

Prolećni semestar 2023/2024

PREDMET: CS130 C/C++ Programski jezik

PROJEKTNI ZADATAK

IGRA PREVOĐENJA

STUDENT

Stevan Stojanović 5607

Student Softverskog inženjerstva

ASISTENT

Sofija

Niš, 20.04.2024.

Sadržaj

|  |  |  |  |
| --- | --- | --- | --- |
| [1.](#page3) | [Uvod ......................................................................................................................................](#page3) | | [3](#page3) |
| 2. | Analiza .................................................................................................................................. | | 4 |
|  | 2.1. | Cilj programa .................................................................................................................. | 4 |
|  | 2.2. | Elementi programa ......................................................................................................... | 4 |
|  | 2.3. | Izrada programa .............................................................................................................. | 16 |
|  | 2.4. | Funkcionisanje programa i prikazi ekrana ..................................................................... | 16 |
| 3. Zaključak ............................................................................................................................. | | | 10 |
| 4. | Reference ............................................................................................................................. | | 11 |

**1. Uvod**

Ovaj program je dizajniran da pojednostavi upravljanje kolekcijom vozila, omogućavajući korisnicima intuitivan način za dodavanje, prikazivanje, i čuvanje informacija o različitim vrstama vozila kao što su automobili i motocikli. Kroz jednostavno korisničko sučelje, korisnici mogu lako unositi nove podatke o vozilima, pregledati postojeće zapise, izračunati ukupnu vrednost kolekcije, kao i sačuvati ili učitati podatke iz datoteke. Ova funkcionalnost omogućava efikasno upravljanje informacijama, što je posebno korisno u kontekstima kao što su auto saloni, rent-a-car agencije, i privatne kolekcije vozila.

**2. Analiza**

**2.1.Cilj programa**

Cilj ovog programa je upravljanje kolekcijom vozila, omogućavajući korisnicima dodavanje novih vozila (automobila i motocikala), prikazivanje informacija o vozilima, izračunavanje ukupne cene svih vozila, čuvanje podataka o vozilima u datoteku i učitavanje tih podataka iz datoteke. Program pruža jednostavno korisničko iskustvo za interakciju s kolekcijom vozila, što olakšava upravljanje informacijama o različitim tipovima vozila.

**2.2.Elementi programa**

Program čine apstraktna klasa Vehicle, klasa Car, Motorcycle, FileNotFoundException, OutOfMemoryException i VehicleCollector, zajedno sa svim metodama i glava main metoda.

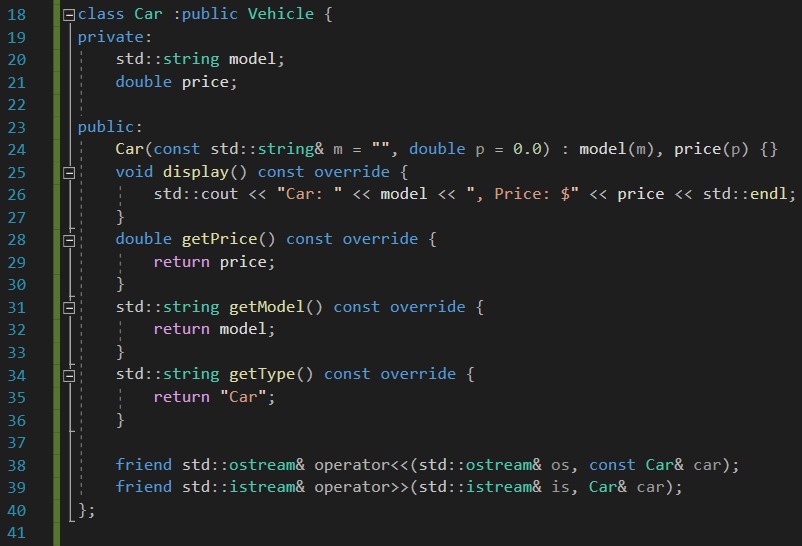
*Apstraktna klasa Vehicle*

Apstraktna klasa Vehicle definiše osnovne karakteristike svih vozila u programu. Čine je sledeći elementi:



1. **Destruktor**
   * virtual ~Vehicle() = default;
   * Ovo je virtualni destruktor koji osigurava pravilno oslobađanje resursa prilikom brisanja objekata izvedenih klasa kroz pokazivače na baznu klasu Vehicle.
2. **Virtualne metode**
   * virtual void display() const = 0;
     + Virtualna metoda koja mora biti implementirana u izvedenim klasama. Cilj ove metode je da prikaže informacije o vozilu.
   * virtual double getPrice() const = 0;
     + Virtualna metoda koja mora biti implementirana u izvedenim klasama. Vraća cenu vozila.
   * virtual std::string getModel() const = 0;
     + Virtualna metoda koja mora biti implementirana u izvedenim klasama. Vraća model vozila.
   * virtual std::string getType() const = 0;
     + Virtualna metoda koja mora biti implementirana u izvedenim klasama. Vraća tip vozila kao string.
3. **Preopterećeni operatori**
   * friend std::ostream& operator<<(std::ostream& os, const Vehicle& vehicle);
     + Funkcija koja preopterećuje operator << za izlazne tokove. Omogućava ispis objekta klase Vehicle i njegovih izvedenih klasa.
   * friend std::istream& operator>>(std::istream& is, Vehicle\*& vehicle);
     + Funkcija koja preopterećuje operator >> za ulazne tokove. Omogućava unos objekta klase Vehicle i njegovih izvedenih klasa.

*Klasa Car*



Klasa Car je izvedena iz apstraktne baze klase Vehicle i predstavlja specifičan tip vozila – automobil. Sadrži informacije o modelu automobila i njegovoj ceni. Klasa Car implementira sve čiste virtualne metode definisane u bazi klasi Vehicle.

**Promenljive**

1. **std::string model**
   * Čuva naziv modela automobila.
   * Koristi se u metodama getModel i display da bi se prikazale informacije o modelu automobila.
2. **double price**
   * Čuva cenu automobila.
   * Koristi se u metodama getPrice i display da bi se prikazala i vratila cena automobila.

