**СОДЕРЖАНИЕ**

Содержание………………………………………………………………..1

Введение…………………………………………………………………...2

1. Постановка задачи……………………………………………………...3

2. Обзор литературы……………………………………………………....4

2.1. Обзор методов и алгоритмов решения поставленной задачи……..4

3. Функциональное проектирование………………………………….....5

3.1. Структура входных и выходных данных……………………….…..5

3.2. Разработка диаграммы классов……………………………………...6

3.3. Описание классов…………………………………………………….6

4 Разработка программных модулей…………………………………...17

4.1. Разработка схем алгоритмов……………………………………….17

4.2. Разработка алгоритмов……………………………………………..17

Результат работы………………………………………………………...19

Заключение……………………………………………………………….21

Список литературы………………………………………………………22

Приложение А………………………………………………………...….23

Приложение Б……………………………………………………………24

Приложение В……………………………………………………………25

Приложение Г……………………………………………………………26

**ВВЕДЕНИЕ**

Первоначально разработанный Бьярне Страуструпом в 1979 году, язык C++ был задуман как преемник языка С. С — процедурный язык, где каждая функция предпринимает определенное действие. Язык C++, напротив, был задуман как объектно-ориентированный, но он реализует такие концепции, как наследование, абстракция, полиморфизм и инкапсуляция. Классы языка C++ используют свойства для содержания данных и методы для обработки этих данных. (Методы являются аналогом функций в языке С.) В результате программист больше думает о данных и о том, что с ними следует сделать.

Компиляторы C++ традиционно поддержали также программы на языке С. Преимуществом этого была совместимость с устаревшим кодом, а недостатком — чрезвычайно высокая сложность компиляторов, вынужденных обеспечивать программистам эту совместимость, одновременно реализуя все новые средства, которых требует развитие языка. Язык C++ — это ключ к современному объектно-ориентированному программированию. Он создан для разработки высокопроизводительного программного обеспечения и чрезвычайно популярен среди программистов.

Сегодня быть профессиональным программистом высокого класса означает быть компетентным в C++. Этот язык не просто популярен. Он обеспечивает концептуальный фундамент, на который опираются другие языки программирования и многие современные средства обработки данных. Не случайно ведь потомками C++ стали такие почитаемые языки, как C# и Java. Поскольку язык C++ предназначен для профессионального программирования, для изучения он не самый простой; тем не менее, C++ — самый лучший язык для изучения. Освоив C++, вы сможете писать профессиональные высокопроизводительные программы. Кроме того, вы сможете легко изучить такие языки программирования, как C# и Java, поскольку они используют тот же базовый синтаксис и те же принципы разработки.

**1 ПОСТАНОВКА ЗАДАЧИ**

Программа “Автопаркинга” должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В программе должна быть предусмотрена возможность хранения информации в нескольких файлах, связанных определенным образом. ㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤㅤ  
Программа должна выполнять:

- выдача информации о свободных местах;

- резервирование места;

- организация труда работников (заработная плата, должность, отпуск)

- другие методы в зависимости 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 в программе классы контейнеров, итераторов и алгоритмов (свои и STL). Производить обработку исключительных ситуаций.

1. **ОБЗОР ЛИТЕРАТУРЫ**

**2.1 Обзор методов и алгоритмов решения поставленной задачи**

Для записи и хранения информации об автомобилях и водителях используется последовательный контейнер двусторонняя очередь. Этот контейнер имеет методы добавить элемент в очередь, удалить элемент из очереди, удалить элемент из очереди с возвратом значения, удалить одинаковые элементы двух очередей, просмотреть.

Для реализации отмены последнего действия используется последовательный контейнер стек. Этот контейнер работает по принципу LIFO («последним пришёл — первым вышел»). Методы стека: push – заносит в вершину стека входящую информацию; pop – удаляет вершину стека и переносит вершину на следующий элемент после вершины. Так же осуществлён метод удаления с возвратом значения.

Перед изменением очереди с данными о водителях и автомобилях мы сохраняем очереди в стек (push), а при отмене достаем вершину (pop).

Алгоритмы сортировки и поиска в очереди. Сортировка использует перегрузку сравнения (<) и сортирует по выбранному полю. Алгоритм сортировки: сортировка вставкой. Поиск использует перегрузку сравнения (==) и ищет по ненулевым параметрам.

Доступы к полям класса осуществленные через “геттер” и “сеттер”, что позволяет изменять параметры класса вне класса.

Для записи и чтения информации из файлов реализован метод прочитать из файла и записать в файл. Эти методы используют перегрузки чтения из файла и записи в файл соответственно.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ**

В данном разделе пояснительной записки детально рассмотрим функционирование программы.

