# Оглавление

[Оглавление 1](#_Toc154158721)

[Задание 2](#_Toc154158722)

[Структурные схемы алгоритмов 4](#_Toc154158723)

[Функция main 4](#_Toc154158724)

[Функция create\_stations 5](#_Toc154158725)

[Функция switch\_add 7](#_Toc154158726)

[Функция add\_stations 10](#_Toc154158727)

[Функция switch\_del 11](#_Toc154158728)

[Функция switch\_move 13](#_Toc154158729)

[Функция makeStation 15](#_Toc154158730)

[Функция DeleteStation 16](#_Toc154158731)

[Функция findElemStation 18](#_Toc154158732)

[Функция find\_id\_station 19](#_Toc154158733)

[Функция find\_route\_stationID 20](#_Toc154158734)

[Тестирование программы 22](#_Toc154158735)

[Тестирование некорректных режимов 22](#_Toc154158736)

[Тестирование корректных режимов 29](#_Toc154158737)

[Вывод 39](#_Toc154158738)

# Задание

**Схема организации данных**

Линейные и кольцевые двусвязные списки

**Тема курсовой работы**

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.

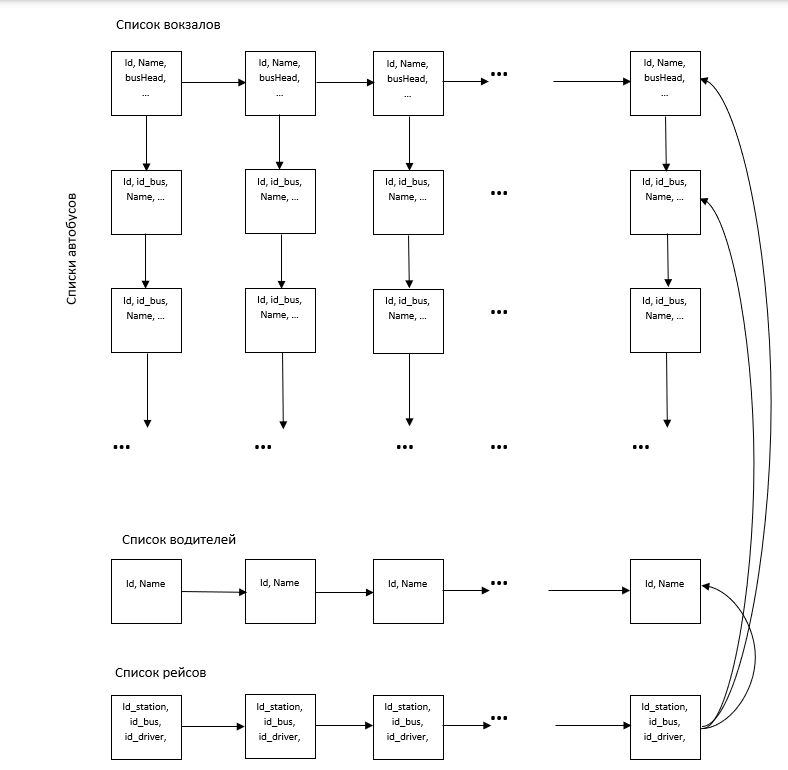
В каждой работе предусмотреть функции добавления, удаления элементов, поиска, печати, перемещение в прямом и обратном направлениях (для двусвязного списка), обеспечить уникальность значений полей и защиту от одновременного использования одного ресурса, где это необходимо. Возможно несколько узлов с одинаковыми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

# Схема списков



# Описание структур

Station

1. Назначение: Структура, содержащая данные вокзале.

2. Прототип структуры:

struct Station // Двунаправленный кольцевой АВТОВОКЗАЛЫ

{

Station\* prev;

Station\* next;

int id; // ID

string name; // Назв автовокзала

Bus\* busHead = NULL; // Голова списка автобусов для данного вокзала

Bus\* busEnd = NULL; // Хвост списка автобусов

};

3. Описание полей:

|  |  |  |
| --- | --- | --- |
| Имя параметра | Тип параметра | Назначение |
| next | Station\* | Указатель на след. эл-т |
| prev | Station\* | Указатель на пред. Элемент |
| id | int | Id данного вокзала |
| name | string | Название данного вокзала |
| busHead | Bus\* | Указатель на первый элемент списка автобусов, относящихся к данному вокзалу |
| busEnd | Bus\* | Указатель на последний элемент списка автобусов, относящихся к данному вокзалу |

Bus

1. Назначение:

Структура, содержащая данные о марках (автобусах), относящихся к конкретному вокзалу

2. Прототип структуры:

struct Bus // Двунаправленный кольцевой АВТОБУСЫ

{

Bus\* prev;

Bus\* next;

int id; // ID автовокзала

int id\_bus; // ID автобуса

string name; // Назв марки

};

3. Описание полей:

|  |  |  |
| --- | --- | --- |
| Имя параметра | Тип параметра | Назначение |
| next | Bus\* | Указатель на след. эл-т |
| prev | Bus\* | Указатель на пред. элемент |
| id | int | Id вокзала, к которому относится данный автобус |
| Id\_bus | int | Id данного автобуса (уникальный) |
| name | string | Название марки (уникальна в данном списке) |

Driver

1. Назначение:

Структура, содержащая данные водителях

2. Прототип структуры:

struct Driver // Водитель (линейный)

{

Driver\* next;

string name; // ФИО

int id; // ID водителя

};

3. Описание полей:

|  |  |  |
| --- | --- | --- |
| Имя параметра | Тип параметра | Назначение |
| next | Driver\* | Указатель на след. эл-т |
| id | int | Id данного водителя |
| name | string | ФИО водителя |

Route

1. Назначение:

Структура, содержащая данные о рейсах

2. Прототип структуры:

struct Route { // Линейный (рейсы)

Route\* next;

int id\_station; // ID автовокзала

int id\_bus; // ID автобуса

int id\_driver; // ID водителя

int route\_number; // Номер рейса

string time; // Время отправления

int tickets; // Количество проданных билетов

int passengers; // Количество пассажиров

string end\_route; // Место назначения

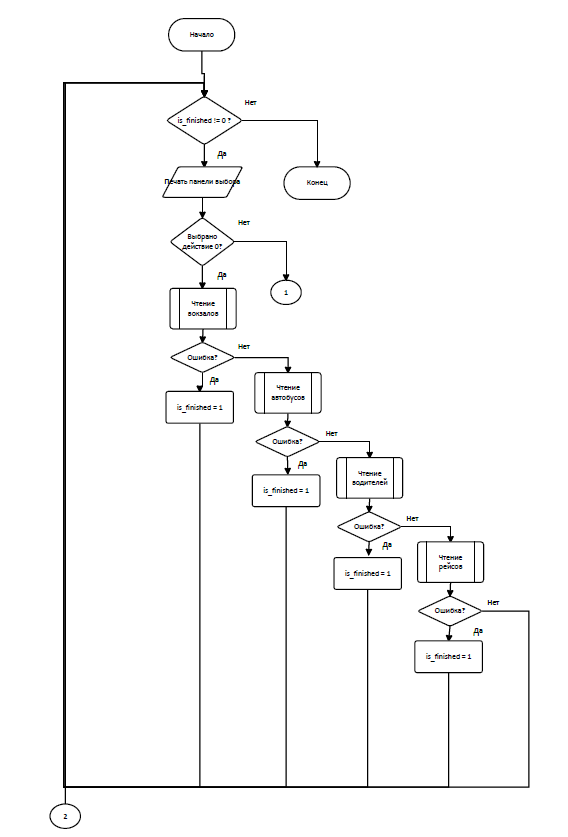
};

3. Описание полей:

|  |  |  |
| --- | --- | --- |
| Имя параметра | Тип параметра | Назначение |
| next | Route\* | Указатель на следующий элемент |
| Id\_station | int | Id вокзала |
| Id\_bus | int | Id автобуса |
| Id\_driver | int | Id водителя |
| Route\_number | int | Номер рейса (уникален) |
| time | string | Время отправления |
| tickets | int | Количество проданных билетов |
| passengers | int | Количество пассажиров |
| End\_route | string | Место назначения |

