Języki Programowania, Czwartek 24.10.2013

Zadanie 3

Ideą programu jest stworzenie "bazy danych" w C++ dla komisu z używanymi samochodami. Sklep ma miejsce na 70 samochodów (można zadeklarować tablicę statyczną na [70] pojazdów), a każdy samochód ma: markę, nazwę, kolor, wiek oraz cenę. Nowe samochody można dodawać, oraz wypisywać wszystko, co komis ma na dostępne.

1) Stwórz klasę Samochod zawierającą pięć składników prywatnych: marka (std::string), nazwa (std::string), kolor (char*), wiek (int) oraz cenę (double). Tablica znaków przeznaczona na nazwę koloru w klasie powinna być alokowana dynamicznie (operator **new**). Klasa powinna być podzielona na dwa oddzielne pliki: samochod.h i samochód.cpp. W trzecim pliku powinna znajdować się funkcja główna.

Klasa powinna mieć dwie funkcje składowe (tzw. metody):

- `SetMarka(const string marka)` - ustawiająca markę danego samochodu
- `SetNazwa(const string nazwa)` - ustawiająca nazwę (model) danego samochodu
- `SetKolor(const char* color)` - zapisująca dany kolor do odpowiedniego składnika klasy
- `wypisz()` - wypisująca na ekran informacje o danym samochodzie (po spacjach: marka, nazwa, kolor, wiek oraz cenę)

2) Należy również zaimplementować konstruktor domyślny, główny, kopiujący oraz destruktor. Destruktor należy opatrzeć komentarzem - wypisywaniem na domyślne wyjście „Został użyty destruktor".

3) Od teraz **zawsze** należy owijać definicje klas w plikach nagłówkowych w strukturę „ifndef":

```
#ifndef _NAZWA_TOKENU
#define _NAZWA_ TOKENU
      class klasa{...}; - definicja naszej klasy
#endif
```

Działa to w ten sposób: jeśli token _NAZWA_TOKENU nie był jeszcze definiowany: wejdź do środka (zdefiniuj token i zapisz go sobie w pamięci, po czym wykonaj wszystkie instrukcje znajdujące się w środku). Jeśli już go deklarowałeś – przejdź od razu do #endif. W ten sposób już nigdy nie będziemy się przejmować kolejnością ładowanych bibliotek – jeśli zostały one wcześniej załadowane, kompilator w ogóle pominie taką instrukcje.

4) W programie głównym, należy:

• stworzyć obiekty napisanej klasy używając konstruktora domyślnego, głównego i kopiującego, wypisać je na ekran (1,5 p.)

1: (Mazda, 636, czerwony, 13 letni, 3000 zł) 2: (Ford, Focus, niebieski, 2 letni, 25000 zł) 3: (Opel, Astra, zielony, 5 letni, 17000 zł)

• stworzyć wskaźniki do obiektów (operator new) napisanej klasy używając konstruktora domyślnego, głównego i kopiującego, wypisać je na ekran (0,5 p.)

• użyć operatora delete do usunięcia wskaźnika na obiekty klasy Samochod (0.5 p.)

5) Należy przygotować „bazę danych" z możliwością dodawania, usuwania oraz wypisywania samochodów używając w tym celu tablicy statycznej na 70 pojazdów. W głównym programie (funkcji main) wymagana jest instrukcja typu switch - case, pytająca w pętli o podanie liczby od 1-3 lub 0.

- Wpisanie 1 - wypisanie wszystkich zapisanych samochodów na ekran. (0.5p) (`cout<<zmienna`)
- Wpisanie 2 - dodanie nowego samochodu do "bazy danych". (0.5 p) (`cin>>zmienna`)
- Wpisanie 3 - usuwanie samochodu o konkretnym indeksie z „bazy danych" (należy pamiętać o przesuwaniu kolejnych samochodów w tablicy). (1p)
- Wpisanie 4 - zmiana ceny samochodu o konkretnym indeksie z „bazy danych" (0.5p)
- Wpisanie 0 - powinno przerwać pętle oraz spowodować opuszczenie programu.
- Wpisanie liczby innej niż dozwolone - program zakomunikuje, iż została wprowadzona błędna wartość.

Przydatne nagłówki (biblioteki):

```
#include <iostream>
#include <cstring>  (przydatne funkcje: strlen, strcpy)
```

6)Program powinien być kompilowany przy pomocy komendy „`make`” (należy stworzyć odpowiedni Makefile!)

Makefile

```
CXX = g++
CXXFLAGS = -Wall
LFLAGS =

OBJS = samochod.o main.o

all: prog

prog: $(OBJS)
	$(CXX) $(CXXFLAGS) $^ -o $@

clean:
	rm -f *.o prog

.PHONY: all clean
```

Zadania dodatkowe:

- Dynamiczna „baza danych”: użycie klasy vector lub queue.