**Konstruktor**

* **Konstruktor sa podrazumevanim vrednostima**
  + Inicijalizuje članove model i price sa prosleđenim vrednostima. Ako vrednosti nisu prosleđene, koristi se prazan string za model i 0.0 za price.

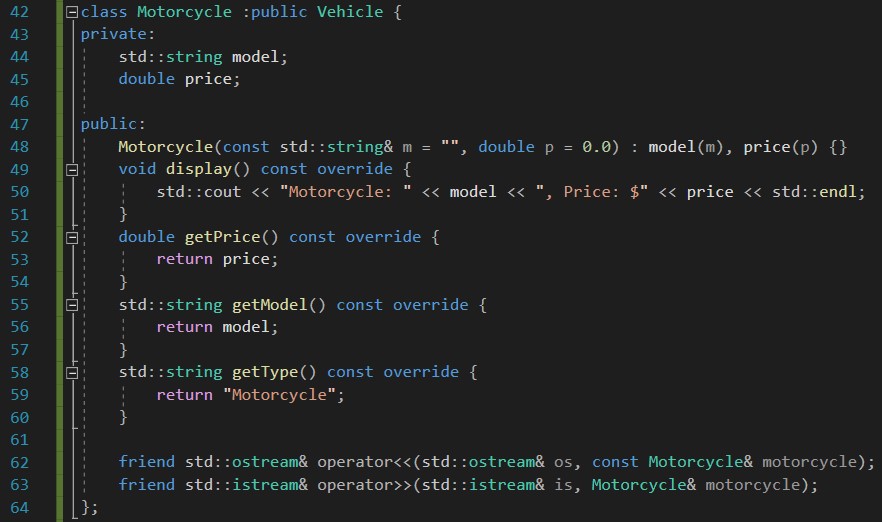
**Metode (implementacije apstraktnih metoda iz bazne Vehicle)**

1. **void display() const override**
   * Prikazuje informacije o automobilu (model i cena) na konzoli.
   * Koristi std::cout za ispis.
2. **double getPrice() const override**
   * Vraća cenu automobila.
   * Koristi se za izračunavanje ukupne cene svih vozila u kolekciji.
3. **std::string getModel() const override**
   * Vraća naziv modela automobila.
   * Koristi se za prikaz i upravljanje informacijama o modelu.
4. **std::string getType() const override**
   * Vraća string "Car" koji označava tip vozila.
   * Koristi se za razlikovanje tipova vozila u funkcijama kao što su preopterećeni operatori.

**Prijateljske funkcije (preopterećeni operatori)**

1. **friend std::ostream& operator<<(std::ostream& os, const Car& car)**
   * Preopterećuje operator << za ispis informacija o automobilu.
   * Ispisuje model i price u dati izlazni tok.
2. **friend std::istream& operator>>(std::istream& is, Car& car)**
   * Preopterećuje operator >> za unos informacija o automobilu.
   * Učitava model i price iz datog ulaznog toka.

*Klasa Motorcycle*

 Klasa Motorcycle je izvedena iz apstraktne baze klase Vehicle i predstavlja specifičan tip vozila – motocikl. Sadrži informacije o modelu motocikla i njegovoj ceni. Klasa Motorcycle implementira sve čiste virtualne metode definisane u bazi klasi Vehicle.

**Promenljive**

1. **std::string model**
   * Čuva naziv modela motocikla.
   * Koristi se u metodama getModel i display da bi se prikazale informacije o modelu motocikla.
2. **double price**
   * Čuva cenu motocikla.
   * Koristi se u metodama getPrice i display da bi se prikazala i vratila cena motocikla.

**Konstruktor**

* **Konstruktor sa podrazumevanim vrednostima**
  + Inicijalizuje članove model i price sa prosleđenim vrednostima. Ako vrednosti nisu prosleđene, koristi se prazan string za model i 0.0 za price.

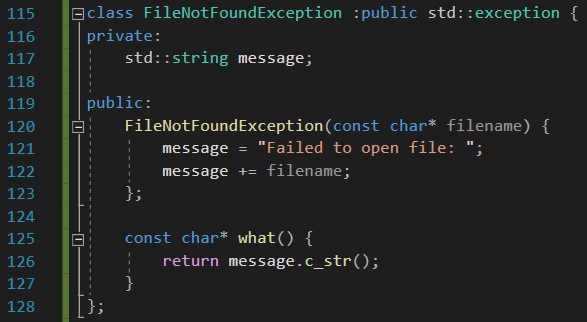
**Metode (implementacije apstraktnih metoda iz Vehicle)**

1. **void display() const override**
   * Prikazuje informacije o motociklu (model i cena) na konzoli.
   * Koristi std::cout za ispis.
2. **double getPrice() const override**
   * Vraća cenu motocikla.
   * Koristi se za izračunavanje ukupne cene svih vozila u kolekciji.
3. **std::string getModel() const override**
   * Vraća naziv modela motocikla.
   * Koristi se za prikaz i upravljanje informacijama o modelu.
4. **std::string getType() const override**
   * Vraća string "Motorcycle" koji označava tip vozila.
   * Koristi se za razlikovanje tipova vozila u funkcijama kao što su preopterećeni operatori.

**Prijateljske funkcije (preopterećeni operatori)**

1. **friend std::ostream& operator<<(std::ostream& os, const Motorcycle& motorcycle)**
   * Preopterećuje operator << za ispis informacija o motociklu.
   * Ispisuje model i price u dati izlazni tok.
2. **friend std::istream& operator>>(std::istream& is, Motorcycle& motorcycle)**
   * Preopterećuje operator >> za unos informacija o motociklu.
   * Učitava model i price iz datog ulaznog toka.

