Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Fundamentals of Computer Programming

Cars

---

author	Szymon Uszok
instructor	Wojciech Dudzik
year	2019/2020
lab group	Friday, 13:45 – 15:15
deadline	2020-01-27

---

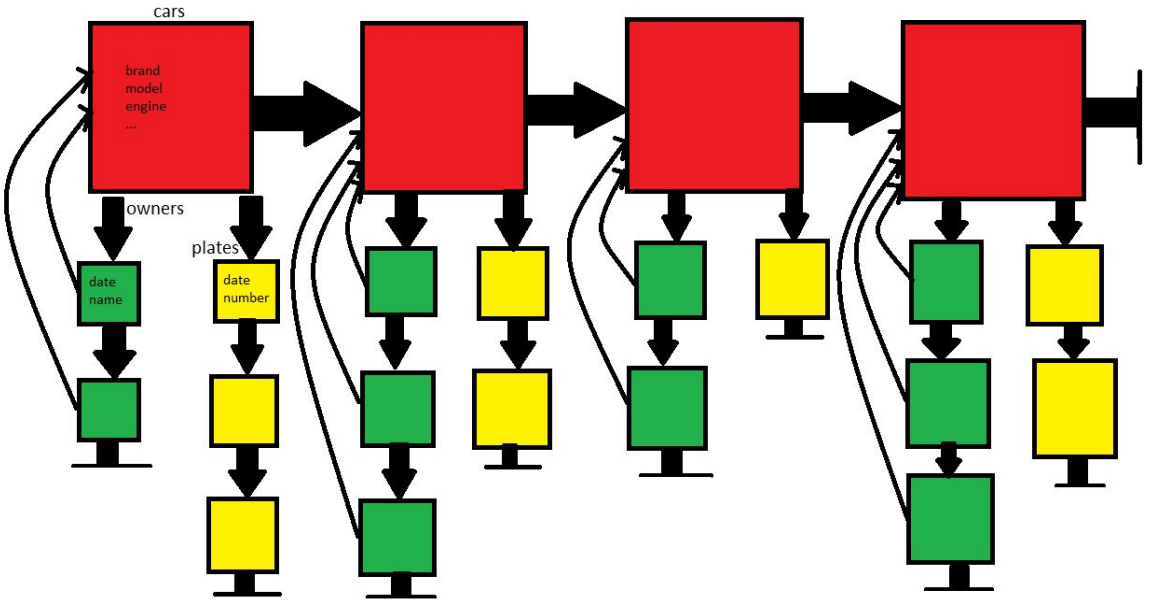
1 Project's topic

The program elaborates a report for each owner with all cars possessed now and in the past with car details and possession period. This is a command line program with switches:
-i input file
-o output file

2 Analysis of the task

The task focuses on reading details from an input file and creating a report.

2.1 Data structures



The image shows how my structures are connected. The main body of my structures is a cars list. Each car has some car details (brand, model, etc.) and pointers to an owners list and a plates list. Each owner has a pointer to his car.

3 External specification

This is a command line program. The program requires the user to specify the names of input and output files. Put input file name after -i switch and output file name after -o switch, eg:

program -i input.txt -o output.txt
program -o output.txt -i input .txt

Both files are text files. The switches may be provided in any order. Program called with incorrect parameters prints an error message:

"Please provide correct arguments for switches -i and -o"

4 Internal specification

The program is implemented with structural paradigm. User interface is separated from program's logic.

4.1 Program overview

The main function calls checkSwitches that verifies parameters of the program. If the verification is negative, an appropriate message is printed. In case of positive verification, data are read with getInfoFromLine function. The function opens a file stream, reads details from the stream, and inserts car, owner or plate details into singly-linked lists with add function. Finally the program generates a owners report and removes the singly-linked car list from the memory.

4.2 Description of types and functions

Description of types and functions is moved to the appendix.

5 Testing

The program has been tested with various types of files. Incorrect files are detected and an error message is printed. An empty input file does not cause failure – an empty output file is created. Maximal number value in an input file depends on a compiler (int may be implemented as 2 or 4 B type). The program has no memory leaks.

6 Conclusions

The program reads a complicated input file and adds new cars, owners and plates to appropriate singly-linked lists. The lists are then used to create a owners report with all of their cars details. The most challenging task was creating functions for reading the input file's contents.

Cars

Generated by Doxygen 1.8.17

1 README 1

2 Class Index 2

2.1 Class List 2

3 File Index 2

3.1 File List 2

4 Class Documentation 2

4.1 car Struct Reference 2

4.1.1 Detailed Description 3

4.2 owner Struct Reference 3

4.2.1 Detailed Description 4

4.3 plate Struct Reference 4

4.3.1 Detailed Description 4

5 File Documentation 4

5.1 functions.h File Reference 4

5.1.1 Function Documentation 5

5.2 structures.h File Reference 11

Index 13

1 README

Use "g++ main.cpp functions.cpp memory/nvwa/bool_array.cpp memory/nvwa/file_line_reader.cpp memory/nvwa/mmap↩
_reader_base.cpp memory/nvwa/debug_new.cpp memory/nvwa/static_mem_pool.cpp memory/nvwa/mem_pool↩
_base.cpp -o cars" to compile.

