод

Разработка программы остановлена на том основании, что ожидаемые результаты тестов совпали с полученными.