*Klasa FileNotFoundException*

 Klasa FileNotFoundException je posebna vrsta izuzetka koja se koristi za rukovanje greškama pri otvaranju fajlova. Ova klasa nasleđuje standardnu C++ klasu std::exception i omogućava korisniku da sazna koji fajl nije mogao biti otvoren putem prilagođene poruke.

**Promenljive**

1. **std::string message**
   * Ova varijabla čuva poruku greške koja se generiše kada se fajl ne može otvoriti.
   * Inicijalizuje se u konstruktoru i koristi u metodi what.

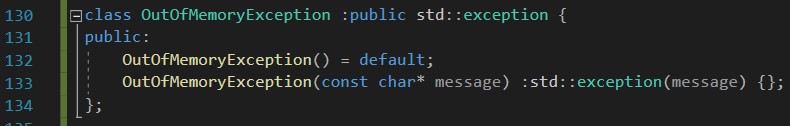
**Konstruktor**

1. **FileNotFoundException(const char\* filename)**
   * Konstruktor prima ime fajla koji nije mogao biti otvoren i formira poruku greške.
   * Konkretno, postavlja vrednost varijable message na "Failed to open file: " praćeno imenom fajla.

**Metode**

1. **const char\* what()** 
   * Ova metoda je prepisana verzija metode what iz standardne klase std::exception.
   * Vraća poruku greške koja je sačuvana u promenljivoj message.

*Klasa OutOfMemoryException*

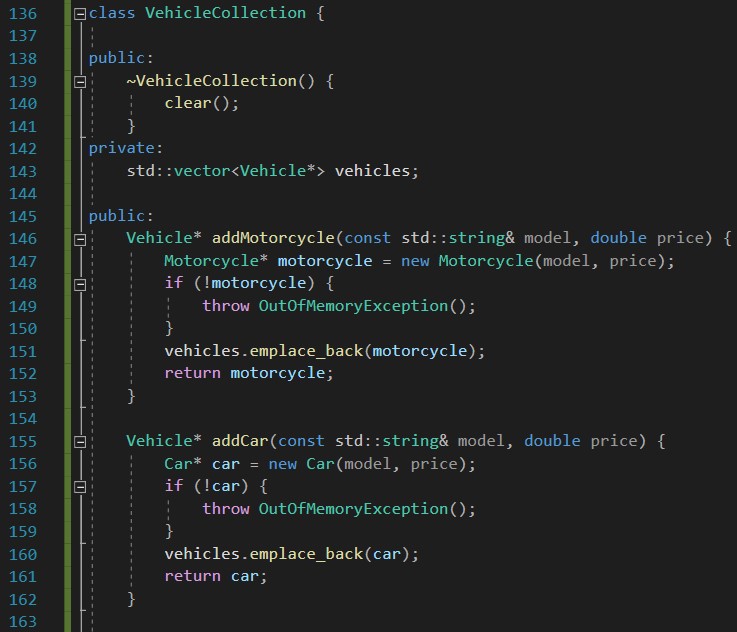
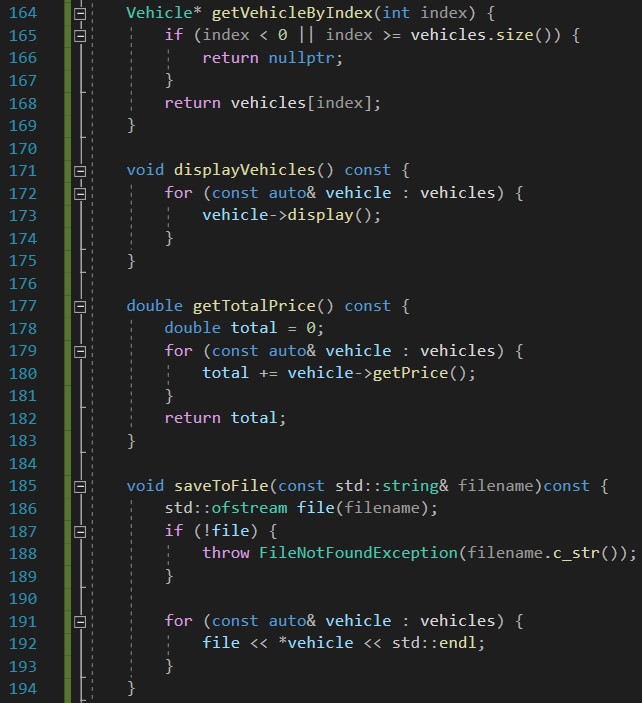
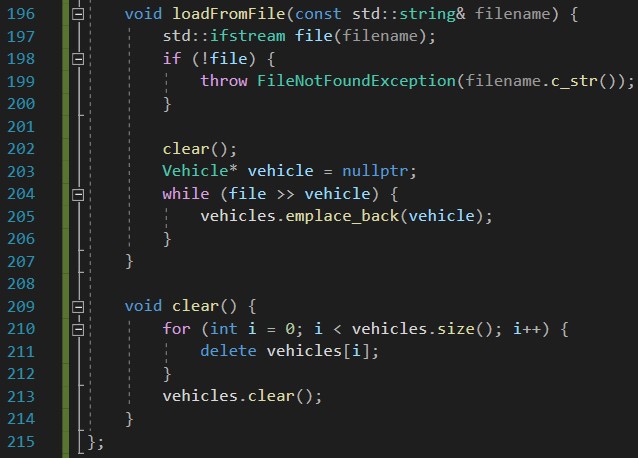
 Klasa OutOfMemoryException je posebna vrsta izuzetka koja se koristi za rukovanje greškama pri alokaciji memorije. Ova klasa nasleđuje standardnu C++ klasu std::exception i omogućava korisniku da prepozna kada program nije uspeo da alocira potrebnu memoriju.

**Konstruktori**

1. **OutOfMemoryException()**
   * Podrazumevani konstruktor koji ne prima argumente.
   * Poziva podrazumevani konstruktor bazne klase std::exception.
   * Ovaj konstruktor se koristi kada nije potrebno proslediti prilagođenu poruku greške.
2. **OutOfMemoryException(const char\* message)**
   * Konstruktor koji prima C-string (const char\*) kao argument i prosleđuje ga konstruktoru bazne klase std::exception.
   * Omogućava korisniku da specificira prilagođenu poruku greške.

