МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

МЕЖИНСТИТУТСКАЯ БАЗОВАЯ КАФЕДРА

**КУРСОВАЯ РАБОТА**

по дисциплине

«Информационные технологии и программирование»

на тему:

«Разработка программы моделирования работы автобусного парка на языке программирования С++»

**Выполнил:**

Дондаев Абу Умар-Пашаевич

студент 1 курса

группы ПИЖ-б-о-23-1(1)

направления подготовки 09.03.04 «Программная инженерия»

направленность (профиль) «Разработка и сопровождение программного обеспечения»

очной формы обучения

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Руководитель работы:

И. В. Мандрица, профессор межинститутской базовой кафедры

Работа допущена к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя) (дата)

Работа выполнена и

защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата защиты

Члены комиссии:

зав. межинститутской

базовой кафедрой Е. Н. Новикова

(подпись)

профессор МИБК И.В. Мандрица

(подпись)

доцент МИБК З.М. Альбекова

(подпись)

Ставрополь, 2024 г.

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Цифрового развития

Кафедра межинститутская базовая

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) «Разработка и сопровождение программного обеспечения»

ЗАДАНИЕ

на курсовую работу

студента Дондаева Абу Умар-Пашаевича

*(фамилия, имя, отчество)*

по дисциплине «Информационные технологии и программирование»

1. Тема работы: Разработка программы моделирования работы автобусного парка на языке программирования С++

2. Цель: повышение уровня профессиональной подготовки путем углубления и закрепления теоретических знаний и практических навыков, приобретенных в результате изучения дисциплины «Информационные технологии и программирование»; подготовка к самостоятельной разработке программного обеспечения с использованием современных информационных технологий.

3. Задачи:

3.1 Анализ прикладной задачи и методов ее решения. Обоснование выбора средств, технологий и алгоритмов решения прикладной задачи.

3.2 Разработка алгоритмов решения задачи.

3.3 Реализация программного кода

3.4 Отладка и тестирование программного кода

4. Перечень подлежащих разработке вопросов:

а) теоретической части: изучение и анализ литературы, постановка условия задачи, выбор и описание методов и библиотек для ее решения и технологий, среды программирования

б) проектная часть: проектирование UML-диаграмм классов, разработка алгоритмов решения задачи, методов классов

в) реализация: описание структуры проекта, описание файлов проекта, разработка программного кода, тестирование программы, создание файла .exe

5. Исходные данные:

а) по литературным источникам: ГОСТы, международные стандарты, программные средства, используемые при разработке программного обеспечения

б) исходные данные, подготовленные для тестирования объекта профессиональной деятельности (информационной системы/приложения/программного продукта)

6. Список рекомендуемой литературы монографии, диссертации, научные статьи, учебно-методические материалы, ссылки на официальные сайты, содержащие информацию необходимую для решения поставленных в работе задач:

1. Липпман С., Лажойе Ж. Язык программирования C++. Полное руководство / Пер. с англ. - 2-е эл. изд. - Саратов: Профобразование, 2019. -1104 с., ил.

2. Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование: Практикум. - СПб .: Питер, 2008. - 265 с .: ил.

3. Павловская Т.А. С/С++. Программирование на языке высокого уровня - СПб .: Питер, 2003. - 461 с .: ил.

7. Контрольные сроки представления отдельных разделов курсовой работы:

25 % - предоставление первого раздела «07» марта 2024 г.

50 % - предоставление второго раздела «15» апреля 2024 г.

75 % - предоставление третьего раздела «29» апреля 2024 г.

100 % - предоставление работы на отзыв «10» мая 2024 г.

8. Срок защиты студентом курсовой работы «23» мая 2024 г.

Дата выдачи задания «14» февраля 2024 г.

Руководитель курсовой работы

Д. эконом. н., профессор И. В. Мандрица

*(ученая степень, звание) (личная подпись) (инициалы, фамилия)*

Задание принял к исполнению студент очной формы обучения 1 курса  
группы ПИЖ-б-о-23-1(1) Дондаев А.У-П.

*(личная подпись) (инициалы, фамилия)*

**Содержание**

[Введение 5](#_Toc167468682)

[1 Теоретическая часть 6](#_Toc167468683)

[1.1 Техническое задание 6](#_Toc167468684)

[1.2 Этапы разработки 7](#_Toc167468685)

[1.3 График выполнения работ 8](#_Toc167468686)

[Вывод 8](#_Toc167468687)

[2 Проектная часть 9](#_Toc167468688)

[2.1 Алгоритм реализации программы 9](#_Toc167468689)

[2.2 Структура программы 10](#_Toc167468690)

[2.3 UML диаграммы 13](#_Toc167468691)

[Вывод 16](#_Toc167468692)

[3 Практическая часть 17](#_Toc167468693)

[3.1 Заголовочный файл «program.h» 17](#_Toc167468694)

[3.2 Основной файл «program.cpp» 26](#_Toc167468695)

[3.3 Интерфейс программы 28](#_Toc167468696)

[3.4 Тестирование 29](#_Toc167468697)

[3.5 Руководство пользователя программы 33](#_Toc167468698)

[3.6 Ссылка на репозиторий Git Hub 34](#_Toc167468699)

[Вывод 34](#_Toc167468700)

[Заключение 35](#_Toc167468701)

[Список использованных источников 36](#_Toc167468702)

# Введение

Цели: повышение уровня профессиональной подготовки путем углубления и закрепления теоретических знаний и практических навыков, приобретенных в результате изучения дисциплины «Информационные технологии и программирование»; подготовка к самостоятельной разработке программного обеспечения с использованием современных информационных технологий.

Задачи:

- анализ прикладной задачи и методов ее решения. Обоснование выбора средств, технологий и алгоритмов решения прикладной задачи.

- разработка алгоритмов решения задачи.

- реализация программного кода

- отладка и тестирование программного кода

Перечень подлежащих разработке вопросов:

а) теоретической части: изучение и анализ литературы, постановка условия задачи, выбор и описание методов и библиотек для ее решения и технологий, среды программирования;

б) проектной части: проектирование UML-диаграмм классов, разработка алгоритмов решения задачи, методов классов;

в) реализации: описание структуры проекта, описание файлов проекта, разработка программного кода, тестирование программы, создание файла .exe.

# 1 Теоретическая часть

## 1.1 Техническое задание

Название: программа моделирования работы автобусного парка.

