

Języki programowania, Czwartek 30.10.2014

Zadanie 4

Zadanie ma na celu dalsze przećwiczenie użycia wielu klas, tworzenia konstruktorów, destruktorów, metod i pól statycznych w klasach. Program demonstruje też wykorzystanie biblioteki STL języka C++. Do tego stworzymy klasę `Triangle`.

Należy stworzyć klasę `Triangle`, która będzie zawierać pola:

```
char *name,  
double a,  
double b,  
double c,  
oraz pole statyczne int fCount.
```

Klasa powinna zawierać również dwa konstruktory:

`Triangle()` - konstruktor bez parametrów (ustawia pole `name` na "nazwa" oraz `a=b=c=0`),

`Triangle(char* Name, double A, double B, double C)` – konstruktor z parametrami, konstruktor kopiujący oraz destruktor. Każde użycie dowolnego konstruktora zwiększa ilość trójkątów o 1, każde użycie destruktora zmniejsza ilość trójkątów o 1.

Oprócz tego, należy stworzyć metody "set" i "get" do każdego z pól i statyczną metodę zwracającą pole statyczne `fCount`.

Część I

Do wykonania:

1. Stworzenie klasy `Triangle` i użycie jej w programie (stworzenie obiektu, użycie konstruktora, wypisanie elementów składowych). **1 p.**
2. Stworzenie wskaźnika na obiekt `Triangle`, użycie konstruktora wraz z operatorem `new` i usunięcie obiektu operatorem `delete`. **1 p.**

Po każdym stworzeniu i usunięciu obiektu używamy metody statycznej do wypisania ilości trójkątów.

W drugiej części zadania będziemy chcieli stworzyć listę obiektów typu `Triangle`. Do tego celu wykorzystamy listę podwójnie linkowaną (`double-linked list`) z tzw. Standardowej Biblioteki Szablonów języka C++ (STL). Przykład ten pokazuje możliwości wykorzystania gotowych algorytmów i struktur danych dostępnych w języku C++.

Użycie listy z biblioteki standardowej pokazuje przykład poniżej:

```
#include <string>  
#include <list>  
#include <iostream>  
  
using namespace std;  
  
class A  
{  
public:  
    string fCiagZnakow;  
    int fLiczba;  
};  
  
int main()  
{  
    A a1;  
    A a2;  
  
    list<A> lista; //lista obiektow klasy A  
  
    //wstawianie elementow na koniec listy  
    lista.push_back(a1);  
    lista.push_back(a2);
```

```

//iterowanie po elementach listy
list<A>::iterator i;
for(i=lista.begin();i!=lista.end();i++)
{
    cout<<(*i).fCiagZnakow<<endl;
    cout<<(*i).fLiczba<<endl;
    cout<<endl;
}

//usuwanie elementu z konca listy
lista.pop_back();

//zwrocenie elementu z konca listy
A a3 = lista.back();

return 0;
}

```

Część II

Do wykonania:

1. Zadeklarować listę obiektów typu `Triangle`, wstawić minimum 3 elementy (deklarując je wcześniej) i wypisać listę iterując po nich, po czym usunąć ostatni element (użyć metody statycznej do wypisania ilości trójkątów). **1 p.**
2. Tworzymy listę wskaźników na obiekt `Triangle`, wstawić do listy trzy wskaźniki: jeden przez referencję do obiektu, który stworzyliśmy na początku zadania, drugi tworząc wskaźnik i alokując mu pamięć operatorem `new` i korzystając z konstruktora, trzeci, wstawiając bezpośrednio w metodzie `push_back` operator `new` oraz konstruktor, np. w podobny sposób:
`lista.push_back(new A());`
 Następnie usuwamy ostatni element z listy i wypisujemy ilość elementów danego typu oraz ponownie iterujemy po liście. Dlaczego liczba elementów jest inna niż ilość elementów w liście (przedyskutować z prowadzącym)? Tworzymy wskaźnik na obiekt i pobieramy ostatni element z listy metodą `back` (ustawiamy to przed usunięciem ostatniego elementu!). Następnie, na końcu, po usunięciu ostatniego elementu z listy, używamy na tym wskaźniku operatora `delete` i ponownie sprawdzamy ilość obiektów danego typu. **1 p.**
3. Na koniec tworzymy bazę danych trójkątów – używamy instrukcji `switch-case`. Program prosi użytkownika o podanie co ma zrobić: 0 – dodanie nowego trójkąta do końca listy, 1 – usunięcie trójkąta z końca listy, 2 – wypisanie wszystkich elementów listy, 3 – wyjście z programu. **1 p.**

Uwagi:

1. Do kompilacji używamy kompilatora `g++`:
`g++ -Wall main.cpp klasa1.cpp klasa2.cpp -o main`
1. Strony z dokumentacją oraz przykładami:
<http://www.cplusplus.com/reference/stl/list/>
<http://www.yolinux.com/TUTORIALS/LinuxTutorialC++STL.html>