Napisać program realizujący obliczanie wyrażeń zadanych za pomocą notacji postfiksowej. W wyrażeniach mogą występować stałe numeryczne, symbole zmiennych oraz wybrane funkcje mające implementacje w standardowej bibliotece numerycznej C++ (np. sin, log itp.). Dla celów wyliczania wartości wyrażenia oraz przekształcenia do postaci infiksowi należy przekształcić wyrażenie do postaci drzewa. W drzewie węzeł-liść reprezentuje stałą lub zmienną a węzeł wewnętrzny – operator. W zależności od tego czy operator jest jedno czy dwuargumentowy to odpowiadający mu węzeł ma jeden lub dwa następniki.

Program powinien realizować następujące funkcje:

* Wprowadzenie wyrażenia w postaci postfiksowej (z możliwością wielokrotnego wprowadzania ale w danej chwili przechowywane jest tylko jedno – ostatnio wprowadzone)
* Wyprowadzenie na konsolę wyrażenia w postaci infiksowiej
* Zadanie wartości zmiennej oraz zapamiętanie zadanej wartości
* Wyliczenie wartości wyrażenia dla aktualnego wartościowania zmiennych.

Przy wprowadzaniu wyrażenia dla ułatwienia założyć że:

* Elementy postaci infiksowej (operandy i operatory) rozdzielane są białymi znakami
* Nazwy zmiennych składają się tylko z liter
* Wprowadzane liczby są nieujemne i bez znaku (ale mogą być to liczby z częścią dziesiętną oraz należy dopuścić jednoargumentowy operator ~odpowiadający minusowi unarnemu)

Uwagi: Za powyższe zadanie można otrzymać dodatkowo 10 punktów jeśli:

1. Program będzie umiał wczytywać również wyrażenia w postaci infiksowej
2. Program będzie automatycznie, na podstawie treści wyrażenia sam rozpoznawał, czy postać jest infiksowi czy postfiksowa.

Lista instrukcji do obsłużenia:

+,-,/,\*,~ (tylda – funkcjonalność jak wyżej), sin, cos, tg, ctg, log

Proponowany algorytm konwersji wyrażenia postfiksowego do postaci drzewa wykorzystuje stos wskaźników do utworzonych poddrzew. Algorytm jest następujący

Opróżnij stos wskaźników; dopóki istnieją jeszcze nie przeczytane elementy postaci postfiksowej:

{

pobierz kolejny element wyrażenia w postaci postfiksowej; jeśli jest to zmienna lub stała

{

utwórz węzeł-liść drzewa wyrażenia; zapisz w nim informację o stałej/zmiennej; połóż wskaźnik do tego węzłą na stosie;

}

w przeciwnym przypadku

jeśli jest to operator jednoargumentowy

{

utwórz wewnętrzny drzewa wyrażenia; zapisz w nim informację o operatorze; pobierz element ze stosu i dołącz go jako następnik nowo utworzonego węzłą; połóż wskaźnik do nowego węzła na stosie;

}

w przeciwnym przypadku

{

utwórz wewnętrzny drzewa wyrażenia; zapisz w nim informację o operatorze; pobierz dwa element ze stosu i dołącz pobrane wskaźniki jako następniki nowo utworzonego węzłą; połóż wskaźnik do nowego węzła na stosie;

}

}

Po przeanalizowaniu całego wyrażenia n stosie znajduje się tylko jeden wskaźnik wskazujący na korzeń drzewa wyrażenia.

Algorytm rekursywnego wartościowania wyrażenia zapisanego w postaci drzewa jest następujący:

jeśli węzeł jest węzłem-liściem

{

odczytaj wartość zmiennej/stałej reprezentowanej przez węzeł zwróć tę wartość jako wartość wyrażenia;

}

w przeciwnym przypadku

{

oblicz wartości podwyrażeń reprezentowanych przez następniki tego węzłą; zastosuj operator reprezentowany przez węzeł do obliczonych wartości podwyrażeń; zwróć obliczoną wartość jako wartość wyrażenia;

}

W tym celu należy zaimplementować następujące komponenty:

1. Klasa TablicaSymboli implementująca zestaw funkcjonalności związanych z przechowywaniem zbioru zdefiniowanych zmiennych wraz z ich wartościami (tzw. tablica symboli):
   * Określenie wartości zmiennej o zadanym identyfikatorze,
   * Pobranie wartości zmiennej o zadanym identyfikatorze
   * Sprawdzenie czy zmienna o zadanym identyfikatorze jest zdefiniowana
   * Usunięcie wszystkich zdefiniowanych zmiennych
2. Klasa DrzewoWyrazenia reprezentująca węzeł w drzewie wyrażenia i implementująca zestaw funkcjonalności obsługujących drzewo wyrażenia:
   * Tworzenie drzewa z zapisu wyrażenia w postaci postfiksowej (w postaci napisu),
   * Dostarczanie listy zmiennych występujących w wyrażeniu a nie nie zdefiniowanych w tablicy symboli,
   * Obliczanie wartości wyrażenia dla aktualnego wartościowania zmiennych
3. Klasa Kalkulator przechowująca tablicę symboli, aktualne wyrażenie w postaci drzewa oraz zarządzająca tym zestawem danych (powinna implementować cztery publiczne motody odpowiadające funkcjom programu wymienionym na początku ALE NIE POWINNA IMPLEMENTOWAĆ ŻADNEGO INTERFEJSU UŻYTKOWNIA a jedynie dostarczać odpowiednio sparametryzowanych metod)
4. Pomocnicza klasa Stos przechowująca stos wskaźników – powinna być użyta przy konwersji postaci postfiksowej do postaci drzewa

Dialog z użytkownikiem należy zaimplementować w programie demonstracyjnym wykorzystującym zaimplementowaną klasę Kalkulator.

UWAGA:

Za względów dydaktycznych w implementacji nie dopuszcza się używania klas szablonowych – należy samodzielnie zaimplementować wszystkie wymienione funkcjonalności stosując tylko podstawowe techniki.