Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машинных

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

Программа «Расписание городского транспорта»

БГУИР КП 1-40 02 01 325 ПЗ

Студент: гр. 950503 Сякачёв П.В.

Руководитель: старший преподаватель кафедры ЭВМ Ковальчук А.М.

Минск 2020

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc60823124)

[**1** **ОБЗОР ИСТОЧНИКОВ** 5](#_Toc60823125)

[**1.1** **Анализ аналогов программного средства** 5](#_Toc60823126)

[**1.2** **Постановка задачи** 5](#_Toc60823127)

[**2.** **СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ** 7](#_Toc60823128)

[**4.** **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ** 9](#_Toc60823129)

[**5.** **РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ** 29](#_Toc60823130)

[**5.1** **Схема алгоритма функции void delstops (int type) класса Ground\_Trans** 29](#_Toc60823131)

[**5.2** **Схема алгоритма функции void sort() класса Ring** 29](#_Toc60823132)

[**5.3** **Алгоритм по шагам функции vector<Time> fromtotime (bool a, bool b, int c) класса Ground\_Trans** 29](#_Toc60823133)

[**5.4** **Алгоритм по шагам функции SearchStop(T& st)** 31](#_Toc60823134)

[**5.5** **Код программы** 31](#_Toc60823135)

[**6.** **ТЕСТИРОВАНИЕ** 32](#_Toc60823136)

[**ЗАКЛЮЧЕНИЕ** 36](#_Toc60823137)

[**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ** 37](#_Toc60823138)

[**ПРИЛОЖЕНИЕ А** 38](#_Toc60823139)

[**ПРИЛОЖЕНИЕ Б** 39](#_Toc60823140)

[**ПРИЛОЖЕНИЕ В** 40](#_Toc60823141)

[**ПРИЛОЖЕНИЕ Г** 41](#_Toc60823142)

[**ПРИЛОЖЕНИЕ Д** 42](#_Toc60823143)

[**ПРИЛОЖЕНИЕ Е** 43](#_Toc60823144)

# **ВВЕДЕНИЕ**

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является всё ещё одним из самых популярных и распространенных языков.

Этот язык поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщенное программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности.

Причины популярности С++ следующие:

1. С++ – это универсальный язык программирования. Запустить приложение, написанное на этом языке, можно на практически любой вычислительной технике, также, как и написать приложение. На этом языке были написаны многие игры, драйвера, даже операционные системы.
2. С++ – это развивающийся и активно поддерживающийся язык программирования. Каждые три года обновляются стандарты этого языке (например, в сентябре 2020 года был утверждён стандарт С++20). Также, сейчас существует большое количество программного обеспечения облегчающее написание кода и большая база знаний, так многие программисты делятся своим кодом.
3. С++ – это один из базовых языков программирования. Этот язык произошёл от С, а Java, JavaScript, C# появились под влиянием С++. С++ хорош как первый язык программирования, который позволяет понять всю необходимую логику и базу.

На сегодняшний день технологии достаточно глубоко внедрились в нашу жизнь, пытаясь как можно больше упростить её и улучшить. Было и будет создано много программ, которые позволяют упростить получаемую нами информацию. На сегодняшний день существует множество приложений-ежедневников, приложений с расписаниями занятий, графиками работы, расписаниями городского и междугороднего транспорта и др. Все они упрощают нашу жизнь, позволяя не держать большое количество информации в голове, вместо чего мы просто можем заглянуть в приложение и узнать нужную нам информацию.

Таким образом, написание программы расписания городского транспорта на языке программирования С++ – хорошая возможность для создания своего приложения, которое со временем можно модифицировать, и применения навыков, полученных в ходе изучения курса Конструирование программ и языки программирования (КПиЯП).

1. **ОБЗОР ИСТОЧНИКОВ**
   1. **Анализ аналогов программного средства**

На данный момент существует множество примеров приложений с расписанием различных типов транспорта, в том числе и городского. С развитием технологий появилась возможность переносить расписания с табличек и записей в цифровое пространство. Со временем у подобного рода приложений появилась графическая оболочка, блоки анализа, позволяющие выполнять различный функционал, от простого просмотра расписаний и расчёта времени до прокладывания маршрута и т.д. Появились приложения представляющее из себя карту, на которой можно прокладывать кратчайшие маршруты на разных видах транспорта из одного места в другое. А многие из этих приложений бесплатные.

В подобного рода приложениях есть возможности менять внешний вид приложения, менять настройки, добавлять удобные маршруты.

В разрабатываемом приложении не будет особенных функций. Оно будет имеет вид консольного приложения и позволит узнать расписание и маршруты разных видов городского транспорта.

* 1. **Постановка задачи**

Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Информация должна хранится в различных файлах, связанных определённым образом. Например, в файлах содержится информация о маршрутах: откуда – куда, по каким дням. Реализовать функции выдачи информации о нужном рейсе, добавления новой информации. Другие методы, в зависимости от специфики задачи.

При реализации операции редактирования, добавления, удаления информации, необходимо предусмотреть операцию отмены последних действий. Разработать иерархию классов с использованием наследования (не меньше 3-х уровней наследования). Разработать и использовать классы контейнеров, итераторов и алгоритмов (свои и STL). Производить обработку исключительных ситуаций.