# Структурные схемы алгоритмов

## Функция main



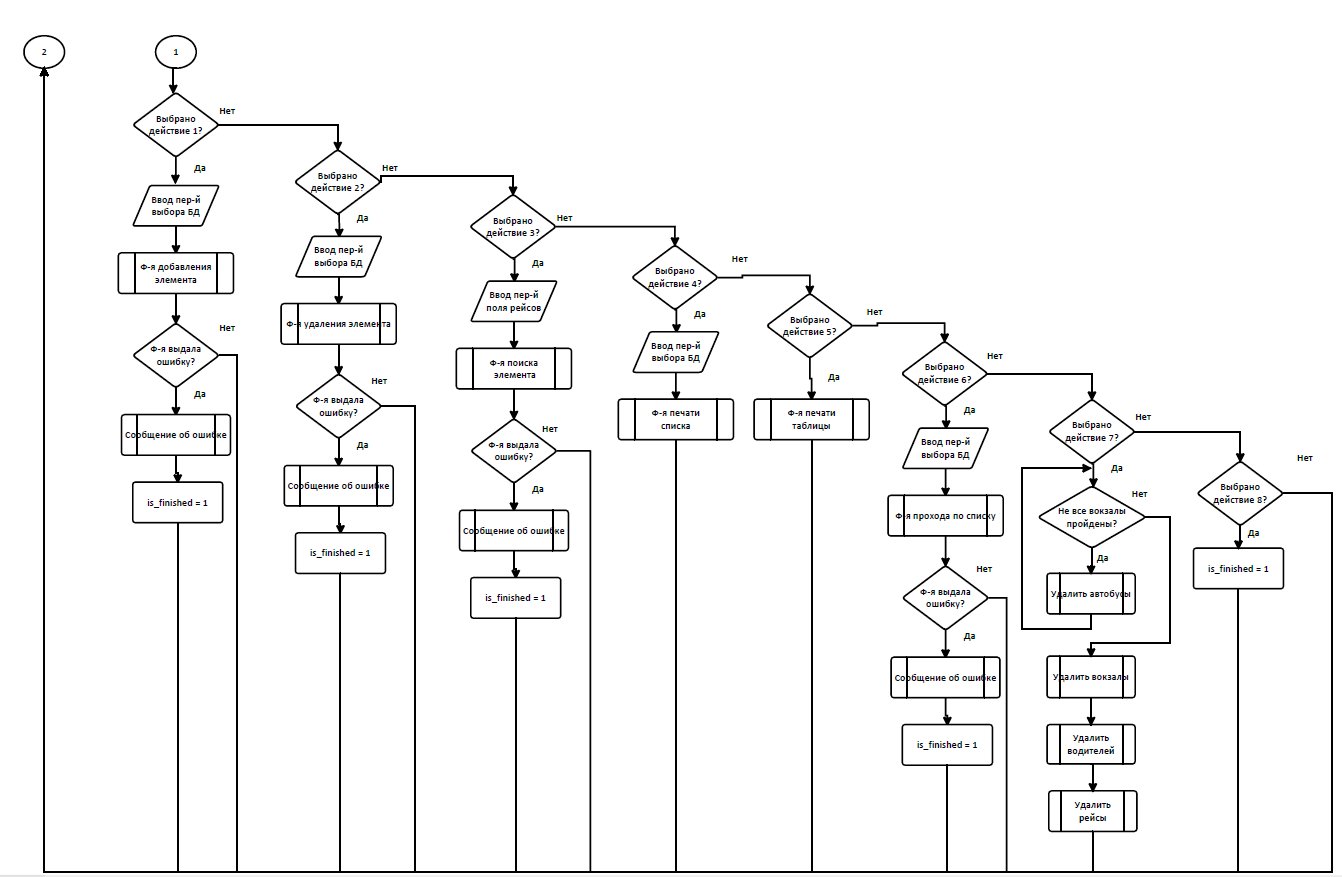


Рисунок 1. Структурная схема алгоритма функции main.

## Функция create\_stations

1. Назначение:

Создание списка вокзалов, представленных в файле

2. Прототип функции:

int create\_stations(const char\* fname, Station\*& head, Station\*& end)

3. Обращение к функции:

code = create\_stations(fstations, station\_head, station\_end)

4. Блок-схема:

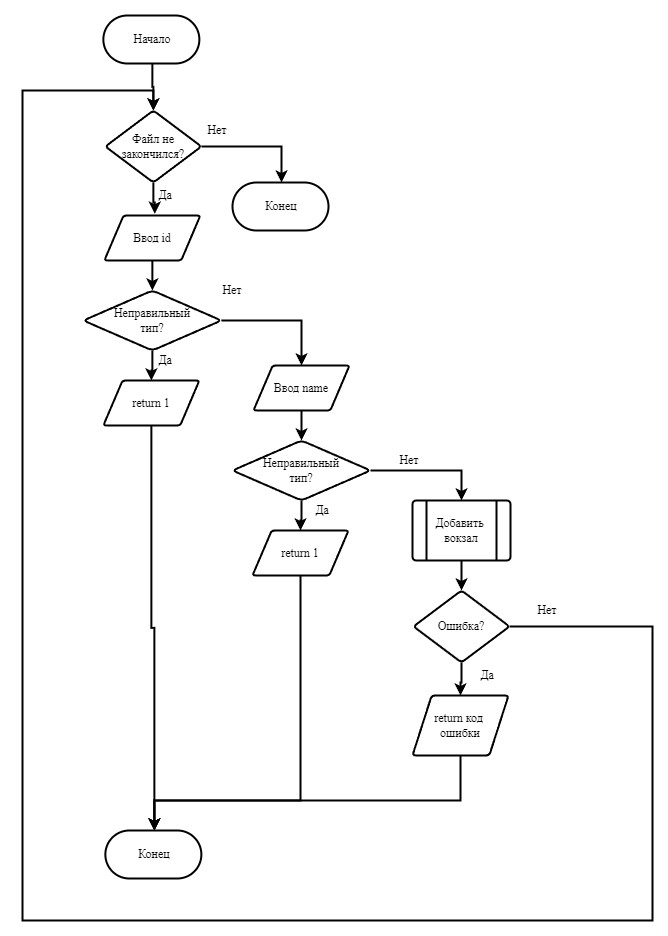


Рисунок 2. Структурная схема алгоритма функции create\_stations

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | Station\* | Указатель на первый элемент списка вокзалов | Вход |
| end | Station\* | Указатель на последний элемент списка вокзалов | Вход |
| fname | Const char\* | Имя файла | Вход |
| is\_mistake | int | Код ошибки добавления вокзала | Выход |

## Функция switch\_add

1. Назначение:

Запрос пользователю на выбор списка для добавления нового элемента

2. Прототип функции:

int switch\_add(Station\*& s\_head, Station\*& s\_end, Driver\*& d\_head, Driver\*& d\_end, Route\*& r\_head, Route\*& r\_end)

3. Обращение к функции:

code = switch\_add(station\_head, station\_end, driver\_head, driver\_end, route\_head, route\_end)

4. Блок-схема:

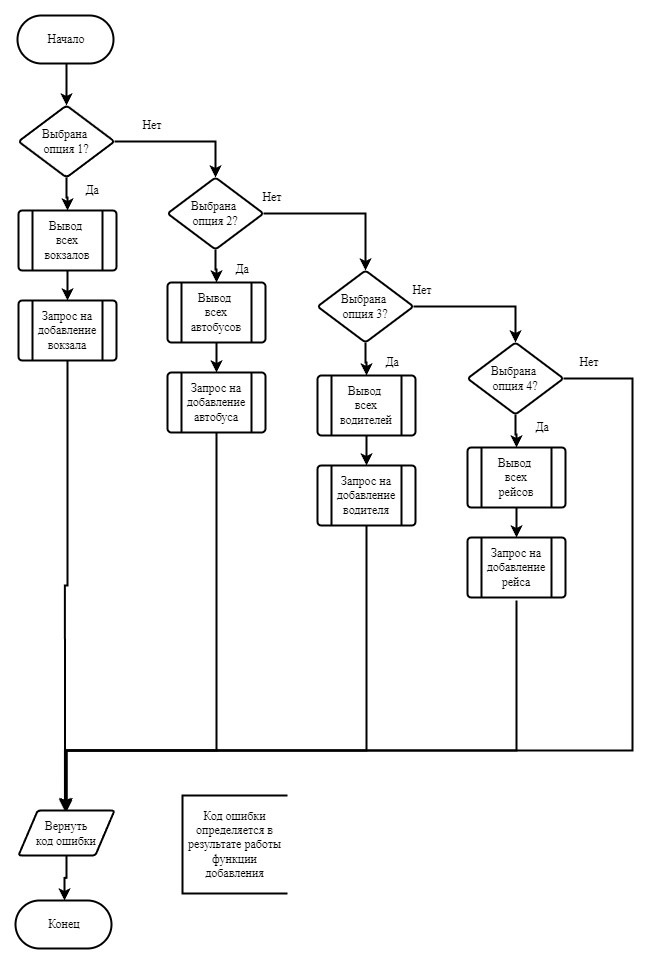


Рисунок 3. Структурная схема алгоритма функции switch\_add

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| s\_head | Station\* | Указатель на первый элемент списка вокзалов | Вход |
| s\_end | Station\* | Указатель на последний элемент списка вокзалов | Вход |
| d\_head | Driver\* | Указатель на первый элемент списка водителей | Вход |
| d\_end | Driver\* | Указатель на последний элемент списка водителей | Вход |
| r\_head | Route\* | Указатель на первый элемент списка рейсов | Вход |
| r\_end | Route\* | Указатель на последний элемент списка рейсов | Вход |

## Функция add\_stations

1. Назначение:

Запрос пользователю на добавление нового вокзала

2. Прототип функции:

int add\_stations(Station\*& head, Station\*& end)

3. Обращение к функции:

mistake\_code = add\_stations(s\_ head, s\_end)

4. Блок-схема:

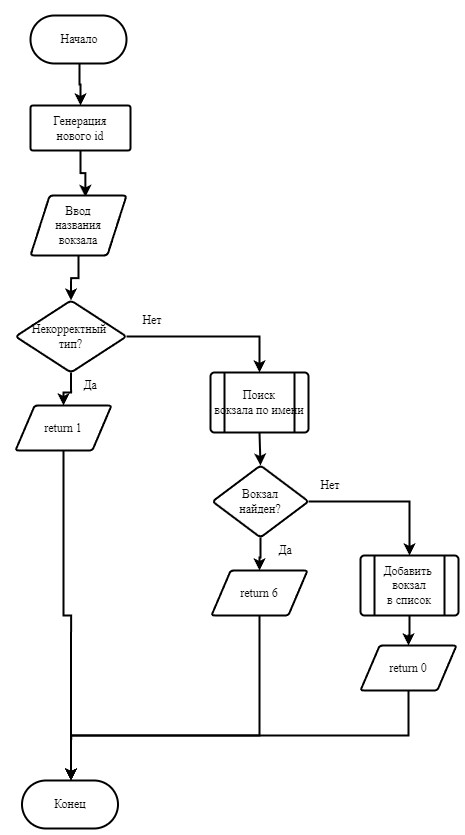


Рисунок 4. Структурная схема алгоритма функции add\_stations

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| head | Station\* | Указатель на первый элемент списка вокзалов | Вход |
| end | Station\* | Указатель на последний элемент списка вокзалов | Вход |
| loc\_choice | int | Глобальная переменная выбора опции | Вход |

## Функция switch\_del

1. Назначение:

Запрос пользователю на выбор списка для удаления элемента

2. Прототип функции:

int switch\_del(Station\*& s\_head, Station\*& s\_end, Driver\*& d\_head, Driver\*& d\_end, Route\*& r\_head, Route\*& r\_end)

3. Обращение к функции:

code = switch\_del(station\_head, station\_end, driver\_head, driver\_end, route\_head, route\_end)

4. Блок-схема:

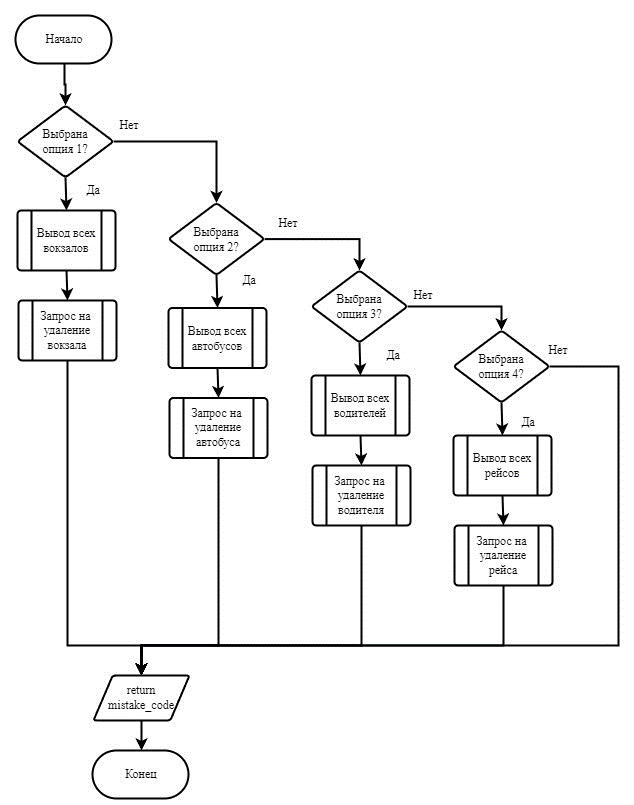


Рисунок 5. Структурная схема алгоритма функции switch\_del

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| s\_head | Station\* | Указатель на первый элемент списка вокзалов | Вход |
| s\_end | Station\* | Указатель на последний элемент списка вокзалов | Вход |
| d\_head | Driver\* | Указатель на первый элемент списка водителей | Вход |
| d\_end | Driver\* | Указатель на последний элемент списка водителей | Вход |
| r\_head | Route\* | Указатель на первый элемент списка рейсов | Вход |

## Функция switch\_move

1. Назначение:

Запрос пользователю на выбор списка для перемещения

2. Прототип функции:

int switch\_move(Station\* s\_head, Station\*& curr\_station, Bus\*& curr\_bus)

3. Обращение к функции:

code = switch\_ move(station\_head, curr\_station, curr\_bus)

4. Блок-схема:

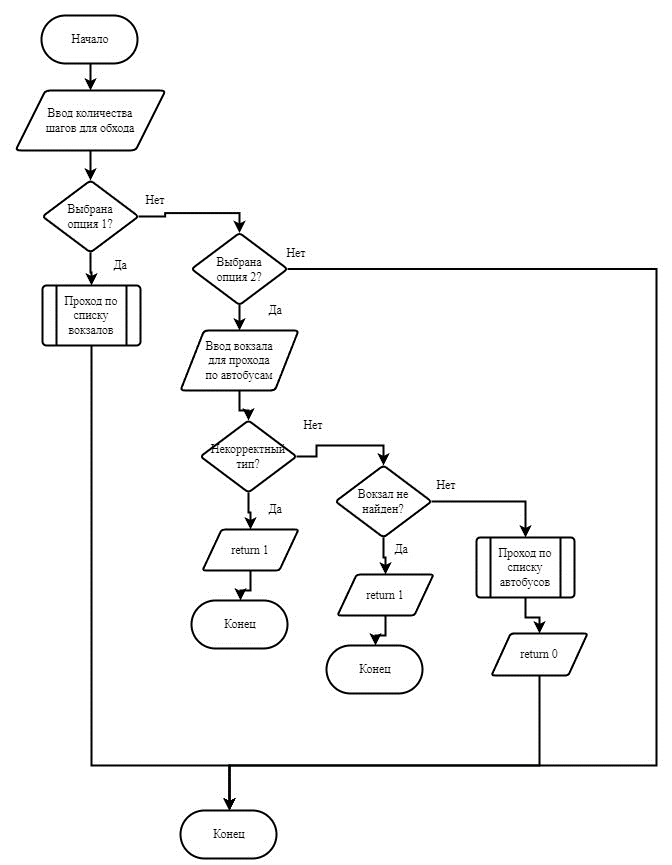


Рисунок 6. Структурная схема алгоритма функции switch\_move

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| s\_head | Station\* | Указатель на первый элемент списка вокзалов | Вход |
| curr\_station | Station\* | Указатель на последний посещенный вокзал | Вход |
| curr\_bus | Bus\* | Указатель на последний посещенный автобус | Вход |

## Функция makeStation

1. Назначение:

Добавление нового вокзала в список

2. Прототип функции:

int makeStation(int id, string name, Station\*& end, Station\*& head)

3. Обращение к функции:

is\_mistake = makeStation(id, name, end, head)

4. Блок-схема:

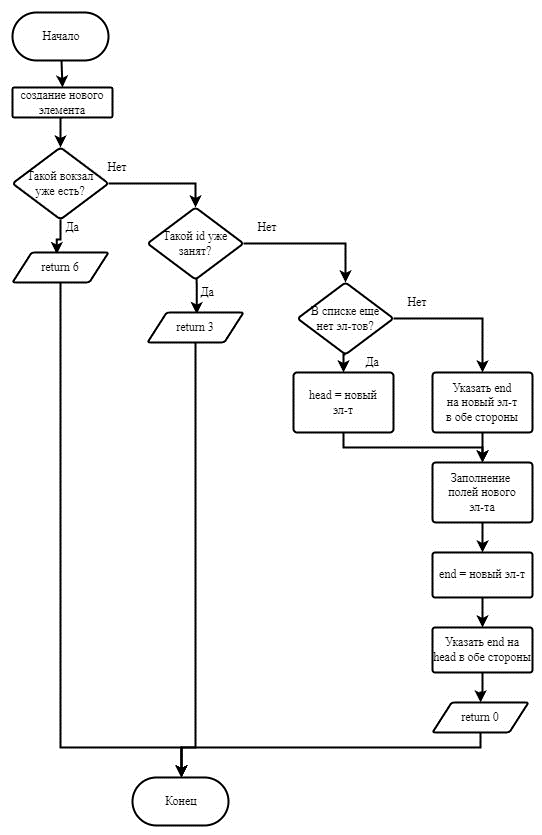


Рисунок 7. Структурная схема алгоритма функции makeStation

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| id | Int | id нового вокзала | Вход |
| name | string | Название нового вокзала | Вход |
| end | Station\* | Указатель на последний вокзал | Вход |
| head | Station\* | Указатель на первый вокзал | Вход |

## Функция DeleteStation

1. Назначение:

Удаление вокзала по его id

2. Прототип функции:

void DeleteStation(Station\*& head, Station\*& end, int id)

3. Обращение к функции:

DeleteStation(head, end, id)

4. Блок-схема:

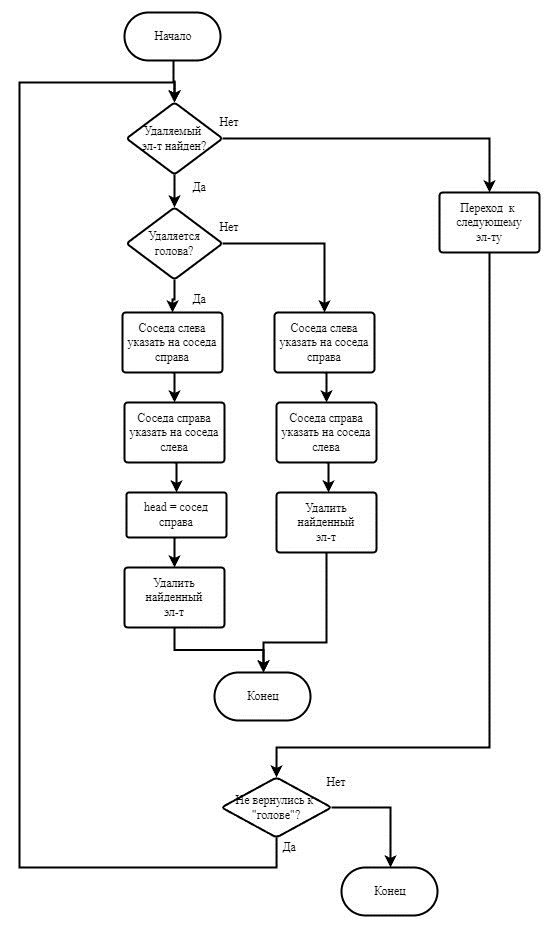


Рисунок 8. Структурная схема алгоритма функции makeStation

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| id | Int | id вокзала | Вход |
| end | Station\* | Указатель на последний вокзал | Вход |
| head | Station\* | Указатель на первый вокзал | Вход |

## Функция findElemStation

1. Назначение:

Поиск указателя на искомый вокзал по его id

2. Прототип функции:

Station\* findElemStation(int id, Station\* head)

3. Обращение к функции:

findElemStation(id, head)

4. Блок-схема:

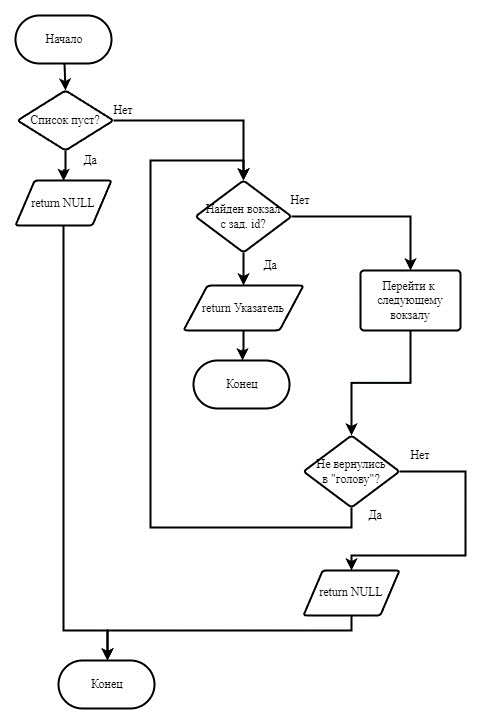


Рисунок 9. Структурная схема алгоритма функции findElemStation

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| id | Int | id вокзала | Вход |
| head | Station\* | Указатель на первый вокзал | Вход |
| ptr | Station\* | Указатель на найденный вокзал | Выход |

## Функция find\_id\_station

1. Назначение:

Поиск указателя на искомый вокзал по его названию

2. Прототип функции:

int find\_id\_station(string name, Station\* head)

3. Обращение к функции:

id = find\_id\_station(str\_filter, st\_head)

4. Блок-схема:

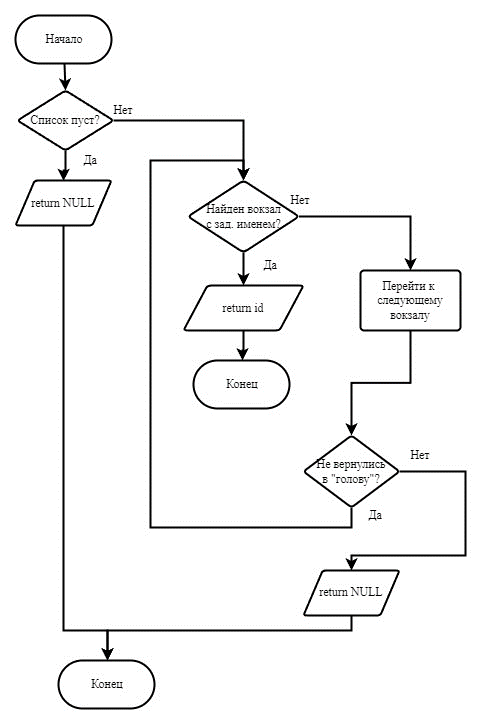


Рисунок 10. Структурная схема алгоритма функции find\_id\_station

5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| name | string | название вокзала | Вход |
| head | Station\* | Указатель на первый вокзал | Вход |
| Id | int | Id найденного вокзала | Выход |

## Функция find\_route\_stationID

1. Назначение:

Поиск указателя на искомый вокзал, находящийся в одном из рейсов, по его id

2. Прототип функции:

Route\* find\_route\_stationID(int id\_station, Route\* head)

3. Обращение к функции:

found = find\_route\_stationID(id, route\_head)

4. Блок-схема:

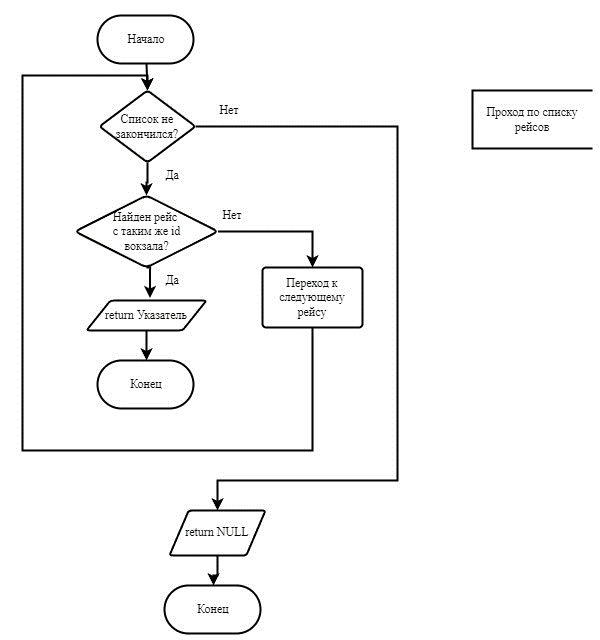
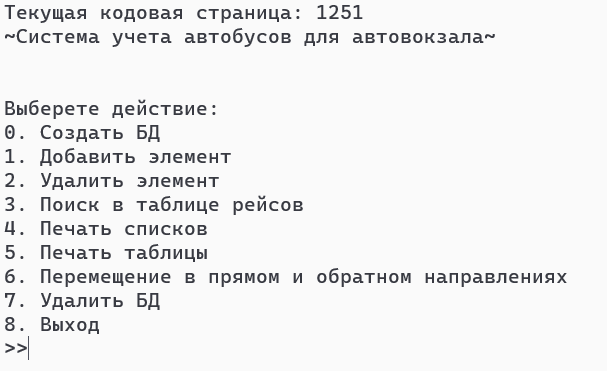


Рисунок 11. Структурная схема алгоритма функции find\_route\_stationID

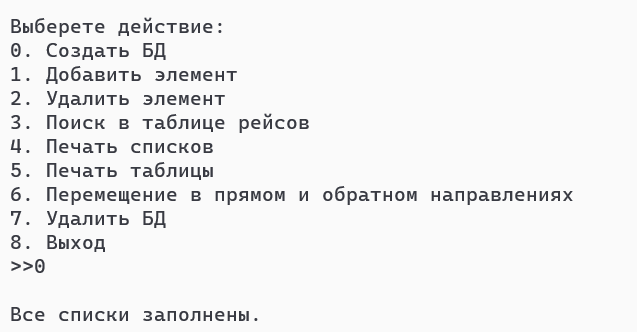
5. Описание параметров:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя параметра | Тип параметра | Назначение | Вход или выход |
| ptr | Route\* | Указатель на рейс с искомым вокзалом | Выход |
| head | Route\* | Указатель на первый рейс | Вход |
| Id\_station | int | Id искомого вокзала | Вход |

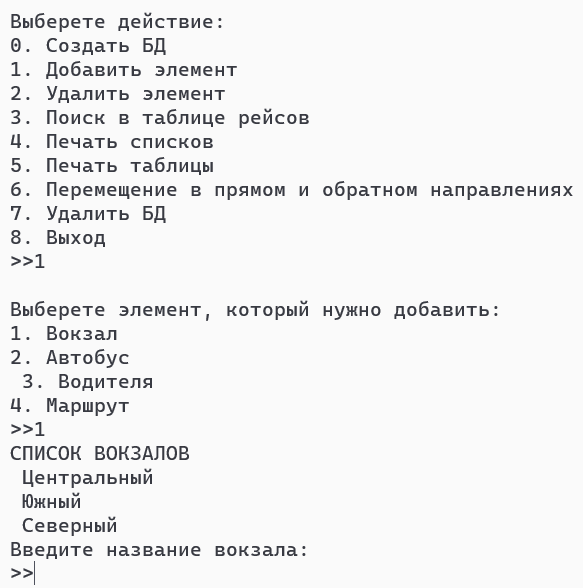
# Описание интерфейса



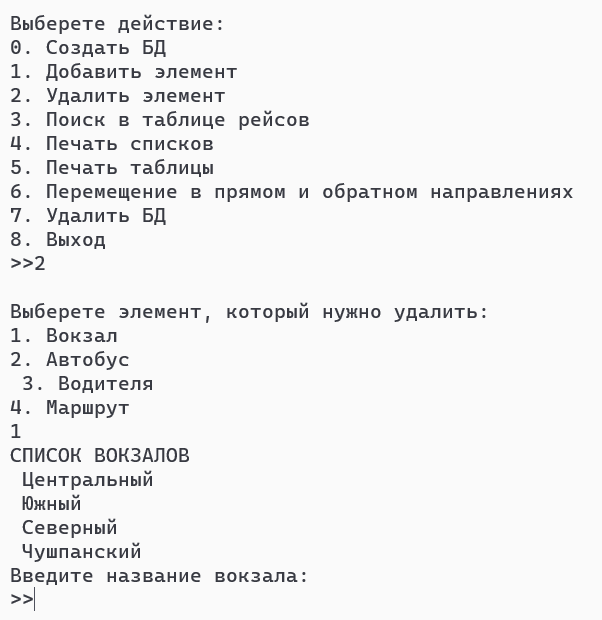
Опция 0 заполняет все списки данными из файлов