Fundamentals of Computer Programming (18) Cars (max grade: 4.0)

1. The objective of the project is to practise the implementation and application of dynamic data structures (eg lists, trees etc) and memory management. Using of dynamic data structures is a sine qua non condition.
2. Application of STL containers (eg vector, list etc) does not fulfil the condition stated in p. 1.
3. Dynamically allocated arrays do not fulfil the condition stated in p. 1.
4. The std::string type can only be used for strings (eg file names). It is not allowed to keep sequences of data (eg numbers) as strings.
5. A data structure has to accepted by a laboratory instructor before implementation.
6. Source code has to be split into header (∗.h) and source (∗.cpp) files.
7. All functions have to be commented in the doxygen style.
8. Any permutation of command line switches has to be handled.

A town hall keeps a list of cars and their owners. Each car has a record:

— car — brand: skoda model: citigo year of construction: 2010 engine capacity: 1300 engine number: 45VG342↩ EW34 VIN number: 324987TR854FR321 first registration: 2010-05-03

vehicle registration plates: 2010-05-03 SG12345 2012-11-03 ZS434U 2014-12-05 NOL3421

owners: 2010-05-03 Zbigniew Szpak 2010-12-12 Anna Romaryn 2012-10-04 Teodor Puma 2014-11-15 Elzbieta Fiuk, Grzegorz Fiuk

The program elaborates a report for each owner with all cars possessed now and in the past with car details and possession period. This is a command line program with switches: -i input file -o output file

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

car 2

owner 3

plate 4

3 File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

functions.h 4

structures.h 11

4 Class Documentation

4.1 car Struct Reference

`#include <structures.h>`

Public Attributes

- std::string brand
  car brand
- std::string model
  car model
- std::string constructionYear
  car year of construction
- std::string engineCapacity
  car engine capacity
- std::string engineNumber
  car engine number
- std::string VIN
  car VIN number
- std::string firstRegistration
  car first registration date
- plate * plates = nullptr
  pointer to car plates
- owner * owners = nullptr
  pointer to car owners
- car * next = nullptr
  next car in a list

4.1.1 Detailed Description

A car of a singly linked list.

The documentation for this struct was generated from the following file:

- structures.h

4.2 owner Struct Reference

```
#include <structures.h>
```

Public Attributes

- std::string date
  owner date
- std::string name
  owner name
- car * car
  pointer to owner's car
- owner * next = nullptr
  next owner in a list

4.2.1 Detailed Description

An owner of a singly linked list.

The documentation for this struct was generated from the following file:

- structures.h

4.3 plate Struct Reference

`#include <structures.h>`

Public Attributes

- std::string date
  plate date
- std::string number
  plate number
- plate * next = nullptr
  next plate in a list

4.3.1 Detailed Description

A plate of a singly linked list.

The documentation for this struct was generated from the following file:

- structures.h

5 File Documentation

5.1 functions.h File Reference

`#include "structures.h"`

Functions

- bool checkSwitches (int argc, char ∗argv[ ])
- std::string switchArgument (std::string switchName, int argc, char ∗argv[ ])
- car ∗ lastCar (car ∗p)
- void newCar (car ∗&p)
- void getCarInfo (std::string line, car ∗&p)
- plate ∗ lastCarLastPlate (car ∗p)
- void newPlate (std::string line, car ∗&p)
- owner ∗ lastCarLastOwner (car ∗p)
- void newOwner (std::string line, car ∗&p)
- void inputMode (std::string line, int &mode)
- void getInfoFromLine (std::ifstream &input, car ∗&p, std::string line, int &mode)
- std::string ownerPeriodOfTime (owner ∗p)
- void outputPlatesDetails (std::ofstream &output, car ∗p)
- void outputCarDetails (std::ofstream &output, car ∗p)
- void deletePlates (plate ∗&p)
- void deleteOwners (owner ∗&p)
- void deleteCars (car ∗&p)
- void newReportedOwner (std::string name, owner ∗&p)
- void createReport (std::ofstream &output, car ∗p)

5.1.1 Function Documentation

5.1.1.1 checkSwitches()

```
bool checkSwitches (
        int argc,
        char * argv[] )
```

The function checks if the user has put correct switch arguments.

Parameters

argc	number of arguments
argv	array of arguments

Returns

true if the switches are correct

5.1.1.2 createReport()

```
void createReport (
        std::ofstream & output,
        car * p )
```

The function creates a report of owners and their cars.

Parameters

output	file where the report is saved at
p	pointer to the car list

5.1.1.3 deleteCars() `void deleteCars (`

```
car *& p )
```

The function deletes a car list.