Для реализации проекта используется язык программирования С++, среда разработки Visual Studio 2019, для визуального отображения используется консоль. Операционная система Windows 10 Pro.

1. **СТРУКТУРНОЕ ПРОЕКТИРОВАНИЕ**

Структурная схема программы продемонстрирована в приложении А.

В программе можно выделить три основных элемента: блок визуализации, блок взаимодействия пользователя с программой и блок логики.

Блок визуализации отвечает за открытие консоли, отображение всех элементов меню программы. Блок создан для того, чтобы пользователь видел текущее состояние программы. Блок визуализации связан с блоком логики.

Блок взаимодействия пользователя с игрой работает с блоком визуализации и блоком логики. Этот блок создан для изменения состояния программы по желания пользователя.

Блок логики взаимодействует с блоком взаимодействия пользователя с игрой и отвечает за изменения в расписании, расчёт времени, затрачиваемого транспортом, чтобы проехать от одной станции до другой, следит за соблюдением границ контейнеров и т.д. Без этого блока игровой процесс невозможен.

Каждый из этих блоков необходим, возможно добавление новых блоков для разгрузки уже существующих блоков.

1. **СТРУКТУРА ВХОДНЫХ И ВЫХОДНЫХ ДАННЫХ**

В программе используется 8 текстовых файлов.

Файлы GrId.txt и UndId.txt хранят данные о наземных и подземных станциях (номер и название).

|  |
| --- |
| 1 Station\_1  2 Station\_2  3 Station\_3 |

Файлы GrStops.txt и UndStops.txt хранят данные о взаимном расположении станций между собой (для каждой – количество соседних станций, их номера и промежуток времени, за который можно добраться).

|  |
| --- |
| 1 1 2 0.04  2 2 1 0.04 3 0.05  3 4 2 0.05 4 0.06 9 0.04 13 0.03  4 2 3 0.06 5 0.08 |

|  |
| --- |
| 1 11 1 2 3 13 14 15 16 8 7 5 6  107 6.41 6.53 7.04 … 1.46 1.57 1.58  105 6.48 6.58 7.08 … 1.30 1.39 1.52  44 7.03 7.25 7.49 … 22.46 23.07 23.21  22 7.25 7.33 7.44 … 20.19 20.55 23.44 |

Файлы Buses.txt, Trolleybuses.txt, Trams.txt содержат в себе информацию о маршрутах автобусов, троллейбусов и трамваев (номер маршрута, количество остановок, их номера по порядку, время отправления с первой и последней станции в будние и выходные дни).

|  |
| --- |
| 1 4 1 2 3 4 0.05 0.11 |

Файл Trains.txt содержит в себе информацию о ветках метро (номер, количество станций, их номера и интервалы отправления в будние и выходные дни)

1. **ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе будут рассмотрены основные функциональные составляющие приложения. Диаграмма классов продемонстрирована в приложении Б.

Класс Time – класс, предназначенный для работы со временем

Таблица 4.1 – Класс Time

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| h | int | | часы |
| m | int | | минуты |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Time | − | − | Конструктор |
| Time | double | − | Конструктор с параметрами |
| Time | const Time& | − | Конструктор копирования |
| ~Time | − | − | Деструктор |
| now | − | − | Возвращает время на данный момент |
| geth | − | int | Возвращает часы |
| getm | − | int | Возвращает минуты |
| seth | int | − | Принимает часы |
| setm | int | − | Принимает минуты |
| operator- | Time& | Time | Вычитание |
| operator+ | Time& | Time | Сложение |
| operator-= | Time& | Time& | Вычитание с приравниванием |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| operator+= | Time& | Time& | Сложение с приравниванием |
| operator= | double | Time& | Присваивание |
| operator= | const Time& | Time& | Присваивание |
| operator> | bool | Time | Знак «больше» |
| operator< | bool | Time | Знак «меньше» |
| operator>= | bool | Time | Знак «меньше или равно» |
| operator<= | bool | Time | Знак «больше или равно» |
| operator== | bool | Time | Сравнение |
| operator!= | bool | Time | Сравнение с отрицанием |
| operator>> | istream&, Time& | istream& | Присваивание |
| operator<< | ostream&, Time& | ostream& | Инициализация через консоль |
| operator>> | ifstream&, Time& | ifstream& | Вывод объекта класса на экран |
| operator<< | ofstream&, Time& | ofstream& | Чтение объекта класса из файла |

Класс Company – базовый класс, содержащий в себе имя транспортной компании.

Таблица 4.2 – Класс Company

|  |  |  |
| --- | --- | --- |
| Private поля класса | | |
| Имя | Тип | Описание |
| CompanyName | string | Имя компании |

Продолжение таблицы 4.2

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Company | − | − | Конструктор |
| Company | string | − | Конструктор с параметрами |
| ~Company | − | − | Деструктор |
| getcname | − | string | Возвращает имя компании |
| setcname | string | − | Принимает имя компании |

Класс Transport – класс второго уровня в иерархии классов, созданных классом Company. Содержит в себе номер маршрута и номера станций, через которые этот маршрут проходит.