*Klasa VehicleCollection*

Klasa VehicleCollection predstavlja kolekciju vozila (objekata klase Vehicle) sa funkcionalnostima za dodavanje, uklanjanje, prikaz i manipulaciju vozilima u kolekciji.



**Promenljive**

1. **std::vector<Vehicle\*> vehicles**
   * Vektor pokazivača na objekte klase Vehicle, koji čuva sve dodate vozile.
   * Koristi se za dinamičko upravljanje objektima klase Vehicle u kolekciji.
   * Svi elementi su pokazivači jer omogućava efikasno upravljanje resursima (kreiranje, brisanje, čuvanje).

**Konstruktor i destruktor**

1. **VehicleCollection()**
   * Podrazumevani konstruktor koji inicijalizuje praznu kolekciju.
   * Automatski se poziva prilikom kreiranja objekta klase VehicleCollection.
2. **~VehicleCollection()**
   * Destruktor koji oslobađa sve resurse zauzete pokazivačima na objekte vozila u kolekciji.
   * Iterira kroz sve elemente u vektoru vehicles i dealocira memoriju za svaki objekat klase Vehicle.

**Metode**

1. **Vehicle\* addMotorcycle(const std::string& model, double price)**
   * Dodaje novi motocikl u kolekciju sa specificiranim modelom i cenom.
   * Kreira novi objekat klase Motorcycle sa prosleđenim parametrima, dodaje ga u vektor vehicles i vraća pokazivač na njega.
2. **Vehicle\* addCar(const std::string& model, double price)**
   * Dodaje novi automobil u kolekciju sa specificiranim modelom i cenom.
   * Kreira novi objekat klase Car sa prosleđenim parametrima, dodaje ga u vektor vehicles i vraća pokazivač na njega.
3. **Vehicle\* getVehicleByIndex(int index)**
   * Vraća pokazivač na vozilo u kolekciji na osnovu prosleđenog indeksa.
   * Omogućava pristup vozilima u kolekciji preko indeksa, koristeći nullptr za nevažeće indekse.
4. **void displayVehicles() const**
   * Prikazuje sve vozile u kolekciji pozivajući display metod za svako vozilo.
   * Koristi se za ispis svih vozila na standardnom izlazu.
5. **double getTotalPrice() const**
   * Računa i vraća ukupnu cenu svih vozila u kolekciji.
   * Iterira kroz sve elemente u vektoru vehicles i sabira cene vozila pozivajući getPrice metod za svako vozilo.
6. **void saveToFile(const std::string& filename) const**
   * Čuva informacije o svim vozilima u kolekciji u datoteku sa datim imenom.
   * Otvara datoteku za pisanje, iterira kroz sve elemente u vektoru vehicles i koristi operator << da bi upisao informacije o svakom vozilu u datoteku.
7. **void loadFromFile(const std::string& filename)**
   * Učitava informacije o vozilima iz datoteke sa datim imenom i dodaje ih u kolekciju.
   * Otvara datoteku za čitanje, čita informacije o svakom vozilu pomoću operatora >> i dodaje novo kreirane objekte vozila u vektor vehicles.
8. **void clear()**
   * Oslobađa sve resurse zauzete pokazivačima na objekte vozila u kolekciji.
   * Iterira kroz sve elemente u vektoru vehicles i dealocira memoriju za svaki objekat klase Vehicle.
   * Nakon ovog koraka, vektor vehicles je prazan.

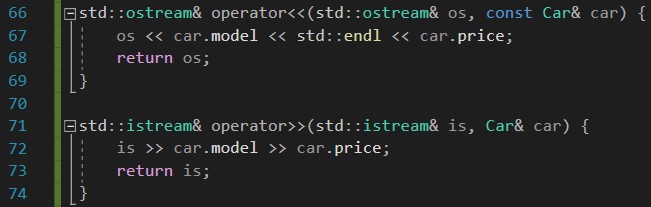
*Preklapanje operatora*

**Operatori za klasu Car**

**Operator << (za ispis u ostream)**

Ovaj operator omogućava ispisivanje informacija o automobilu (Car) u izlazni tok (ostream). Implementacija operatora je definisana kao prijateljska funkcija unutar klase Car.

* **Argumenti:**
  + os: Referenca na izlazni tok (ostream) u koji se ispisuje informacija.
  + car: Konstantna referenca na objekat klase Car čije se informacije ispisuju.
* **Detalji implementacije:**
  + Operator << ispisuje model automobila (car.model) i cenu (car.price) u odgovarajući izlazni tok (os).
  + Na kraju se vraća referenca na izlazni tok kako bi se omogućilo lančano korišćenje operatora.



**Operator >> (za unos iz istream)**

Ovaj operator omogućava čitanje informacija o automobilu (Car) iz ulaznog toka (istream). Implementacija operatora je takođe definisana kao prijateljska funkcija unutar klase Car.

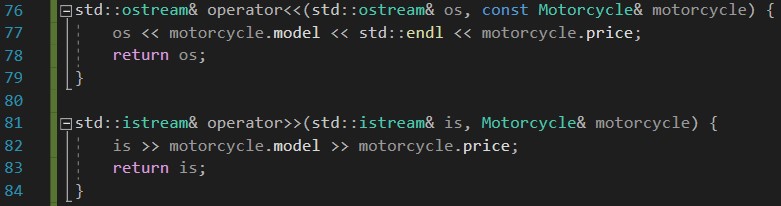
* **Argumenti:**
  + is: Referenca na ulazni tok (istream) iz kojeg se čitaju informacije.
  + car: Referenca na objekat klase Car u koji se čuvaju učitane informacije.
* **Detalji implementacije:**
  + Operator >> čita redom model automobila (car.model) i cenu (car.price) iz ulaznog toka (is).
  + Na kraju se vraća referenca na ulazni tok kako bi se omogućilo lančano korišćenje operatora.

