Zadanie 6 *25–28 kwietnia 2017 r.*

JĘZYK PROGRAMOWANIA C++

WYRAŻENIA ARYTMETYCZNE

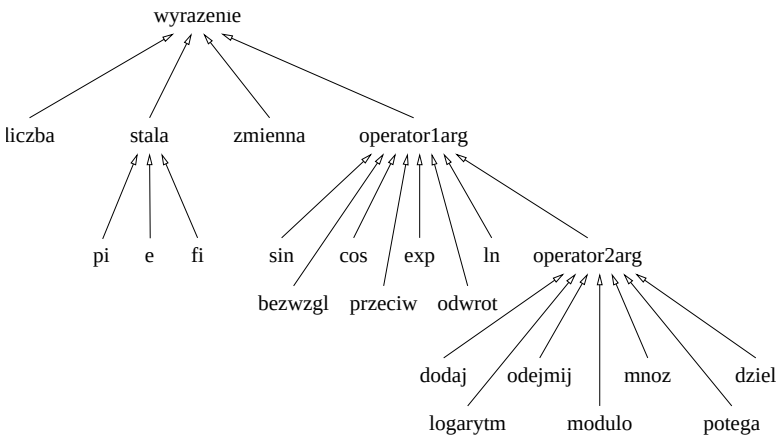
Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Wyrażenie arytmetyczne w języku programowania to dowolne wyrażenie typu liczbowego. Może być ono złożone ze zmiennych, liczb, symboli działań, nawiasów, funkcji, itp.

W matematyce podobne znaczenie ma wyrażenie algebraiczne, które jest złożone z jednego lub większej liczby symboli algebraicznych, czyli stałych lub zmiennych, połączonych znakami działań (dodawania, odejmowania, mnożenia, dzielenia, potęgowania, itp.) i ewentualnie nawiasów, zgodnie z regułami notacji matematycznej. Nie są natomiast wyrażeniami algebraicznymi zapisy, w których uczestniczą symbole funkcji albo relacji.

Zadanie.

Zdefiniuj abstrakcyjną klasę bazową `wyrazenie`, reprezentującą wyrażenie arytmetyczne. W klasie tej umieść deklaracje abstrakcyjnych metod `oblicz()` (jej zadaniem w klasach potomnych będzie obliczenie wartości wyrażenia i przekanie wyniku typu `double`) oraz `opis()` (ta metoda ma zwracać napis typu `string` reprezentujący całe wyrażenie z dopisanymi niezbędnymi nawiasami — należy uwzględnić priorytety operatorów, na przykład priorytet mnożenia jest wyższy niż priorytet dodawania, oraz ich łączność, na przykład mnożenie jest lewostronnie łączne a potęgowanie prawostronnie).



Następnie zdefiniuj klasy dziedziczące po klasie `wyrazenie`, które będą reprezentowały operandy i operatory. Do operandów zaliczamy liczby (stała zmiennopozycyjna typu `double`),

zmienne (zmienna ma mieć określoną nazwę `string`, przez którą będzie można odwołać się do zbioru zmiennych i stamtąd odczytać wartość) oraz stałe (stałe mają określoną nazwę typu `string`, za którą kryje się pewna ustalona wartość). Operatory natomiast to podstawowe symbole operacji arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie oraz jednoargumentowa operacja zmiany znaku na przeciwny) i wybrane funkcje matematyczne (sinus, cosinus, logarytm,...). Klasy te powinny być tak zaprojektowane, aby można z nich było zbudować drzewo wyrażenia: obiekty klas `liczba`, `zmienna` czy stałe dziedziczące po `stala` to liście a operatory i funkcje unarne albo binarne to węzły wewnętrzne w takim drzewie. W klasach potomnych ponadpisuj metody `oblicz()` oraz `opis()`.

Na koniec napisz krótki program testowy, sprawdzający działanie obiektów tych klas. W swoim programie skonstruuj następujące drzewa obliczeń z wykorzystaniem zmiennej `x`:

```
((x-1)*x)/2
(3+5)/(2+x*7)
2+x*7-(y*3+5)
cos((x+1)*x)/e^x^2
```

Wypisz te wyrażenia korzystając z metody `opis()` a potem oblicz i wypisz wartości tych wyrażeń dla wartości zmiennej `x` z zakresu od 0 do 1 ze skokiem co 0.01 stosując metodę `oblicz()`.

Uzupełnienie.

Zmienne pamiętaj w zbiorze asocjacyjnym, czyli w obiekcie typu `vector<pair<string,double>>`. Zbiór ten umieść jako prywatne pole statyczne w klasie `zmienna` i dopisz kilka publicznych statycznych metod pozwalających zarządzać tym zbiorem.

Przykład.

Wyrażenie `pi-(2+x*7)` należy zdefiniować następująco:

```
wyrazenie *w = new odejmij(
    new pi(),
    new dodaj(
        new liczba(2),
        new mnoz(
            new zmienna("x"),
            new liczba(7)
        )
    )
);
```

Potem można obliczać wartość takiego wyrażenia nadając zmiennej `x` różne wartości .

Elementy w programie, na które należy zwrócić szczególną uwagę.

- Podział programu na pliki nagłówkowe i źródłowe.
- Definicja abstrakcyjnej klasy `wyrazenie` z czysto wirtualnymi metodami abstrakcyjnymi.
- Nadpisanie metod `oblicz()` i `opis()` w klasach potomnych.
- W funkcji `main()` należy przetestować obiekty wszystkich klas nieabstrakcyjnych.