Требования к функциональным характеристикам программы:

Сведения о каждом автобусе содержат: номер автобуса, фамилию и инициалы водителя, номер маршрута. Программа должна обеспечивать выбор с помощью меню выполнение одной из следующих функций:

- начальное формирование данных о всех автобусах в парке в виде списка (ввод с клавиатуры или из файла);

- имитация выезда автобуса из парка: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся в парке, и записывает эти данные в список автобусов, находящихся на маршруте;

− имитация въезда автобуса в парк: вводится номер автобуса; программа удаляет данные об этом автобусе из списка автобусов, находящихся на маршруте, и записывает эти данные в список автобусов, находящихся в парке;

− вывод сведений об автобусах, находящихся в парке, и об автобусах, находящихся на маршруте.

Для представления необходимых списков использовать контейнерный класс 1ist.

Требования к программным характеристикам программы:

Для функционирования программы необходимо наличие на запускающем устройстве компилятора C++.

## 1.2 Этапы разработки

1. Составление алгоритма решения поставленной задачи.
2. Рассмотрение возможных способов реализации методов, указанных в требованиях к функциональным характеристикам программы.
3. Создание класса Bus (Автобус), хранящего данные о номере автобуса, фамилии и инициалах водителя автобуса и номере маршрута. Реализация функции вывода данных об объекте данного класса. Реализация функций, возвращающих значения атрибутов класса, находящихся в приватном доступе.
4. Создание класса Park (Автобусный парк), хранящего списки автобусов, находящихся в парке или на маршруте. Реализация всех функций, указанных в требованиях к функциональным характеристикам программы.
5. Реализация меню в основном файле программы.
6. Проверка работоспособности программы.
7. Тестирование программы с целью выявить возможные ошибки и исправить их. Проверка работоспособности каждой функции файла классов.

Для разработки программы использовался текстовый редактор кода Visual Studio Code. Расширения, использованные в написании кода: C/C++, C/C++ Extension Pack, C/C++ Runner.

Выбор текстового редактора Visual Studio Code заключается в простоте его использования, в возможности подключения необходимых расширений и разработке программного продукта на разных языках программирования, в кроссплатформенности и в опыте пользования данной технологией, приобретенного в результате изучения дисциплины «Информационные технологии и программирование».

## 1.3 График выполнения работ

Контрольные сроки представления отдельных разделов курсовой работы представлены в таблице 1.3.1.

|  |  |
| --- | --- |
| Наименование этапа работы | Срок выполнения |
| Завершение теоретической части | 14 февраля – 07 марта |
| Завершение проектной части | 08 марта – 15 апреля |
| Завершение практической части | 16 апреля – 29 апреля |
| Доработка семестровой курсовой работой в соответствии с замечаниями | 29 апреля – 10 мая |

Рисунок 1.3.1 - Этапы работы семестровой курсовой работы

## Вывод

Итогом курсовой работы должен быть готовый и реализованный в соответствии с техническим заданием продукт. Реализация программы должна основываться на ООП.

Программа должна автоматизировать учет автобусов в автобусном парке, позволяя сохранять списки автобусов, находящихся в парке или на маршруте, а также предоставлять ее пользователям доступ к информации об автобусах и маршрутах. Целевой аудиторией программы являются водители автобусов и потенциальные пассажиры.

# 2 Проектная часть

## 2.1 Алгоритм реализации программы

Алгоритм реализации программы будет представлен в виде общей блок-схемы программы, представленной на рисунке 2.1.1.

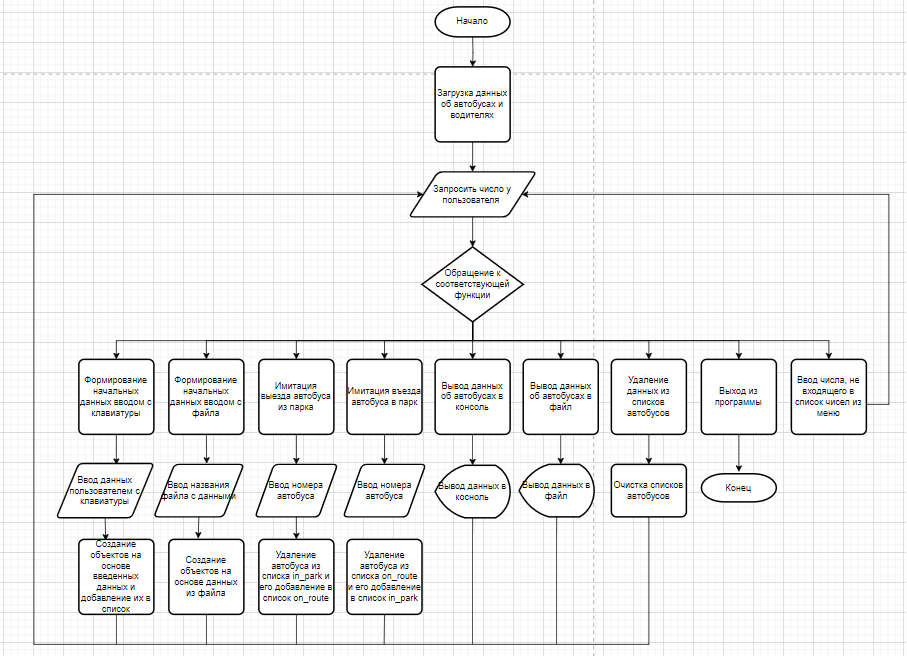


Рисунок 2.1.1 - Общая блок-схема программы

## 2.2 Структура программы

Структура программы представлена в виде интеллектуальной карты, иллюстрирующей, на какие элементы делится программа, и поясняющей способ реализации методов классов. См. рисунки 2.2.1 – 2.2.3.