Таблица 4.3 – Класс Transport

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | | Тип | | | Описание | |
| number | | int | | | Номер маршрута | |
| st | | vector<int> | | | Номера станций | |
| Public методы класса | | | | | |
| Имя | Принимаемые параметры | | Возвращаемое значение | Описание | |
| Transport | − | | − | Конструктор | |
| Transport | int, vector<int> | | − | Конструктор с параметрами | |
| Transport | Transport& | | − | Конструктор копирования | |
| ~Transport | − | | − | Деструктор | |
| setnumber | int | | − | Принимает номер маршрута | |
| setst | vector<int> | | − | Принимает номера остановок | |

Продолжение таблицы 4.3

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| pushst | int | − | Принимает номер одной из остановок |
| getnumber | − | int | Возвращает номер маршрута |
| getst | − | vector<int> | Возвращает номера остановок |
| operator= | Transport& | Transport& | Присваивание |
| operator>> | istream&, Transport& | istream& | Инициализация через консоль |
| operator<< | ostream&, Transport& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Transport& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Transport& | ofstream& | Запись объекта в файл |

Класс Ground\_Trans – класс третьего уровня в иерархии классов, созданных классом Company. Наследуется от класса Transport. Содержит в себе данные, нужные для описания наземного транспорта и методы для работы с ними.

Таблица 4.4 – Класс Ground\_Trans

|  |  |  |
| --- | --- | --- |
| Private поля класса | | |
| Имя | Тип | Описание | |
| stops | Ring<class Ground\_Stops\*> | Указатели на объекты остановок, через которые проходит маршрут | |
| time1 | vector<Time> | Время отправления с первой остановки | |
| time2 | vector<Time> | Время отправления с последней остановки | |
| wtime1 | vector<Time> | Время отправления с первой остановки по выходным | |

Продолжение таблицы 4.4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Имя | Тип | | | Описание | |
| wtime2 | vector<Time> | | | Время отправления с последней остановки по выходным | |
| Public методы класса | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | Описание |
| Ground\_Trans | | − | − | Конструктор |
| Ground\_Trans | | int, Ring<Ground\_Stops\*>, vector<Time>, vector<Time>, vector<Time>, vector<Time>, vector<int> | − | Конструктор с параметрами |
| Ground\_Trans | | Ground\_Trans& | − | Конструктор копирования |
| ~Ground\_Trans | | − | − | Деструктор |
| setstops | | Ring<Ground\_Stops\*> | − | Принимает кольцо остановок маршрута |
| setstops | | Ring<Ground\_Stops>&, int | − | Создаёт новый маршрут |
| settime1 | | vector<Time> | − | Принимает время отправления с первой остановки |
| settime2 | | vector<Time> | − | Принимает время отправления с последней остановки |
| setwtime1 | | vector<Time> | − | Принимает время отправления с первой остановки по выходным |
| setwtime2 | | vector<Time> | − | Принимает время отправления с последней остановки по выходным |

Продолжение таблицы 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| pushstops | Ground\_Stops\* | − | Принимает одну остановку маршрута |
| delstops | int | − | Очищает список остановок маршрута |
| fromtotime | bool, bool, int | vector<Time> | Вычисление расписания маршрута на конкретной станции |
| getstops | − | Ring<Ground\_Stops\*> | Возвращает остановки маршрута |
| gettime1 | − | vector<Time> | Возвращает время отправления с первой остановки |
| gettime2 | − | vector<Time> | Возвращает время отправления с последней остановки |
| getwtime1 | − | vector<Time> | Возвращает время отправления с первой остановки по выходным |
| getwtime2 | − | vector<Time> | Возвращает время отправления с последней остановки по выходным |
| operator= | Ground\_Trans& | Ground\_Trans& | Присваивание |
| operator> | Ground\_Trans& | bool | Знак «больше» |
| operator< | Ground\_Trans& | bool | Знак «меньше» |
| info | − | − | Информация о маршруте |
| stopinfo | int | − | Расписание маршрута на конкретной остановке |

Продолжение таблицы 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| operator>> | istream&, Ground\_Trans& | istream& | Инициализация через консоль |
| operator<< | ostream&, Ground\_Trans& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Ground\_Trans& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Ground\_Trans& | ofstream& | Запись объекта в файл |

Класс Underground\_Trans – класс третьего уровня в иерархии классов, созданных классом Company. Наследуется от класса Transport. Содержит в себе данные, нужные для описания подземного транспорта и методы для работы с ними.

