METODY PROGRAMOWANIA – LABORATORIUM 4

PROGRAM 01

Napisz i przetestuj program, który przedstawi sposób implementacji stosu:

A) w tablicy dynamicznej jednowymiarowej

B) za pomocą listy jednokierunkowej

C) za pomocą adaptera stosu (wykorzystaj std::stack ze standardowej biblioteki szablonów języka C++) *

Każdy ze sposobów powinien realizować następujące operacje: informacja czy stos jest pusty, zwrócenie ilości elementów umieszczonych na stosie, wartość szczytowego elementu na stosie, umieszczenie nowego elementu na szczycie stosu, zdjęcie istniejącego elementu ze szczytu stosu.

WEJŚCIE

Plik testowy o dowolnej nazwie (standardowe wejście) ma następujący format:

Pierwsza linia zawiera liczbę całkowitą n określająca ilość elementów.

W kolejnej linijce znajduje się n liczb całkowitych będących kolejnymi elementami

WYJŚCIE

Demonstracja działania poszczególnych operacji.

PROGRAM 02

Napisz efektywny pamięciowo i obliczeniowo program realizujący następujące operacje:

A) przekształcenie wyrażeń arytmetycznych i instrukcji przypisania z tradycyjnej notacji infiksowej do ONP.

B) przekształcenie wyrażeń arytmetycznych i instrukcji przypisania z ONP do notacji infiksowej z minimalną liczbą użytych nawiasów.

Instrukcja przypisania ma postać: operand = wyrażenie arytmetyczne.

Wyrażenia arytmetyczne mogą zawierać jedynie:

- nawiasy: (,) – tylko w notacji infiksowej
- operandy: małe litery alfabetu angielskiego

Operatory:

Operator	Priorytet	Łączność	Rodzaj operatora
=	0	prawostronna	przypisania
< >	1	lewostronna	relacyjny
+ -	2	lewostronna	addytywny
* / %	3	lewostronna	multiplikatywny
^	4	prawostronna	potęgowania
~	5	prawostronna	unarny

WEJŚCIE

Dane są umieszczone w pliku tekstowym zawierającym kolejne linie, przy czym pierwsza linia w pliku zawiera liczbę całkowitą zakresu od 1 do 2^{15}, oznaczającą liczbę linii zawierających wyrażenia arytmetyczne, których opisy występują kolejno po sobie

Każda linia zawiera co najmniej 6 znaków i nie przekracza 256 znaków, może mieć jedną z dwóch postaci:
`INF`: wyrażenie arytmetyczne lub instrukcja przypisania, zapisane w notacji infiksowej
`ONP`: wyrażenie arytmetyczne lub instrukcja przypisania zapisane w notacji ONP

Przy czym wyrażenia mogą zawierać dowolne znaki. Program najpierw usuwa znaki niewystępujące w wyrażeniach, w tym spacje oraz sprawdza poprawność wyrażeń.

Można założyć, że po usunięciu błędnych symboli wyrażenia wejściowe w postaci `INF` są poprawne jeśli są poprawne w C++. Natomiast wyrażenia w postaci `ONP` są poprawne jeśli są wykonalne.

WYJŚCIE

Wyrażenie poprzedzone na wejściu napisem "`INF:` " musi być na wyjściu poprzedzone napisem „`ONP:` " i analogicznie wyrażenie poprzedzone na wejściu napisem "`ONP:` " musi być na wyjściu poprzedzone napisem "`INF:` " . W przypadku błędnego wyrażenia, na wyjściu, zamiast skonwertowanego wyrażenia pojawi napis `error`.

W przypadku konwersji do notacji infiksowej, wyjściowe wyrażenie musi zawierać minimalną liczbę nawiasów gwarantującą taką kolejność operacji, jak w wejściowym wyrażeniu, np. `ONP: abc**` zostanie przekształcone do `INF: a*(b*c)` a nie do `INF: a*b*c`. Nawiasy obejmujące `b*c` w wyrażeniu wyjściowym wymuszają taką kolejność operacji mnożenia jaka jest w zapisie `ONP` w wyrażeniu wejściowym.

W przypadku wyrażeń w postaci infiksowej, np. `INF:( a,+ b)/..[c3` , program pozostawia jedynie: `(a+b)/c`, pozostałe znaki, w tym spacje – odrzuca, dodatkowo sprawdza poprawność wyrażenia, po czym dokonuje konwersji, wypisując na wyjściu: `ONP: ab+c/`.

W przypadku wyrażeń w notacji ONP, np. `ONP: ( a,b,.).c;-,*` program pozostawia jedynie: `abc-*`, dodatkowo sprawdza, czy wyrażenie jest poprawne, po czym dokonuje konwersji, wypisując na wyjściu: `INF: a*(b-c)`.

PRZYKŁADY:

wejście:	**wyjście:**
18 INF: a)+(b ONP: ab+a~a-+ INF: a+b+(~a-a) INF: x=~~a+b*c INF: t = ~ a < x < ~b INF: ~a-~~b<c+d&!p|!!q INF: a^b*c-d<xp|q+x INF: x=~a*b/c-d+e%~f ONP: xabcdefg++++++= ONP: ab+c+d+e+f+g+ ONP: abc++def++g++ ONP: abc++def++g+++ INF: x=a=b=c ONP: xabc=== INF: x=a^b^c INF: x=a=b=c^d^e ONP: xabcde^^=== INF: x=(a=(b=c^(d^e)))	ONP: error INF: a+b+(~a-a) ONP: ab+a~a-+ ONP: xa~~bc*+= ONP: ta~x<b~<= ONP: error ONP: error ONP: xa~b*c/d-ef~%+= INF: x=a+(b+(c+(d+(e+(f+g))))) INF: a+b+c+d+e+f+g INF: a+(b+c)+(d+(e+f)+g) INF: error ONP: xabc=== INF: x=(a=(b=c)) ONP: xabc^^= ONP: xabcde^^=== INF: x=(a=(b=c^(d^e))) ONP: xabcde^^===

ZASADY ODDAWANIA GOTOWYCH PROGRAMÓW:

Pliki ***.cpp*** o nazwach:

Nazwisko_Imie_Program_01.cpp
Nazwisko_Imie_Program_02.cpp

wraz z wszystkimi plikami tekstowymi powinny być zamieszczone w katalogu: ***Nazwisko_Imie_Laboratorium_4***

Katalog powinien być spakowany w formacie ***.rar*** lub ***.zip*** i przesłany do folderu: ***Programy - laboratorium 4 – Poniedziałek godzina 17.15*** dostępnego na stronie kursu MP (elf3.pk.edu.pl)