**Operatori za klasu Motorcycle**

**Operator << (za ispis u ostream)**

Ovaj operator omogućava ispisivanje informacija o motociklu (Motorcycle) u izlazni tok (ostream). Implementacija operatora je definisana kao prijateljska funkcija unutar klase Motorcycle.

* **Argumenti:**
  + os: Referenca na izlazni tok (ostream) u koji se ispisuje informacija.
  + motorcycle: Konstantna referenca na objekat klase Motorcycle čije se informacije ispisuju.
* **Detalji implementacije:**
  + Operator << ispisuje model motocikla (motorcycle.model) i cenu (motorcycle.price) u odgovarajući izlazni tok (os).
  + Na kraju se vraća referenca na izlazni tok kako bi se omogućilo lančano korišćenje operatora.



**Operator >> (za unos iz istream)**

Ovaj operator omogućava čitanje informacija o motociklu (Motorcycle) iz ulaznog toka (istream). Implementacija operatora je takođe definisana kao prijateljska funkcija unutar klase Motorcycle.

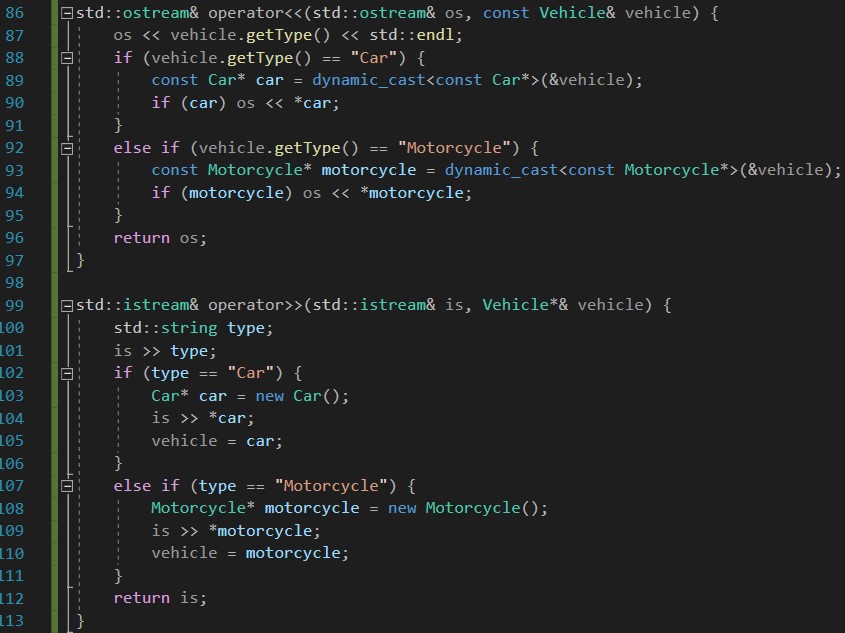
* **Argumenti:**
  + is: Referenca na ulazni tok (istream) iz kojeg se čitaju informacije.
  + motorcycle: Referenca na objekat klase Motorcycle u koji se čuvaju učitane informacije.
* **Detalji implementacije:**
  + Operator >> čita redom model motocikla (motorcycle.model) i cenu (motorcycle.price) iz ulaznog toka (is).
  + Na kraju se vraća referenca na ulazni tok kako bi se omogućilo lančano korišćenje operatora.

**Operatori za klasu Vehicle**

**Operator << (za ispis u ostream)**

Ovaj operator omogućava ispisivanje informacija o vozilu (Vehicle) u izlazni tok (ostream). Implementacija operatora je definisana kao prijateljska funkcija izvan svih klasa.

* **Argumenti:**
  + os: Referenca na izlazni tok (ostream) u koji se ispisuje informacija.
  + vehicle: Konstantna referenca na objekat klase Vehicle čije se informacije ispisuju.
* **Detalji implementacije:**
  + Operator << prvo ispisuje tip vozila (vehicle.getType()) u izlazni tok.
  + Zatim koristi dinamičko prebacivanje tipova (dynamic\_cast) da bi pristupio specifičnim podacima (modelu i ceni) vozila, bez obzira na to da li je vozilo tipa Car ili Motorcycle.
  + Na kraju se vraća referenca na izlazni tok kako bi se omogućilo lančano korišćenje operatora.