**3.1 Структура входных и выходных данных**

Таблица 3.1 - файл легковых автомобилей *car.txt*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| id | Парковочное место | Дата  въезда | Дата  выезда | Оплата,  бел.руб. | Марка | Номер  машины |
| 2 | 5 А | 03 12 2022 15 00 | 03 12 2022 23 00 | 8 | BMW | 1452DA2 |

Таблица 3.2 - файл грузовиков *truck.txt*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | Парковочное место | Дата въезда | Дата  выезда | Оплата,  бел.руб. | Марка | Номер  машины | Высот,  см | Длина,  см |
| 5 | 3 B | 01 12 2022 09 00 | 07 12 2022 09 00 | 114 | MAZ | 1425LM4 | 210 | 600 |

Таблица 3.3 - файл электросамокатов *scooter.txt*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| id | Парковочное место | Дата  въезда | Дата  выезда | Оплата,  бел.руб. | Марка | Объем  батареи | Статус  зарядки |
| 7 | 1 D | 03 12 2022 10 00 | 03 12 2022 19 00 | 9 | Kugoo | 2500 | - |

Таблица 3.4 – файл электромобилей *electrocar.txt*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | Парковочное место | Дата  въезда | Дата  выезда | Оплата,  бел.руб. | Марка | Номер машины | Объем  батареи | Статус  зарядки |
| 6 | 2 С | 03 12 2022 16 00 | 04 12 2022 04 00 | 12 | Tesla | 1435NG7 | 20000 | + |

Таблица 3.5 - файл сотрудников employee*.txt*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Логин | Пароль | Фамилия | Имя | Отчество | Должность | Зарплата | Начало отпуска | Конец отпуска |
| 1234 | 1234 | Гаращук | Никита | Васильевич | Директор | 2000 | 30 12 2022 | 05 01 2023 |

**3.2 Разработка диаграммы классов**

Диаграмма классов данной работы показана в приложении А.

**3.3 Описание классов**

Класс Queue<T> - шаблонный класс контейнера двунаправленной очереди. Этот класс является основным классов проекта и используется для хранения объектов различных классов.

Описание полей классa:

nodeQ<T>\* head – указатель на голову очереди

nodeQ<T>\* tail – указатель на хвост очереди

int size – количество элементов очереди

Описание методов:

queue() – конструктор по умолчанию

queue(T\* arr, int count) – конструктор с параметрами

queue(const queue<T>& other) – конструктор копирования

~queue() – деструктор

void put(T& value) - метод добавления элемента в очередь

void show() – метод вывода очереди на экран

void del() – метод удалить элемент с головы

void del(T del\_value) – метод удалить нужный элемент

void del\_any(queue<T> other) – метод удалить одинаковые элементы очередей

void clear() – метод очистки очереди

bool isEmpty() – метод проверки пустая ли очереди

int getSize() – метод получить количество элементов очереди

queue<T> operator=(const queue<T>& other) – метод приравнять очереди

queue<T> operator+(const queue<T>& other) – метод сложить очереди

Iterator Begin() – метод переместить итератора очереди в начало

Iterator End() – метод переместить итератор очереди в конец

Класс nodeQ<T> - шаблонный класс элемента очереди.

Описание полей классa:

T inform – информация

nodeQ\* nx – указатель на следующий элемент

nodeQ\* pr – указатель на предыдущий элемент

Описание методов:

nodeQ() – конструктор по умолчанию

~nodeQ() – деструктор

Класс stack<T> - шаблонный класс контейнера стека. Этот класс используется для реализации отмены последнего действия.

Описание полей классa:

nodeS<T>\* Top - указатель на вершину стека

int size – количество элементов стека

Описание методов:

int getSize() – метод получить количество во элементов в стеке

void Push(T& obj) – метод добавить элемент

void Pop(T& obj) – метод удалить с вершины с возвратом значения

void Pop() – метод удалить с вершины

bool isEmpty() – метод проверки пуст ли стек

Класс nodeS<T> - шаблонный класс элемента стека.

Описание полей классa:

T inform – информация

nodeS\* nx – указатель на следующий элемент

Описание методов:

nodeS() – конструктор по умолчанию

~nodeS() – деструктор

Класс algorithm<T> – шаблонный класс алгоритмов контейнеров, включающий в себя сортировки и поиск.

Описание методов:

void sortAscendingOrder(queue<T>::template Iterator itBeg, queue<T>::template Iterator itEnd) – сортировка по возрастанию

void sortDescendingOrder(queue<T>::template Iterator itBeg, queue<T>::template Iterator itEnd) – сортировка по убыванию

queue<T>::template Iterator searchObj(queue<T>::template Iterator itBeg, queue<T>::template Iterator itEnd,T obj) – поиск объекта

queue<T> searchQueue(queue<T>::template Iterator itBeg, queue<T>::template Iterator itEnd, T obj) – поиск очереди

Класс myInterface – класс интерфейса, включающий в себя различные меню для работы с программой.

Описание методов:

myInterface() – конструктор по умолчанию

~myInterface() – деструктор

void menu(complect<car>& sectorA, complect<truck>& sectorB, complect<electrocar>& sectorC, complect<scooter>& sectorD, queue<driver>& drivers) – меню работника

void menuAdmin(queue<employee>& workers) – меню администратора

friend void update(queue<T>& q, place\* places, int count) – обновление мест

friend void menuSector(complect<T>& sector, queue<driver>& drivers) – меню сектора

Класс complect – класс сектора.

