TRANSPORT AND TELECOMMUNICATION INSTITUTE

ENGINEERING FACULTY

**CAr Database**

COURSE PROJECT

in Programming

by

Daniels Bleiders

st81712

3202BNA

Riga 2023

# Table of contents

Contents

[Table of contents 2](#_Toc136428013)

[Introduction 3](#_Toc136428014)

[Design 4](#_Toc136428015)

[Testing 13](#_Toc136428016)

[Conclusion 15](#_Toc136428017)

[Pseudocode 16](#_Toc136428018)

# Introduction

The subject area of this project is the development of a Car database application in C++ language. The Car Database is a software application designed to manage car information for an owner. It allows users to store, retrieve, update, and delete car records.

The car database is very important nowadays. It allows us to organise all the necessary information about cars and store them ensuring that this information is secured and won’t be lost. Also, it saves a lot of time on managing cars and organising all the necessary information. That enhances productivity in choosing the needed car.

The program uses some data structures to achieve its functionality. Two main structs are Car and Owner. The Car class stores information about individual car information, such as their Id, name, their owner Id. The Owner class stores information about owner, such as owner Id, name, number of cars.

To connect all structs together and to establish belongings of each struct, the program uses unique Id to owner and car. This allows for efficient data retrieval and manipulation.

The program offers various functionalities to manipulate with data. Users can add new cars and owners to the system, providing essential information such as names and id. The program allows users to print desired information, such as sorted and filtered cars and owners. The program enables users to delete cars and owners providing a comprehensive data management system.

In summary, the Car database program in C++ is for effective management. Through the utilization of appropriate data structures, relationships, and functionalities, the program offers a reliable and user-friendly solution. The following sections will provide more detailed information about the implementation and features of the program, showcasing its potential to enhance productivity and organization.

# Design

UI is developed using simple style in console command and while statement.

Time to go through all the files in the program.

**Libraries.h**

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <algorithm>

#include <cstring>

using namespace std;

This file runs all the necessary libraries for the program.

After we define three classes:

**Car.h**

#pragma once

#include "Libraries.h"

struct Car

{

private:

    unsigned car\_id = 0, owner\_id = 0;

    string car\_name;

    static const unsigned Max\_cars = 100;

public:

    //getters

    unsigned get\_car\_id();

    //func for creating deleting and printing deps

    void create\_cars(Car\* cars, unsigned& num\_car, const unsigned Max\_cars);

    void delete\_cars(Car\* cars, unsigned& num\_car, unsigned Max\_cars);

    void print\_cars(Car\* cars, unsigned num\_car);

    //func for choosing a dep

    unsigned choose\_Car(Car\* cars, unsigned& num\_car, const unsigned Max\_cars);

    unsigned choose\_Car\_id(Car\* cars, unsigned& num\_car, const unsigned Max\_cars);

    //func for setting and removing deps

    void set\_car\_to\_owner(Car\* cars, unsigned num\_car, unsigned owner\_id, unsigned Max\_cars);

    void remove\_car\_from\_owner(Car\* cars, unsigned num\_car, unsigned owner\_id);

    void filter\_cars\_by\_name(Car\* cars, unsigned num\_car, string \_car\_name);

    //func for sort

    void sort\_cars\_ascend(Car\* cars, unsigned num\_car);

    void sort\_cars\_descend(Car\* cars, unsigned num\_car);

    //functiuons to calculate Car count for Owner

    unsigned calculate\_car\_counts(Car\* cars,unsigned num\_car, unsigned \_owner\_id);

    //read and write to file

    void write\_Car\_to\_file(Car\* cars, unsigned num\_car);

    void read\_Car\_to\_file(Car\* cars, unsigned& num\_car);

};

Class department shows department id, its name, number of members and its company Id.

