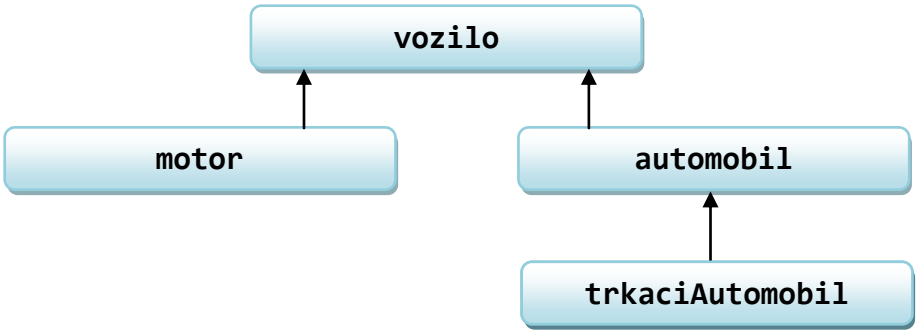
Zadatak:

Napišite sučelje i implementaciju za klase **`vozilo, motor, automobil i trkaciAutomobil`**. Sljedeći dijagram opisuje relacije među klasama. Sučelja za klase spremite u datoteku **`vozilo.h`**, a implementacije u datoteku **`vozilo.cpp`**.



Klasa `vozilo`

Ova klasa opisuje vozilo. Svako vozilo ima tri parametra koji ga određuju: snaga motora, broj kotača i registracija.

- **`vozilo(int snagaMotora, int brojKotaca, string registracija)`**
  Konstruktor za klasu vozilo. Registracija će biti jedinstvena za svako vozilo.

- **`int snagaMotora()`**
  Vraća snagu motora vozila.
  .
- **`int brojKotaca()`**
  Vraća broj kotača vozila

- **`string registracija()`**
  Vraća registraciju vozila

- **`void odvoziKrug()`**
  Simulacija vožnje vozila. Nakon odvoženog kruga gume na vozilu se istroše 10%. Na početku je istrošenost guma 0%.

- **`int istrosenostGuma()`**
  Vraća istrošenost guma vozila u postocima (gume se troše prilikom poziva funkcije `odvoziKrug`).

- **`vozilo& iduciBrzi()`**
  Vraća referencu na iduće brže vozilo (s obzirom na sva vozila koja trenutno postoje u programu), tj. od svih vozila koji su strogo brža od vozila koji je pozvao funkciju treba uzet najsporijeg. Jedno vozilo je brže od drugog ako ima veću snagu motora. Ako postoji više takvih vozila vraća se jedno od mogućih rješenja po proizvoljnom izboru. Ako ne postoji brže vozilo onda se vraća referenca na samog sebe.

- **`static vozilo& brzeVozilo(vozilo &voz1, vozilo &voz2)`**
  Uspoređuje vozila voz1 i voz2 te vraća referencu od bržeg vozila.

Klasa `motor`

Ova klasa opisuje motor. Podrazumijeva se da motor ima dva kotača.

- **`motor(int snagaMotora, string registracija)`**
  Konstruktor za klasu motor.

- **`motor& kupiBolji()`**
  Povećava snagu motora za 100 te vraća referencu na samog sebe.

Klasa `automobil`

Ova klasa opisuje automobil. Podrazumijeva se da automobil ima četiri kotača. Također, svaki automobil odlikuje i broj vrata.

- **`automobil(int snagaMotora, int brojVrata, string registracija)`**
  Konstruktor za klasu automobil.

- **`int brojVrata()`**
  Vraća broj vrata automobila.

Klasa `trkaciAutomobil`

Ova klasa opisuje trkaći automobil. Podrazumijeva se da trkaći automobil ima dvoja vrata.

- **`trkaciAutomobil(int snagaMotora, string registracija)`**
  Konstruktor za klasu trkaciAutomobil.

- **`void odvoziKrug();`**
  Nakon odvoženog kruga gume na vozilu se istroše 20%.

Primjer klijentskog programa

```
#include <iostream>
#include "vozilo.h"

int main(void)
{
    vozilo A(200, 2, "AAAA"), B(200, 4, "BBBB"), C(400, 4, "CCCC");
    motor M(350, "MMMM");
    trkaciAutomobil T(800, "TTTT");
    vozilo *nizVozila[5];
    nizVozila[0] = new automobil(200, 3, "0000");
    nizVozila[1] = new motor(200, "1111");
    nizVozila[2] = new automobil(250, 5, "2222");
    nizVozila[3] = new trkaciAutomobil(900, "3333");
    nizVozila[4] = new motor(125, "4444");

    cout << A.iduciBrzi().registracija() << endl;
    //  MMMM

    cout << B.iduciBrzi().registracija() << endl;
    //    2222

    vozilo &v = vozilo::brzeVozilo(A, C);
    cout << v.snagaMotora() << " " << v.brojKotaca() << " " << v.registracija() << endl;
    //    400 4 CCCC

    for(int i = 0; i < 5; i++)
    {
        nizVozila[i]->odvoziKrug();
        nizVozila[i]->odvoziKrug();
        nizVozila[i]->odvoziKrug();
    }

    for(int i = 0; i < 5; i++)
        cout << nizVozila[i]->registracija() << " " << nizVozila[i]->istrosenostGuma() << endl;
    //    0000 30
    //    1111 30
    //    2222 30
    //    3333 60
    //    4444 30

    cout << M.kupiBolji().snagaMotora() << endl;
    //    450

    return 0;
}
```

Opće napomene

- Struktura, funkcije i datoteke koje šaljete moraju se zvati točno onako kako je zadano u zadatku. Pazite na mala i velika slova!
- Trebate poslati samo sučelje i implementaciju. U datotekama koje šaljete ne smije se nalaziti funkcija main()!
- nijedna funkcija ne smije ništa učitavati s tipkovnice ili neke datoteke, niti išta ispisivati na ekran ili u neku datoteku.
- Svaki od main-ova pomoću kojih testiramo ispravnost vašeg programa neće pozivati sve gore navedene funkcije. Stoga, ako neku od funkcija ne znate napisati ipak možete dobiti koji bod (u tom slučaju tu funkciju nemojte navesti niti u .h niti u .cpp datoteci ili napravite neku trivijalnu implementaciju).

Ispravnost implementacija koje napišete bit će provjerena tako da ćemo mi napisati razne klijentske programe koji će deklarirati nekoliko varijabli zadane strukture, i na njima pozivati funkcije koje ste trebali napisati. Ako se poslani programi ne budu uspješno povezivali (linkali) s našim klijentskim programima, smatrat će se neispravnima.

Neki klijentski programi provjeravat će samo neke jednostavnije funkcije, dok će neki provjeravati sve funkcije koje trebate napisati. Provjera je potpuno automatska, tako da je od presudne važnosti da se pridržavate specifikacije. Nepridržavanje lako može uzrokovati osvojenih 0 bodova iz zadaće!

Naravno, za provjeru radi li implementacija prije nego što je pošaljete, preporučuje se da je testirate pomoću nekog klijentskog programa. No taj klijentski program ne šaljete!

Za sva pitanja vezana uz ovu zadaću javite se asistentu Vujčiću na jvujcic+rp1@gmail.com.