Описание полей классa:

char nameSector – название сектора

int count – количество мест

place\* places – указатель на массив мест

queue<T> q – очередь транспорта

Описание методов:

complect(char nameSector, int count, place\* places, queue<T>& q) – конструктор по умолчанию

complect(complect& other) – конструктор копирования

~complect() - деструктор

queue<T>& getQueue() – метод получить очередь

place\* getPlaces() – метод получить массив мест

int getCount() – метод получить количество мест

Класс MYFILE<T> – шаблонный класс для работы с файлом

Описание полей:

ofstream \_outFile – поток ввода

ifstream \_inFile – поток вывода

char nameFile[80] – название файла

char mode – режим работы

Описание методов:

MYFILE(char\* nameFile) – конструктор с параметрами

MYFILE(const char\* nameFile) – конструктор с параметрами

MYFILE(const MYFILE& other) – конструктор копирования

void fileOpen(const char\* mode) – открыть файл в выбранном режиме

void fileClose() – закрыть файл

void fileWrite(T& inform) – метод записи объекта

void fileRead(T& inform) – метод чтения объекта

void remote() – метод переместиться в начало

bool isEnd() – метод проверки достигнут ли конец файла

Класс MyException – базовый класс ошибок разного типа.

Описание полей:

int code – код ошибки

char message[80] – что за ошибка

Описание методов:

MyException() – конструктор по умолчанию

MyException(int code, const char\* message) – конструктор с параметрами

MyException(const MyException& other) – конструктор копирования

~MyException() - деструктор

virtual void what() – виртуальная функция вывода ошибки на экран

Класс MyExceptionNUM – класс ошибок для работы с числами

Описание полей:

char message2[80] – строка для помощи устранить ошибку

Описение методов:

MyExceptionNUM(int code, const char\* message) – конструктор с параметрами

MyExceptionNUM(const MyException& other) – конструктор копирования

~MyExceptionNUM() - деструктор

virtual void what() – виртуальная функция вывода ошибки на экран

Класс MyExceptionFILE – класс ошибок для работы с файлами

Описание полей:

char message2[80] – строка для помощи устранить ошибку

Описание методов:

MyExceptionFILE (int code, const char\* message) – конструктор с параметрами

MyExceptionFILE (const MyException& other) – конструктор копирования

~MyExceptionFILE () - деструктор

virtual void what() – виртуальная функция вывода ошибки на экран

Класс MyExceptionSTR – класс ошибок для работы со строками

Описание полей:

char message2[80] – строка для помощи устранить ошибку

Описание методов:

MyExceptionSTR (int code, const char\* message) – конструктор с параметрами

MyExceptionSTR (const MyException& other) – конструктор копирования

~MyExceptionSTR () - деструктор

virtual void what() – виртуальная функция вывода ошибки на экран

Класс date – класс даты

Описание полей:

int dd – день

int mm – месяц

int yyyy – год

Описание методов:

date() – конструктор по умолчанию

date(int day, int month, int year) – конструктор с параметрами

date(const date& other) – конструктор копирования

~date() - деструктор

date operator= (const date& other) – перегрузка оператора =

date operator= (const int ms[3]) – перегрузка оператора =

bool operator==(date& other) – перегрузка оператора ==

bool operator<(date& other) – перегрузка оператора <

int operator-(date& other) – перегрузка оператора -

bool isNull() – метод проверки на пустой объект

bool leap() – метод проверки на високосный год

void setDay(int dd) – метод установить день

void setMonth(int mm) – метод установить месяц

void setYear(int yyyy) – метод установить год

int getDay() – метод получить день

int getMonth() – метод получить месяц

int getYear() – метод получить год

friend ostream& operator<<(ostream&, date&) – перегрузка оператора вывода

friend istream& operator>>(istream&, date&) – перегрузка оператор ввода

friend ofstream& operator<<(ofstream&, date&) – перегрузка оператора записи в файл

friend ifstream& operator>>(ifstream&, date&) - перегрузка оператора чтения из файла

Класс human – базовый класс человека

Описание полей:

int id – уникальный номер

char name[15] – имя

char surname[15] – фамилия

char midname[15] – отчество

Описание методов:

human() – конструктор по умолчанию

human(const human& other) – конструктор копирования

~human() – деструктор

void setId(int id) – метод становить id

void setName(const char\* name) – метод установить имя

void setSurname(const char\* surname) – метод установить фамилию

void setMidname(const char\* midname) – метод установить отчество

int getId() – метод получить значение id

char\* getName() – метод получить имя

char\* getSurname() – метод получить фамилию

char\* getMidname() – метод получить отчество

virtual void tableHead() – виртуальная функция вывода шапки таблицы

void replace(const human& other) – метод копирования не нулевых полей

human ChooseParams() – метод для выбора параметра сортировки

bool operator==(human& other) – перегрузка оператора ==

human operator=(const human& other) – перегрузка оператора =

bool operator>(human& other) – перегрузка оператора >

friend istream& operator>>(istream& in, human& other) – перегрузка ввода

friend ostream& operator<<(ostream& out, human& other) – перегрузка вывода

friend ifstream& operator>>(ifstream& inf, human& other) – перегрузка чтения из файла

friend ofstream& operator<<(ofstream& outf, human& other) – перегрузка записи в файл