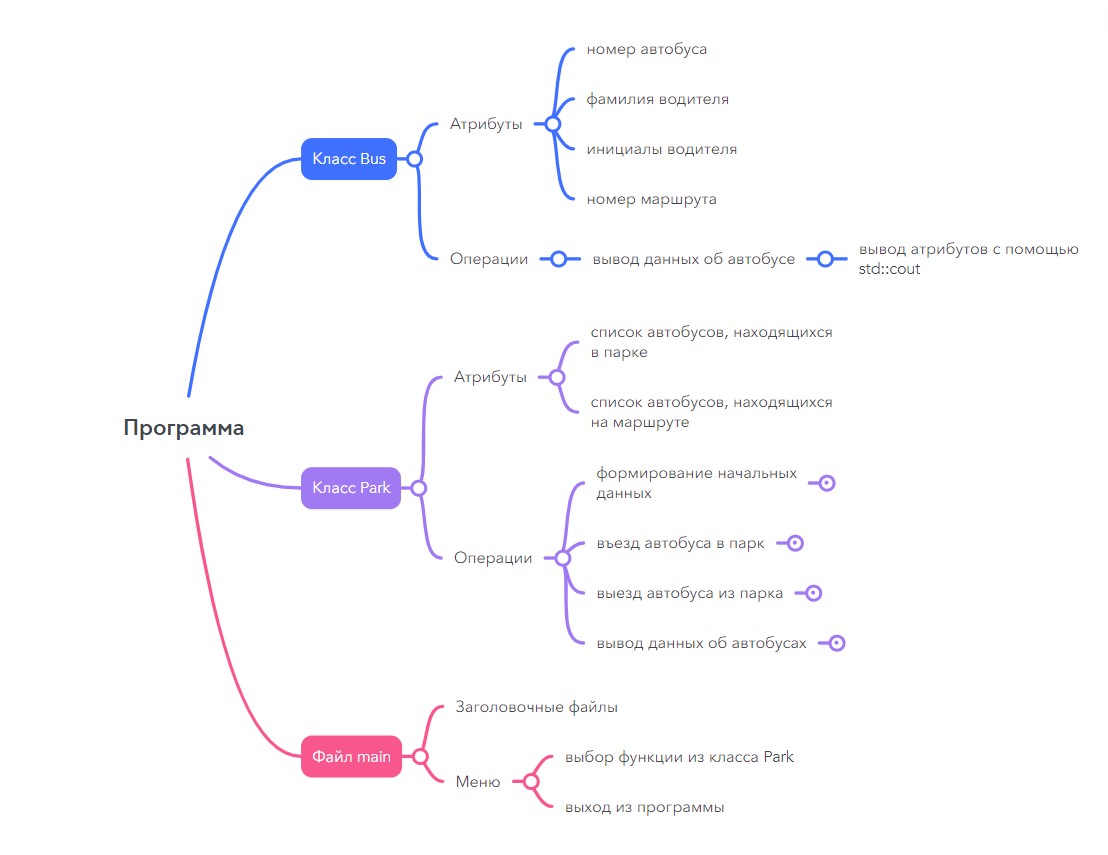


Рисунок 2.2.1 - Интеллект-карта по структуре программы

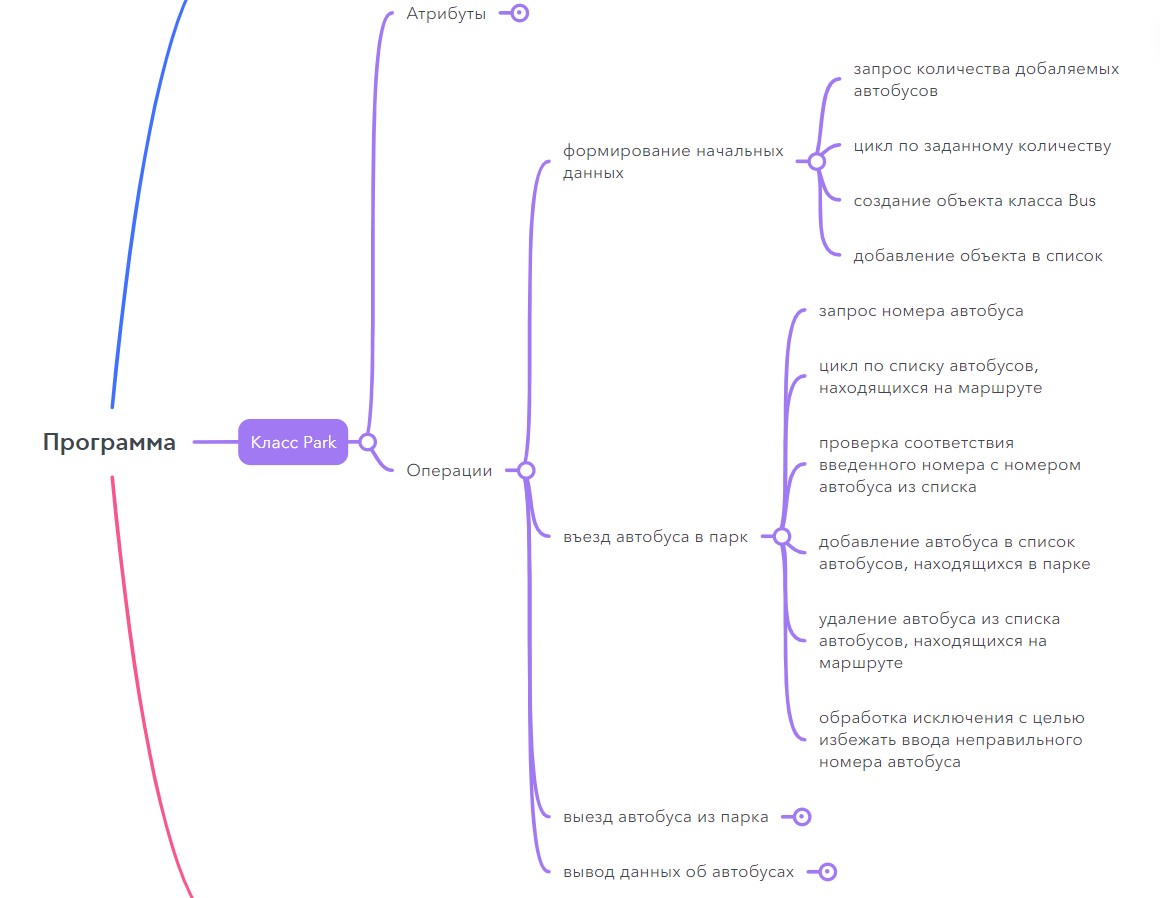


Рисунок 2.2.2 - Алгоритм функций формирования начальных данных и въезда автобуса в парк

