Zadanie 10 *30 maja – 2 czerwca 2017 r.*

JĘZYK PROGRAMOWANIA C++

ODWROTNA NOTACJA POLSKA

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

ONP czyli *odwrotna notacja polska* to sposób zapisu wyrażeń arytmetycznych, w którym znak wykonywanej operacji umieszczony jest po operandach (zapis postfiksowy), a nie pomiędzy nimi jak w konwencjonalnym zapisie algebraicznym (zapis infiksowy). Zapis ten pozwala na całkowitą rezygnację z użycia nawiasów w wyrażeniach, jako że jednoznacznie określa kolejność wykonywanych działań. Odwrotna notacja polska została opracowana przez australijskiego naukowca Charlesa Hamblina jako odwrócenie beznawiasowej notacji polskiej Jana Łukasiewicza na potrzeby zastosowań informatycznych.

Zadanie.

Napisz program interaktywnego kalkulatora. 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 (abstrakcyjna klasa `symbol`) w wyrażeniu ONP. Wyrażenie to ciąg operandów (klasa `operand`) i operatorów/funkcji (klasa `funkcja`). Operandy to liczby rzeczywiste (klasa `liczba` pamiętająca wartość typu `double`), zmienne (klasa `zmienna` z nazwą zmiennej) albo stałe (klasa `stala` z nazwą stałej i skojarzoną z nią wartością) jak na przykład `e` i `pi`. W klasie `zmienna` umieść *kolekcję asocjacyjną* ze zmiennymi w postaci niepublicznego pola statycznego (na przykład `map<string,double>` albo `unordered_map<string,double>`) — wartość zmiennej odczytujemy szukając w tym zbiorze pary z odpowiednią nazwą. Funkcje to przede wszystkim dwuargumentowe operatory dodawania, odejmowania, mnożenia i dzielenia; należy też zaimplementować funkcje dwuargumentowe `modulo`, `min`, `max`, `log` i `pow` oraz jednoargumentowe `abs`, `sgn`, `floor`, `ceil`, `frac`, `sin`, `cos`, `atan`, `acot`, `ln` i `exp`.

Symbole występujące w wyrażeniu należy najpierw sparsować, potem utworzyć odpowiednie obiekty a na koniec umieścić je w wybranej *kolekcji sekwencyjnej* (na przykład `vector<>` albo `forward_list<>`). Parametrem tego obiektu niech będzie `shared_pointer<symbol>`, czyli sprytny wskaźnik będący opakowaniem dla różnych symboli, które mogą się pojawić w wyrażeniu (nie można umieszczać klas pochonych w kolekcji).

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

- `print` *wyrażenieONP*
  Obliczenie wartości wyrażenia *wyrażenieONP* i wypisanie jej na standardowym wyjściu. Wyrażenie będzie zapisane w postaci postfiksowej (*Odwrotna Notacja Polska*). Czytając kolejne tokeny wyrażenia program powinien je zamieniać na konkretne symbole i umieszczać w *kolejce* (klasa `queue<>`). Przy obliczaniu wartości wyrażenia należy się posłużyć *stosem* (klasa `stack<>`).

- `assign` *wyrażenieONP* `to` *zm*
  Utworzenie nowej zmiennej *zm* i przypisanie jej warości obliczonego wyrażenia *wyrażenieONP*. Wartość obliczonego wyrażenia należy wypisać na standardowym wyjściu. Jeśli zmienna *zm* była zdefiniowana już wcześniej, to należy tylko zmodyfikować zapisaną w niej wartość.

- `clear`
  Usunięcie wszystkich zminnych zapamiętanych do tej pory w zbiorze zmiennych. Do kolekcji mogą trafiać tylko zmienne o nazwach będących poprawnymi identyfikatorami i różnych od nazw standardowych dla tego programu funkcji.

- `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*, *assign*, *to*, *clear*, *exit* itp.

Wskazówka.
Wykorzystaj kolekcje zdefiniowane w STL.

Uzupełnienie.
Definicje klas opakowujących pliki umieść w przestrzeni nazw `kalkulator`.

Uwaga.
Podziel program na pliki nagłówkowe i źródłowe.

Elementy w programie, na które należy zwrócić szczególną uwagę.

- Użycie kolekcji standardowych.
- Wykorzystanie iteratorów do sekwencyjnego przeglądania kolekcji.
- Interaktywna procedura interpretująca polecenia użytkownika.
- Obsługa błędów za pomocą wyjątków.
- Podział programu na pliki nagłówkowe i źródłowe.