

# kurs języka C++

## wrażenia 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 i nawiasów. Wyrażenia arytmetyczne nie stanowią samoistnych instrukcji ale są ich częścią.

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

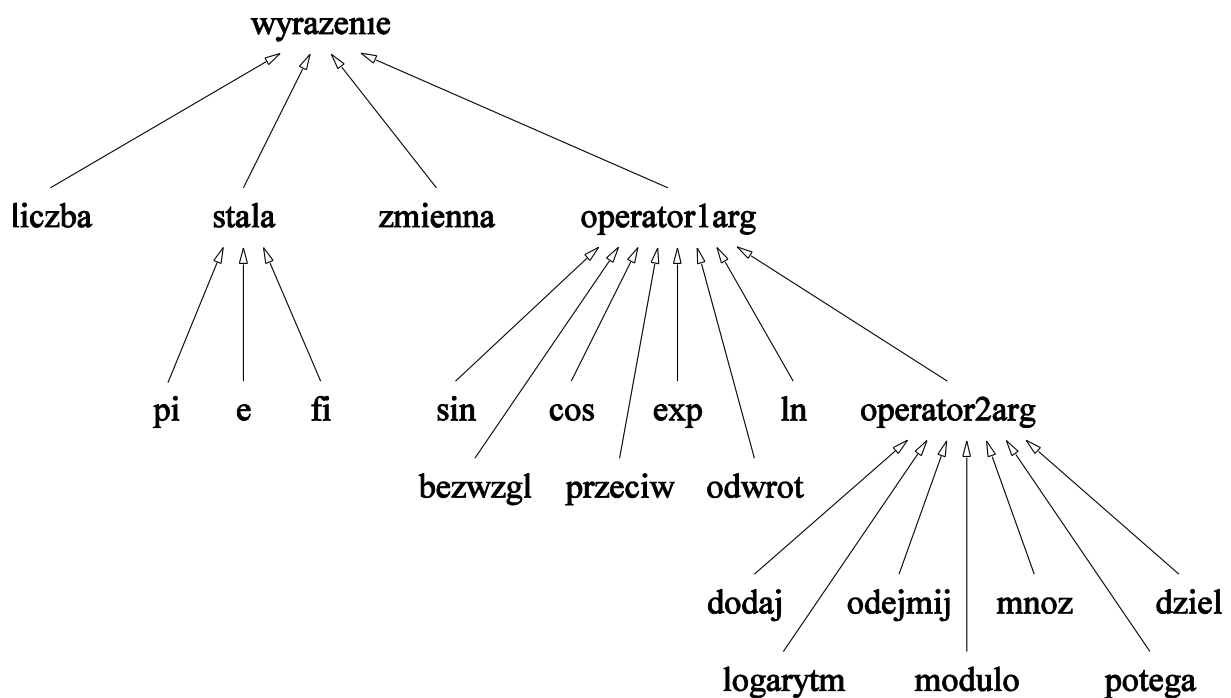
Zdefiniuj abstrakcyjną klasę bazową `wyrazenie`, reprezentującą wyrażenie arytmetyczne. W klasie tej umieść deklaracje abstrakcyjnych metod `oblicz()` oraz `zapis()`. Metoda `oblicz()` doprecyzowana w klasach potomnych będzie obliczać wartość wyrażenia i zwracać wynik typu `double`; metoda `zapis()` ma zwracać napis typu `string` reprezentujący całe wyrażenie wraz z dopisanymi niezbędnymi nawiasami – należy przy tym 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 prawo-stronnie).

Następnie zdefiniuj klasy dziedziczące po klasie `wyrazenie`, które będą reprezentowały operandy i operatory. Do operandów zaliczamy liczby (opakowana wartość zmiennopozycyjna typu `double`), zmienne (zmienna ma mieć określoną nazwę typu `string`, przez którą będzie można odwołać się do zbioru zmiennych i stamtąd pobrać skojarzoną wartość) oraz stałe (stałe mają określoną nazwę typu `string`, za którą kryje się pewna ustalona wartość typu `double`).

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).

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



Hierarchia dziedziczenia dla tych klas powinna być tak zaprojektowana, aby można z nich było zbudować drzewo wyrażenia: obiekty klas reprezentujących liczbę, zmienną czy stałą liście w drzewie a operatory i funkcje unarne albo binarne to węzły wewnętrzne. Można rozważyć odseparowanie operatorów i funkcji (aby uprościć implementację metody zapisującej wyrażenie w postaci łańcucha znaków typu `string`). W klasach potomnych w stosunku do klasy wyrażenie ponadpisuj metody `oblicz()` oraz `zapis()`. Zablokuj też możliwość kopiowania wyrażień.

Na koniec napisz program testowy, sprawdzający działanie obiektów tych klas. W swoim programie skonstruuj kilka drzew obliczeń z wykorzystaniem zmiennych  $x$  i  $y$ , na przykład:

$$((x-1) * x) / 2$$

$$(3+5) / (2+x*7)$$

$$2+x*7 - (y*3+5)$$

$$\cos((x+1) * \pi) / e^{x^2}$$

Wypisz te wyrażenia korzystając z metody `zapis()` a potem oblicz i wypisz ich wartości dla zmiennych  $x$  i  $y$  z zakresu od 0 do 1 ze skokiem co 0.1 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.

## Uwaga

Podziel program na pliki nagłówkowe (definicje klas) i źródłowe (definicje metod zadeklarowanych w klasach). Funkcję `main()` z testami umieść w osobnym pliku źródłowym.

## Ważne 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 operatorów 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.