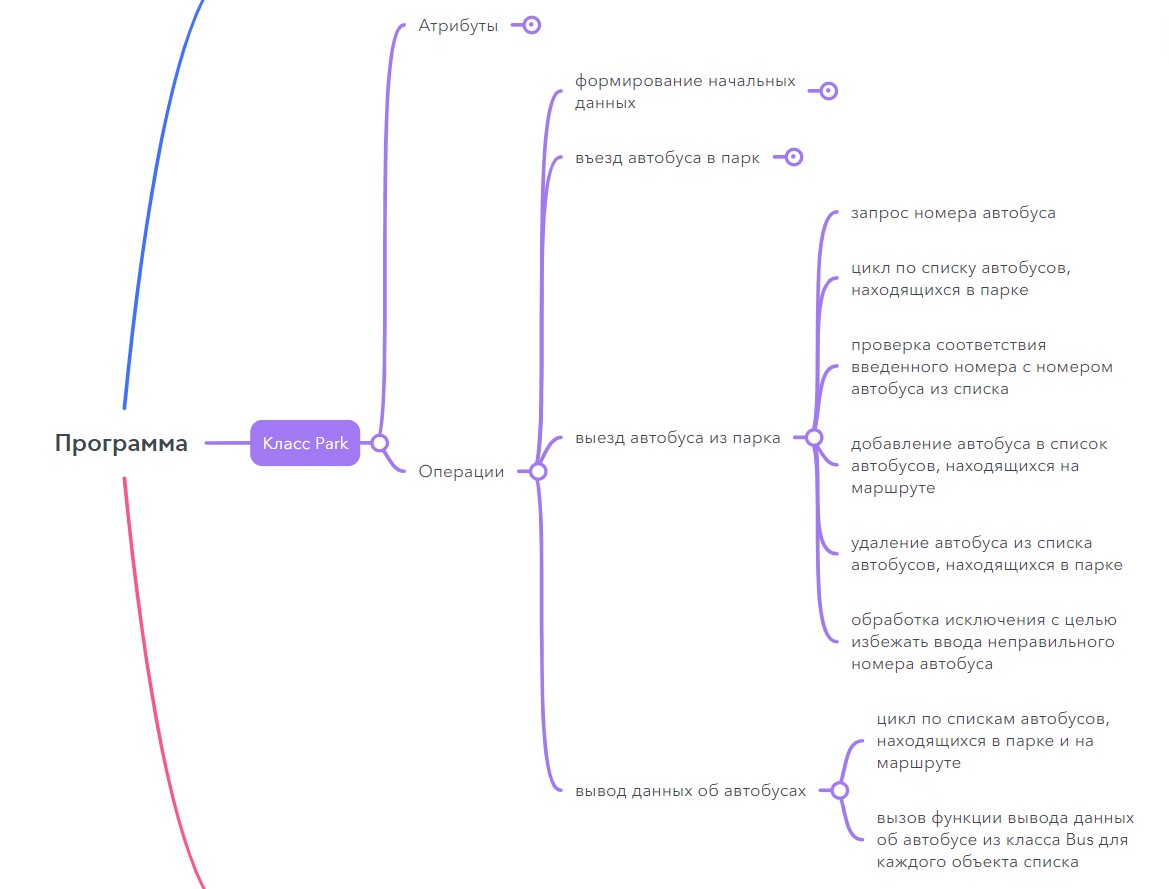


Рисунок 2.2.3 - Алгоритм функций выезда автобуса из парка и вывода данных об автобусах

Программа состоит из трёх файлов: основного файла программы (файл функции main) – program.cpp, заголовочного файла, содержащего реализацию классов Bus и Park – program.h, и файла формата .exe – program.exe. Для формирования начальных данных и вывода конечных данных используются текстовые документы data.txt и information.txt соответственно.

Основной файл программы содержит в себе меню программы. В нем в качестве заголовочного файла подключен файл «program.h».

Заголовочный файл классов содержит в себе инициализацию классов Bus и Park, их атрибутов и операций, а также реализацию операций.

Пользователя после запуска программы встречает меню, предлагающее ему ввести число – номер функции, которой он хочет воспользоваться. После ввода числа запускается функция под выбранным номером и просит ввести пользователя необходимую информацию. Результат функции выводится на экран, после чего повторно выводится меню. Предусмотрена возможность выхода из программы вводом числа «8».

