Języki Programowania, Zadanie 3, 20.10.2014

Ideą programu jest stworzenie “bazy danych” w C++ dla sklepu z używanymi rowerami. Sklep ma miejsce na 40 rowerów (można zadeklarować tablicę statyczną na [40] rowerów), a każdy rower ma: kolor, wiek oraz cenę. Można nowe rowery dodawać, oraz wypisywać wszystko, co sklep ma na składzie.

1) Stwórz klasę Rower zawierającą trzy składniki prywatne: kolor (string), wiek (int) oraz cenę (double). Klasa powinna być podzielona na dwa oddzielne pliki: rower.h i rower.cpp. W trzecim pliku powinna znajdować się funkcja główna.

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

- `SetKolor(std::string color)` - zapisująca dany kolor do odpowiedniego składnika klasy
- `Wypisz()` - wypisująca na ekran informacje o danym rowerze (po spacjach: 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) W programie głównym, należy:

• stworzyć obiekty napisanej klasy używając konstruktora domyślnego, głównego i kopiującego, wypisać je na ekran (0,5 p.)
1: (czerwony, 3 letni, 300.5 zł) 2: (niebieski, 2 letni, 400 zł) 3: (niebieski, 2 letni, 400 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 Rower (0.5 p.)

4) Należy do klasy Rower dodać pole statyczne int fIlosc zliczające ilość rowerów utworzonych w programie. Każde użycie dowolnego konstruktora zwiększa ilość rowerów o 1, każde użycie destruktora zmniejsza ilość rowerów o 1. Ponadto należy dodać statyczną metodę zwracającą pole statyczne fIlosc. W programie należy trzy razy użyć metody statycznej do wypisania ilości rowerów: na samym początku (gdy żaden obiekt nie jest stworzony), po stworzeniu wszystkich obiektów i wskaźników, oraz po usunięciu obiektu przy użyciu delete. (1 p)

5) Należy przygotować „bazę danych” z możliwością dodawania, usuwania oraz wypisywania rowerów używając w tym celu tablicy na 40 rowerów. W głównym programie (funkcji main) wymagana jest instrukcja typu switch - case, pytająca w pętli o podanie liczby od 1-4 lub 0.

- Wpisanie liczby innej niż dozwolone - program zakomunikuje, iż została wprowadzona błędna wartość.
- Wpisanie 0 - powinno przerwać pętle oraz spowodować opuszczenie programu. (0.5 p)
- Wpisanie 1 - dodanie nowego roweru do “bazy danych”. (wczytywanie z klawiatury: `cin>>zmienna`) (0.5 p)
- Wpisanie 2 - wypisanie wszystkich zapisanych rowerów na ekran. (0.5p)
- Wpisanie 3 - zmiana koloru roweru o konkretnym indeksie z „bazy danych” (0.5p)

6) 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.***

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

Makefile

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

OBJS = rower.o main.o

all: prog

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

clean:
	rm -f *.o prog

.PHONY: all clean
```

Zadania dodatkowe:

- Wpisanie 4 - usuwanie roweru o konkretnym indeksie z „bazy danych” (należy pamiętać o przesuwaniu kolejnych rowerów w tablicy).
- Zamienić „string” na char*. || `#include <cstring>  (przydatne funkcje: strlen, strcpy)`
- Dynamiczna „baza danych”: użycie klasy vector lub queue.