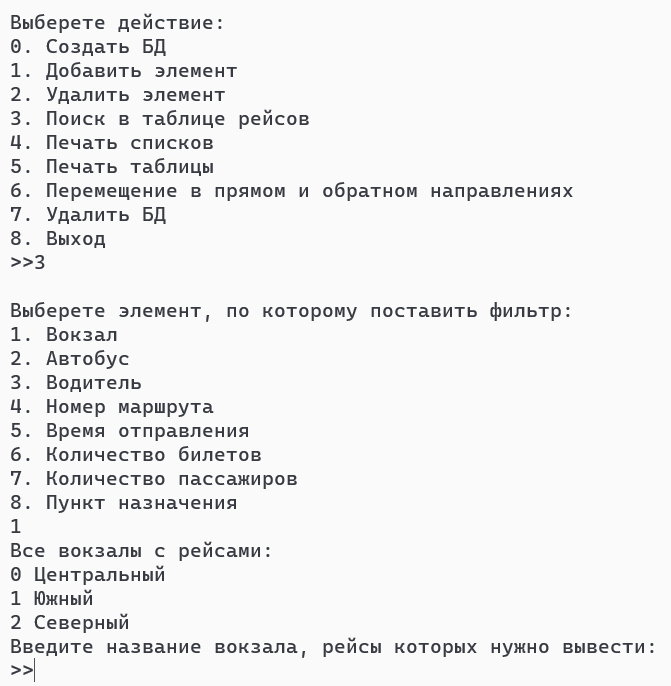
Опция 1 запускает функцию добавления элемента, предлагая предварительно выбрать список для добавления.



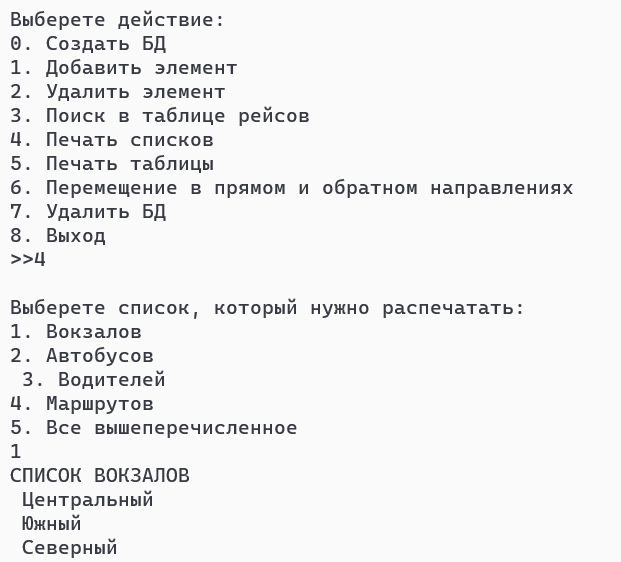
Опция 2 запускает функцию удаления элемента, предлагая предварительно выбрать список для удаления.



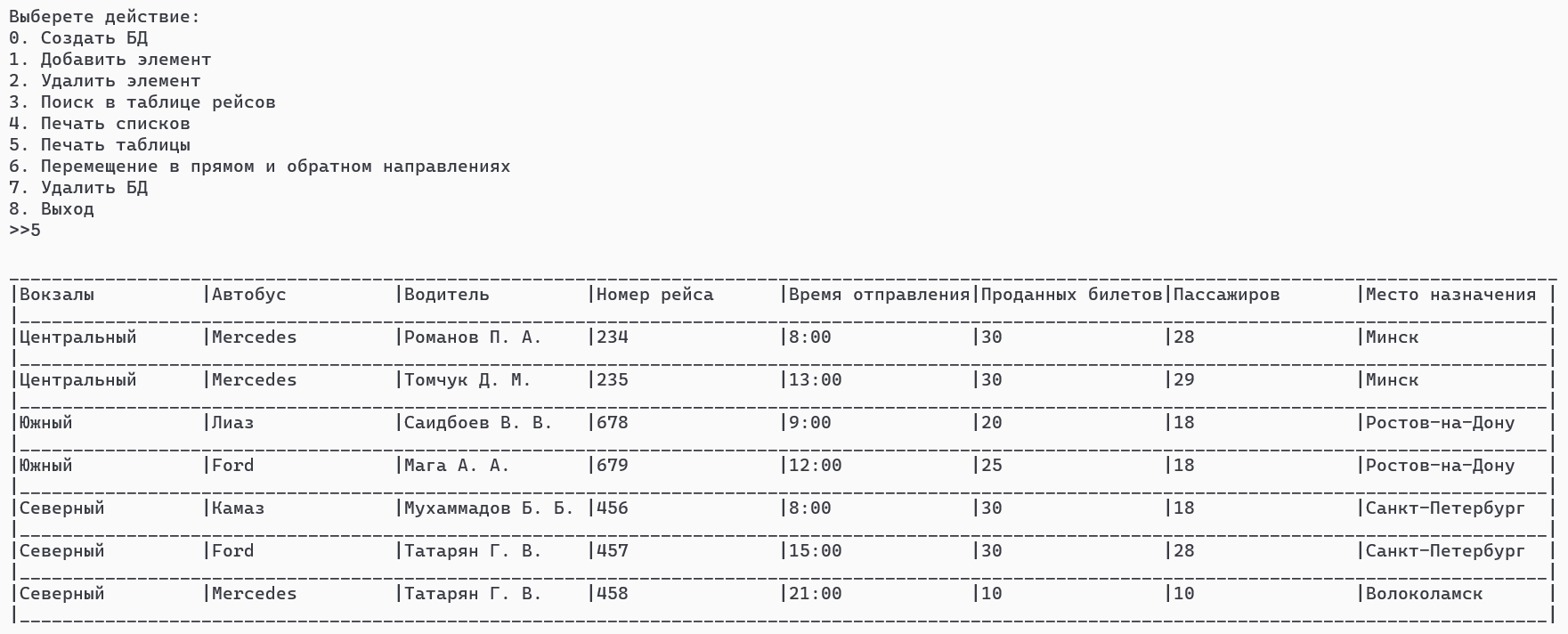
Опция 3 запускает функцию поиска элемента среди списка рейсов.



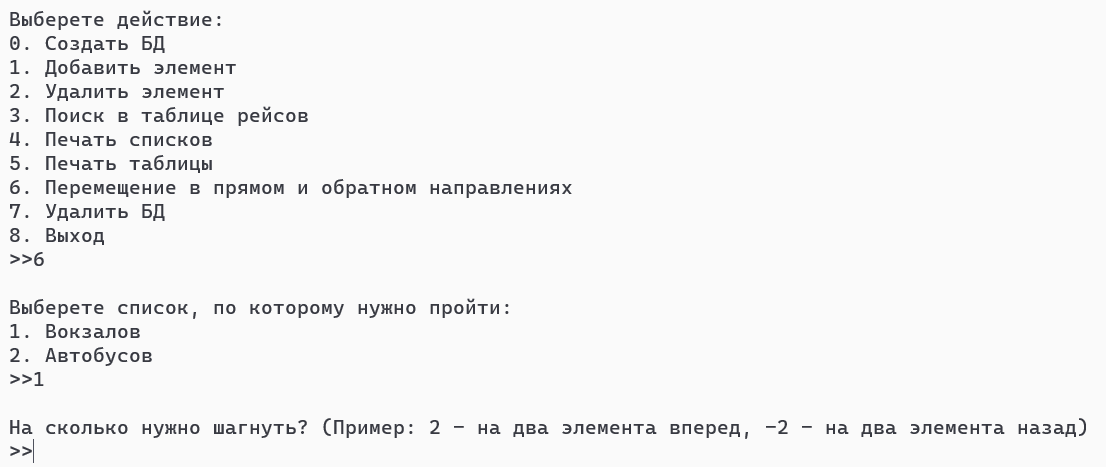
Опция 4 запускает функцию печати списков, предлагая предварительно выбрать сам список.



Опция 5 запускает функцию печати таблицы рейсов.



Опция 6 запускает функцию обхода кольцевых списков (вокзалов и автобусов) в прямом и обратном направлениях.



Опция 7 удаляет все списки.

Опция 8 совершает выход из программы.