## 2.3 UML диаграммы

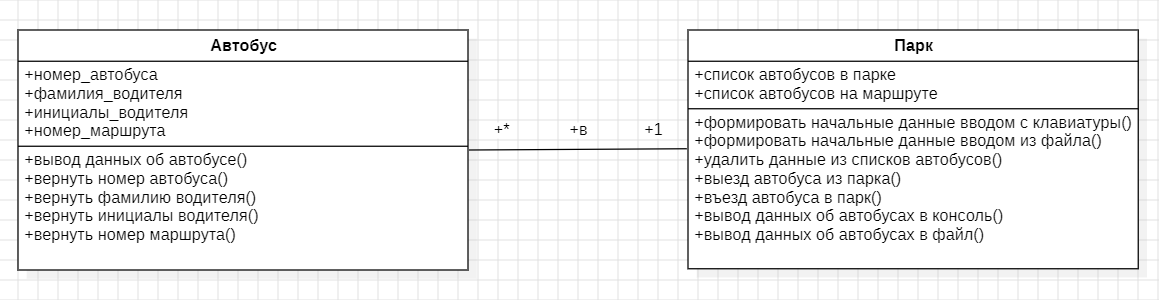


Рисунок 2.3.1 - Диаграмма классов программы

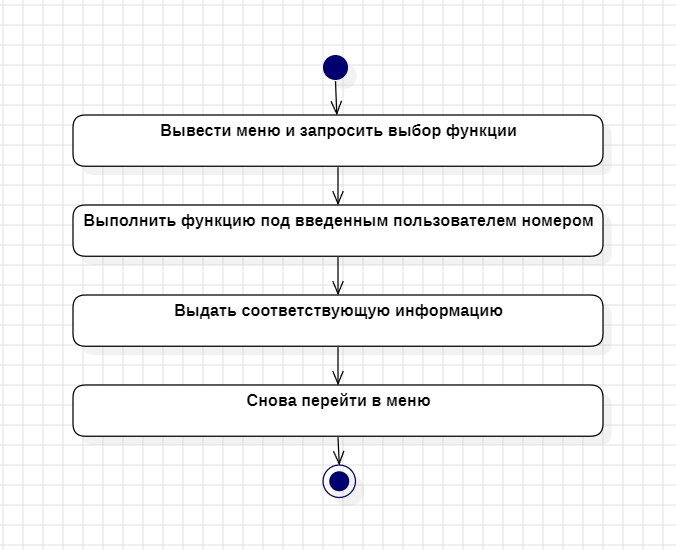


Рисунок 2.3.2 - Диаграмма активности программы

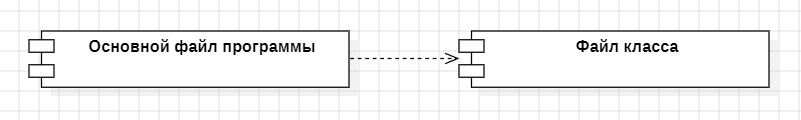


Рисунок 2.3.3 - Диаграмма компонентов программы

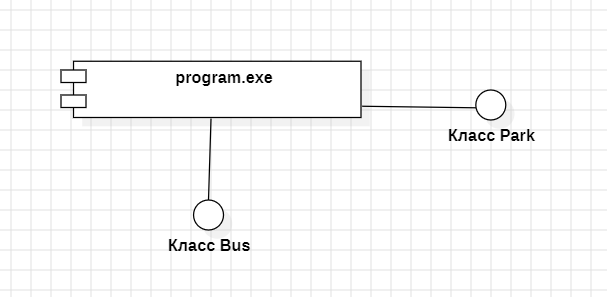


Рисунок 2.3.4 - Диаграмма интерфейсов программы



Рисунок 2.3.5 - Диаграмма последовательности программы

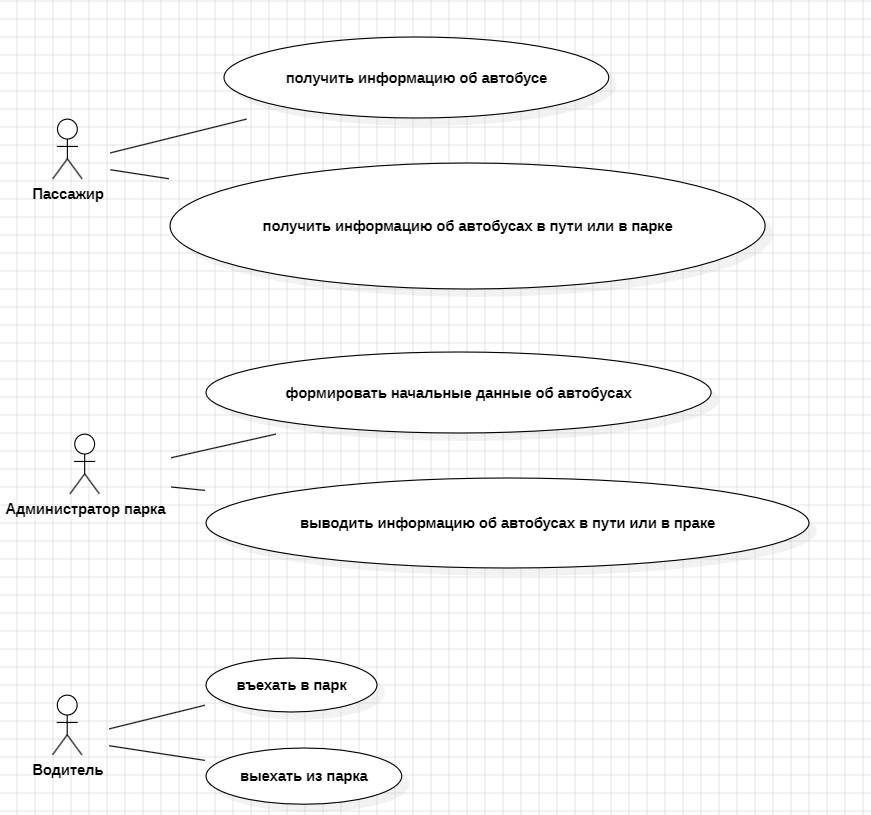


Рисунок 2.3.6 - Диаграмма вариантов использования программы

## Вывод

В этом разделе расписаны алгоритмы реализации методов классов программы, ее общая блок-схема, составлены UML-диаграммы. Программа состоит из классов Bus и Park, что позволяет грамотно управлять функционалом программы и дополнять ее по мере необходимости. Все перечисленное позволяет четко увидеть, как программа будет реализована в практической части.

# 3 Практическая часть

## 3.1 Заголовочный файл «program.h»

Заголовочный файл «program.h» состоит из реализации двух классов: Bus и Park.

В классе Bus в приватном доступе инициализированы переменные: bus\_number, surname, initials, route\_number; в публичном доступе: функции display(), get\_bus\_number(), get\_surname(), get\_initials(), get\_route\_number(). Также инициализирован конструктор класса Bus() по умолчанию и конструктор класса, задающий параметры атрибутам класса.

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип переменной | Значение переменной |
| bus\_number | int | Хранит номер автобуса |
| surname | string | Хранит фамилию водителя |
| initials | string | Хранит инициалы водителя |
| route\_number | int | Хранит номер маршрута |

Таблица 3.1.1 - Атрибуты класса Bus

|  |  |  |
| --- | --- | --- |
| Метод | Тип возвращаемого значения | Функционал |
| display() | void | Выводит данные об автобусе в консоль |
| get\_bus\_number() | int | Возвращает номер автобуса |
| get\_surname() | string | Возвращает фамилию водителя |
| get\_initials() | string | Возвращает инициалы водителя |
| get\_route\_number() | int | Возвращает номер маршрута |

Таблица 3.1.2 - Методы класса Bus

В классе Park в приватном доступе расположено два пустых списка in\_park и on\_route. Они хранят в себе данные об автобусах, находящихся в парке и на маршруте соответственно. В приватном доступе реализованы функции: make\_start\_data(), make\_start\_data\_from\_file(), delete\_data(), departure(), entry(), display\_data(), display\_data\_in\_file().

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип переменной | Значение переменной |
| in\_park | list<Bus> | Список автобусов, находящихся в парке |
| on\_route | list<Bus> | Список автобусов, находящихся на маршруте |

Таблица 3.1.3 - Атрибуты класса Park

|  |  |  |
| --- | --- | --- |
| Метод | Тип возвращаемого значения | Функционал |
| make\_start\_data() | void | Формирует начальные данные об автобусах через ввод данных с клавиатуры |
| make\_start\_data\_from\_file() | void | Формирует начальные данные об автобусах через ввод данных с файла |
| delete\_data() | void | Удаляет данные из списков автобусов |
| departure() | void | Имитирует выезд автобуса из парка |
| entry() | void | Имитирует въезд автобуса в парк |
| display\_data() | void | Выводит данные об автобусах в консоль |
| display\_data\_in\_file() | void | Выводит данные об автобусах в файл |

Таблица 3.1.4 - Методы класса Park

#pragma once

#include <iostream>

#include <string>

#include <list>

#include <fstream>

//инициализация класса Bus

class Bus {

private:

std::string surname; //переменная, хранящая фамилию водителя

std::string initials; //переменная, хранящая инициалы водителя

int route\_number; //переменная, хранящая номер маршрута

int bus\_number; //переменная, хранящая номер автобуса

public:

Bus(); //конструктор класса по-умолчанию

Bus(int b\_num, std::string s\_name, std::string init, int r\_num){ //конструктор класса

bus\_number = b\_num;

surname = s\_name;

initials = init;

route\_number = r\_num;

}

void display(){ //функция вывода данных об автобусе

std::cout << "Номер автобуса: " << bus\_number << "; инициалы водителя: " << surname << " " << initials << "; номер маршрута: " << route\_number << "." << std::endl;

}

int get\_bus\_number(){ //функция возвращает номер автобуса

return bus\_number;

}

std::string get\_surname(){ //функция возвращает фамилию водителя

return surname;

}

std::string get\_initials(){ //функция возвращает инициалы водителя

return initials;