1. **Unsigned Get\_car\_id()** – This function takes a car as input and searches for the corresponding car Id and returns its Id.
2. **Void Create\_car(**Car\* cars, unsigned& num\_car, const unsigned Max\_cars) – This function creates new car to the database. You can set its name and Id. After that function adds object to the array cars.
3. **Void Delete\_car(**Car\* cars, unsigned& num\_car, unsigned Max\_cars) – This function asks you to choose one of the cars and then deletes it from the array of cars. You choose one of the cars by **Choose\_car** function, which will be described in the next step.
4. **Void Print\_car** (Car\* cars, unsigned num\_car) – This function prints all existing cars in the database and the information about it.
5. **Unsigned Choose\_car (**Car\* cars, unsigned& num\_car, const unsigned Max\_cars) – This function allows you to choose one car in the enumerated list of all cars from the cars array and returns the number of this car from the list.
6. **Unsigned Choose\_car\_id(**Car\* cars, unsigned& num\_car, const unsigned Max\_car) – This function allows you to choose one car in the enumerated list of all cars from the cars array and returns the id of the selected car.
7. **Void Set\_car\_to\_owner(**Car\* cars, unsigned num\_car, unsigned owner\_id, unsigned Max\_cars) – This function calls **Choose\_car** function and takes the needed car from the array, and then gives to this car owner id.
8. **Void remove\_car\_from\_owner**(Car\* cars, unsigned num\_car, unsigned owner\_id, unsigned Max\_cars) – This function creates filtered cars array to pickup those cars, who got the same owner\_id as the \_owner\_id we got to our function. After that we create car\_num variable and assign it a function **Choose\_car** which takes filtered cars array. Then we create variable car\_id\_to\_remove and assign to it one of the filtered cars id with the index of car\_num. Then we go through all the array of cars and assign its owner Id to 0;
9. **Void filter\_cars\_by\_name(** Car\* cars, unsigned num\_car, string \_car\_name) – we create array of filtered cars and watch, if there is a car who got the needed name from car array, then we assign this car to array of filtered cars and print this array.
10. **Void sort\_cars\_ascend(** Car\* cars, unsigned num\_car) – we use the function sort from algorithm library and sorts them by first letter from beginning to end using lambda function, which takes two member objects and compare if the first object is Unicode is bigger than second. Lambda function is a bool type function.
11. **Void sort\_cars\_descend(** Car\* cars, unsigned num\_car) – we use the function sort from algorithm library and sorts them by salary from beginning to end using lambda function, which takes two member objects and compare if the first object number of members is smaller than second. Lambda function is a bool type function.
12. **Unsigned calculate\_car\_counts**(Car\* cars,unsigned num\_car, unsigned \_owner\_id) – function to go through car array, and calculate car count if cars owner id is equal to owner id.
13. **Void write\_car\_to\_file(**Car\* cars, unsigned num\_car) – This function initialise string fn, which will be the name of the file, and then initialise ofstream ofs, which will be binary type. Then program goes through the array. Then the program converts all the field of object to char\*, if the field is type string, program creates size\_t variable, convecrts it to char\*, and then converts string to cstring with the size of size\_t. Then ofs closes.
14. **Void read\_car\_to\_file(**Car\* cars, unsigned& num\_car) – This function clears the array, initialise string fn, which will be the name of the file, and then initialise ifstream ifs, which will be binary type. Then it goes to the end of file and reads all the variables, converts them back from char\*. If the field is type string, program creates size\_t variable, convecrts it to char\*, then creates new string, reads here all the information from size\_t and then moves it to the object. Then object is added to array and ifs closes.

**Owner.h**

#pragma once

#include "Libraries.h"

struct Owner

{

private:

    unsigned owner\_id = 0, num\_car = 0;

    string owner\_name = "";

public:

    //func for creating, deleting and printing owners

    void create\_owners(Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners);

    void delete\_owners(Owner\* owners, unsigned& num\_owners, unsigned Max\_owners);

    void print\_owners(Owner\* owners, unsigned num\_owners);

    //func for choosing one Owner by name and id

    unsigned choose\_Owner(Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners);

    unsigned choose\_Owner\_id(Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners);

    //func for sorting

    void sort\_owners\_ascend(Owner\*owners, unsigned num\_owners);

    void sort\_owners\_descend(Owner\* owners, unsigned num\_owners);

    //func for setting number of cars to Owner

    void update\_car\_count(Owner\* owners, unsigned num\_owners, unsigned \_owner\_id, unsigned car\_count);

    //read and write to file

    void write\_Owner\_to\_file(Owner\* owners, unsigned num\_owners);

    void read\_Owner\_to\_file(Owner\* owners, unsigned &num\_owners);

};

Company class shows company id, its name and number of members and departments in it.

