laboratorium: zadanie 6 termin: 17–21 kwietnia 2023 r.

KURS JĘZYKA C++

WYRAŻENIA ARYTMETYCZNE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Prolog.

Wyrażenia arytmetyczne mają fundamentalne znaczenie w każdym języku programowania — są to dowolne wyrażenia typu liczbowego złożone z liczb, zmiennych, funkcji, operatorów, nawiasów itp. Wyrażenia arytmetyczne nie stanowią samoistnych instrukcji ale są ich częścią składową.

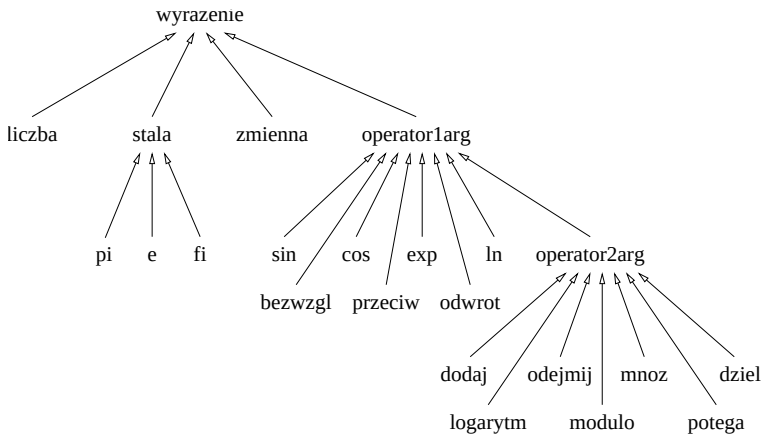
Drzewa wyrażeń pozwalają na dynamiczne budowanie wyrażeń w trakcie działania programu. Każde wyrażenie można przedstawić w postaci drzewa (najczęściej drzewa binarnego), w którym liście reprezentują argumenty a węzły wewnętrzne operacje arytmetyczne (najczęściej operacje dwuargumentowe) albo funkcje matematyczne o dowolnej arności (wtedy też drzewa wyrażeń niekoniecznie muszą być drzewami binarnymi). Argumentami operatorów albo funkcji w takim drzewie są inne wyrażenia (poddrzewa wyrażeń). Sposób łączenia ze sobą poddrzew w jedno wyrażenie determinują priorytety i łączność operatorów oraz rozstawienie nawiasów (operacja wykonująca się na samym końcu w trakcie obliczania wyrażenia zostanie umieszczona w korzeniu drzewa).

Zadanie.

Zdefiniuj abstrakcyjną klasę bazową `wyrazenie`, reprezentującą wyrażenie arytmetyczne. W klasie tej umieść deklaracje abstrakcyjnych metod `oblicz()` oraz `zapis()`. Metoda `oblicz()` doprecyzowana w klasach potomnych będzie obliczać wartość wyrażenia i zwracać wynik typu `double`; metoda `zapis()` ma zwracać napis typu `string` reprezentujący całe wyrażenie wraz 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 jest łączne prawostronnie).

Następnie zdefiniuj klasy dziedziczące po klasie `wyrazenie`, które będą reprezentowały operandy i operatory. Do operandów zaliczamy liczby (wartość zmiennopozycyjna typu `double`), zmienne (zmienna ma mieć unikalną nazwę `string`, przez którą będzie można odwołać się do zbioru zmiennych i stamtąd pobierać skojarzoną wartość) oraz stałe (stałe mają unikalną nazwę typu `string`, za którą kryje się pewna ustalona wartość). Wszystkie operandy (liczby, stałe i zmienne) zdefiniuj tak, aby zablokować możliwość dziedziczenia po tych klasach.

Zmienne pamiętaj w zbiorze asocjacyjnym 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 (dodawanie nowej zmiennej, usuwanie i modyfikacja istniejących zmiennych).



Operatory natomiast to podstawowe operacje arytmetyczne (dodawanie, odejmowanie, mnożenie, dzielenie, potęgowanie oraz jednoargumentowa operacja zmiany znaku na przeciwny) i wybrane funkcje matematyczne (sinus, cosinus, logarytm, funkcja eksponencjalna itp.). Pamiętaj o arności operatorów i funkcji, a dodatkowo o priorytetach i łączności operatorów.

Hierarchia dziedziczenia dla tych klasa powinna być tak zaprojektowana, aby można z nich było zbudować drzewo wyrażenia: obiekty klas reprezentujących liczbę, zmienną czy stałą to liście a operatory i funkcje (unarne albo binarne) to węzły wewnętrzne w takim drzewie. W klasach pochodnych od klasy `wyrazenie` ponadpisuj metody `oblicz()` oraz `zapis()`. Zablokuj też możliwość kopiowania wyrażeń (brak konstruktora i przypisania kopiującego).

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

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

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

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.

Istotne elementy programu.

- Optymalna hierarchia klas pozwalająca definiować różne elementy wyrażenia; na szczycie tej hierarchii ma się znaleźć abstrakcyjna klasa `wyrazenie` z czysto wirtualnymi metodami abstrakcyjnymi `oblicz()` i `zapis()`.
- Nadpisanie metod `oblicz()` i `zapis()` w klasach potomnych; wykorzystanie priorytetów do zminimalizowania liczby wypisywanych nawiasów przez metodę `zapis()`.
- Zablokowanie kopiowania i przenoszenia dla wyrażeń.
- Zgłaszanie wyjątków w konstruktorach i funkcjach składowych.
- Podział programu na pliki nagłówkowe i pliki źródłowe (wyodrębniony osobny plik z funkcją `main()`); w funkcji `main()` należy przetestować obiekty wszystkich klas nieabstrakcyjnych.