Класс driver – класс водителя

Описание полей:

int numPassport – номер паспорта

int driverLicense – номер лицензии

Описание методов:

driver() – конструктор по умолчанию

driver(const driver& other) – конструктор копирования

~driver() – деструктор

void setNumPassport(int numPassport) – метод установить номер паспорта

void setDriverLicense(int driverLicense) – метод установить номер прав

int getNumPassport() – метод получить номер паспорта

int getDriverLicense() – метод получить номер прав

void tableHead() override – виртуальная функция вывода шапки таблицы

void change() - меню изменения полей класса

void replace(const driver& other) – метод копирования не нулевых полей

driver ChooseParams() – метод выбор поля сортировки

void filter() – метод для заполнения фильтра поиска

driver operator = (const driver& other) – перегрузка оператора оператор =

bool operator==(driver& other) – перегрузка оператора ==

bool operator>(driver& other) – перегрузка оператора >

friend istream& operator>>(istream& in, driver& other) – перегрузка ввода

friend ostream& operator<<(ostream& out, driver& other) – перегрузка вывода

friend ifstream& operator>>(ifstream& inf, driver& other) – перегрузка чтение из файл

friend ofstream& operator<<(ofstream& outf, driver& other) – перегрузка записи в файл

Класс employee – класс работника

Описание полей:

date startHoliday – дата начала отпуска

date endHoliday – дата конца отпуска

int salary – зарплата

char jobTitle[15] – должность

access \_access – пароль и логин

Описание методов:

employee() – конструктор по умолчанию

employee(employee& other) – конструктор копирования

~employee() – деструктор

void setSalary(int salary) – метод установить зарплату

void setJobTitle(const char\* jobTitle) – метод установить должность

void setAccess(access& otherAccess) – метод установить пароль и логин

int getSalary() – метод получить значение зарплаты

char\* getJobTitle() – метод получить должность

access getAccess() – метод получить пароль и логин

void filter() – метод заполнения фильтра поиска

void change() – меню изменения полей

bool isNull() – метод проверки нулевой ли класс

void start() – метод загрузки работника

void registration() – метод регистрации

void authorization() – метод авторизация

void greeting() – приветствие

void tableHead() override – виртуальная функция вывода шапки таблицы

void replace(const employee& other) – метод копирования не нулевых полей

employee ChooseParams() – метод выбора поля для сортировки

employee operator=(const employee& other) – перегрузка оператора =

bool operator==(employee& other) – перегрузка оператора ==

bool operator>(employee& other) – перегрузка оператора >

friend istream& operator>>(istream& in, employee& other) – перегрузка ввода

friend ostream& operator<<(ostream& out, employee& other) – перегрузка вывод

friend ofstream& operator<<(ofstream& outf, employee& other) – перегрузка записи в файл

friend ifstream& operator>>(ifstream& inf, employee& other) – перегрузка чтения из файла

Класс place – класс парковочного места

Описание полей:

int id – уникальный номер

int num – номер места

char sector – секция

bool taken – статус занят/свободен

fullDate timeOfEntry – дата въезда

fullDate timeOfExit – дата выезда

int payment – оплата

Описание методов:

place() – конструктор по умолчанию

place(const place& other) – конструктор копирования

~place() – деструктор

void setTaken(bool taken) – метод установить статус

void setNum(int num) – метод установить номер

void setId(int Id) – метод установить id

void setSector(char sector) – метод установить сектор

void setPayment(int payment) – метод установить оплату

void setTimeOfEntry(const fullDate timeOfEntry) – метод установить дату въезда

void setTimeOfExit(const fullDate timeOfExit) – метод установить дату выезда

bool getTaken() – метод получить статус

int getNum() – метод получить номер места

int getId() – метод получить id

char getSector() – метод получить сектор

fullDate getTimeOfEntry() – метод получить дату въезда

fullDate getTimeOfExit() – метод получить дату выезда

int getPayment() – метод получить оплату

place operator=(const place& other) - перегрузка оператор =

friend void show\_places(place\* other, int count) – метод вывода мест в виде рисунка

friend istream& operator>>(istream& in, place& other) – перегрузка ввода мест

friend ifstream& operator>>(ifstream& inf, place& other) – перегрузка чтения из файла

friend ostream& operator<<(ostream& out, place& other) – перегрузка вывода

friend ofstream& operator<<(ofstream& outf, place& other) – перегрузка записи в файла