# Тестирование программы

## Тестирование некорректных режимов

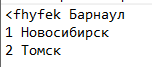
Тестирование функций записи из файлов

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**:

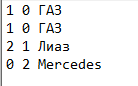


Тест 2

**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

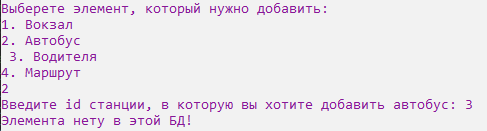
 

Id несуществующего вокзала = 3

**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

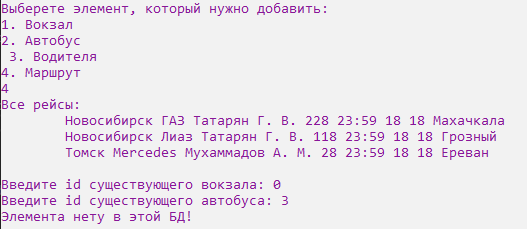
 

Id несуществующего автобуса = 3

**Ожидаемый результат**:

«Элемента нету в этой БД!»

**Полученный результат**:



Тестирование функций удаления элементов

Тест 7

**Цель**: Проверить работу программы при попытке удалить несуществующий вокзал.

**Исходные данные**:

Вокзалы Марки

Водители Рейсы

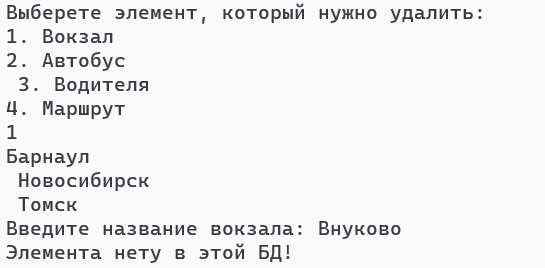
 

Несуществующий вокзал - Внуково

**Ожидаемый результат**:

«Элемента нету в этой БД!»

**Полученный результат**:



Тест 8

**Цель**: Проверить работу программы при попытке удалить несуществующий автобус

**Исходные данные**:

Вокзалы Марки

Водители Рейсы

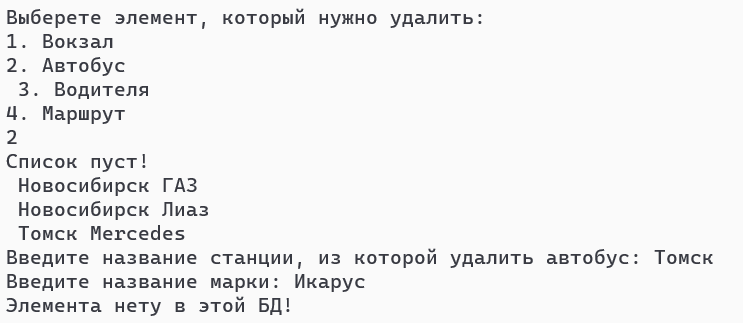
 

Несуществующий автобус - Икарус

**Ожидаемый результат**:

«Элемента нету в этой БД!»

**Полученный результат**:



Тестирование функций поиска элементов

Тест 9

**Цель**: Проверить работу программы при попытке найти несуществующий автобус

**Исходные данные**:

Вокзалы Марки

Водители Рейсы

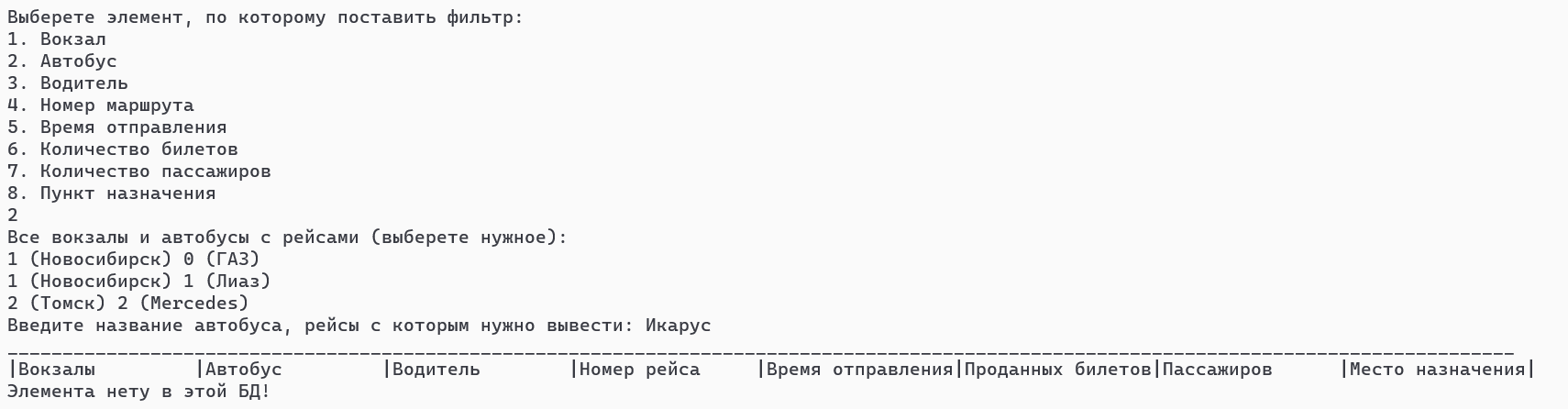
 

Несуществующий автобус - Икарус

**Ожидаемый результат**:

«Элемента нету в этой БД!»

**Полученный результат**:



Тест 10

**Цель**: Проверить работу программы при попытке найти рейс с неправильно заданным временем

**Исходные данные**:

Вокзалы Марки

Водители Рейсы

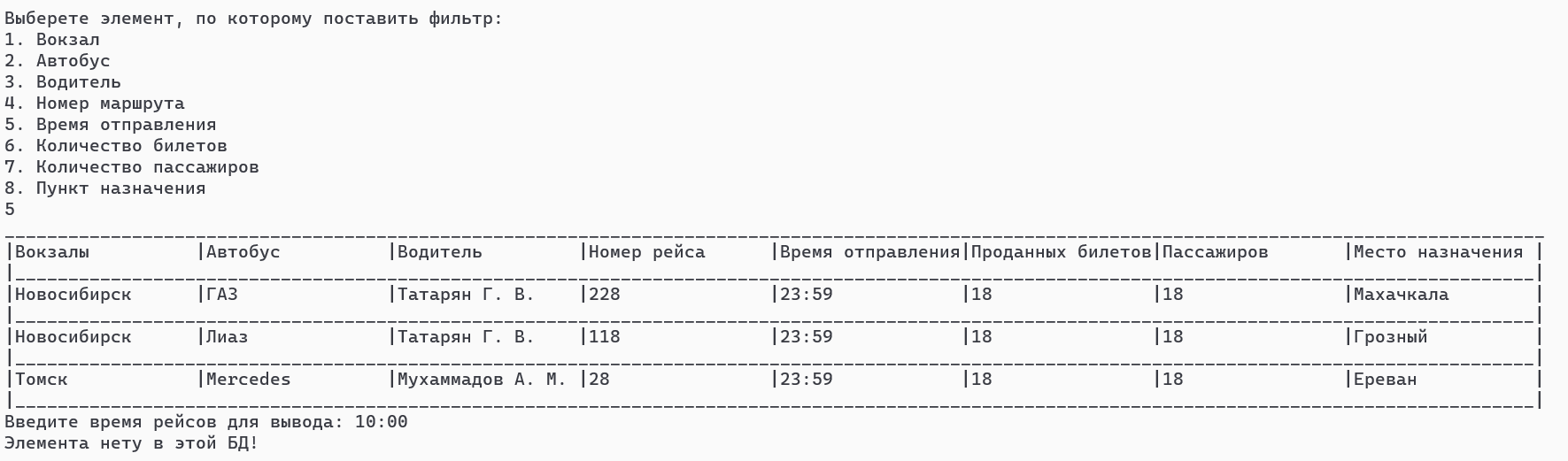
 

Запрашиваемое время – 10:00

**Ожидаемый результат**:

«Элемента нету в этой БД!»

**Полученный результат**:



## Тестирование корректных режимов

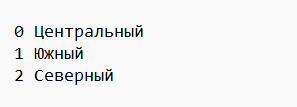
Тестирование функции вывода таблицы рейсов.