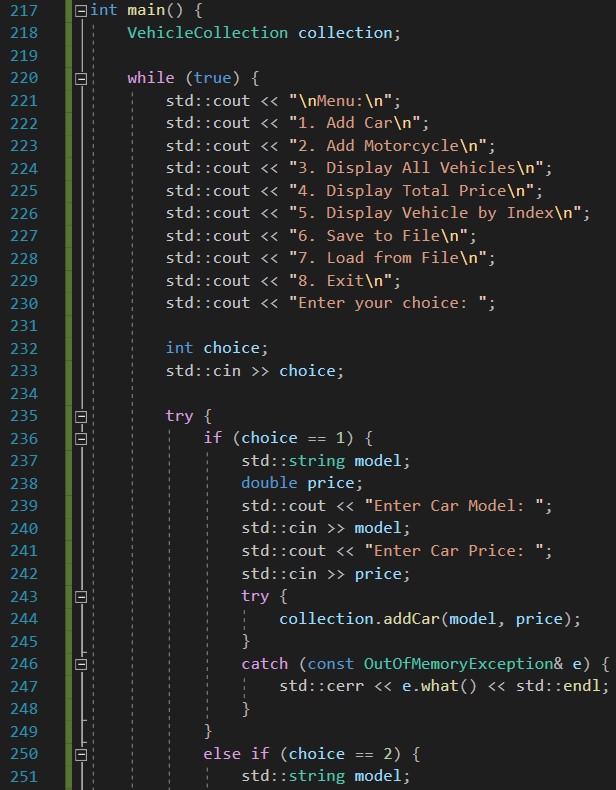


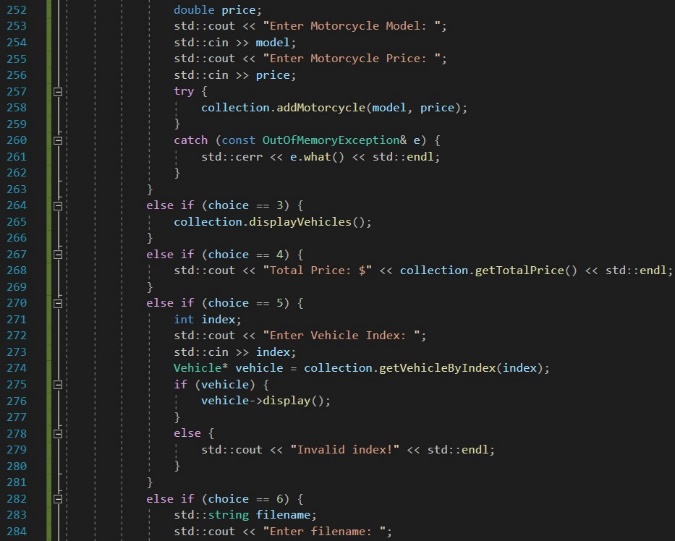
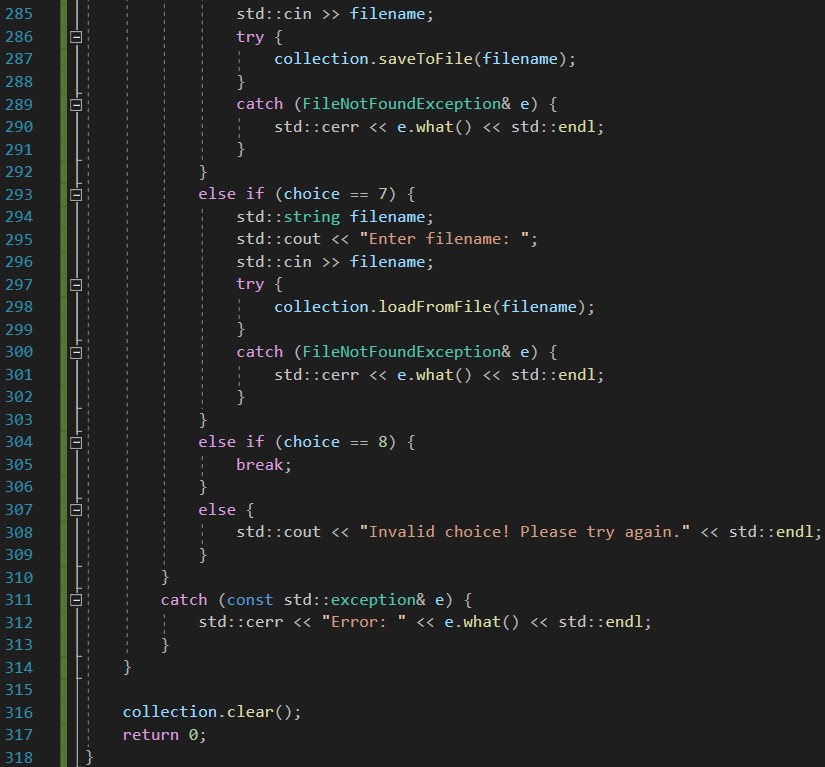
**Operator >> (za unos iz istream)**

Ovaj operator omogućava čitanje informacija o vozilu (Vehicle) iz ulaznog toka (istream). Implementacija operatora je takođe definisana kao prijateljska funkcija izvan svih klasa.

* **Argumenti:**
  + is: Referenca na ulazni tok (istream) iz kojeg se čitaju informacije.
  + vehicle: Pokazivač na pokazivač na objekat klase Vehicle u koji se smešta učitano vozilo.
* **Detalji implementacije:**
  + Operator >> prvo čita tip vozila (type) iz ulaznog toka (is).
  + Na osnovu tipa vozila (Car ili Motorcycle), dinamički alocira novi objekat odgovarajuće klase (Car ili Motorcycle).
  + Zatim koristi operator >> da bi učitao specifične podatke o vozilu (model i cena) u novi objekat.

*Glavna MAIN metoda*





Objašnjenje main metode:

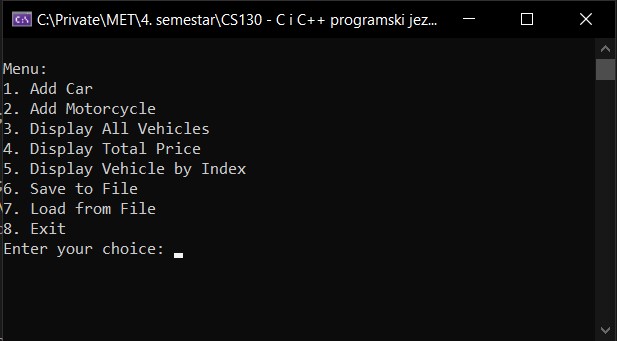
1. **Inicijalizacija VehicleCollection objekta:**
   * VehicleCollection collection; stvara instancu klase VehicleCollection, koja će se koristiti za upravljanje kolekcijom vozila tokom izvršavanja programa.
2. **Glavna while petlja:**
   * while (true) ova petlja služi kao osnovni meni aplikacije, koji se izvršava sve dok korisnik ne izabere opciju za izlaz (choice == 8).
3. **Prikaz menija:**
   * Korisniku se prikazuje meni sa opcijama koje može izabrati.
   * std::cout se koristi za ispisivanje opcija na standardni izlaz.
4. **Unos i obrada izbora korisnika:**
   * std::cin >> choice; čeka unos korisnika kako bi odabrao željenu opciju iz menija.
5. **Obrada izbora korisnika (try-catch blok):**
   * Svaka opcija u meniju je odgovarajuće implementirana unutar try-catch bloka kako bi se hvatale i obradile moguće izuzetne situacije (npr. nedovoljno memorije, greška pri radu sa datotekom).
   * choice == 1 i choice == 2: Dodavanje novog automobila odnosno motocikla u kolekciju.
   * choice == 3: Prikaz svih vozila u kolekciji.
   * choice == 4: Prikaz ukupne cene svih vozila u kolekciji.
   * choice == 5: Prikaz specifičnog vozila na osnovu indeksa unetog od strane korisnika.
   * choice == 6: Čuvanje celokupne kolekcije vozila u datoteku.
   * choice == 7: Učitavanje celokupne kolekcije vozila iz datoteke.
   * choice == 8: Izlaz iz glavne while petlje i završetak programa.
6. **Klirovanje:**
   * Pre nego što se program završi (return 0;), poziva se collection.clear(); kako bi se oslobodili svi resursi (memorija) koji su zauzeti prilikom dodavanja i učitavanja vozila.