Таблица 4.5 – Класс Underground\_Trans

|  |  |  |  |
| --- | --- | --- | --- |
| Private поля класса | | | |
| Имя | Тип | | Описание |
| stops | vector<Underground\_Stops\*> | | Номер маршрута |
| interval | Time | | Номера станций |
| winterval | Time | |  |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Underground\_Trans | − | − | Конструктор |
| Underground\_Trans | int, Time, vector<int>, vector<class Underground\_Stops> | − | Конструктор с параметрами |
| Underground\_Trans | Underground\_Trans& | − | Конструктор копирования |
| ~Underground\_Trans | − | − | Деструктор |
| setstops | vector<class Underground\_Stops\*> | − | Принимает указатели на станции линии |

Продолжение таблицы 4.5

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| setinterval | Time | − | Принимает интервал движения |
| setwinterval | Time | − | Принимает интервал движения по выходным |
| info | − | − | Выводит на экран информацию о ветке метро |
| pushstops | Underground\_Stops\* | − | Принимает станцию метро |
| getstops | − | vector<class Underground\_Stops\*> | Возвращает станции ветки |
| getinterval | − | Time | Возвращает интервал движения |
| getwinterval | − | Time | Возвращает интервал движения по выходным |
| operator= | Underground\_Trans& | Underground\_Trans& | Присваивание |
| operator> | bool | Underground\_Trans& | Знак «больше» |
| operator< | bool | Underground\_Trans& | Знак «меньше» |
| operator>> | istream&, Underground\_Trans& | istream& | Инициализация через консоль |
| operator<< | ostream&, Underground\_Trans& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Underground\_Trans& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Underground\_Trans& | ofstream& | Запись объекта в файл |

Структура Nearest\_St используется для описания ближайших остановок

Таблицa 4.6 – Структура Nearest\_St

|  |  |  |  |
| --- | --- | --- | --- |
| Поля структуры | | | |
| Имя | Тип | | Описание |
| id | int | | Номер ближайшей остановки |
| time | Time | | Время, за которое можно доехать |
| Методы | | | |
| Имя | Принимаемые  параметры | Возвращаемые значения | Описание |
| operator>> | istream&, Nearest\_St& | istream& | Инициализация через консоль |
| operator<< | ostream&, Nearest\_St& | ostream& | Вывод на экран |
| operator>> | ifstream&, Nearest\_St& | ifstream& | Чтение из файла |
| operator<< | ofstream&, Nearest\_St& | ofstream& | Запись в файл |

Структура StopsInfo содержит в себе имя и номер станции

Таблицa 4.7 – Структура StopsInfo

|  |  |  |  |
| --- | --- | --- | --- |
| Поля структуры | | | |
| Имя | Тип | | Описание |
| id | int | | Номер остановки |
| name | string | | Имя остановки |
| Методы | | | |
| Имя | Принимаемые  параметры | Возвращаемые значения | Описание |
| operator>> | istream&, StopsInfo& | istream& | Инициализация через консоль |
| operator<< | ostream&, StopsInfo& | ostream& | Вывод на экран |
| operator>> | ifstream&, StopsInfo& | ifstream& | Чтение из файла |
| operator<< | ofstream&, StopsInfo& | ofstream& | Запись в файл |

Класс Stops – класс второго уровня в иерархии классов, созданных классом Company. Содержит в себе номер остановки, её имя и информацию о соседних остановках.

Таблица 4.8 – Класс Stops

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| Info | StopsInfo | | | Номер маршрута | | |
| nearest | vector<Nearest\_St> | | | Номера станций | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Stops | | − | − | | Конструктор |
| Stops | | int, string, vector<Nearest\_St> | − | | Конструктор с параметрами |
| Stops | | Stops& | − | | Конструктор копирования |
| ~Stops | | − | − | | Деструктор |
| setid | | int | − | | Принимает номер станции |
| setname | | string | − | | Принимает имя станции |
| pushnstops | | Nearest\_St | − | | Принимает информацию о соседней станции |
| getid | | − | int | | Возвращает номер станции |
| getname | | − | string | | Возвращает имя станции |
| getnstops | | − | vector<Nearest\_St> | | Возвращает информацию о соседних станциях |
| operator> | | bool | Stops& | | Знак «больше» |
| operator< | | bool | Stops& | | Знак «меньше» |
| operator= | | Stops& | Stops& | | Присваивание |

Продолжение таблицы 4.8

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| operator>> | istream&, Stops& | istream& | Инициализация через консоль |
| operator<< | ostream&, Stops& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Stops& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Stops& | ofstream& | Запись объекта в файл |

Класс Ground\_Stops – класс третьего уровня в иерархии классов, созданных классом Company. Наследуется от класса Stops. Содержит в себе указатели на транспорт, маршрут которого проходит через остановку

Таблица 4.9 – Класс Ground\_Stops

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| buses | Ring<Ground\_Trans\*> | | | Указатели на автобусы | | |
| trams | Ring<Ground\_Trans\*> | | | Указатели на трамваи | | |
| trolleybuses | Ring<Ground\_Trans\*> | | | Указатели на троллейбусы | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Ground\_Stops | | − | − | | Конструктор |
| Ground\_Stops | | int, string, vector<Nearest\_St>, Ring<Ground\_Trans\*>, Ring<Ground\_Trans\*>, Ring<Ground\_Trans\*> | − | | Конструктор с параметрами |
| Ground\_Stops | | Ground\_Stops& | − | | Конструктор копирования |
| ~Ground\_Stops | | − | − | | Деструктор |
| settrans | | int, Ring<Ground\_Trans\*> | − | | Принимает указатели на транспорт |