}

int get\_route\_number(){ //функция возвращает номер маршрута

return route\_number;

}

};

//инициализация класса Park

class Park {

private:

std::list<Bus> in\_park; //пустой список автобусов, находящихся в парке

std::list<Bus> on\_route; //пустой список автобусов, находящихся на маршруте

public:

void make\_start\_data(){ //функция формирования начальных данных вводом с клавиатуры

int num; //количество автобусов, которое необходимо добавить в список

std::cout << "Введите количество автобусов, данные которых необходимо добавить: ";

std::cin >> num;

std::cout << "Вводите данные по следующему образцу:\nНомер\_автобуса Фамилия\_водителя Инициалы\_водителя Номер\_маршрута\nПример: 1 Иванов И.И. 1\n" << std::endl;

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

int b\_num, r\_num;

std::string s\_name, init;

std::cin >> b\_num >> s\_name >> init >> r\_num;

Bus object(b\_num, s\_name, init, r\_num); //создание объекта класса Bus

in\_park.push\_back(object); //добавление объекта в список автобусов, находящихся в парке

}

std::cout << "Данные приняты." << std::endl;

}

void make\_start\_data\_from\_file(){ //функция формирования начальных данных вводом из файла

std::string name\_of\_file; //переменная-название файла

std::cout << "Расположите текстовый документ в папку с файлами программы.\nВажно: информация в текстовом документе должна быть представлена в следуюем виде: Номер\_автобуса Фамилия\_водителя Инициалы\_водителя Номер\_Маршрута. Например: 1 Иванов И.И. 48.\nВведите название файла с добавлением .txt в конец названия файла: " << std::endl;

std::cin >> name\_of\_file;

std::ifstream file(name\_of\_file.c\_str()); //открыть файл

if (file.is\_open()){

int b\_num, r\_num;

std::string s\_name, init;

while (file >> b\_num >> s\_name >> init >> r\_num){

Bus object(b\_num, s\_name, init, r\_num); //инициализация объекта класса Bus

in\_park.push\_back(object); //добавление объекта в массив

}

file.close(); //закрытие файла

std::cout << "Данные приняты." << std::endl;

}

else{

std::cout<<"Файл невозможно открыть. Возможно, вы ввели некорректное имя файла!" << std::endl; //обработка исключения

}

}

void delete\_data(){ //функция очистки массивов

in\_park.clear(); //удалить данные из массива in\_park

on\_route.clear(); //удалить данные из массива on\_route

std::cout << "Данные из массивов удалены!" << std::endl;

}

void departure(){ //функция имитации выезда автобуса из парка

int num, count = 0; //num - номер автобуса, выезжающего из парка, count - переменная для проверки наличия автобуса с заданным номером в списке

std::cout << "Введите номер автобуса, который выезжает из парка: ";

std::cin >> num;

auto iter = in\_park.begin(); //итератор на начало списка

for (Bus obj : in\_park){

if (obj.get\_bus\_number() == num){ //проверка соответствия номеров

count++;

on\_route.push\_back(obj); //добавление автобуса в список автобусов, находящихся на маршруте

in\_park.erase(iter); //удаление автобуса из списка автобусов, находящихся в парке

std::cout << "Автобус номер " << num << " выехал из парка." << std::endl;

break;

}

++iter; //смещение итератора

}

if (count == 0){ //вывод на случай, если автобуса с заданным номером нет в списке; обработка исключения

std::cout << "Автобуса с номером " << num << " нет в парке!" << std::endl;

}

}

void entry(){ //функция имитации въезда автобуса в парк

int num, count = 0; //num - номер автобуса, въезжающего в парк, count - переменная для проверки наличия автобуса с заданным номером в списке

std::cout << "Введите номер автобуса, который въезжает в парк: ";

std::cin >> num;

auto iter = on\_route.begin(); //итератор на начало списка

for (Bus obj : on\_route){

if (obj.get\_bus\_number() == num){ //проверка соответствия номеров

count++;

in\_park.push\_back(obj); //добавление автобуса в список автобусов, находящихся в парке

on\_route.erase(iter); //удаление автобуса из списка автобусов, находящихся на маршруте

std::cout << "Автобус номер " << num << " въехал в парк." << std::endl;

break;

}

++iter; //смещение итератора

}

if (count == 0){ //вывод на случай, если автобуса с заданным номером нет в списке; обработка исключения

std::cout << "Автобуса с номером " << num << " нет в списке автобусов, находящихся на маршруте!" << std::endl;

}

}

void display\_data(){ //вывод информации об автобусах

std::cout << "Сведения об автобусах, находящихся в парке: " << std::endl;

int count = 1;

if (in\_park.empty()) std::cout << "Список пуст!" << std::endl; //переменная для нумерации

for (Bus obj : in\_park){

std::cout << count << ". ";

obj.display(); //вывод данных об автобусе из списка

++count;

}

count = 1; //обновление переменной нумерации

std::cout << std::endl << "Сведения об автобусах, находящихся на маршруте: " << std::endl;

if (on\_route.empty()) std::cout << "Список пуст!" << std::endl;

for (Bus obj : on\_route){

std::cout << count << ". ";

obj.display(); //вывод данных об автобусе из списка

++count;

}

}

void display\_data\_in\_file(){ //вывод данных в файл

std::ofstream file("information.txt"); //инициализация файла для вывода данных

if (file.is\_open()){

file << "Сведения об автобусах, находящихся в парке: " << std::endl;

int count = 1; //переменная для нумерации

if (in\_park.empty()) file << "Список пуст!" << std::endl;

for (Bus obj : in\_park){

file << count << ". ";

file << "Номер автобуса: " << obj.get\_bus\_number() << "; инициалы водителя: " << obj.get\_surname() << " " << obj.get\_initials() << "; номер маршрута: " << obj.get\_route\_number() << "." << std::endl; //вывод данных об автобусе из списка

++count;

}

count = 1; //обновление переменной нумерации

file << std::endl << "Сведения об автобусах, находящихся на маршруте: " << std::endl;

if (on\_route.empty()) file << "Список пуст!" << std::endl;

for (Bus obj : on\_route){

file << count << ". ";

file << "Номер автобуса: " << obj.get\_bus\_number() << "; инициалы водителя: " << obj.get\_surname() << " " << obj.get\_initials() << "; номер маршрута: " << obj.get\_route\_number() << "." << std::endl; //вывод данных об автобусе из списка

++count;

}

}

else{

std::cout << "Невозможно создать файл! Попробуйте снова." << std::endl; //обработка исключения

}

}

};

## 3.2 Основной файл «program.cpp»

Основной файл программы содержит подключение всех необходимых для работоспособности программы заголовочных файлов. В их числе: iostream, program.h.