**2.3.Izrada programa**

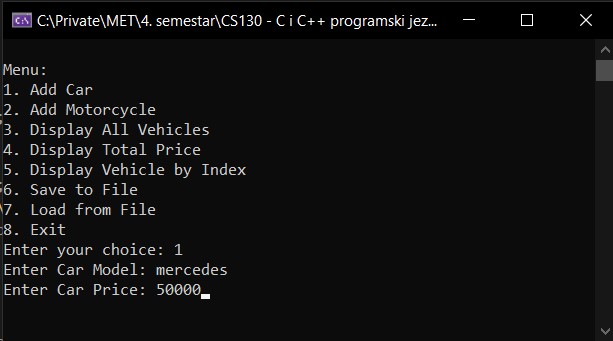
Prvenstveno je kreirana apstraktna klasa Vehicle, a zatim i klase Car i Motorcycle sa svim neophodnim elementima. Potom su definisane klase FileNotFoundException i OutOfMemoryException kako bi se obradile moguće greške prilikom rada programa, a onda i klasa VehicleCollections za upravljanje kolekcijom vozila. Usput, kako su se kreirale klase, definisali su se i odgovarajući operatori za unos i iznos. Na kraju se sve to objedinilo u glavnoj metodi programa main gde se definisao ceo koncept programa i povezivanje svih komponenti u smislenu celinu.

**2.4. Funkcionisanje programa i prikazi ekrana**

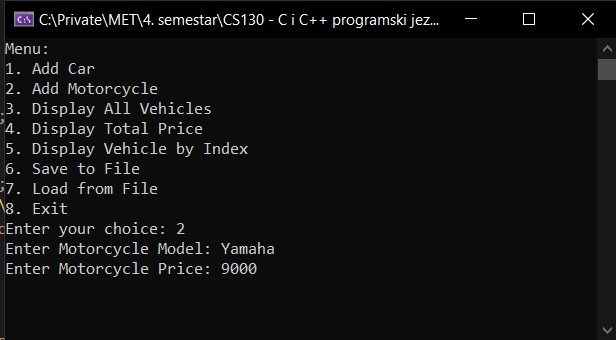
Prolikom pokretanja programa otvara se meni gde korisnik bira neku od opcija unoseći broj.



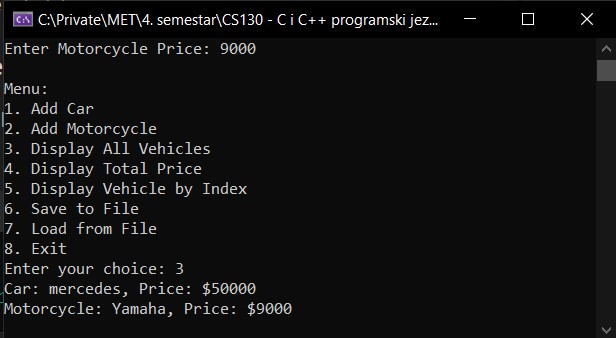
Ukoliko korisnik unese broj 1, otvara mu se mogućnost da doda u kolekciju automobil tako što unosi model automobila i njegovu cenu.



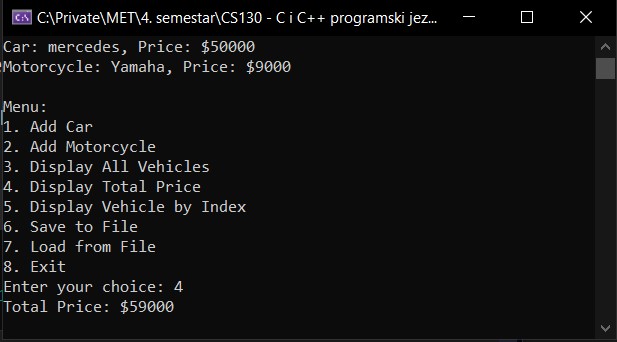
Ukoliko korisnik unese broj 2, otvara mu se mogućnost da doda u kolekciju motocikl tako što unosi model motocikla i njegovu cenu.



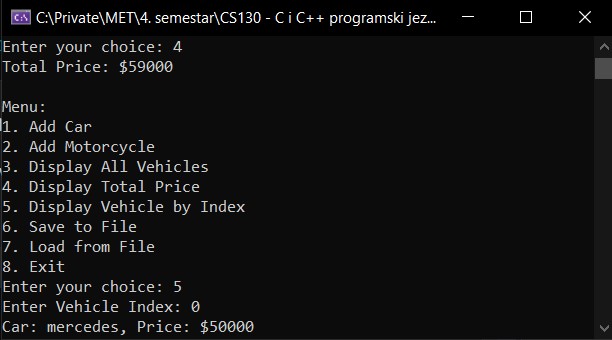
Ukoliko korisnik unese broj 1, prikazuju mu se sva vozila koja se trenutno nalaze u kolekciji tako što se za svako vozilo prikaže tip vozila (Car ili Motorcycle), model i cena.