Продолжение таблицы 4.9

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| pushtrans | int, Ground\_Trans\* | − | Принимает указатель на проходящий маршрут |
| setnstops | Ring<Ground\_Stops>& | − | Изменение соседних станций |
| gettrans | int | Ring<Ground\_Trans\*> | Возвращает проходящие маршруты |
| info | − | − | Выводит информацию об остановке |
| operator= | Ground\_Stops& | Ground\_Stops& | Присваивание |
| operator>> | istream&, Ground\_Stops& | istream& | Инициализация через консоль |
| operator<< | ostream&, Ground\_Stops& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Ground\_Stops& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Ground\_Stops& | ofstream& | Запись объекта в файл |

Класс Underground\_Stops – класс третьего уровня в иерархии классов, созданных классом Company. Наследуется от класса Stops. Содержит в себе указатели на транспорт, маршрут которого проходит через остановку

Таблица 4.10 – Класс Underground\_Stops

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| line | int | | | Номер маршрута | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Underground\_Stops | | − | − | | Конструктор |
| Underground\_Stops | | int, string, vector<Nearest\_St>, int | − | | Конструктор с параметрами |

Продолжение таблицы 4.10

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Underground\_Stops | Underground\_Stops& | − | Конструктор копирования |
| ~Underground\_Stops | − | − | Деструктор |
| setline | int | − | Принимает номер ветки метро |
| setnstops | Ring<Underground\_Stops\*> | − | Изменение соседних станций |
| getline | − | int | Возвращает номер ветки |
| operator= | Underground\_Stops& | Underground\_Stops& | Присваивание |
| operator>> | istream&, Underground\_Stops& | istream& | Инициализация через консоль |
| operator<< | ostream&, Underground\_Stops& | ostream& | Вывод объекта класса на экран |
| operator>> | ifstream&, Underground\_Stops& | ifstream& | Чтение объекта класса из файла |
| operator<< | ofstream&, Underground\_Stops& | ofstream& | Запись объекта в файл |

Класс Exception – родительский класс обработки исключений

Таблица 4.11 – Класс Underground\_Stops

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| message | string | | | Сообщение об ошибке | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Exception | | string | − | | Конструктор |
| ~Exception | | − | − | | Деструктор |
| what | | − | − | | Вывод сообщения |

Класс Input\_Except – класс обработки исключений ввода

Таблица 4.12 – Класс Input\_Exept

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| code | int | | | Код ошибки | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Input\_Exept | | string, int | − | | Конструктор |
| ~Input\_Exept | | − | − | | Деструктор |
| what | | − | − | | Вывод сообщения |

Класс File\_Except – класс обработки исключений файла

Таблица 4.13 – Класс File\_Exept

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| code | int | | | Код ошибки | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| File\_Exept | | string, int | − | | Конструктор |
| ~File\_Exept | | − | − | | Деструктор |
| what | | − | − | | Вывод сообщения |

Класс Search\_Except – класс обработки исключений поиска

Таблица 4.14 – Класс Search\_Exept

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| code | int | | | Код ошибки | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Search\_Exept | | string, int | − | | Конструктор |
| ~Search\_Exept | | − | − | | Деструктор |
| what | | − | − | | Вывод сообщения |

Шаблонный класс File – родительский класс, содержащий нужную информацию для работы с файлами

Таблица 4.15 – Класс File

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| fname | string | | | Имя файла | | |
| fin | ifstream | | | Поток чтения | | |
| fout | ofstream | | | Поток записи | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| File | | − | − | | Конструктор |
| File | | string | − | | Конструктор с параметром |
| ~File | | − | − | | Деструктор |

Шаблонный класс Text\_File предназначен для работы с текстовыми файлами

Таблица 4.16 – Класс Text\_File

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Text\_File | − | − | Конструктор |
| Text\_File | string | − | Конструктор с параметром |
| ~Text\_File | − | − | Деструктор |
| Open\_in | − | − | Открывает файл для чтения |
| Open\_out | − | − | Открывает файл для записи |
| Read | T& | − | Читает из файла объект |
| Write | T& | − | Записывает в файл объект |
| Remote | − | − | Устанавливает курсор в начало/конец |
| End | − | bool | Проверка на конец файла |
| IsEmpty | − | bool | Проверка на пустой файл |

Шаблонная структура Node предназначена для работы с шаблонным классом-контейнером Ring

Таблица 4.17 – Структура Node

|  |  |  |
| --- | --- | --- |
| Поля структуры | | |
| Имя | Тип | Описание | |
| obj | T | Объект | |
| next | Node\* | Ссылка на следующий элемент | |
| prev | Node\* | Ссылка на предыдущий элемент | |

Шаблонный класс Ring предназначен для хранения данных разных типов

Таблица 4.18 – Класс Ring

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| head | Node<T>\* | | | Указатель на голову | | |
| current | Node<T>\* | | | Указатель на узел | | |
| size | int | | | Размер кольца | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Ring | | − | − | | Конструктор |
| Ring | | T | − | | Конструктор с параметром |
| Ring | | const Ring<T>& | − | | Конструктор с параметром |
| Ring | | Ring& | − | | Конструктор копирования |
| ~Ring | | − | − | | Деструктор |
| push | | T | − | | Добавление элемента в кольцо |
| pop | | − | T | | Удаление элемента из кольца |
| IsNotEmpty | | − | int | | Проверка на пустоту |
| PrintRing | | − | − | | Вывод всего кольца на экран |
| DeleteNode | | int | − | | Удаление одного элемента |
| Delete | | − | − | | Удаление кольца |
| getsize | | − | int | | Возвращает размер кольца |

