

# kurs języka C++

## kalkulator ONP

Instytut Informatyki  
Uniwersytetu Wrocławskiego

Paweł Rzechonek

---

### Prolog

Notacja Polska to beznawiasowy sposób zapisu wyrażeń logicznych i arytmetycznych, w którym najpierw występuje operator (funkcja) a za nim operandy (argumenty). Taka prefiksowa notacja została przedstawiona w 1920 roku przez polskiego logika Jana Łukasiewicza. Pozwala ona na łatwiejsze przeprowadzanie operacji na długich formułach logicznych czy wyrażeniach arytmetycznych.

ONP czyli Odwrotna Notacja Polska to sposób zapisu wyrażeń arytmetycznych, w którym operator umieszczony jest za operandami. Jest to więc notacja postfiksowa. Zapis ten pozwala na całkowitą rezygnację z użycia nawiasów w wyrażeniach, jako że jednoznacznie określa kolejność wykonywanych działań (podobnie jak notacja Łukasiewicza).

Odwrotna notacja polska została opracowana przez Arthura Burksa, Dona Warrena i Jessego Wrighta w 1954 roku. Sam algorytm i notacja zostały dopracowane przez australijskiego filozofa i informatyka Charlesa L. Hamblina w połowie lat 50'tych XX wieku. Notacja postfiksowa została odkryta na nowo przez Friedricha L. Bauera i Edsgera W. Dijkstrę na początku lat 60'tych XX wieku, kiedy chcieli oni wykorzystać stos obsługiwany przez procesor do przyspieszenia obliczania wyrażeń arytmetycznych (notacja postfiksowa idealnie nadawała się do tego celu).

### Zadanie

Napisz program interaktywnego kalkulatora postfiksowego. Kalkulator ten powinien interpretować i obliczać wyrażenia zapisane w Odwrotnej Notacji Polskiej. Program ma odczytywać polecenia ze standardowego wejścia `cin`, wykonywać obliczenia i wypisywać wyniki na standardowe wyjście `cout`. Wszelkie komentarze i uwagi program ma wysyłać na standardowe wyjście dla błędów `clog`. Dodatkową funkcjonalnością tego kalkulatora ma być możliwość zapamiętywania wyników obliczeń w zmiennych.

Zaprojektuj hierarchię klas, która umożliwi łatwą i elegancką klasyfikację poszczególnych symboli w wyrażeniu ONP (abstrakcyjna klasa `symbol`). Wyrażenie to ciąg operandów (klasa `operand`) i operatorów albo funkcji (klasa `funkcja`). Operandy to liczby (klasa `liczba` pamiętająca wartość typu `double`), zmienne (klasa `zmienna` z nazwą zmiennej) albo stałe (klasa `stala` z nazwą stałej i skojarzoną z nią wartością typu `double`). Dobrze znane przykłady stałych, które powinny się znajdować w Twoim kalkulatorze to `e` (2,718281828459),

$\pi$  (3,141592653589) i  $\phi$  (1,618033988750). W klasie `zmienna` umieść statyczną kolekcję asocjacyjną zawierającą zbiór ze zmiennymi (na przykład `map<string, double>` albo `unordered_map<string, double>`) – zmienną odszukujemy po nazwie a wartość skojarzoną ze zmienną odczytujemy z drugiego pola. Funkcje to przede wszystkim dwuargumentowe operatory dodawania, odejmowania, mnożenia i dzielenia; należy też zaimplementować funkcje dwuargumentowe `mod` (reszta z dzielenia), `log` (logarytm) i `pow` (potęgowanie) oraz jednoargumentowe `neg` (zmiana znaku), `ln` (logarytm naturalny) i `exp` (funkcja eksponencjalna).

Symbole występujące w wyrażeniu należy najpierw podzielić za pomocą białych znaków (separatorem niech będzie ciąg spacji i tabulacji), potem dopasować i utworzyć odpowiednie obiekty a na koniec umieścić je w wybranej kolekcji sekwencyjnej (na przykład `forward_list<symbol>`).

Program kalkulatora ma pracować z użytkownikiem interaktywnie i powinien rozpoznawać trzy rodzaje poleceń:

- `print wyrażenieONP`  
Obliczenie wartości wyrażenia ONP i wypisanie jej na standardowym wyjściu. Wyrażenie `wyrażenieONP` będzie oczywiście zapisane w postaci postfiksowej. Czytając kolejne symbole w wyrażeniu program powinien je zamieniać na konkretne obiekty i umieszczać w kolejce (klasa `queue<>`). Przy obliczaniu wartości wyrażenia należy się posłużyć stosem (klasa `stack<>`).
- `set zmienna to wyrażenieONP`  
Utworzenie nowej zmiennej `zmienna` i przypisanie jej wartości obliczonego wyrażenia `wyrażenieONP`. Wartość obliczonego wyrażenia należy wypisać na standardowym wyjściu. Jeśli zmienna `zmienna` była zdefiniowana już wcześniej, to należy tylko zmodyfikować zapisaną w niej wartość.
- `clear`  
Usunięcie wszystkich zmiennych zapamiętanych do tej pory w zbiorze zmiennych. Do kolekcji mogą trafić tylko zmienne o nazwach będących poprawnymi identyfikatorami i różnych od nazw funkcji, którymi posługuje się program.
- `exit`  
Zakończenie działania programu. Zamknięcie strumienia wejściowego również powinno zakończyć działanie programu.

Jeśli w wyrażeniu ONP zostanie wykryty błąd (nieznana komenda, źle sformułowane wyrażenie, błędna nazwa, błędny literał stałopozycyjny, czy nierozpoznany operator, funkcja lub zmienna) to należy wypisać stosowny komunikat o błędzie, ale nie przerywać działania programu. Zadbaj o to by nazwa każdej zmiennej nie była dłuższa niż 7 znaków oraz aby była różna od słów kluczowych `print`, `set`, `to`, `clear` i `exit`.

Do zaprogramowania tego zadania wykorzystaj kolekcje standardowe zdefiniowane w STL. Definicje klas reprezentujących różne symbole w wyrażeniu ONP umieść w przestrzeni nazw `kalkulator`.

Więcej informacji na temat ONP znajdziesz w Internecie na stronie:

[https://pl.wikipedia.org/wiki/Odwrotna\\_notacja\\_polska](https://pl.wikipedia.org/wiki/Odwrotna_notacja_polska)

### **Ważne elementy programu**

- Podział programu na pliki nagłówkowe i źródłowe.
- Użycie kolekcji standardowych.
- Wykorzystanie iteratorów do sekwencyjnego przeglądania kolekcji.
- Interaktywne przyjmowanie poleceń od użytkownika.
- Implementacja algorytmu obliczającego wartość wyrażenia ONP.
- Obsługa błędów za pomocą wyjątków.