Класс vehicle – базовый класс средства передвижения

Описание полей:

int id – уникальный номер

char brand[10] – марка

place Place – место

Описание методов:

vehicle() – конструктор по умолчанию

vehicle(const vehicle& other) – конструктор копирования

~vehicle() – деструктор

void setId(int id) – метод установить id

void setBrand(const char\* brand) – метод установить марку

void setPlace(int num, char sector) – метод установить место

int getId() – метод получить значение id

place getPlace() – метод получить место

char\* getBrand() – метод получить марку

void operator=(const vehicle& other) – перегрузка оператора =

bool operator ==(vehicle& other) – перегрузка оператора ==

bool operator>(vehicle& other) – перегрузка оператор >

virtual void tableHead() – виртуальная функция вывода шапки таблицы

void replace(vehicle& other) – метод копирование не нулевых полей

vehicle ChooseParams() – метод выбора параметра сортировки

friend ofstream& operator<<(ofstream& outf, vehicle& other) – перегрузка записи в файл

friend istream& operator>>(istream& in, vehicle& other) – перегрузка ввод

friend ostream& operator<<(ostream& out, vehicle& other) – перегрузка вывода

friend ifstream& operator>>(ifstream& in, vehicle& other) – перегрузка чтения из файла

Класс car – класс машины

Описание полей:

char numCar[8] – номер машины

Описание методов:

car () – конструктор по умолчанию

car(const car& other) – конструктор копирования

~car() – деструктор

void setNumCar(char num[4], char series[2], char region) – метод установить номер машины

char\* getNumCar() – метод получить номер машины

car operator=(const car& other) – перегрузка оператор =

bool operator ==(car& other) – перегрузка оператор ==

bool operator>(car& other) – перегрузка оператор >

void tableHead() override – виртуальная функция вывода шапки таблицы

void replace(car& other) – метод копирования не нулевых полей

car ChooseParams() – метод выбора поля сортировки

void change() – меню изменения поля

void filter() – метод ввода фильтра поиска

friend istream& operator>>(istream& in, car&) – перегрузка ввода

friend ostream& operator<<(ostream& out, car& other) – перегрузка вывод

friend ofstream& operator<<(ofstream& out, car& other) – перегрузка записи в файл

friend ifstream& operator>>(ifstream& inf, car& other) – перегрузка чтения из файла

Класс scooter – класс электросамоката

Описание полей:

bool charge – статус зарядки

int volumeСharge – объем зарядки

Описание методов:

scooter () – конструктор по умолчанию

scooter(const scooter& other) – конструктор копирования

~scooter() – деструктор

void setCharge(bool charge) – метод установить статус

void setVolumeCharge(int volumeCharge) – метод установить объем батареи

bool getCharge() – метод получить статус

int getVolumeCharge() – метод получить объем батареи

scooter operator=(const scooter& other) – перегрузка оператор =

bool operator==(scooter& other) – перегрузка оператор ==

bool operator>(scooter& other) – перегрузка оператор >

void tableHead() override – виртуальная функция вывода шапки таблицы

void replace(scooter& other) – копирование не нулевых полей

scooter ChooseParams() – метод выбора поля сортировки

void change() – меню изменения полей

void filter() – ввода фильтра поиска

friend ofstream& operator<<(ofstream& out, scooter& other) – перегрузка записи в файл

friend istream& operator>>(istream& in, scooter& other) – перегрузка ввода

friend ostream& operator<<(ostream& out, scooter& other) – перегрузка вывода

friend ifstream& operator>>(ifstream& inf, scooter& other) – перегрузка чтения из файла

Класс truck – класс грузовика

Описание полей:

int height – высота

int length – длина

Описание методов:

truck() – конструктор по умолчанию

truck(const truck& other) : car(other) – конструктор копирования

~truck() – деструктор

void setHeight(int height) – метод установить высоту

void setLength(int length) – метод установить длину

int getHeight() – метод получить высоту

int getLength() – метод получить длину

truck operator=(const truck& other) – перегрузка оператора =

bool operator==(truck& other) – перегрузка оператора ==

bool operator>(truck& other) – перегрузка оператора >

void tableHead() override – виртуальная функция вывода шапки таблицы

void replace(truck& other) – копирование не нулeвых полей

truck ChooseParams() – метод выбора поля для сортировки

void change() – меню изменения полей

void filter() – ввод фильтра поиска

friend ofstream& operator<<(ofstream& out, truck& other) – перегрузка записи в файл

friend istream& operator>>(istream& in, truck& other) – перегрузка ввода

friend ostream& operator<<(ostream& out, truck& other) – перегрузка вывода

friend ifstream& operator>>(ifstream& inf, truck& other) – перегрузка чтения из файла

Класс electrocar – класс электромобиля

Описание полей:

bool charge – статус зарядки

int volumeСharge – объем зарядки

Описание методов:

electrocar () – конструктор по умолчанию

electrocar(const electrocar& other) – конструктор копирования

~electrocar() - деструктор

void setCharge(bool charge) – метод установить статус

void setVolumeCharge(int volumeCharge) – установить объем батареи

bool getCharge() – метод получить статус

int getVolumeCharge() – метод получить объем батареи

electrocar operator=(const electrocar& other) – перегрузка оператора =

bool operator==(electrocar& other) – перегрузка оператора ==

bool operator>(electrocar& other) – перегрузка оператора >

void tableHead() override – виртуальная функция вывода шапки таблицы

void replace(electrocar& other) - копирование не нулевых полей

electrocar ChooseParams() – метод выбора поля сортировки

void change() – меню изменения полей

void filter() – метод ввода фильтра поиска

friend ofstream& operator<<(ofstream& out, electrocar& other) – перегрузка записи в файл

friend istream& operator>>(istream& in, electrocar& other) – перегрузка ввода

friend ostream& operator<<(otream& out, electrocar& other) –перегрузка вывода

friend ifstream& operator>>(ifstream& inf, electrocar& other) – перегрузка чтения из файла

**4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

В данном разделе курсовой работы рассмотрены алгоритмы по шагам 2

методов (sortAscendingOrder() и searchObj()), а также схемы двух других методов (saveInFile() и put()).