Продолжение таблицы 4.18

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| sort | − | − | Сортирует кольцо по возрастанию |
| operator[] | int | T& | Возвращает объект |
| begin | − | Node<T>\* | Возвращает ссылку на голову кольца |
| tail | − | Node<T>\* | Возвращает ссылку на хвост кольца |

Класс Iterator – вложенный класс, предназначенный для работы с шаблонным классом Ring

Таблица 4.19 – Класс Iterator

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Private поля класса | | | | | |
| Имя | Тип | | | Описание | | |
| current | Node<T>\* | | | Указатель на выбранный элемент | | |
| Public методы класса | | | | | |
| Имя | | Принимаемые параметры | Возвращаемое значение | | Описание |
| Iterator | | Node<T>\* | − | | Конструктор с параметром |
| Iterator | | Iterator | − | | Конструктор копирования |
| operator+ | | int | − | | Передвижение на несколько узлов вперёд |
| operator- | | int | − | | Передвижение на несколько узлов назад |
| operator++ | | int | − | | Передвижение на узел вперёд |
| operator-- | | int | − | | Передвижение на узел назад |

Продолжение таблицы 4.19

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| operator= | T | − | Присваивание |
| operator\* | − | T& | Возвращает объект |
| operator== | − | bool | Сравнение |
| GetCurrent | − | Node<T>\* | Возвращает указатель на узел |

Класс Interface – класс, созданный для взаимодействия пользователя с программой

Таблица 4.20 – Класс Interface

|  |  |  |
| --- | --- | --- |
| Private поля класса | | |
| Имя | Тип | Описание | |
| gid | vector<Stops\_Info> | Имя и номер каждой наземной остановки | |
| uid | vector<Stops\_Info> | Имя и номер каждой подземной остановки | |
| gs | Ring<Underground\_Stops> | Информация о наземных остановках | |
| us | Ring<Ground\_Stops> | Информация о подземных остановках | |
| train | Ring<Underground\_Trans> | Информация о поездах | |
| bus | Ring<Ground\_Trans> | Информация об автобусах | |
| trolleybus | Ring<Ground\_Trans> | Информация о троллейбусах | |
| tram | Ring<Ground\_Trans> | Информация о трамваях | |

Продолжение таблицы 4.20

|  |  |  |  |
| --- | --- | --- | --- |
| Public методы класса | | | |
| Имя | Принимаемые параметры | Возвращаемое значение | Описание |
| Interface | − | − | Конструктор |
| ~Interface | − | − | Деструктор |
| open | − | − | Чтение из файлов |
| save | − | − | Сохранение в файл |
| Choose | − | − | Главное меню |
| Change | − | − | Меню изменений |
| Add | − | − | Меню добавления |
| ChangeSmth | − | − | Меню изменения данных |
| ChangeStops | − | − | Меню изменения остановок |
| ChangeTrans | − | − | Меню изменения транспорта |
| Stops | − | − | Меню остановок |
| GrStops | − | − | Наземные остановки |
| UndStops | − | − | Подземные остановки |
| Trans | − | − | Меню транспорта |
| GrTrans | − | − | Наземный транспорт |
| UndTrans | − | − | Подземный транспорт |

1. **РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**
   1. **Схема алгоритма функции void delstops (int type) класса Ground\_Trans**

Схема алгоритма представлена в приложении В.

* 1. **Схема алгоритма функции void sort() класса Ring**

Схема алгоритма представлена в приложении Г.

* 1. **Алгоритм по шагам функции vector<Time> fromtotime (bool a, bool b, int c) класса Ground\_Trans**
     + 1. Начало.
       2. Создаём вектор объектов класса Time vector<Time> NewTime.
       3. Создаём объект класса Time t.
       4. Создаём две int переменные d и i.
       5. Проверяем направление движения. Если a = 1, то направление обратное - шаг 6, иначе шаг 13.
       6. Переменной d присваиваем значение -1.
       7. Переменной i присваиваем значение stops.getsize() – 1.
       8. Проверяем, какой сегодня день (будний/выходной). Если b = 0, то будний - шаг 9, иначе шаг 11.
       9. Переменной NewTime присваиваем значение \*this.time1.
       10. Переходим в шаг 21.
       11. Переменной NewTime присваиваем значение \*this.wtime1.
       12. Переходим в шаг 21.
       13. Переменной d присваиваем значение -1.
       14. Переменной d присваиваем значение 1.
       15. Переменной i присваиваем значение 0.
       16. Проверяем, какой сегодня день (будний/выходной). Если b = 0, то будний - шаг 17, иначе шаг 19.
       17. Переменной NewTime присваиваем значение \*this.time2.
       18. Переходим в шаг 21.
       19. Переменной NewTime присваиваем значение \*this.wtime2.
       20. Переходим в шаг 21.
       21. Если номер остановки совпадает с номером той, котору мы ищем (\*this.stops[i]->getid() = с), то переходим в шаг 34.
       22. Создаём переменную l и инициализируем её нулём.
       23. Если l больше или равно \*this.stops[i]->getnstops().size(), то переходим в шаг 34.
       24. Если номер станции совпадает с номером следующей станции маршрута (stops[i]->getnstops()[l].id == stops[i + d]->getid()), то принимаем в переменную t время, нужное, чтобы доехать до следующей станции, и переходим в шаг 27.
       25. Прибавляем к l единицу.
       26. Переходим в шаг 23.
       27. Создаём переменную k и инициализируем её нулём.
       28. Если k больше или равно NewTime.size(), то переходим в шаг 32.
       29. К времени отъезда от станции NewTime[k] прибавляем время t.
       30. К переменной k прибавим 1.
       31. Переходим в шаг 28.
       32. К переменной i прибавляем d.
       33. Переходим в шаг 21.
       34. Возвращаем NewTime.
       35. Конец.
  2. **Алгоритм по шагам функции SearchStop(T& st)**

