Необходима документация обхващаща курсов проект по „Програмни езици“

Изготвил:

Васил Стойчев Стойков

Факултет: ФКСТ

Факултетен номер: 121222094

# Съдържание:

## Преглед на основни функционалност

## Употреба и подготовка за употреба

## Инструкции за употреба

## Структура на кода

## Подробна информация за някои от по-важните функции

## Допълнителна информация

## Преглед на основните функционалности

Програмата представлява конзолно приложение, което е приложимо за управление на всякакъв вид организиран транспорт, който включва редовни маршрути, като самото приложение би било използваемо и за нередовни такива. То може да запазва информация за превозни средства и маршрути, като може отделните превозни средства да бъдат отреждани към определени от потребителя маршрути и тази информация отново да бъде запазена в паметта. Информацията, която се запазва за превозните средства е както следва: марка, модел, година на производство, брой места, товароносимост, разход на гориво, както и идентификационен номер, който се задава автоматично и е уникален за всяко превозно средство, което прави информацията по лесна за разбиране при наличие на множество еднакви превозни средства. Информацията, която се запазва за маршрутите е: спирките, на които превозното средство спира (за слизане и качване на пътници или за товарене и разтоваряне на товари), дължина на маршрута и повторенията на маршрута на определена база (по подразбиране базата е дневна, тъй като приложението е разработено главно за често повтарящи се маршрути, но това не пречи то да бъде използвано за други подобни цели). Тук отново имаме наличие на идентификационен номер, който се използва повече за улеснено навигиране из менютата, но отново може да имаме наличие на два подобни маршрута с леки разлики помежду си и с различни превозни средства отредени към тях. При присвояването на превозно средство от маршрут потребителят вижда колко гориво би било необходимо на превозното средство за изпълнението на съответната задача след което той може да избере дали да откаже. Като цяло употребата на приложението е интуитивна и разбираема. Всяко подаване на данни е свързано с проверяване на данните, което го прави ненаказуемо при грешки. По всяко време потребителят може да напише “STOP”, при което той ще се върне едно меню нагоре. При използването му потребителят бива насочван към това какво точно се очаква от него, за да работи всичко безпроблемно.

## Употреба и подготовка за употреба

За употребата на конзолното приложение не е необходима никаква подготовка. Нужно е само цялата папка да бъде изтеглена, след което може да се премине към отварянето на “main.exe”, което е основния файл на приложението. След това ще се отвори конзола от където потребителят може да започне работа. От друга страна ако потребител или разработчик иска да отвори Source code-а и да прави промени ще му бъде необходимо да компилира наново всички класове във main файла. Това може да стане след отваряне на терминал, навигиране до папката на приложението и изпълнението на следния prompt:

g++ main.cpp src/core/route.cpp src/core/vehicle.cpp src/menus/mainMenu.cpp src/manager.cpp src/fileManager.cpp src/menus/createMenu.cpp src/menus/displayMenu.cpp src/menus/assignMenu.cpp src/menus/unassignMenu.cpp src/menus/editMenu.cpp src/menus/deleteMenu.cpp src/regex/regex.cpp -o main.exe

След което за да бъде отворено промененото приложение, то може да бъде заредено чрез отваряне на новия “main.exe” файл или чрез въвеждането на следния prompt в терминала:

./main

## Инструкции за употреба

Както вече стана ясно употребата на приложението е интуитивно и потребителят постоянно бива „подбутван“ към това какво трябва да въведе. Нека прегледаме всички функционалности по отделно:

1. A diagram of a software development

   Description automatically generatedDisplay information
   1. Display All vehicles
   2. Display All routes
   3. Display All pairs
   4. Return to main menu
2. Create new…
   1. New Vehicle
   2. New Route
   3. Return to main menu
3. Assign new pair
4. Unassign new pair
5. Edit…
   1. Edit Vehicle
   2. Edit Route
   3. Return to main menu
6. Delete…
   1. Delete Vehicle
   2. Return to main menu
7. Exit
   1. Показва всички превозни средства

Обектите се четат от оперативната памет в момента на отваряне на менюто.