Тест 1

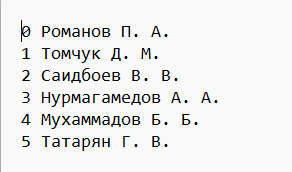
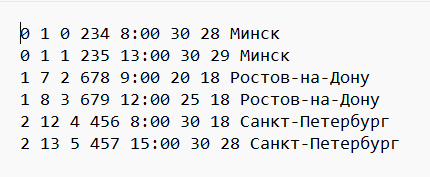
**Цель**: Проверить функцию вывода таблицы при корректных входных данных

**Исходные данные**:

Вокзалы Марки

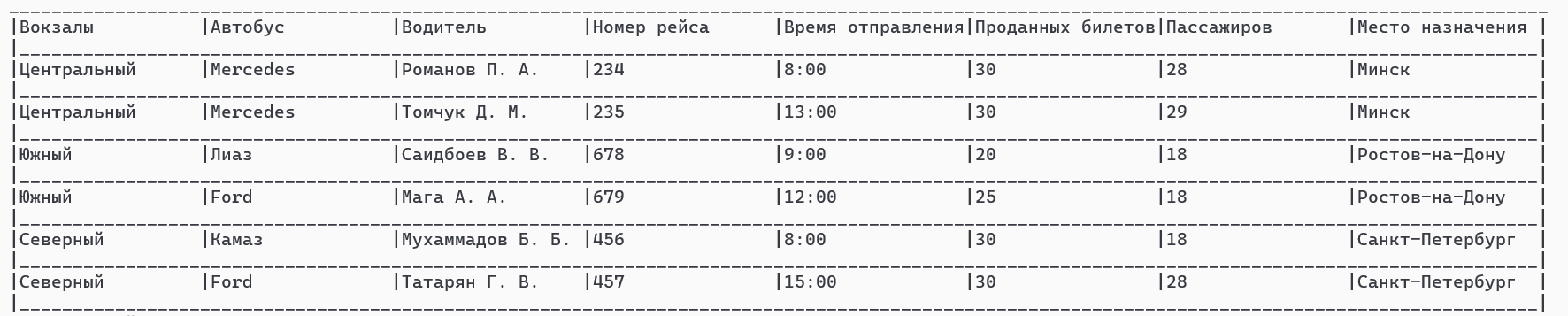
Водители Рейсы

**Ожидаемый результат**:

Таблица рейсов с id, замененными на соответствующие названия.

**Полученный результат**:



Проверка функции добавления

Тест 2

**Цель**: Проверить работу программы при попытке вставить нового водителя.

**Исходные данные**: Те же самые, водитель – Бустиренко Г. В.

**Ожидаемый результат**:

Обновленный список водителей:

Романов П. А.

Томчук Д. М.

Саидбоев В. В.

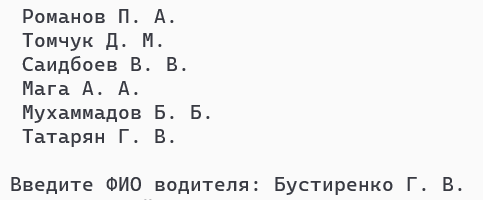
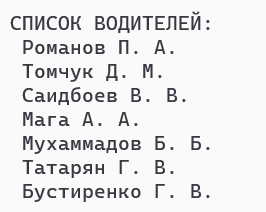
Мага А. А.

Мухаммадов Б. Б.

Татарян Г. В.

Бустиренко Г. В.

**Полученный результат**:

Тест 3

**Цель**: Создать новый рейс с добавленным водителем

**Исходные данные**:

Те же; поля нового рейса:

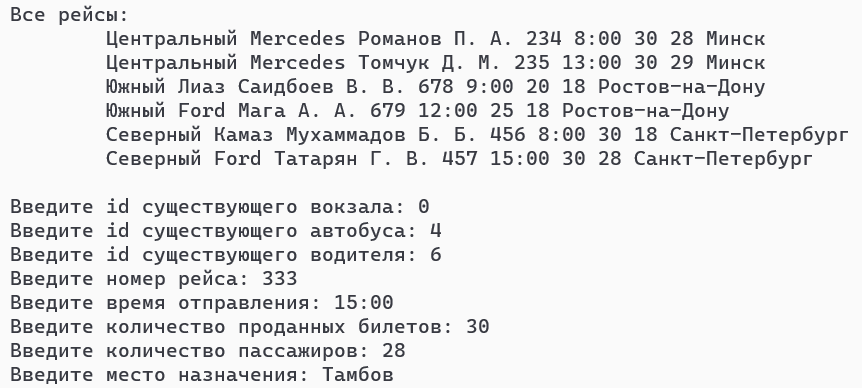
0 4 6 333 15:00 30 28 Тамбов

**Ожидаемый результат**:

Новая строка таблицы:

Центральный Камаз Бустиренко Г. В. 333 15:00 30 28 Тамбов

**Полученный результат**:





Тестирование функций удаления

Тест 4

**Цель**: Проверить работу программы при запросе удалить автобус, не находящийся ни в одном из рейсов, из существующего вокзала

**Исходные данные**:

Те же, что в тесте 1.

Вокзал: Центральный;

Удаляемый автобус: Камаз

**Ожидаемый результат**:

Обновленный список всех автобусов:

Центральный Mercedes

Центральный Hyondai

Центральный Лиаз

Южный Mercedes

Южный Hyondai

Южный Лиаз

Южный Ford

Северный Mercedes

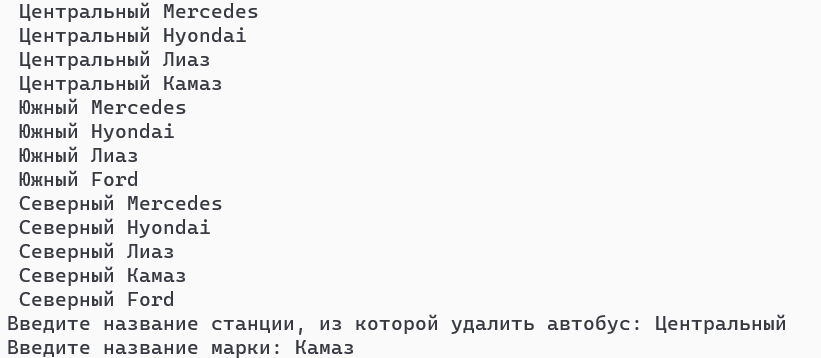
Северный Hyondai

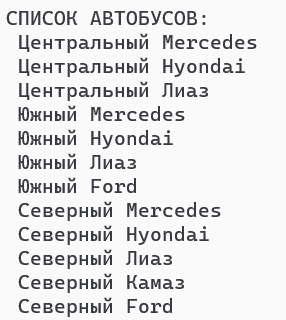
Северный Лиаз

Северный Камаз

Северный Ford

**Полученный результат**:





Тест 5

**Цель**: Проверить работу программы при запросе удалить автобус, находящийся в одном из рейсов, из существующего вокзала

**Исходные данные**:

Те же, что в тесте 1.

Вокзал: Центральный;

Удаляемый автобус: Mercedes

**Ожидаемый результат**:

Обновленный список всех автобусов:

Центральный Hyondai

Центральный Лиаз

Южный Mercedes

Южный Hyondai

Южный Лиаз

Южный Ford

Северный Mercedes

Северный Hyondai

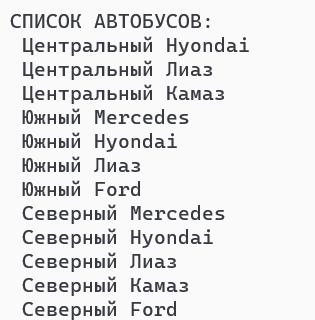
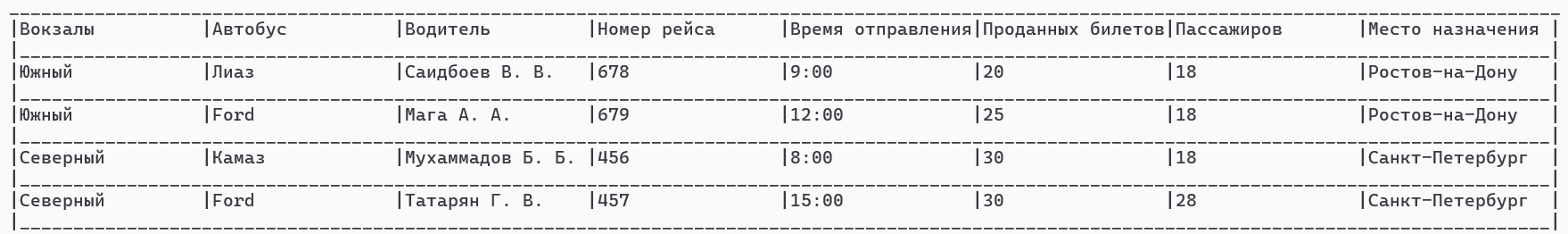
Северный Лиаз

Северный Камаз

Северный Ford

А также таблица рейсов без строк с удаленным автобусом.