В функции main реализовано меню, выводящее на экран пользователя список методов. Меню состоит из переменной choice, предназначенной для хранения введенного пользователем числа, обозначающего выбранную им функцию, цикла do-while, обеспечивающего повторный вывод меню после использования одной из функций, и оператора ветвления switch-case, вложенного в цикл, позволяющего по вводу номера функции воспользоваться ей. Выход из программы обеспечивается вводом числа «8».

#include <iostream>

#include "program.h"

using namespace std;

int main(){

setlocale(LC\_ALL, "Russian");

Park park; //инициализация объекта класса Park

//Меню

int choice; //переменная для выбора функции

do {

cout << "\nМеню\nВыберите функцию\n";

cout << "1. Сформировать начальные данные вводом с клавиатуры\n";

cout << "2. Сформировать начальные данные вводом с файла\n";

cout << "3. Выезд автобуса из парка\n";

cout << "4. Въезд автобуса в парк\n";

cout << "5. Вывод информации об автобусах в консоль\n";

cout << "6. Вывод информации об автобусах в файл\n";

cout << "7. Удалить данные из массивов\n";

cout << "8. Выход из программы\n";

cout << "Введите ваш выбор: ";

cin >> choice;

cout << endl;

switch (choice) {

case 1:

park.make\_start\_data();

break;

case 2:

park.make\_start\_data\_from\_file();

break;

case 3:

park.departure();

break;

case 4:

park.entry();

break;

case 5:

park.display\_data();

break;

case 6:

park.display\_data\_in\_file();

break;

case 7:

park.delete\_data();

break;

case 8:

cout << "Выход из программы...\n";

break;

default:

cout << "Неверный выбор. Пожалуйста, введите существующий вариант.\n";

}

} while (choice != 8);

return 0;

}

## 3.3 Интерфейс программы

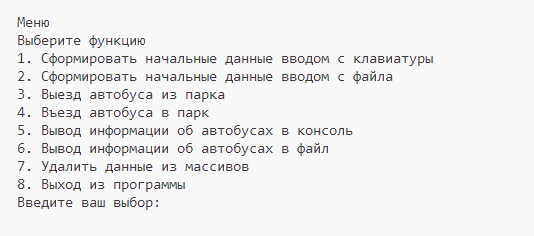


Рисунок 3.3.1 - Меню программы

На фото представлено меню, выводящееся при запуске программы. Непосредственно с ним пользователю предстоит взаимодействовать с программой.

## 3.4 Тестирование

Тестирование работоспособности программы будет проводиться в виде проверки корректности реализации методов, представленных в меню.

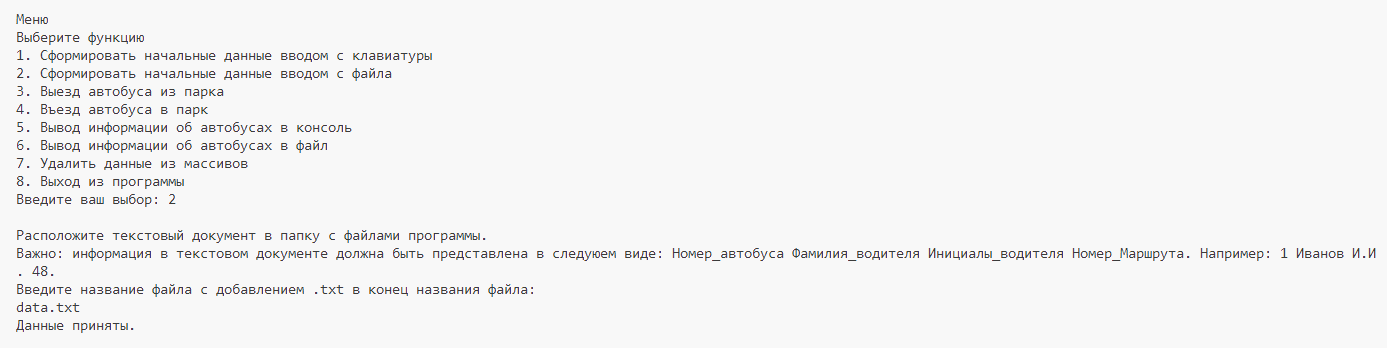


Рисунок 3.4.1 - Тест функции формирования начальных данных вводом из файла data.txt

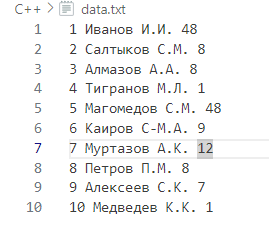


Рисунок 3.4.2 - Данные файла data.txt

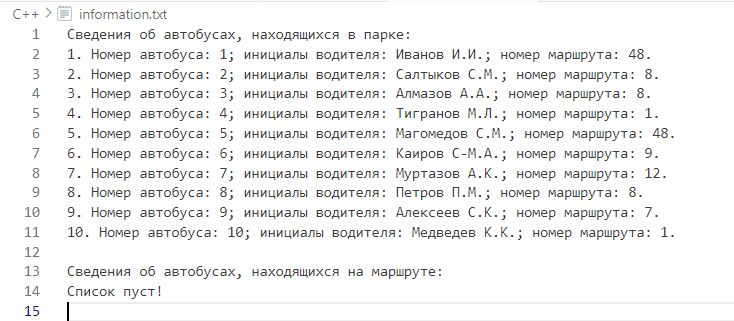


Рисунок 3.4.3 - Тест функции вывода данных в файл и доказательство корректности работы функции формирования начальных данных

