Języki Programowania, Zadanie 4, 31.10.2013

1. Proszę napisać **klasę Samochód** składającą się z:
   - pól prywatnych: `nr_rejestracji` (typu string), `marka` (typu string) oraz `przebieg` (składnik typu zmiennoprzecinkowego),
   - prywatnego pola statycznego `licznik` (typu całkowitego),
   - publicznej metody wypisującej informacje o samochodzie (alternatywnie można skorzystać z operatora strumienia wyjściowego),
   - publicznej metody zapisującej dane (rejestrację, markę, przebieg) do składników klasy
   - publicznej metody statycznej zwracającej wartość pola statycznego licznik

   Klasę należy zapisać w dwóch oddzielnych plikach .cpp i .h. Funkcja główna `main()` ***powinna znajdować się w oddzielnym pliku.***

2. Należy również zaimplementować konstruktor domyślny (Opel, WXF1234, 0), główny, kopiujący oraz destruktor. Dodać konstruktor przyjmujący dwa parametry: markę oraz przebieg zaimplementowany przy użyciu listy inicjalizacyjnej.

3. W programie głównym, należy:
   - stworzyć 4 obiekty napisanej klasy używając wszystkich konstruktorów, (1,5 p.)
   - stworzyć wskaźniki do obiektów oraz obiekty na które wskazują (operator new) używając wszystkich konstruktorów. Konstruktor kopiujący użyć dwa razy (kopiując zwykły obiekt, oraz kopiując obiekt na który wskazuje wskaźnik) - razem 5 nowych obiektów (0,5 p.)
   - użyć operatora delete do usunięcie wskaźników na obiekty klasy Samochod, wypisując przed i po tej operacji liczbę samochodów (korzystając z publicznej metody statycznej klasy Samochod), wypisać również liczbę samochodów w programie przed stworzeniem wszystkich samochodów (powinna być „0”) (1 p.)
   - **zgłosić prowadzącemu wykonanie pierwszej części zadania.**

4. Stworzyć tablicę na co najmniej 3 obiekty klasy Samochod. Należy do co najmniej jeden z nich wpisać dane używając metody „Zapisz” (1 p)
Napisać dwie funkcje o przeładowanej nazwie „Wypisz”:
   - int Wypisz(Samochod* tab, int n, string marka) – zwracającej ilość samochodów w tablicy tab o danej marce (jeśli n to liczba wszystkich samochodów w tablicy), oraz wypisującej informacje o tych samochodach na ekran.
   - int Wypisz(Samochod* tab, int n, double min, double max) – zwracającej ilość samochodów w tablicy tab o liczbie przejechanych kilometrów pomiędzy min a max, oraz wypisującej informacje o tych samochodach na ekran.

5. Zamienić string na char* dla numeru rejestracyjnego. Pamiętać o alokowaniu pamięci dla tablic we wszystkich konstruktorach. Funkcja `strcpy` z biblioteki `<cstring>` umożliwia kopiowanie dwóch ciągów znaków char*. (0.5 p)

6. Zakomentować kod testowy. Teraz należy zasymulować tworzenie bazy danych samochodów. Do zliczania samochodów używać pola statycznego „licznik”. W tym celu należy stworzyć tablicę wskaźników samochodów:

```
Samochod* baza_samochodow[1000];
```

Następnie program powinien w pętli umożliwiać: (0.5 p)
   - dodanie nowego samochodu do bazy danych (po naciśnięciu „1”)
   - wypisanie bazy danych samochodów (po naciśnięciu „2”)

Dodatkowo, dla bardzo odważnych i ambitnych:
Tworzymy program obsługujący bramki na autostradzie. Gdy dany samochód wjeżdża na autostradę zostaje dodany do listy samochodów, jeśli wyjeżdża, zostaje usunięty z tej listy. Ponadto, niektórzy klienci mają wykupione karnety, a ich dane również widnieją w bazie.

Baza danych zarówno samochodów jak i klientów ma formę tablicy wskaźników:
`Samochod* baza_samochodow[1000];`
`Klient* baza_klientow[1000];`
Przy tworzeniu nowych samochodów korzystamy z operatora ***new***. Podobnie postępujemy z klientami.

Program powinien umożliwiać (w pętli):
- *Dodanie nowego samochodu* (kamery przy bramkach automatycznie odczytują znaki na tablicy rejestracyjnej = dane wpisywane z klawiatury). Jeśli samochód z daną tablicą rejestracyjną widnieje już w bazie danych klientów (klient ma wykupiony karnet), to nowo stworzony samochód ma ustawioną opłatę równą 0. Jeśli dany samochód nie widnieje w bazie, opłata powinna zostać ustawiona na 7.5. Należy zapisywać opłatę (double) zamiast przebiegu samochodów.
- *Dodanie nowego klienta* (podajemy imię, nazwisko, oraz nr rejestracji samochodu z klawiatury)
- *Wypisanie aktualnie korzystających z autostrady samochodów*
- *Wypisanie bazy danych klientów*
- *Usunięcie samochodu / klienta z bazy (w obu przypadkach wyszukujemy odpowiedniego obiektu bazując na numerach tablicy rejestracyjnej podanej z klawiatury)*

Korzystamy ze stworzonej przed chwilą klasy Samochód oraz tworzymy nową klasę:
`Klient`
Pola składowe (prywatne!):
- `Imię` (std::string)
- `Nazwisko` (std::string)
- `Rejestracja` (std::string)
- `ilosc` (int, pole statyczne**)** - podliczające sumaryczną liczbę klientów w systemie
Konstrukor
- `Klient(string,string,string)` – imię, nazwisko, oraz nr tablicy samochodu klienta. Pole `ilosc` powinno zostać zwiększone o 1.
Destruktor: pole `ilosc` zmniejszone o 1.

- `bool SprawdzRejestracje(std::string r)` – porównuje otrzymaną rejestrację r z rejestracją zapisaną dla danego klienta, zwraca odpowiednio `true` lub `false`;
- `static int IloscKlientow()` - metoda statyczna, zwracająca wartość statycznego pola `ilosc`.
- `void WypiszKlienta()` - wypisuje na ekran imię, nazwisko oraz rejestrację samochodu klienta.

Zadeklarować statyczne tablice wskaźników: `baza_samochodow` oraz `baza_klientow`.
Program powinien w pętli umożliwiać:

- dodanie nowego samochodu do bazy danych
- wypisanie bazy danych samochodów
- dodawanie nowych klientów (wskaźniki!) do bazy (w odpowiednie miejsce tablicy, opierając się na wartości pól statycznych) - użycie operatora `new` **(dojście tutaj: +0.5 p)**
- wypisanie bazy danych klientów
- dodanie nowego samochodu do bazy danych – program pyta o numer rejestracji (Uwaga! Jeśli dana rejestracja istnieje już w bazie danych klientów należy ustawić opłatę na 0 zł! W przeciwnym wypadku koszt przejazdu wynosi 7.5 zł.)
- usunięcie samochodu z listy (bazując na numerze rejestracji) **(dojście tutaj: +0.5 p)**