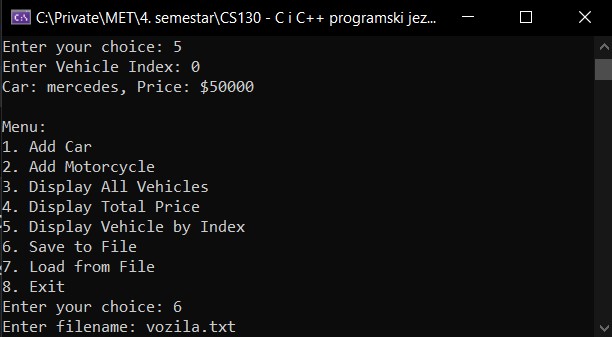
Ukoliko korisnik unese broj 4, prikazuje mu se suma cena svih trenutnih vozila u kolekciji.



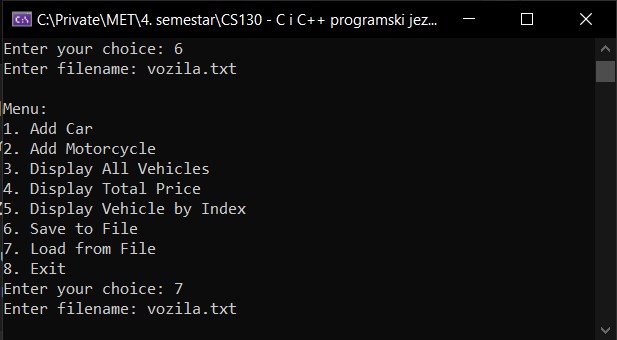
Ukoliko korisnik unese broj 5, otvara mu se mogućnost da pogleda određeno vozilo iz kolekcije tako što će uneti njegov index, i imati priliku da pogleda tip vozila, model i njegovu cenu.



Ukoliko korisnik unese broj 6, otvara mu se mogućnost da trenutnu kolekciju vozila sačuva u fajl, i kasnije ima mogućnost da učitava tu kolekciju sačuvanih vozila iz fajla. Neophodno je uneti željeno ime i ekstenziju fajla koji ce biti sačuvan.



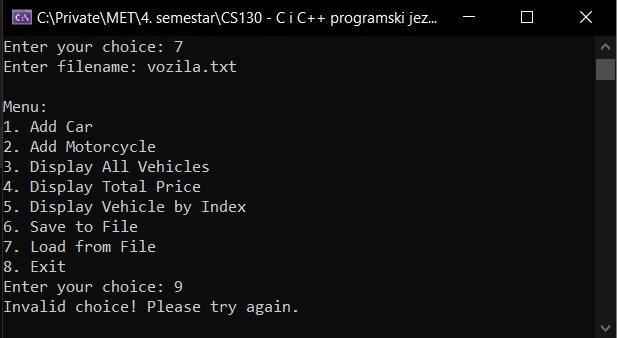
Ukoliko korisnik unese broj 8, otvara mu se mogućnost da učita neku od kolekcija vozila koje su sačuvane u nekom fajlu.



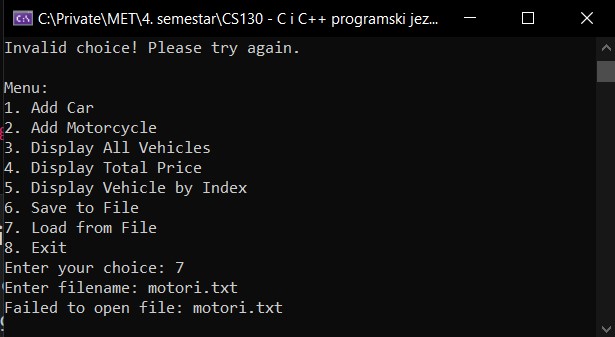
Ukoliko korisnik unese broj 8, napušta program.

**Izuzeci**

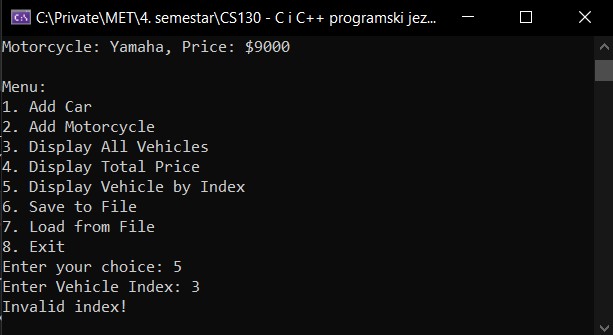
Ukoliko korisnik unese broj koji nije ponuđen kao opcija, prikazuje mu se poruka koja ga obaveštava o nevalidnom unosu.



Ukoliko korisnik pokuša da učita nepostojeći fajl u direktorijumu, prikazaće mu se poruka koja ga obaveštava o nemogućnosti otvaranja takvog fajla.



Ukoliko korisnik pokuša da pregleda neko vozilo koje ne postoji tako što je uneo pogrešan index ili je jednostavno pogrešno uneo broj, prikazaće mu se poruka koja ga obaveštava da vozilo pod tim indexom ne postoji u trenutnoj kolekciji vozila.



**3. Zaključak**

Ova aplikacija je namenjena svima koji žele da efikasno upravljaju kolekcijom informacija o vozilima, bilo da su to privatni kolekcionari automobila i motocikala, prodavci vozila ili bilo ko ko želi organizovati informacije o vozilima koje poseduju ili prodaju.

**4. Reference**

1. Stack Overflow, [http://stackoverflow.com/questions/15388041/how-to-write-stdstring-to-](http://stackoverflow.com/questions/15388041/how-to-write-stdstring-to-file)

[file](http://stackoverflow.com/questions/15388041/how-to-write-stdstring-to-file)

1. Cplusplus, <http://www.cplusplus.com/forum/beginner/7553/>
2. Cplusplus, <http://www.cplusplus.com/doc/tutorial/files/>
3. Cplusplus, <http://www.cplusplus.com/reference/cstdlib/rand/>
4. LAMS, <http://lams.metropolitan.ac.rs:8080/lams/index.do>