Parameters

p	pointer to the car list

5.1.1.4 deleteOwners() `void deleteOwners (`

```
owner *& p )
```

The function deletes an owner list.

Parameters

p	pointer to the owner list

5.1.1.5 deletePlates() `void deletePlates (`

```
plate *& p )
```

The function deletes a plate list.

Parameters

p	pointer to the plate list

5.1.1.6 getCarInfo() `void getCarInfo (`

```
std::string line,
car *& p )
```

The function reads a line containing car details and add details to the last car in a list.

Parameters

line	line of car details
p	pointer to the car list

5.1.1.7 getInfoFromLine()

```
void getInfoFromLine (
        std::ifstream & input,
        car *& p,
        std::string line,
        int & mode )
```

The function creates a new car, an owner or a plate to a car list, depending on the mode variable.

Parameters

input	file containing all car details
p	pointer to the car list
line	line of car, owner or plate details
mode	variable which stores the type of car details

5.1.1.8 inputMode()

```
void inputMode (
        std::string line,
        int & mode )
```

The function reads a line and decides what type of car details it contains.

Parameters

line	line of car details
mode	variable which stores the type of car details

5.1.1.9 lastCar()

```
car* lastCar (
        car * p )
```

The function finds the last car in a list.

Parameters

p	pointer to the car list

Returns

pointer to the last car

5.1.1.10 lastCarLastOwner() `owner* lastCarLastOwner ( car * p )`

The function finds the last owner of the last car in a list.

Parameters

p	pointer to the car list

Returns

pointer to the last owner of the last car

5.1.1.11 lastCarLastPlate() `plate* lastCarLastPlate ( car * p )`

The function finds the last plate of the last car in a list.

Parameters

p	pointer to the car list

Returns

pointer to the last plate of the last car

5.1.1.12 newCar() `void newCar ( car *& p )`

The function adds a new car to a car list.

Parameters

p	pointer to the car list

5.1.1.13 newOwner() `void newOwner (`

```
        std::string line,
        car *& p )
```

The function reads a line containing owner details and adds a new owner to a car list.

Parameters

line	line of owner details
p	pointer to the car list

5.1.1.14 newPlate() `void newPlate (`

```
        std::string line,
        car *& p )
```

The function reads a line containing plate details and adds a new plate to a car list.

Parameters

line	line of plate details
p	pointer to the car list

5.1.1.15 newReportedOwner() `void newReportedOwner (`

```
        std::string name,
        owner *& p )
```

The function adds a new owner to a reported owners list.

Parameters

name	name of the new owner
p	pointer to the owner list

5.1.1.16 outputCarDetails() `void outputCarDetails (`

```
        std::ofstream & output,
        car * p )
```

The function outputs owner details.

Parameters

output	file where the output is saved at
p	pointer to the car list

5.1.1.17 outputPlatesDetails()

```
void outputPlatesDetails (
        std::ofstream & output,
        car * p )
```

The function outputs plate details.

Parameters

output	file where the output is saved at
p	pointer to the car list

5.1.1.18 ownerPeriodOfTime()

```
std::string ownerPeriodOfTime (
        owner * p )
```

The function reads a owner's ownership details.

Parameters

p	pointer to the owner list

Returns

period of ownership

5.1.1.19 switchArgument()

```
std::string switchArgument (
        std::string switchName,
        int argc,
        char * argv[] )
```

The function reads the program's arguments.

Parameters

switchName	name of the switch
argc	number of arguments
argv	array of arguments

Returns

argument of the switch

5.2 structures.h File Reference

Classes

- struct plate
- struct owner
- struct car

Index

car, 2
checkSwitches
    functions.h, 4
createReport
    functions.h, 4

deleteCars
    functions.h, 5
deleteOwners
    functions.h, 5
deletePlates
    functions.h, 5

functions.h, 4
    checkSwitches, 4
    createReport, 4
    deleteCars, 5
    deleteOwners, 5
    deletePlates, 5
    getCarInfo, 5
    getInfoFromLine, 6
    inputMode, 6
    lastCar, 6
    lastCarLastOwner, 7
    lastCarLastPlate, 7
    newCar, 7
    newOwner, 7
    newPlate, 8
    newReportedOwner, 8
    outputCarDetails, 8
    outputPlatesDetails, 9
    ownerPeriodOfTime, 9
    switchArgument, 9

getCarInfo
    functions.h, 5
getInfoFromLine
    functions.h, 6

inputMode
    functions.h, 6

lastCar
    functions.h, 6
lastCarLastOwner
    functions.h, 7
lastCarLastPlate
    functions.h, 7

newCar
    functions.h, 7
newOwner
    functions.h, 7
newPlate
    functions.h, 8
newReportedOwner
    functions.h, 8

outputCarDetails
    functions.h, 8
outputPlatesDetails
    functions.h, 9
owner, 3
ownerPeriodOfTime
    functions.h, 9

plate, 3

structures.h, 10
switchArgument
    functions.h, 9