**СОДЕРЖАНИЕ**

Содержание………………………………………………………………..1

Введение…………………………………………………………………...2

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

2 Обзор методов и алгоритмов решения поставленной задачи………..3

3 Функциональное проектирование…………………………………….23

3.1. Структура входных и выходных данных…………………………..23

3.2. Разработка диаграммы классов………………………………..……23

3.3. Описание классов……………………………………………………23

4 Разработка программных модулей……………………………………23

4.1. Разработка схем алгоритмов………………………………………..23

4.2. Разработка алгоритмов……………………………………………...23

Результат работы………………………………………………………....23

Заключение……………………………………………………………….23

Список литературы………………………………………………………23

Приложения…………………………………………………………...….23

**ВВЕДЕНИЕ**

C++ - компилируемый, структурированный, объектно-ориентированный язык программирования. На сегодняшний день, несмотря на немалый возраст языка C++, он остается одним из самых востребованных и популярных языков программирования в мире.

C++ поддерживает множество стилей программирования (процедурное программирование, абстракцию данных, объектно-ориентированное программирование и обобщенное программирование). Поэтому разработчик может сам выбрать, в каком стиле ему писать программу.

Одним из главных достоинств языка C++ является его быстрота. Этот язык программирования куда чаще других используется в олимпиадном программировании.

Отдельного упоминания заслуживает объектно-ориентированная часть языка. Широкий функционал, включающий в себя различные конструкторы, деструкторы, возможности перегрузки операторов, организация классов – все это заметно упрощает организацию кода и повышает его читаемость, ускоряет процесс разработки программ.

С целью поддержки принципов объектно-ориентированного программирования все объектно-ориентированные языки, включая С++, обеспечивают четыре характерных принципа: инкапсуляцию, полиморфизм, наследование и абстракцию.

Инкапсуляция — это программный механизм, который связывает данные с обрабатывающими их кодами и защищает и те, и другие от внешних воздействий и ошибочных действий.

Полиморфизм обозначает средство, позволяющее посредством единого интерфейса получить доступ к целому классу действий.

Наследование является процессом, который позволяет одному объекту приобретать свойства другого объекта.

Абстракция – это использование только тех характеристик объекта, которые с достаточной точностью представляют его в данной системе. Основная идея состоит в том, чтобы представить объект минимальным набором полей и методов и при этом с достаточной точностью для решаемой задачи

В современном мире язык C++ используется при разработке операционных систем, драйверов устройств, разнообразных прикладных систем.

Сегодня быть профессиональным программистом высокого класса означает быть компетентным в C++. Этот язык обеспечивает концептуальный фундамент, на который опираются другие языки программирования и многие современные средства обработки данных. Освоив C++, 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**

Программный модуль “Автосалон” должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. В программе должна быть предусмотрена возможность хранения информации в нескольких файлах, связанных определенным образом. По желанию можно узнать, где какой автомобиль обслуживается. 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 в программе классы контейнеров, итераторов и алгоритмов (свои и STL). Производить обработку исключительных ситуаций.

**2 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ**

Для записи и хранения информации об автомобилях и станциях обслуживания используется самостоятельно разработанный контейнер бинарное дерево. Этот контейнер имеет методы добавления, удаления и вывода элементов, поиска минимального значения дерева.

Для реализации системы регистрации используется контейнер map из стандартной библиотеки шаблонов. Данный контейнер представляет из себя словарь, в котором каждому ключу соответствует какое-то единственное значение.

В поле ключа заносится логин пользователя, а в поле значения пароль, после чего выполняется проверка на существование данного пользователя в зарегистрированных.

Используются алгоритмы поиска, сортировки, изменения полей контейнера. Сортировка выполнена при помощи перегрузки операторов больше(>) и меньше(<). При выборе сортировки по определенному полю, флаг сортировки устанавливается в нужное значение, после чего происходит пересборка дерева с учетом нового сравниваемого поля. Поиск реализован при помощи перегрузки оператора сравнения (==), который сравнивает только существующие поля. Изменение параметров достигается с помощью методов доступа к полям класса.

Также используется метод копирования бинарного дерева, который поэлементно заносит узлы другого контейнера в новый.