**4.1 Разработка схем алгоритмов**

Метод saveInFile(queue<T> q) сохраняет очередь из объектов класса T в файл. Схема метода saveInFile(queue<T> q) показана в приложении Б.

Метод put(T& value) добавляет элемент класса T в хвост очереди. Схема

метода put(T& value) показана в приложении В.

* 1. **Разработка алгоритмов**

Метод sortAscendingOrder() сортирует очередь в порядке возрастания.

Алгоритм по шагам метода sortAscendingOrder (template queue<T> Iterator itBeg, template queue<T> Iterator itEnd):

1. Начало
2. Входные данные:

template queue<T> Iterator itBeg – итератор на начало контейнера

template queue<T> Iterator itEnd – итератор на конец контейнера

template queue<T> Iterator min – итератор на минимальный элемент

template queue<T> Iterator j – итератор-бегунок

T objMin – объект итератора min

T objJ – объект итератора j

T tmp – объект для перестановки объектов

1. Выходные данные:

Отсортированный контейнер Queue<T> по определённому полю

1. Вызов метода у объекта \*(itBeg) ChooseParams() – выбор параметра сортировки
2. Результат метода занести в объекты objMin и objJ
3. min=itBeg, j=itBeg
4. Вызов метода объекта minObj replace(\*min) – изменение не нулевых полей объекта
5. Вызов метода объекта minJ replace(\*j) – изменение не нулевых полей объекта
6. Если objJ > objMin, переход на шаг 10, иначе на шаг 11
7. min=j;
8. Увеличение итератора j на 1.
9. Если j равен itBeg, переход на шаг 13, иначе на 7
10. Если min равен itBeg, переход на шаг 15, иначе на шаг 14
11. Поменять местами объекты \*min и \*itBeg
12. Увеличить итератор itBeg на 1
13. Если itBeg равен itEnd переход на шаг 17, иначе 7
14. Конец

Метод searchObj() находит первый найденный объект в очереди

Алгоритм по шагам метода searchObj (template queue<T> Iterator itBeg,  
template queue<T> Iterator itEnd, T obj):

1. Начало
2. Входные данные:

template queue<T> Iterator itBeg – итератор на начало контейнера

template queue<T> Iterator itEnd – итератор на конец контейнера

T obj – объект-фильтр для поиска

1. Выходные данные:

template queue<T> Iterator itFind – итератор на найденный объект

1. Цикл по элементам контейнера от itBeg до itEnd с шагом itBeg+1
2. Если объект \*itBeg равен obj, переход на шаг 6, иначе на шаг 7
3. itFind = itBeg
4. Конец

**РЕЗУЛЬТАТ РАБОТЫ**

Результат работы программы показана на рисунках. Рисунки 1 и 2 интерфейс обычного работника. Рисунки 3 и 4 интерфейс администратора.