1. Начало.
2. Создаём int переменные i, f = 0; bool flag = false; string name.
3. Если остановок нет (st.getsize() == 0), возвращаем -1 и переходим к шагу .
4. С помощью функции InString() вводим с клавиатуры переменную name.
5. Значение i приравниваем к 0.
6. Если индекс выйдет за пределы размера кольца (i => st.getsize()), переходим к шагу 12.
7. Если имя выбранной остановки совпадает с введённым (st[i].getname() == name), переходим к шагу 8, иначе шаг 10.
8. Переменную f приравниваем к 1.
9. Переходим к шагу 12.
10. Прибавляем единицу к i.
11. Переходим к шагу 6.
12. Если f не равно 1, функция what() класса исключений поиска сообщит об ошибке, переходим к шагу 14.
13. Переменной flag присваиваем значение true.
14. Если значение переменной flag равно false, переходим к шагу 4.
15. Возвращаем значение переменной i.
16. Конец.
    1. **Код программы**

Код программы представлен в приложении Д.

1. **ТЕСТИРОВАНИЕ**

На этапе тестирования проверяется правильность основных действий, совершаемых во время работы приложения.

Запуск приложения, открытие меню (рисунок 5.1).

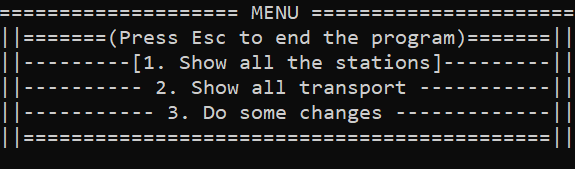


Рисунок 5.1

Выбираем в меню пункт со станциями (рисунок 5.2).

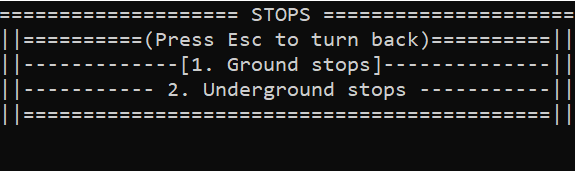


Рисунок 5.2

Получаем список всех наземных остановок (рисунок 5.3)

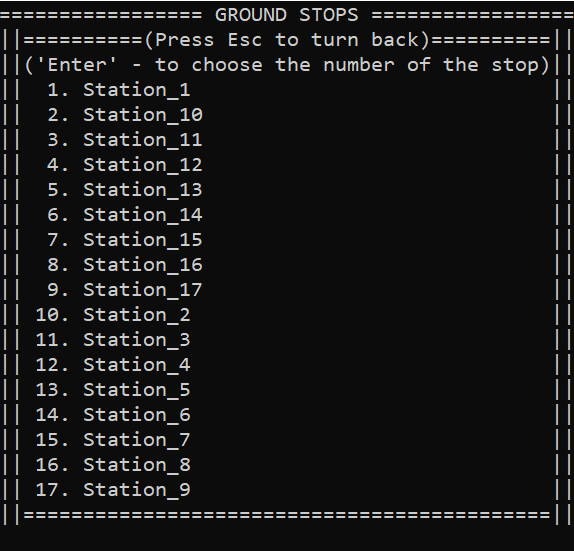


Рисунок 5.3

Получаем информацию о подземных станциях метро (рисунок 5.4)

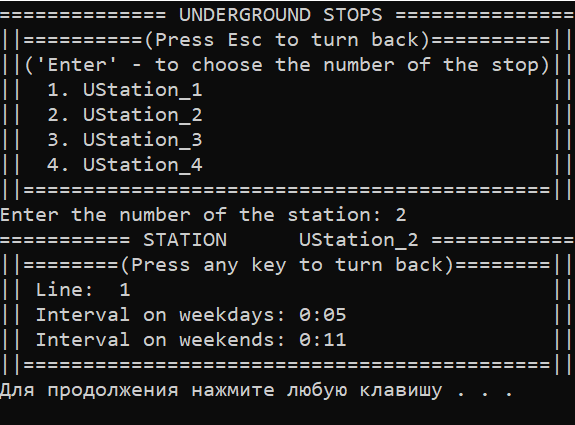


Рисунок 5.4

Если мы захотим посмотреть информацию о каком-нибудь маршруте, мы увидим список остановок и три ближайших времени прибытия транспорта на них (рисунок 5.5)

