zadanie nr 7 *22/24 kwietnia 2008 r.*

KURS PROGRAMOWANIA W C++

KALKULATOR ONP

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie

Napisz program *interaktywnego kalkulatora*. Kalkulator ma interpretować i obliczać wyrażenia zapisane w notacji ONP. Program powinien odczytywać polecenia ze standardowego wejścia `cin` (każde polecenie w osobnym wierszu), wykonywać obliczenia i wypisywać wyniki na standardowe wyjście `cout`. Wszelkie komentarze i informacje o błędach program ma wysyłać na standardowe wyjście dla błędów `clog`.

Program powinien rozpoznawać trzy rodzaje poleceń:

- `set` *zm = wyrażenieONP*
  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. Zmienne pamiętaj w kolekcji asocjacyjnej dostępnej w STL'u (na przykład `map<>` albo `hash_map<>`).

- `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 elementy obliczalne `oblicz` i umieszczać w standardowej kolekcji `queue<>`. Przy obliczaniu wartości wyrażenia należy się posłużyć stosem `stack<>`.

- `clear`
  Usunięcie wszystkich zminnych zapamiętanych do tej pory w kolekcji asocjacyjnej. Do tablicy mogą trafiać tylko zmienne o nazwach różnych od nazw zaprogramowanych przez ciebie funkcji (`pi`, `e`, `abs`, `floor`, `ceil`, `frac`, `min`, `max`, `sin`, `cos`, `atan`, `acot`, `log`, `ln`, `power`, `ex` i innych).

Po wydaniu polecenia `set` lub `print`, 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.

Zaprojektuj całą hierarchię klas, na szczycie której znajdzie się klasa `oblicz`, z której będą dziedziczyć (nie bezpośrednio) operatory, funkcje i operandy (literały zmiennopozycyjne i zmienne). Klasa `oblicz` powinna zawierać abstrakcyjną metodę wirtualną `wartosc()` do obliczania wartości. Metodę tą można będzie bezpiecznie wywołać tylko wtedy, gdy operator lub funkcja (obiekt klasy dziedziczącej po `oblicz`) zostanie nakarmiona wymaganą ilością argumentów.

```
class oblicz
{
    // ...
public:
    virtual double wartosc () throw (logic_error) = 0;
    virtual int arnosc () throw ();
    virtual bool potrzebny_arg () throw () = 0;
    virtual void dodaj_arg (double arg) throw (logic_error) = 0;
};
```

Wskazówka

Do odczytania linii ze standardowego wejścia wykorzystaj funkcję `getline(istream&we,string&wynik)`. Potem posłuż się strumieniem `istringstream`, aby wydobyć poszczególne tokeny z odczytanej linii. Do rozbioru polecenia na tokeny wykorzystaj klasę `lancuch` z poprzedniego zadania.

Uwaga

Zdefiniuj odpowiednie wyjątki dziedziczące po standardowej klasie wyjątku `logic_error`.