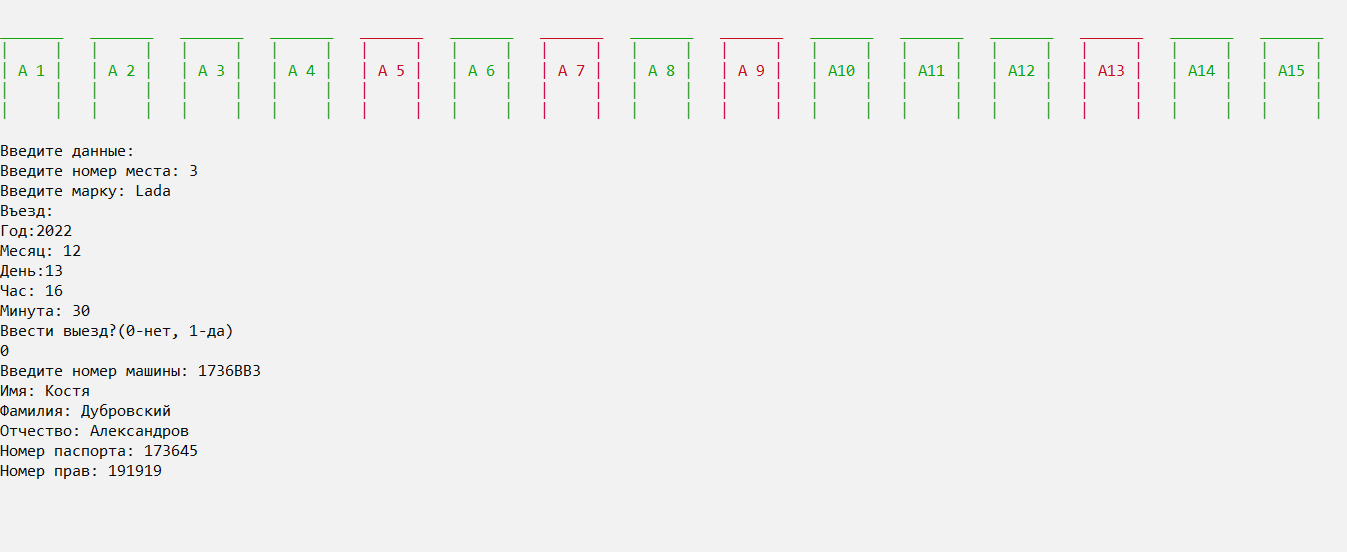


Рисунок 1 – Регистрация въезда на парковку (по факту)

