zadanie nr 7 termin realizacji: 6 kwiecień 2005

C++

elementy obliczalne (dziedziczenie) studia wieczorowe

Zdefiniuj abstrakcyjną klasę bazową dla obiektów, które posiadają swoją nazwę, i które mogą podawać nam wartości rzeczywiste w zależności od tego, jakie dane do obiektów tych klas wcześniej wprowadzimy. Sprawdź w konstruktorze, czy podana nazwa jest poprawnym identyfikatorem (ciąg złożony z liter i cyfr rozpoczynający się od litery).

```
class DajacySiePoliczyc
{
public:
    const string nazwa;
public:
    DajacySiePoliczyc (string naz);
    virtual ~DajacySiePoliczyc ();
public:
    virtual double obliczWynik () = 0;
    virtual void podajDane (double wart) = 0;
};
```

Następnie zdefiniuj ukonkretnienie klasy `DajacySiePoliczyc` w postaci klasy `Zmienna`, służącej do przechowywania wartości skojarzonych z określoną nazwą.

```
class Zmienna: public DajacySiePoliczyc
{
protected:
    double wartosc;
public:
    Zmienna (string naz, double wart=0);
    virtual ~Zmienna ();
public:
    virtual double obliczWynik (); // odczytanie pamiętanej wartości
    virtual void podajDane (double wart); // zapamiętanie nowej wartości
};
```

Zdefiniuj też klasę `Funkcja` jako inne ukonkretnienie klasy `DajacySiePoliczyc`. W konstruktorze należy sprawdzić czy podana arność jest prawidłowa (nieujemna) i przydzielić pamięc na tablicę argumentów wykorzystywanych przy wywoływaniu funkcji `obliczWynik` (tablicę tą należy zwolnić w destruktorze). Funkcję `obliczWynik` można uruchomić dopiero po wprowadzeniu wszystkich argumentów funkcją `podajDane` (trzeba ich wprowadziś dokładnie tyle ile wynosi arność funkcji). Obliczenie wyniku funkcji powoduje usunięcie wszystkich argumentów. Tak więc ponowne obliczenie musi być poprzedzone wprowadzeniem nowych danych.

```
class Funkcja: public DajacySiePoliczyc
{
public:
    const int arnosc; // arność funkcji (liczba argumentów)
protected:
    double *argumenty; // argumenty
    int liczba; // liczba dostarczonych argumentów
public:
    Funkcja (string naz, int arn);
    virtual ~Funkcja ();
public:
    virtual double obliczWynik () // sprawdzenie liczby argumentów
    {
        if (liczba!=arnosc) throw Blad("za malo argumentow");
        liczba = 0;
    }
    virtual void podajDane (double wart) // zapamiętanie nowej wartosci
    {
        if (liczba==arnosc) throw Blad("za duzo argumentow");
        argumenty[liczba++] = wart;
    }
};
```

Następnie zdefiniuj kilka klas pochodnych od klasy `Funkcja`, obliczających podstawowe funkcje arytmetyczne i trygonometryczne: `Pi`, `E`, `Abs`, `Min`, `Max`, `Ln`, `Log`, `Exp`, `Power`, `Sin`, `Cos`, `Atan`, `Acot`. Na przykład:

```
class Pi: public Funkcja
{
public:
    Pi (): Funkcja("pi",0) {}
public:
    virtual double obliczWynik ()
    {
        Funkcja::obliczWynik();
        return 3.14159265358979323746264;
    }
};
```

Do zgłaszania wyjątków zdefiniuj specjalną klasę `Blad`.

```
struct Blad
{
    const string opis;
    Blad (string op): opis(op) {}
    Blad (const Blad &bl): opis(bl.opis) {}
};
```

Definicje wszystkich klas podziel na część nagłówkową i źródłową. Następnie w osobnym pliku umieść program testowy, który sprawdzi poprawność zdefiniowanych przez Ciebie klas reprezentujących funkcje i zmienne.

Uwaga! Klasy zdefiniowane w tym zadaniu będą wykorzystywane w następnych zadaniach, a zatem warto napisać je bardzo starannie.

Uniwersytet Wrocławski
Instytut Informatyki
ul. Przesmyckiego 20
PL-51-151 Wrocław

Paweł Rzechonek