Techniki programowania (INP001002Wl) – laboratorium

Lista 4

Minimalny próg zaliczenia listy: 12 punktów.

Rozwiązania należy oddać do 27 maja. Praca nad rozwiązaniami powinna być udokumentowana w repozytorium.

1. (4 pkt.) Zaprojektuj i zaimplementuj klasę reprezentującą liczby zespolone. Interfejs klasy powinien obejmować przeciążone operatory pozwalające na:
   - dodawanie dwóch liczb zespolonych,
   - dodawanie liczby zespolonej i liczby rzeczywistej,
   - dodawanie liczby rzeczywistej i liczby zespolonej,
   - odejmowanie dwóch liczb zespolonych,
   - odejmowanie liczby zespolonej i liczby rzeczywistej,
   - odejmowanie liczby rzeczywistej i liczby zespolonej,
   - mnożenie dwóch liczb zespolonych,
   - mnożenie liczby zespolonej przez liczbę rzeczywistą,
   - mnożenie liczby rzeczywistej przez liczbę zespoloną,
   - sprzężenie liczby zespolonej,
   - wypisanie do strumienia **`ostream`**.

   Aby uniknąć kolizji z biblioteką **`complex`** umieść deklarację klasy w pliku nagłówkowym **`complex0.h`**. [1]

2. (4 pkt.) Zaprojektuj i zaimplementuj klasę bazową **`Figure`**, oraz klasy pochodne: **`Circle`**, **`Triangle`**, **`Rectangle`**. Interfejs publiczny klasy bazowej powinien zawierać czysto wirtualne metody:

   ```
   virtual double area() const = 0;
   virtual std::ostream& print(std::ostream&) const = 0;
   ```

   oraz operator:

   ```
   std::ostream& operator<<(std::ostream& out, Figure& f).
   ```

   Przykładowy kod wykorzystujący wymienione klasy:

```
#include <iostream>
#include "Figure.h"
#include "Circle.h"
#include "Triangle.h"
#include "Rectangle.h"

using namespace std;

int main()
{
    Figure** figures = new Figure*[3];
```

```
    figures[0] = new Circle(1);
    figures[1] = new Triangle(1,1,1);
    figures[2] = new Rectangle(2,3);

    for(int i=0; i<3; i++){
        cout << *figures[i] << endl;
    }
    cout << "Pola figur:" << endl;
    for(int i=0; i<3; i++){
        cout << figures[i]->area() << endl;
    }
    for(int i=0; i<3; i++){
        delete figures[i];
    }
    delete[] figures;
}
```

3. (8 pkt.) **STL, plikowe wejście/wyjście**
   Zaimplementuj program obsługujący niewielką książkę adresową, która umożliwia użytkownikowi wprowadzanie nazw oraz adresów e-mail, usuwanie oraz zmienianie wpisów, wyświetlanie zawartości książki adresowej. Do przechowywania danych wykorzystaj szablon mapy z biblioteki STL (**`map<string, string>`**). Dane powinny być przechowywane na dysku – przy uruchomieniu program powinien wczytywać dane z pliku, natomiast przy zamknięciu plik powinien być aktualizowany. [2]

4. (9 pkt.) **Szablony**
   Zaimplementuj szablon klasy pozwalającej na przechowywanie informacji w postaci par: klucz/wartość, w którym typ klucza i typ wartości są parametrami szablonu. Klasa powinna przechowywać dane w postaci drzewa poszukiwań binarnych (dla każdego węzła *x* klucze w jego lewym poddrzewie są mniejsze niż klucz w *x*, natomiast klucze w jego prawym poddrzewie są większe niż klucz w *x* – w konsekwencji klucze są unikatowe). Zaimplementuj metody pozwalające na:
   - dodawanie nowej pary do zbioru (lub nadpisanie istniejącej wartości jeśli klucz istnieje w zbiorze),
   - usuwanie pary o zadanym kluczu,
   - wyświetlenie zawartości zbioru w rosnącej kolejności kluczy.

[1] S. Prata, *Szkoła programowania C++*, rozdział 11.
[2] A. Allain, *Przewodnik dla początkujących C++*, rozdziały 18 i 28.