1. **Void Create\_owner (**Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners) – This function creates new owner to the database. You can set its name and Id. After that function adds object to the array owners.
2. **Void Delete\_owner(**Owner\* owners, unsigned& num\_owners, unsigned Max\_owners) – This function asks you to choose one of the owners and then deletes it from the array of owners. You choose one of the owner by **Choose\_owner** function, which will be described in the next step.
3. **Void Print\_owner**(Owner\* owners, unsigned num\_owners) – This function prints all existing owners in the database and the information about it.
4. **Unsigned Choose\_owner(**Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners) – This function allows you to choose one owner in the enumerated list of all owners from the owners array and returns the number of this owner from the list.
5. **Unsigned Choose\_owner\_id** **Owner\* owners, unsigned& num\_owners, const unsigned Max\_owners**) – This function allows you to choose one owner in the enumerated list of all ownerss from the owners array and returns the id of the selected owner.
6. **Void sort\_owners\_ascend(** Owner\*owners, unsigned num\_owners) – we use the function sort from algorithm library and sorts them by first letter from beginning to end using lambda function, which takes two member objects and compare if the first object Unicode is smaller than second. Lambda function is a bool type function.
7. **Void sort\_owners\_descend(** Owner\*owners, unsigned num\_owners) – we use the function sort from algorithm library and sorts them by first letter from beginning to end using lambda function, which takes two member objects and compare if the first object is Unicode is bigger than second. Lambda function is a bool type function.
8. **Void Update\_car\_count(**Owner\* owners, unsigned num\_owners, unsigned \_owner\_id, unsigned car\_count) **–** This function goes through the owners array and searches the owner with the needed id and then assigns to it number of cars.
9. **Void write\_owner\_to\_file(**Owner\* owners, unsigned num\_owners) – This function initialise string fn, which will be the name of the file, and then initialise ofstream ofs, which will be binary type. Then program goes through the array. Then the program converts all the field of object to char\*, if the field is type string, program creates size\_t variable, convecrts it to char\*, and then converts string to cstring with the size of size\_t. Then ofs closes.
10. **Void read\_owner\_to\_file(**Owner\* owners, unsigned num\_owners) – This function clears the array, initialise string fn, which will be the name of the file, and then initialise ifstream ifs, which will be binary type. Then it goes to the end of file and reads all the variables, converts them back from char\*. If the field is type string, program creates size\_t variable, convecrts it to char\*, then creates new string, reads here all the information from size\_t and then moves it to the object. Then object is added to array and ifs closes.

# Testing

Case 1: Correct input

I create car and owner, Then I add car to owner ,I’ve checked if the car got the needed id and number of cars in owner. Everything was allright. Then I wrote data to file and read. Program worked correctly.

Case 2: Incorrect input in unsigned int

If I try to input a letter in Id number, I’ll get “Exception caught Error, incorrect Id input”.

Case 3: Remove car from an empty owner.

If I press remove car and there are no downera connected to car, I’ll get: “There is no cars for owner.”

Case 4: Set car to empty array of owners.

I’ll get “Firstly you should create owner”

Case 5: trying to delete car connected to owner.

I’ll get: “Firstly you should remove car from owner”.

Case 6: Trying to sort an array of 1 owner.

Will get “You should have at least 2 owners to sort”

Case 7: Trying to read non existing file

Delete file owners.bin and trying to read it will get to “Error, cannot read a file!”

Case 8: Choosing non existing ownert:

Will get “Exception caught: Error, owner\_num cannot exceed the given number of owners!”

Case 9: Entering negative owner number

Will get “Exception caught: Error, owner\_num cannot exceed the given number of owners!”

Case 10: Entering incorrect option in menu.

If a enter ‘a’ in the menu, I’ll get “Error, no such an option available!”

Case 11: Print an empty list of owners.

I’ll get “Firstly you should create owner.”

# Conclusion

In conclusion I can say that the program works normally and catches all the incorrect inputs and strange result, such as letters in id, no existing files, trying to connect non existing objects and to sort one object. Program needs better quality of life improvements such as not terminating process while you choose incorrect option and make a better menu for the program, it’s very big and not so understandable. Also need to make a more convenient list of programs for quality-of-life improvement. Despite that program works normally and completes its functionality, it will be usable for better management and saving time.

# Pseudocode

1. Print cars(car, num\_cars)

Count

For I in num\_cars do

Count + cars[i]car\_id + cars[i]car\_name + cars[i]car\_name+ coms[i]owner\_id

Count ++

1. Create cars(cars, num\_cars, max\_cars)

Car car

If num\_cars == max\_cars do

Error, max\_cars reached.

“Enter car id”

If type car.car\_id do

Else

“Error, incorrect id.”

“Enter car name

Type car.car\_name

Cars += car.

Num\_cars++