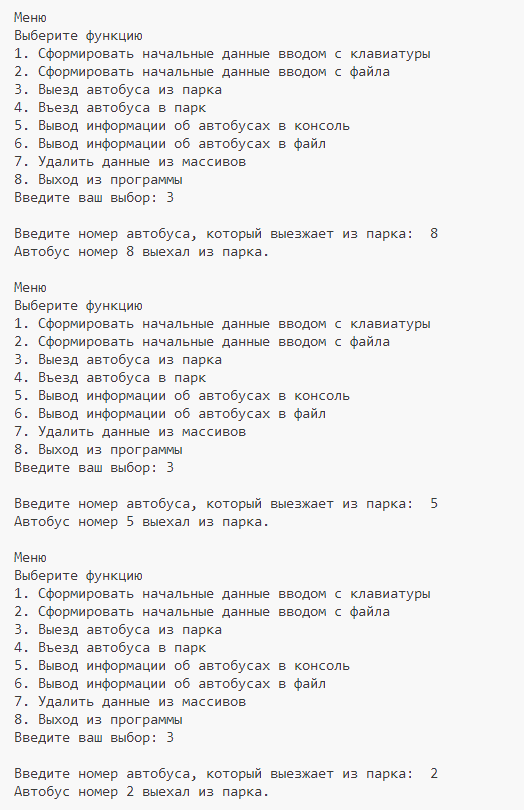


Рисунок 3.4.4 - Тест функции имитации выезда автобуса из парка

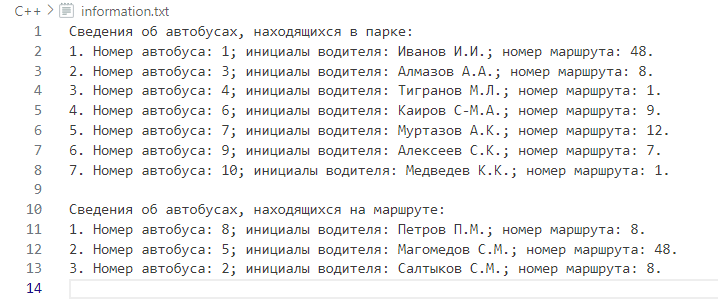


Рисунок 3.4.5 - Тест корректности работы функции выезда автобуса из парка

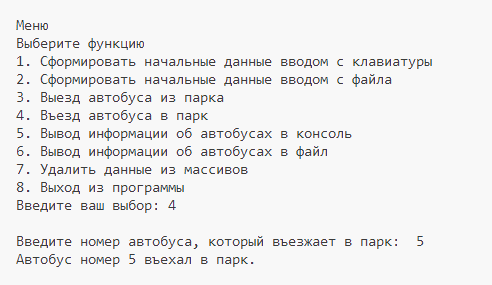


Рисунок 3.4.6 - Тест функции имитации въезда автобуса в парк

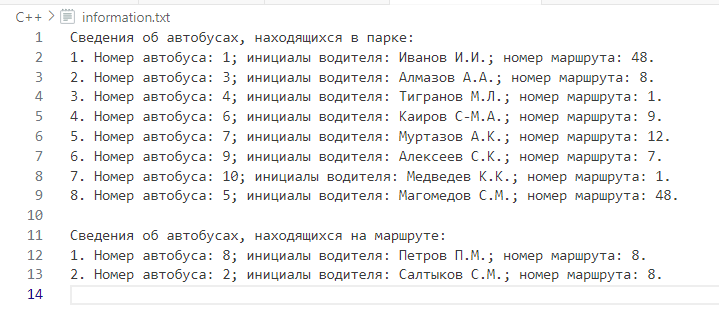


Рисунок 3.4.7 - Тест корректности работы функции въезда автобуса в парк

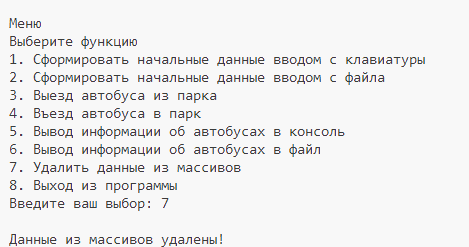


Рисунок 3.4.8 - Тест функции удаления данных из списков автобусов

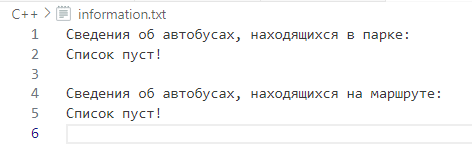


Рисунок 3.4.9 - Тест корректности работы функции удаления данных из списков автобусов

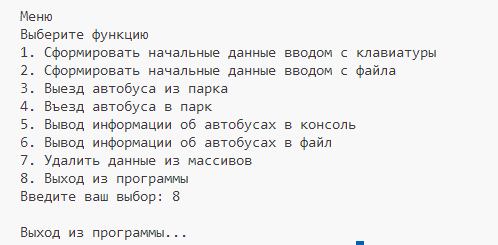


Рисунок 3.4.10 - Тест выхода из программы

Тестирование показало, что каждая функция работает корректно, то есть выдает правильные результаты на основе введенных пользователем данных. Это позволяет с уверенностью заявить о работоспособности и эффективности данной программы.

## 3.5 Руководство пользователя программы

После запуска программы на экран выводится меню, содержащее список всех доступных программе функций. Пользователю необходимо ввести номер функции, которой он хочет воспользоваться.

ВАЖНО! Необходимо при запуске программы в первую очередь формировать начальные данные, так как по умолчанию они не заданы. То есть необходимо в первую очередь воспользоваться функцией под номером «1» или «2».

Функция имитации выезда из парка (в списке под номером «3») и въезда в парк (в списке под номером «4») автобуса реализованы логически одинаково, поэтому достаточно ввести номер автобуса в консоль.

Функция удаления данных из списков автобусов (в списке под номером «7») не требует ввода дополнительной информации, она очищает двусвязные списки, хранящие данные об автобусах.

Функции вывода данных в консоль (в списке под номером «5») или в файл (в списке под номером «6») так же не требуют ввода дополнительной информации, они выводят информацию об автобусах в консоль программы или в файл information.txt соответственно.

Выход из программы осуществляется вводом в консоль числа «8».

## 3.6 Ссылка на репозиторий Git Hub

<https://github.com/Abu9541/LaboratoryRab_ITP/tree/main/%D0%9A%D1%83%D1%80%D1%81%D0%BE%D0%B2%D0%B0%D1%8F>

## Вывод

В этом разделе проведена полная реализация программы.

Программа оснащена понятным интерфейсом. Все задачи, объявленные в теоретической части, решены. Программа соответствует данным проектной части. Проведено тестирование всех функций программы.

На основании вышеперечисленного программа моделирования работы автобусного парка готова к эксплуатации.

# Заключение

Во время выполнения данной курсовой работы получен опыт работы с классами и двусвязными списками на языке программирования C++, закреплены знания, полученные при прохождении курса «Информационные технологии и программирование» и приобретены умения в сфере создания программ сложной структуры.

Цели и задачи, объявленные во введении, выполнены в полном объеме.

Итогом работы над курсовой работой является программный продукт, полностью соответствующий требованиям, выдвинутым в исходном техническом задании.

# Список использованных источников

1. Липпман С., Лажойе Ж. Язык программирования C++. Полное руководство / Пер. с англ. - 2-е эл. изд. - Саратов: Профобразование, 2019. -1104 с., ил.

2. Павловская Т. А., Щупак Ю. А. С++. Объектно-ориентированное программирование: Практикум. - СПб .: Питер, 2008. - 265 с .: ил.

3. Павловская Т.А. С/С++. Программирование на языке высокого уровня - СПб .: Питер, 2003. - 461 с .: ил.