* 1. Показва всички маршрути

Обектите се четат от оперативната памет в момента на отваряне на менюто.

* 1. Показва всички двойки (превозно средство – маршрут)

Обектите се четат от оперативната памет в момента на отваряне на менюто.

2.1. Създава нов обект от тип превозно средство

Потребителят бива насочван към това да създаде нов обект от тип превозно средство, като въвежда необходимата информация за него. Съществуват ограничения за отделните полета, като те са зададени от регулярни изрази. Пример за тези ограничения е това, че не може годината на производство да е преди 1886 (годината на създаване на първия автомобил) и 2024 (годината на създаване на приложението). Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”, което не запаметява обекта.

2.2. Създава нов обект от тип маршрут

Потребителят бива насочван към това да създаде нов обект от тип маршрут, като въвежда необходимата информация за него. Съществуват ограничения за отделните полета, като те са зададени от регулярни изрази. Пример за тези ограничения е това, че не може повторенията да са повече от 999. Спирките се записват една по една, като е необходимо натискането на enter между отделните спирки. Интервалът е забранен символ. Той би записал една спирка като 2. Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”, което не запаметява обекта.

5.1. Променя обект от тип превозно средство

Потребителят отваря меню, което позволява промяната на елемент от обекта, избран от него. Това се случва, като на потребителят му биват показани всички превозни средства, след което той избира кой обект да промени, посочвайки неговият идентификационен номер. Промяната става чрез изтриване на старата информация и записване на новата. Стандартните ограничения важат отново. Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”, което не запаметява обекта.

5.2. Променя обект от тип маршрут

Потребителят отваря меню, което позволява промяната на елемент от обекта, избран от него. Това се случва, като на потребителят му биват показани всички маршрути, след което той избира кой обект да промени, посочвайки неговият идентификационен номер. Промяната става чрез изтриване на старата информация и записване на новата. Стандартните ограничения важат отново. Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”, което не запаметява обекта.

6.1. Изтрива обект от тип превозно средство

Потребителят отваря меню, което позволява изтриването на обекта, избран от него. Това се случва, като на потребителят му биват показани всички превозни средства, след което той избира кой обект да изтрие, посочвайки неговият идентификационен номер. Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”.

6.2. Изтрива обект от тип маршрут

Потребителят отваря меню, което позволява изтриването на обекта, избран от него. Това се случва, като на потребителят му биват показани всички маршрути, след което той избира кой обект да изтрие, посочвайки неговият идентификационен номер. Възможно е излизане от това меню по всеки момент чрез въвеждане на ключовата дума “STOP”.

1.4., 2.3., 5.3. и 6.3. Връщат потребителя едно ниво нагоре

## Структура на кода

Файловата структура на проекта е както следва:

Имаме папка, в която се намира цялото приложение (.exe файла, както и сорс кода). В нея са поместени 3 основни папки. Едната се казва “data”, където са поместени данните за отделните обекти. Другата се казва “docs”, където са поместени както диаграмите, така и този файл. И третата се нарича “src”, където е поместен целия сорс код. Тук имаме още няколко разделения. Папка “core” съхранява основните обекти на приложението, а именно “vehicle” и “route”. Тези класове са поместени в по 2 файла, а те са “.h” файла, в който са поместени декларациите на функциите, както и техните член-променливи, а в “.cpp” файла са поместени техните дефиниции. Това важи за всеки един клас. След това в папка “menus” са поместени всичките менюта, чиито функции се използват за всичко свързано с визуализация на информацията. Всички менюта се намират тук включително и главното “mainMenu”. Следва папка “regex” където са поместени всички функции, извършващи проверка за това, дали данните са въведени правилно (дали са спазени определени граници и дали е спазен исканият формат). Това се извършва чрез регулярни изрази. Последните 4 файла в папка “src” са “manager.h”, “fileManager.h”, както и техните “.cpp” файлове. “manager.h”, е отговорен за това информацията която трябва да се съхрани да бъде запазена в определената за целта “data structure”, както и да може да се осъществяват всички операции, които се изискват от приложението докато то работи. Другият важен файл е “fileManager.h”. Той е отговорен за това папките необходими на приложението да бъдат създадени при всяко пускане на програмата. След това той създава необходимите файлове в папката “data” ,ако не са създадени, като те са именно: “vehicles.txt”, “routes.txt” и “pairs.txt”. Ако обаче те са създавани приложението минава към следващата стъпка, а именно четене на данните от тези файлове. То ги прочита и записва всеки запис в оперативната памет, подготвяйки приложението за работа. Следващата стъпка е той да затвори тези файлове. Втората и последна задача на този клас е да запише данните в файловете, като той първо ги изпразва и след това ги презаписва. Това се случва след като потребителят инициира затварянето на приложението. Накрая файловете се затварят и приложението излиза от своят “main.cpp” файл и затваря приложението, който отново се намира в папка “scr”.

## Подробна информация за някои от по-важните методи

Претоварените оператори << и >> за класовете “vehicle” и “route”.

std::ostream& operator<<(std::ostream& os, const Vehicle& vehicle)

std::istream& operator>>(std::istream& is, Vehicle& vehicle)

std::ostream& operator<<(std::ostream& os, const Route& route)

std::istream& operator>>(std::istream& is, Route& route)

Това са функции намиращи се в съответните класове, а именно “vehicle” и “route”. Тяхната роля е да систематизират данните в даден формат с цел по-лесно боравене с тях. Те се използват при извеждането на информация на конзолата, както и при четенето и записа на данните във текстовите файлове, където данните се съхраняват, когато програмата не работи. Ето и примери за това как изглеждат данните след това форматиране:

[5, Honda, Civic, 2002, 5 seats, 1200 kg, 5 l/100km, untaken]

[3, {sofiaRingMall, Mladost, unss}, 35 km, 50 repetitions]

MainMenu::mainMenu()

Това е главния метод, в класа “MainMenu” от където се избира каква операция потребителят иска да извърши. Това се случва посредством “switch case” като входа е число и според числото потребителят бива препратен към друго подменю. Освен това този метод е отговорен и за правилното зареждане на файловете. Това се реализира чрез методите “FileManager::initializeFiles()” и “FileManager::loadData()”. Когато потребителят иска да излизе от програмата методът извиква метод “FileManager::closeFiles()”. Това е и единствения мотод, които се извиква във класа “main”.

FileManager::initializeFiles() и FileManager::loadData()

Първия метод проверява дали файловете съществуват и ако не съществуват ги създава. Той връща булева стойност като така можем да направим проверка дали всичко е минало наред. Втория метод отваря файловете, зарежда данните и ги затваря след това.

FileManager::closeFiles()

Това е метод в класа “FileManager”, който отваря файловете, записва данните във файловете и след това ги затваря. Записването се осъществява чрез 3 метода във клас “Manager”, който са отговорни съответно за двата класа по отделно, както и двойките превозно средство-маршрут. Тези методи записват данните чрез цикли, които обхождат структурите по отделно и ги записват във празните файлове.

## Допълнителна информация

Някой от проблемите, с които потребителят трябва да е наясно са:

1. Неправилното инициализиране на ID след изтриване на междинен запис

Това се случва тъй като за инициализиране на ID се използва последния запис. Именно това е и причината когато се изтрие запис, който не е последен неговото ID да не се използва никога повече (освен ако всички записи след него не бъдат изтрити също)

2. Запаметяване на данните

Запаметяването на данните се извършва само при излизане от програмата от главното меню. Само тогава приложението ще запамети данните в съответните файлове. Това означава, че всяко неправилно излизане от приложението би довело до загуба на работата на потребителя.