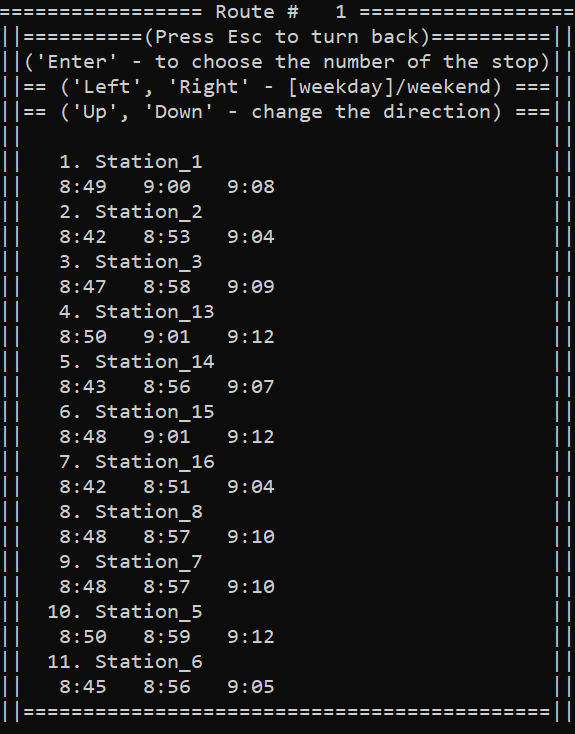


Рисунок 5.5

Если мы захотим узнать расписание маршрута на конкретной остановке, мы можем увидеть времена отправления, в зависимости от направления движения и того, какой сегодня день (будний или выходной) (рисунок 5.6)

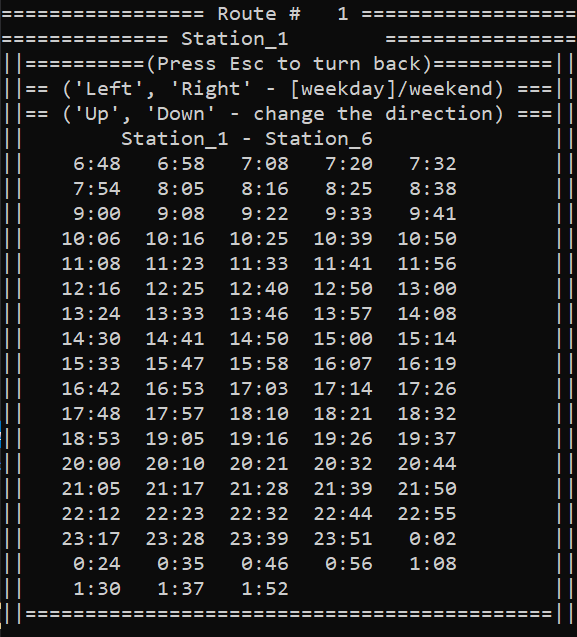


Рисунок 5.6

# **ЗАКЛЮЧЕНИЕ**

В данном курсовом проекте была разработана программа, имитирующая работу программы расписания городского транспорта. Данный опыт позволил углубить знания об этом языке, приобрести новые навыки, отработать материал, пройденный в семестре. В ходе разработки программы пришлось изучить достаточное количество различных материалов по языку программирования С++.

В программе удалось реализовать возможность добавлять и редактировать остановки и маршруты, выводить на экран расписание наземного и подземного транспорта как для конкретного маршрута, так и для нескольких маршрутов на конкретной станции.

Данное приложение в дальнейшем можно усовершенствовать:

* Добавить возможность прокладывать маршруты из одной точки в другую.
* Добавить возможность получать информацию о расписании при помощи сети интернет.
* Добавить удобный графический интерфейс.
* Добавлять другие виды транспорта, выйти за рамки городского транспорта и добавить междугородние маршруты.

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Шилдт, Г. Самоучитель C++ (3-е издание)/ Г. Шилдт — СПб.: БХВ-Петербург, 2005. – 679 с.
2. Лафоре, Р. Объектно-ориентированное программирование в C++/ Р. Лафоре. – СПб. : Питер, 2004. – 923 с.
3. Язык программирования С++ / Бьерн Страуструп — Москва : Бином, 2011. – 1135 с. :
4. Луцик, Ю. А. Объектно-ориентированное программирование на языке С++ : учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. – Минск : БГУИР, 2008. – 266 с.

# **ПРИЛОЖЕНИЕ А**

(обязательное)

Структурная схема

# **ПРИЛОЖЕНИЕ Б**

(обязательное)

Диаграмма классов

# **ПРИЛОЖЕНИЕ В**

(обязательное)

Схема алгоритма функции void delstops (int type) класса Ground\_Trans

# **ПРИЛОЖЕНИЕ Г**

(обязательное)

Схема алгоритма функции bool analizeUpLeftForFight (Checker\*\*\* board) класса WhiteQueen

# **ПРИЛОЖЕНИЕ Д**

(обязательное)

Код программы

# **ПРИЛОЖЕНИЕ Е**

(обязательное)

Ведомость документов