laboratorium: zadanie 5 termin: 20–23 kwietnia 2021 r.

# KURS JĘZYKA C++

## WYRAŻENIA ARYTMETYCZNE

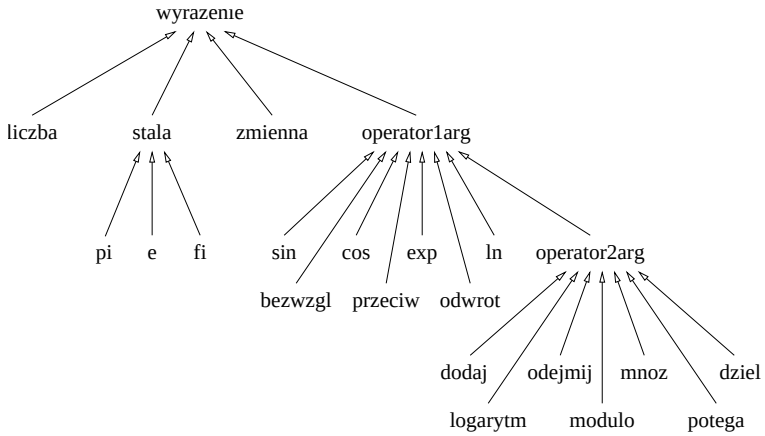
Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

**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.

**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 określoną nazwę `string`, przez którą będzie można odwołać się do zbioru zmiennych i stamtąd pobierać skojarzoną 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, itp). 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 `zapis()`.

Zablokuj możliwość kopiowania wyrażeń.

Na koniec napisz 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 `zapis()` a potem oblicz i wypisz ich wartości dla zmiennej `x` z zakresu od 0 do 1 ze skokiem co 0.01 stosując metodę `oblicz()`.

**Uzupełnienie.**

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, usuwanie i modyfikacja zmiennych).

**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.**

- Podział programu na pliki nagłówkowe i pliki źródłowe (wyodrębniony osobny plik z funkcją `main()`).
- Zdefiniowanie odpowiedniej hierarchii klas pozwalających zdefiniować różne elementy wyrażenia; na szczycie tej hierarchii ma się znaleźć abstrakcyjna klasa `wyrazenie` z czysto wirtualnymi metodami abstrakcyjnymi.
- 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.
- W funkcji `main()` należy przetestować obiekty wszystkich klas nieabstrakcyjnych.