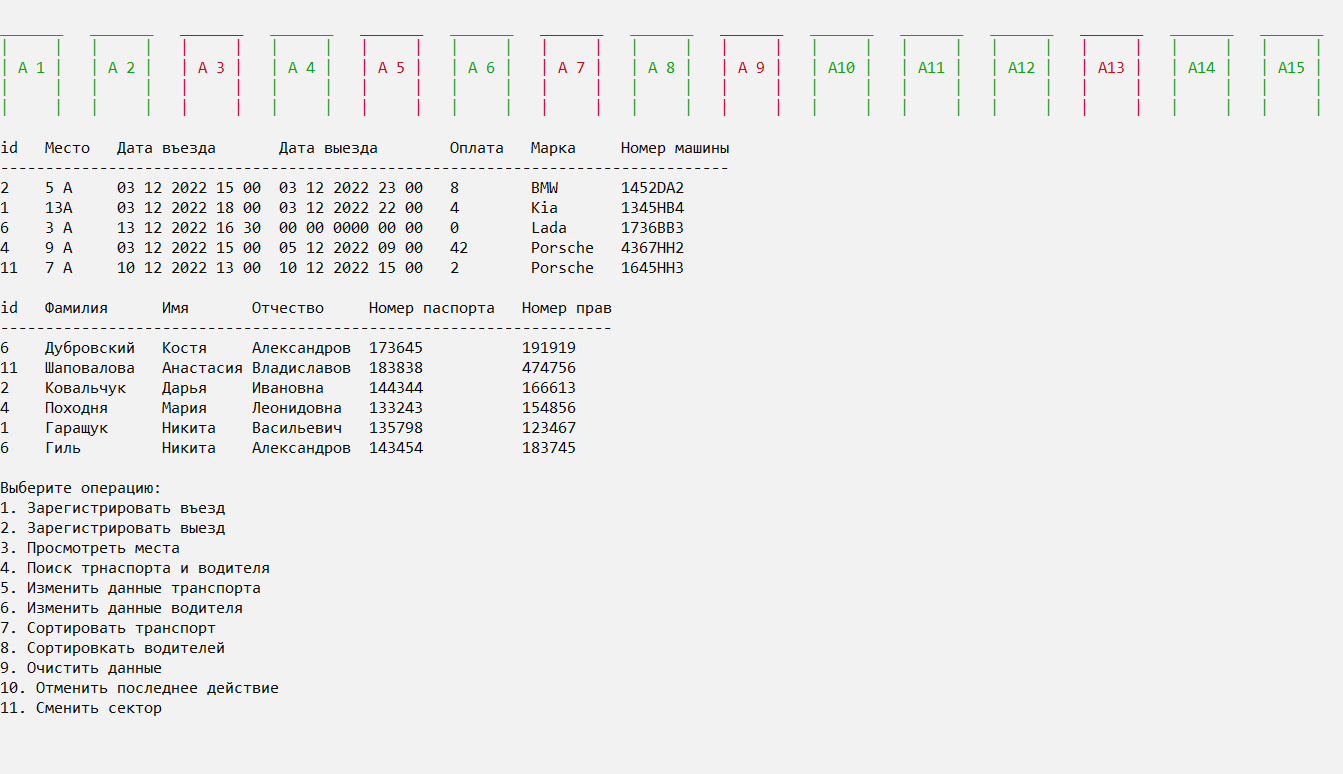


Рисунок 2 – Обновление информации о местах и вывод информации на экран

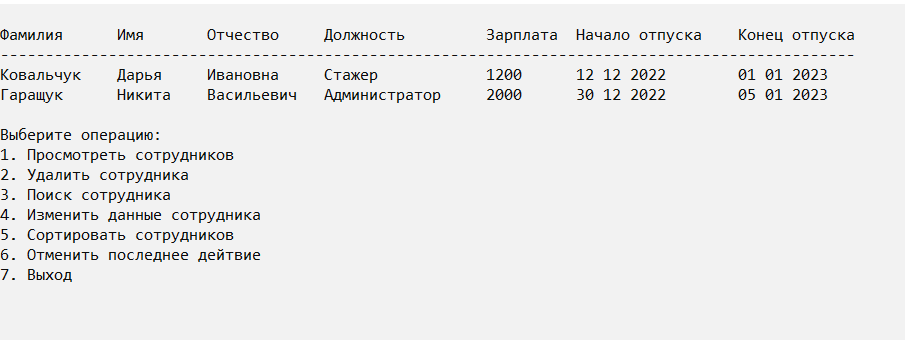


Рисунок 3 – Меню администратора и вывод сотрудников на экран

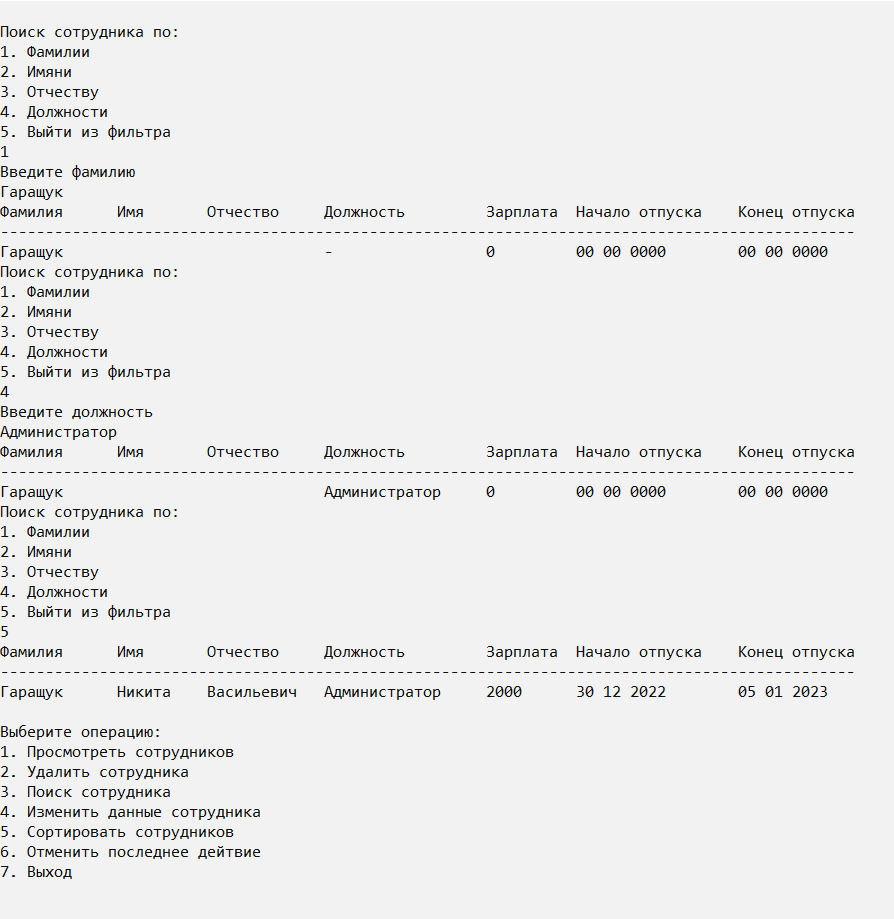


Рисунок 4 – Поиск сотрудника по параметрам

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения курсового проекта были выполнены первоначально заданные цели, а именно реализована программа авто паркинга с удобным интерфейсом. Так же были реализованы следующие алгоритмы: отмена последнего действия, сортировка и поиск объектов разных классов по определённым полям.

Информационные системы ежедневно встречаются в нашей жизни – дома, на работе, на улице, в транспорте. И сегодня только представить жизнь без таких систем довольно сложно! На предприятиях активно используются информационные системы управления. С помощью таких систем жизнь человечества существенно облегчается, это огромная и неоценимая помощь, ведь один или несколько человек не могут держать в голове или же на бумажных носителях данные, которые в компьютере занимают терабайты оперативной памяти. Однако просто хранить такую информацию – это мало, ее нужно систематизировать и адаптировать для удобного использования.

ОС – Windows 10

Среда разработки – Microsoft Visual Studio 2022

Минимальные системные требования:

Занимаемая память процессора – x мб

Время выполнения – x c

**СПИСОК ЛИТЕРАТОРЫ**

1. Объектно-ориентированное программирование на языке С++: учеб. пособие / Ю. А. Луцик, В. Н. Комличенко. – Минск : БГУИР, 2008.

2. Конструирование программ и языки программирования: метод. указания по К65 курсовому проектированию для студ. спец. I-40 02 01 “Вычислительные машины, системы и сети” для всех форм обуч. / сост. А. В. Бушкевич,   
А. М. Ковальчук, И. В. Лукьянова. – Минск : БГУИР, 2009.

**ПРИЛОЖЕНИЕ А**

Диаграмма классов

**ПРИЛОЖЕНИЕ Б**

Блок-схема метода saveInFile()

**ПРИЛОЖЕНИЕ В**

Блок-схема метода put()

**ПРИЛОЖЕНИЕ Г**

Код программы