**Полученный результат**:

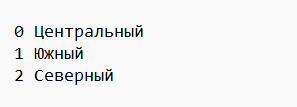
Тестирование функций поиска

Тест 6

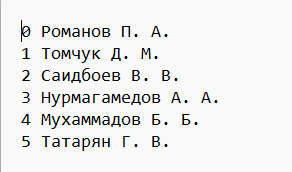
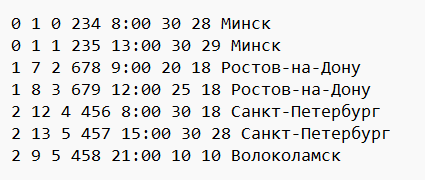
**Цель**: Проверить работу программы при поиске всех рейсов с заданным названием автобуса

**Исходные данные**:

Вокзалы Марки

Водители Рейсы

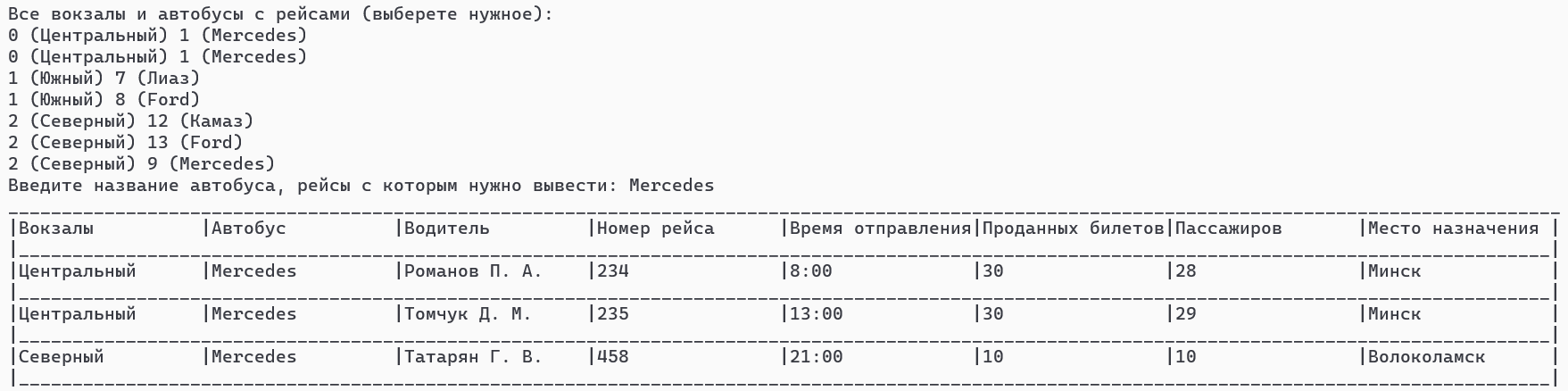
 

Автобус: Mercedes

**Ожидаемый результат:**

Таблица с автобусами Mercedes

**Полученный результат**:



Тест 7

**Цель**: Проверить работу программы при поиске всех рейсов с заданным временем отправления

**Исходные данные**:

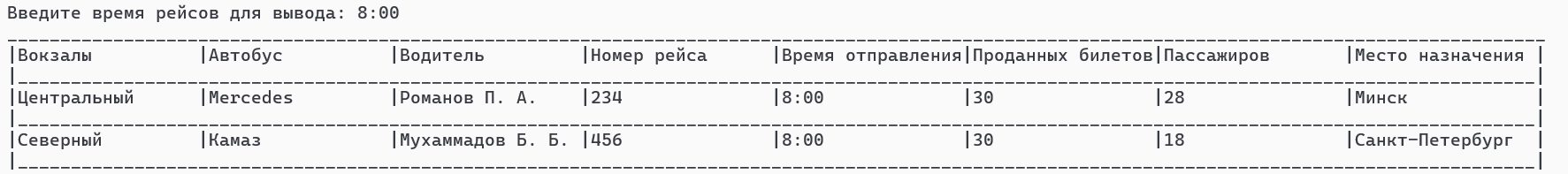
Те же, что в тесте 6

Время – 8:00

**Ожидаемый результат:**

Таблица рейсов с временем отправления 8:00

**Полученный результат**:



Тестирование функций печати

Тест 8

**Цель**: Проверить работу программы при запросе печати всех списков

**Исходные данные**:

Те же, что в тесте 6

**Ожидаемый результат:**

СПИСОК ВОКЗАЛОВ:

Центральный

Южный

Северный

СПИСОК АВТОБУСОВ:

Центральный Mercedes

Центральный Hyondai

Центральный Лиаз

Центральный Камаз

Южный Mercedes

Южный Hyondai

Южный Лиаз

Южный Ford

Северный Mercedes

Северный Hyondai

Северный Лиаз

Северный Камаз

Северный Ford

СПИСОК ВОДИТЕЛЕЙ:

Романов П. А.

Томчук Д. М.

Саидбоев В. В.

Мага А. А.

Мухаммадов Б. Б.

Татарян Г. В.

СПИСОК МАРШРУТОВ:

Центральный Mercedes Романов П. А. 234 8:00 30 28 Минск

Центральный Mercedes Томчук Д. М. 235 13:00 30 29 Минск

Южный Лиаз Саидбоев В. В. 678 9:00 20 18 Ростов-на-Дону

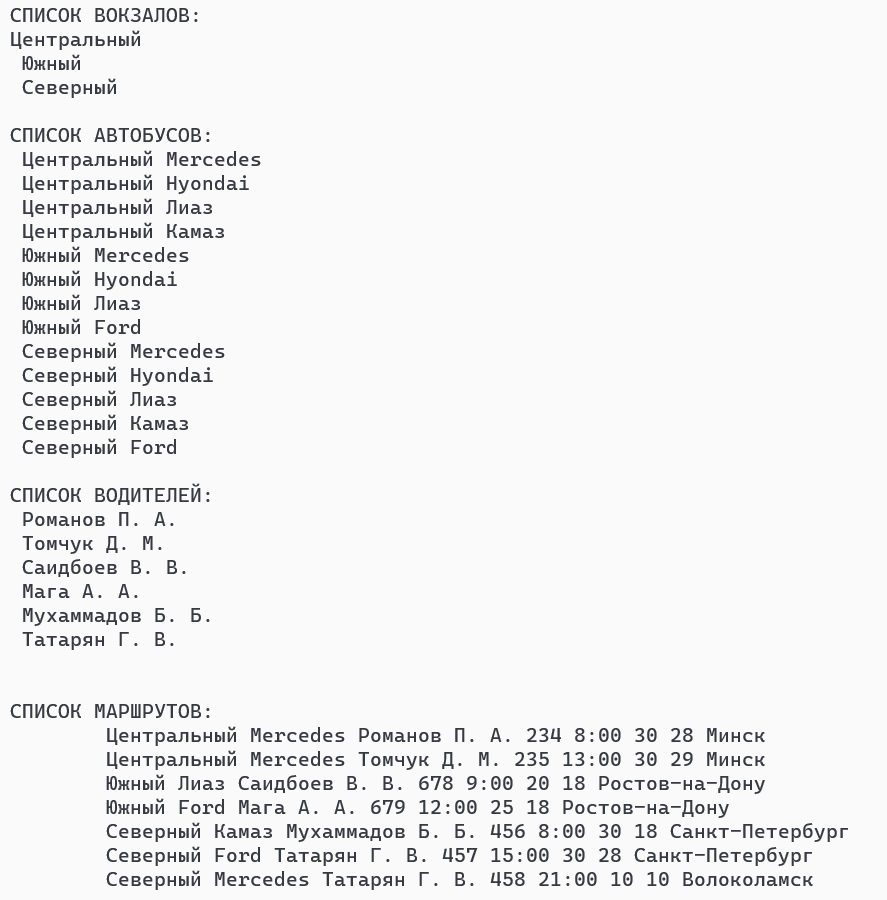
Южный Ford Мага А. А. 679 12:00 25 18 Ростов-на-Дону

Северный Камаз Мухаммадов Б. Б. 456 8:00 30 18 Санкт-Петербург

Северный Ford Татарян Г. В. 457 15:00 30 28 Санкт-Петербург

Северный Mercedes Татарян Г. В. 458 21:00 10 10 Волоколамск

**Полученный результат**:



Тестирование функции прохода по спискам

Тест 9

**Цель**: Проверить работу программы при запросе совершить проход в положительном направлении по кольцевому списку автобусов внутри Центрального вокзала

**Исходные данные**:

Те же, что в тесте 6

Id вокзала = 0

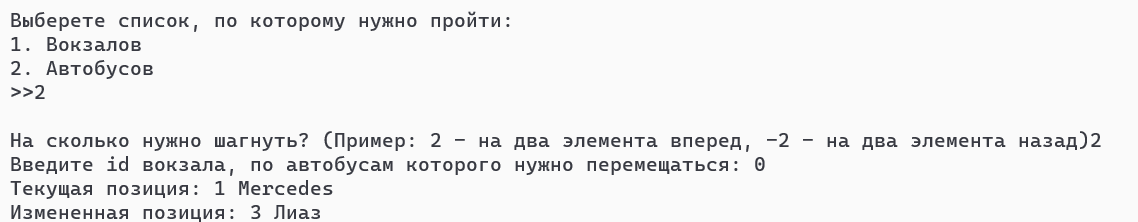
Шагов прохода: 2

**Ожидаемый результат:**

Текущая позиция: 1 Mercedes

Измененная позиция: 3 Лиаз

**Полученный результат:**

****

Тест 10

**Цель**: Проверить работу программы при запросе совершить проход в отрицательном направлении по кольцевому списку автобусов внутри Центрального вокзала, вернувшись тем самым в изначальную позицию в тесте 9.

**Исходные данные**:

Те же, что в тесте 6

Id вокзала = 0

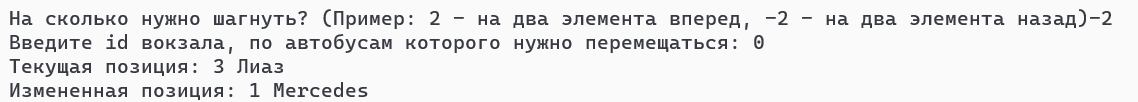
Шагов прохода: -2

**Ожидаемый результат:**

Текущая позиция: 3 Лиаз

Измененная позиция: 1 Mercedes

**Полученный результат:**

****

# Вывод

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