**3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ**

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

**3.1 Структура входных и выходных данных**

Таблица 3.1 - файл легковых автомобилей *pass.txt*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Марка авто | Цена($) | Вес(т) | Мощность двигателя(Л/с) | Количество  пассажирских мест | Объем багажника(л) |
| Audi A6 | 5000 | 1.8 | 150 | 5 | 50 |

Таблица 3.2 - файл спортивных автомобилей *sport.txt*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Марка авто | Цена  ($) | Вес  (т) | Мощность двигателя  (кВт/л) | Количество  пассажирских мест | Наличие багажника | Время разгона до 100 км/ч(с) |
| Bugatti Veyron | 100000 | 1.2 | 420 | 2 | Нет | 5 |

Таблица 3.3 - файл грузовых автомобилей *cargo.txt*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Марка авто | Цена  ($) | Вес  (т) | Мощность двигателя  (кВт/л) | Количество  пассажирских мест | Объем  Прицепа  (м^3) | Максимальная грузоподъемность  (т) |
| Hyundai HD - 120 | 30000 | 30 | 120 | 2 | 80 | 20 |

Таблица 3.4 - файл легковых автомобилей *vintage.txt*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Марка авто | Цена аренды($) | Вес(т) | Мощность двигателя(кВт/л) | Дата выпуска |
| Mercedes-Benz W100 | 50000 | 1.8 | 150 | 11.10.1982 |

Таблица 3.5 - файл спортивных автомобилей *cabriolet.txt*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Марка авто | Цена аренды($) | Вес(т) | Мощность двигателя  (кВт/л) | Наличие складной крыши |
| Ford Model T | 30000 | 1.2 | 420 | Нет |

Таблица 3.6 - файл грузовых автомобилей *limousin.txt*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Марка авто | Цена  ($) | Вес  (т) | Мощность двигателя  (кВт/л) | Количество  пассажирских мест | Объем  Прицепа  (м^3) | Максимальная грузоподъемность  (т) |
| Hyundai HD - 120 | 30000 | 30 | 120 | 2 | 80 | 20 |

Таблица 3.7 - файл автосервисов *autostation.txt*

|  |  |
| --- | --- |
| Средний чек($) | Адрес автосервиса |
| 110 | Ул. Ленина, 15 |

Таблица 3.8 - файл регистрации *registration.txt*

|  |  |
| --- | --- |
| Логин | Пароль |
| admin | admin |

**3.2 Разработка диаграммы классов**

Диаграмма классов для курсового проекта представлена в виде плаката (Приложение N)

**3.3 Описание классов**

Класс **Tree<Type> -** шаблонный класс контейнера бинарного дерева. Является основным классом программы и служит для хранения объектов прочих классов.

Описание полей класса:

Node<Type>\* root – указатель на главный корень дерева

Описание методов:

Tree() – конструктор контейнера

~Tree() – деструктор контейнера

Iterator Begin() – метод возвращающий итератор на начало контейнера

Iterator End() – метод возвращающий итератор на конец контейнера

Node<Type>\* PushNode(Node<Type>\* root, Type data, Node<Type>\* prev = nullptr) – метод вызова добавления элемента в контейнер

Node<Type>\* DeleteNode(Type data) – удаление элемента

void CallDeleteNode(Type data) – вызов метода удаления элемента

void AddNode(Type data) – добавление элемента

void ShowTree(Node<Type>\* root) – вывод контейнера

Node<Type>\* GetRoot() – получение указателя на корень дерева

void DeleteTree(Node<Type>\* root) – очистка всего контейнера

void ClrTreeRoot(Node<Type>\* root) – вызов удаления контейнера

Type\* GetRootInfo() - получение значения корня дерева

Node<Type>\* GetMinEl(Node<Type>\* root) – поиск минимального элемента контейнера

Класс **Node<Type>** - шаблонный класс элемента контейнера

Описание полей класса:

Node\* R – указатель на правый корень дерева

Node\* L – указатель на левый корень дерева

Node\* prev – указатель на родительский корень дерева

Type data – информация, хранящаяся в узле

Описание методов класса:

Node(Type\* \_data) – конструктор класса

~Node() – деструктор класса

Класс **Interface** – класс интерфейса, включающий в себя все основные меню работы с программой

Описание методов класса:

void CallTypeMenu(int selectedMode) – меню выбора типа товара

void CallModeMenu() – меню регистрации

void CallAdminMenu(Tree<Type> obj, const char\* filename) – меню работы для администратора

void CallUserMenu(Tree<Type> obj, const char\* fileName) – меню работы для пользователя

void CallAutoStationAdminMenu(Tree<AutoStation> obj) – меню работы для администратора с автостанцией

void CallAutoStationUserMenu(Tree<AutoStation> obj) – меню работы для пользователя с автостанцией

Класс **File** – базовый класс работы с файлами

Описание полей класса:

ifstream ifStream – поток для чтения из файла

ofstream ofStream – поток для записи в файл

Класс **TxtFile –** класс работы с текстовыми файлами

void OpenTxtFileOut(std::string fileName) – открытие текстового файла для записи

void OpenTxtFileIn(std::string fileName) – открытие текстового файла для чтения

void CloseTxtFileOut() – закрытие текстового файла для записи

void CloseTxtFileIn()– закрытие текстового файла для записи

bool IsEof() – проверка, достигнут ли конец файла

bool IsOpen() – проверка, открылся ли файл

void OutputTxtFile(Type& obj) – запись в текстовый файл

void InputTxtFile(Type& obj) – чтение из текстового файла

Класс **ExceptionType** – базовый класс для ошибок разного типа

Описание полей класса:

char exceptionType[BASE\_LEN] – тип ошибки

Описание методов класса:

ExceptionType(const char\* type) – конструктор класса

~ExceptionType() – деструктор класса

virtual void ShowError() = 0 – абсолютно виртуальная функция вывода ошибки на экран

Класс **InputException** - класс обработки ошибок ввода

Описание полей класса:

char exceptionInputName[BASE\_LEN] – сообщение чтобы помочь исправить ошибку

InputException(const char\* type, const char\* name) – конструктор с параметрами

~InputException() – деструктор класса

void ShowError() – вывод ошибки

Класс **ContainerException** - класс обработки ошибок контейнера

Описание полей класса:

char exceptionContainerName[BASE\_LEN] – сообщение чтобы помочь исправить ошибку

ContainerException(const char\* type, const char\* name) – конструктор с параметрами

~ContainerException() – деструктор класса

void ShowError() – вывод ошибки

Класс **Date** – класс для работы с датами

Описание полей класса:

int day – день

int month – месяц

int year – год

Описание методов класса:

Date() – конструктор класса

Date(Date& other) – деструктор класса

Date& operator=(Date& other) – перегрузка оператора =

friend void InputDate(Date& newDate) – метод обработки ввода даты

friend std::ostream& operator<<(std::ostream& out, Date& someDate) – перегрузка оператора ввода

friend std::ofstream& operator<<(std::ofstream& out, Date& obj) – перегрузка оператора вывода в файл

friend std::ifstream& operator>>(std::ifstream& in, Date& obj) – перегрузка оператора ввода в файл

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

bool operator!=(Date& other) – перегрузка оператора !=

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

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

int GetDay() – метод получения значения дня

Класс **AutoStation** – класс станций обслуживания

Описание полей класса:

char geoLocation[BASE\_LEN] – адрес

double avgCheckPrice – средний чек

Описание методов класса:

AutoStation() – конструктор

~AutoStation() – деструктор

void SetGeoLocation(const char\* newGeo) – установить адрес

void SetAvgCheckPrice(double newPrice); – установить средний чек

const char\* GetGeoLocation() – получить адрес

double GetAvgCheckPrice() – получить средний чек

AutoStation(AutoStation& obj) – конструктор копирования

AutoStation& operator=(AutoStation& obj) – перегрузка оператора =

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

bool operator!=(AutoStation& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, AutoStation& obj) – перегрузка оператор ввода

friend std::ostream& operator<<(std::ostream& out, AutoStation& obj) – перегрузка оператор вывода

friend std::ofstream& operator<<(std::ofstream& out, AutoStation& obj) – перегрузка оператор вывода в файл

friend std::ifstream& operator>>(std::ifstream& in, AutoStation& obj) – перегрузка оператор ввода из файла

void PrintTableHead() – вывод шапки таблицы

void SetFieldValue() – изменить поле

Класс **Car –** основной базовый класс автомобиля

Описание полей класса:

char carModel[BASE\_LEN] – модель автомобиля

double enginePower – мощность двигателя

double weight – вес автомобиля

int compareParam – флаг, регулирующий поле для перегрузки > и <

Описание методов класса:

Car() - конструктор

virtual ~Car() – виртуальный деструктор

void SetCompareParam(int otherParam) – метод установки флага

void SetCarModel(const char\* newModel) – метод установки модели

void SetEnginePower(double power) – метод установки мощности

void SetWeight(double newWeight) – метод установки веса

const char\* GetCarModel() – метод получения модели

double GetEnginePower() – метод получения мощности

double GetWeight() – метод получения веса

Car(Car& other) – конструктор копирования

Car& operator=(Car& obj) – перегрузка оператора =

friend std::istream& operator>>(std::istream& in, Car& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, Car& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, Car& obj) – перегрузка оператора вывода в файл

friend std::ifstream& operator>>(std::ifstream& in, Car& obj) – перегрузка оператора ввода из файла

void virtual PrintTableHead() – вывод шапки таблицы

Класс **CarsForSale –** класс автомобилей, выставленных на продажу

Описание полей класса:

double carPrice – цена автомобиля

Описание методов класса:

CarsForSale() - конструктор

~CarsForSale() - деструктор

void SetCarPrice(double newRentPrice) – метод установки цены

double GetCarPrice() – метод получения цены

CarsForSale(CarsForSale& other) – конструктор копирования

CarsForSale& operator=(CarsForSale& obj) – перегрузка оператора =

friend std::istream& operator>>(std::istream& in, CarsForSale& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, CarsForSale& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, CarsForSale& obj) – перегрузка оператора вывода в файл

friend std::ifstream& operator>>(std::ifstream& in, CarsForSale& obj) – перегрузка оператора ввода из файла

void virtual PrintTableHead() – вывод шапки таблицы

Класс **RentCars –** класс автомобилей для сдачи в аренду

Описание полей класса:

double rentPrice – цена автомобиля

Описание методов класса:

RentCars() - конструктор

~ RentCars() - деструктор

void SetRentPrice(double newPrice) – метод установки цены

double GetRentPrice() – метод получения цены

RentCars(RentCars& other) – конструктор копирования

RentCars& operator=( RentCars& obj) – перегрузка оператора =

friend std::istream& operator>>(std::istream& in, RentCars& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, RentCars& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, RentCars& obj) – перегрузка оператора вывода в файл

friend std::ifstream& operator>>(std::ifstream& in, RentCars& obj) – перегрузка оператора ввода из файла

void virtual PrintTableHead() – вывод шапки таблицы

Класс **PassCars –** класс пассажирских автомобилей, доступных к покупке

Описание полей класса:

int seatsNumber – число сидений

double trunkVolume – объем багажника

Описание методов класса:

PassCars() - конструктор

~PassCars() - деструктор

void SetSeatsNumber(int newSeatsNum) – метод установки числа сидений

void SetTrunkVolume(double newVolume) – метод установки объема багажника

int GetSeatsNumber() – метод получения числа сидений

double GetTrunkVolume() – метод получения объема багажника PassCars(PassCars& other) – конструктор копирования

PassCars& operator=(PassCars& obj) – перегрузка оператора =

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

bool operator!=(PassCars& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, PassCars& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, PassCars& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, PassCars& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, PassCars& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree<PassCars> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

Класс **SportCars –** класс спортивных автомобилей, доступных к покупке

Описание полей класса:

double accelTime – скорость разгона до 100 км/ч

Описание методов класса:

SportCars()- конструктор

~SportCars() - деструктор

void SetAccelTime(int newAccelTime) – метод установки быстроты разгона

double GetAccelTime() – метод получения быстроты разгона

SportCars(SportCars& other) – конструктор копирования

SportCars& operator=( SportCars& obj) – перегрузка оператора =

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

bool operator!=( SportCars& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, SportCars& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, SportCars& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, SportCars& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, SportCars& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree< SportCars> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

Класс **CargoCars –** класс спортивных автомобилей, доступных к покупке

Описание полей класса:

double maxLoad – максимальная грузоподъемность

double trailerVolume – объем прицепа

Описание методов класса:

CargoCars()- конструктор

~CargoCars() - деструктор

void SetAccelTime(int newMaxLoad) – метод установки грузоподъемности

void SetAccelTime(int newTrailerVolume) – метод установки объема прицепа

double GetMaxLoad () – метод получения грузоподъемности

double GetTrailerVolume () – метод получения объема прицепа

SportCars(SportCars& other) – конструктор копирования

SportCars& operator=( SportCars& obj) – перегрузка оператора =

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

bool operator!=( SportCars& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, SportCars& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, SportCars& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, SportCars& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, SportCars& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree< SportCars> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

Класс **Cabriolet –** класс кабриолетов, доступных для взятия в аренду

Описание полей класса:

bool isRoof – наличие складной крыши

Описание методов класса:

Cabriolet()- конструктор

~Cabriolet() - деструктор

void SetIsRoof (int roof) – метод установки грузоподъемности

bool GetTrailerVolume () – метод получения объема прицепа

Cabriolet(Cabriolet& other) – конструктор копирования

Cabriolet& operator=( Cabriolet& obj) – перегрузка оператора =

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

bool operator!=( Cabriolet& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, Cabriolet& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, Cabriolet& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, Cabriolet& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, Cabriolet& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree<Cabriolet> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

Класс **VintageCars –** класс винтажных машин, доступных для взятия в аренду

Описание полей класса:

Date productionDate – время выпуска автомобиля

Описание методов класса:

VintageCars()– конструктор

~VintageCars() – деструктор

void SetProductionDate (int roof) – метод установки времени выпуска

Date GetProductionDate () – метод получения времени выпуска

VintageCars (VintageCars& other) – конструктор копирования

VintageCars& operator=( VintageCars& obj) – перегрузка оператора =

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

bool operator!=( VintageCars& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, VintageCars& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, VintageCars& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, VintageCars& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, VintageCars& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree< VintageCars> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

Класс **Limousin –** класс лимузинов, доступных для взятия в аренду

Описание полей класса:

int seatsNumber – время выпуска автомобиля

Описание методов класса:

Limousin()– конструктор

~ Limousin() – деструктор

void SetSeatsNumber (int seatsN) – метод установки времени выпуска

Date GetSeatsNumber () – метод получения времени выпуска

Limousin (Limousin& other) – конструктор копирования

Limousin& operator=( Limousin& obj) – перегрузка оператора =

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

bool operator!=( Limousin& other) – перегрузка оператора !=

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

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

friend std::istream& operator>>(std::istream& in, Limousin& obj) – перегрузка оператора ввода

friend std::ostream& operator<<(std::ostream& out, Limousin& obj) – перегрузка оператора вывода

friend std::ofstream& operator<<(std::ofstream& out, Limousin& obj) – перегрузка оператора ввода в файл

friend std::ifstream& operator>>(std::ifstream& in, Limousin& obj) – перегрузка оператора вывода из файла

void virtual PrintTableHead() – вывод шапки таблицы

void SubMenu(Tree<Limousin> obj) – вывод подменю для фильтра

void SetFieldValue() – изменение определенного поля объекта

int SortData() – сортировка по определенному полю

**3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

Текст

Текст

Текст

* 1. **Разработка схем алгоритмов**

Текст

Текст

Текст

* 1. **Разработка алгоритмов**

Текст

Текст

Текст

1. **РЕЗУЛЬТАТ РАБОТЫ**

Текст

Текст

Текст

**ЗАКЛЮЧЕНИЕ**

Текст

Текст

Текст

**СПИСОК ЛИТЕРАТОРЫ**

Текст

Текст

Текст

**ПРИЛОЖЕНИЯ**

Текст

Текст

Текст