Laborator 03 - Stive

Responsabili

- Matei Păvălucă [mailto:mateipavaluca@yahoo.com] (2013)
- Laura Vasilescu [mailto:laura.vasilescu@cti.pub.ro] (2012)

Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- înțeleagă principiul de funcționare al unei stive
- implementeze o stivă folosind un vector pentru stocarea elementelor
- transforme o expresie din forma infixată în formă postfixată

Ce este o stivă?

O stivă este o instanță a unui tip de date abstract ce formalizează conceptul de colecţie cu acces restricționat. Restricția respectă regula LIFO (Last In, First Out).

Accesul la elementele stivei se face doar prin vârful acesteia.

Operații:

- *push* – adaugă un element (entitate) în stivă. Adăugarea se poate face doar la vârful stivei.
- *pop* – șterge un element din stivă și îl returnează. Ștergerea se poate face doar la vârful stivei.
- *peek* – consultă (întoarce) elementul din vârful stivei fără a efectua nicio modificare asupra acesteia.
- *isEmpty* – întoarce *1* dacă stiva este goală; *0* dacă are cel puțin un element

Cum se implementează?

O structură de date definește un set de operații și funcționalitatea acestora.
Implementarea efectivă a unei structuri de date poate fi realizată în diverse moduri, cât timp funcționalitatea este păstrată.

O stivă poate fi implementată cu ajutorul unui **vector** sau cu **liste înlănțuite**. O **listă înlănțuită** este o structură de date folosită pentru a stoca un set de elemente folosind zone de memorie discontinue. Listele vor fi studiate în cadrul laboratorului 5 [http://elf.cs.pub.ro/sd/wiki/laboratoare/laborator-01].

În cadrul acestui laborator, ne vom concentra asupra implementării unei stive cu ajutorul unui vector de stocare.

Implementare cu vector

La nivel de implementare, stiva este reprezentată printr-o clasă ce folosește (pe lângă operațile ce pot fi efectuate asupra ei) un vector de stocare (*stackArray*) de o dimensiune maximă dată (*NMAX*) și un indice ce indică vârful stivei(*topLevel*).

stack.h

```
#ifndef __STACK__H
#define __STACK__H

// Primul argument al template-ului este tipul de date T
// Al doilea argument este dimensiunea maxim a stivei N
template<typename T, int N>
class Stack {
        public:
                // constructor
                Stack() {
                        // TODO: initializari
                }

                // destructor
                ~Stack() {
                        // TODO: eliberare resurse, daca este cazul
                }

                // operator de adaugare
                void push(T x) {
                        // TODO: verificari, implementare
                }

                // operatorul de stergere
                T pop() {
                        // TODO: verificari, implementare
                }

                // operatorul de consultare
                T peek() {
                        // TODO: verificari, implementare
                }

                // operatorul de verificare dimensiune
                int isEmpty() {
                        // TODO: implementare
                }

        private:
                // vectorul de stocare
                T stackArray[N];

                // pozitia in vector a varfului stivei
                int topLevel;
};

#endif // __STACK__H
```

Forma poloneză inversă (formă postfixată)

Forma poloneză inversă [http://en.wikipedia.org/wiki/Reverse_Polish_notation] este o notație matematică în care fiecare operator urmează dupa toți operanzii săi.

Cel mai simplu exemplu de notație postfixată este cel pentru doi operanzi și un operator:

5 + 4	**se scrie sub forma**	**5 4 +**

În cazul în care există mai multe operații, operatorul apare imediat după cel de-al doilea operand:

2 + 4 - 5	**se scrie sub forma**	**2 4 + 5 -**

Avantajul major al formei poloneze inverse este faptul că elimină parantezele din cadrul expresilor:

5 + (1 + 4)	**se scrie sub forma**	**5 1 4 + +**

Algoritmul de conversie a unei expresii din formă infixată în formă postfixată

```
1. cât timp există elemente de citit
    1.1 citește un element
    1.2 dacă elementul este un număr, afișare (se adaugă la forma
        postfixată)
    1.3 dacă elementul este o paranteză stângă, adaugă-l în stivă
    1.4 dacă elementul este o paranteză dreaptă, extrage operatorii din
        stivă și adaugă-i la forma postfixată până când vârful stivei
        ajunge o paranteză stângă (care este extrasă, dar nu este
        adăugată la forma postfixată).

    !!! dacă stiva s-a golit fără să fie găsită o paranteză stângă,
        înseamnă că expresia inițială avea paranteze greșite
    1.5 dacă elementul este un operator (fie el O1)
        1.5.1 cât timp există un alt operator în vârful stivei (fie el O2)
              ȘI precedența lui O1 este mai mică decât cea a lui O2 extrage O2
              din stivă, afișare (se adaugă la forma postfixată)
        1.5.2 adaugă O1 în stivă
2. când nu mai există elemente de citit, extrage toate elementele rămase
   în stivă și adaugă-le la forma postfixată (elementele trebuie să fie
   numai operatori; dacă este extrasă o paranteză stângă expresia inițială
   avea parantezele greșite).
```

Exemplu

Fie expresia:

```
1 - 7 * 2 /(3 + 5)^2^5
```

Element	Acțiune	Forma postfixată	Stiva	Observaţii
1	Adaugă element la forma postfixată	1		
-	Pune elementul în stivă	1	-	
7	Adaugă element la forma postfixată	1,7	-	
*	Pune elementul în stivă	1,7	* -	* are precedență mai mare decât -
2	Adaugă element la forma postfixată	1,7,2	* -	
/	Extrage element din stivă	1,7,2*	-	/ și * au aceeași prioritate
	Pune elementul în stivă		/ -	/ are precedență mai mare decât -
(	Pune elementul în stivă	1,7,2*	(/ -	
3	Adaugă element la forma postfixată	1,7,2*3	(/ -	
+	Pune elementul în stivă	1,7,2*3	+ (/ -	
5	Adaugă element la forma postfixată	1,7,2*3,5	+ (/ -	
)	Extrage element din stivă	1,7,2*3,5+	(/ -	Se repeta până când se întâlnește (

	repetă		/ -	(a fost ignorat
^	Pune elementul în stivă	1,7,2*3,5+	^ / -	^ are precedență mai mare decât /
2	Adaugă element la forma postfixată	1,7,2*3,5+2	^ / -	
^	Pune elementul în stivă	1,7,2*3,5+2	^ ^ / -	^ este considerat asociativ-dreapta
5	Adaugă element la forma postfixată	1,7,2*3,5+2,5	^ ^ / -	
Final	Extrage toate elementele din stivă	1,7,2*3,5+2,5^^/-		

Algoritmul de evaluare a unei expresii în formă postfixată

```
1. cât timp există elemente de citit
    1.1 citește un element
    1.2 dacă elementul este o valoare
            1.2.1 pune elementul în stivă
        altfel (elementul este un operator)
            1.2.2 extrage 2 operanzi din stivă
            1.2.3 dacă nu există 2 operanzi în stivă
                EROARE: forma postfixată nu este corectă
            1.2.4 evaluează rezultatul aplicării operatorului asupra celor doi
                  operanzi
            1.2.5 pune rezultatul în stivă
2. dacă există o singură valoare în stivă
        2.1 afișează valoarea ca rezultat final al evaluării expresiei
    altfel
        EROARE: forma postfixată nu este corectă
```

Exercitii

- [**4p**] Pornind de la header-ul definit anterior [http://ocw.cs.pub.ro/courses/sd-ca/laboratoare/laborator-03#implementare_cu_vector], realizați implementarea structurii de date *stivă*.
    - [0.5p] constructor și destructor
    - [1p] metoda push
    - [1p] metoda pop
    - [1p] metoda peek
    - [0.5p] metoda isEmpty
- [**5p**] Implementați conversia unei expresii din formă infixată în formă postfixată.
- [**3p**] Implementați evaluarea unei expresii în formă postfixată.

Interviu

Această secțiune nu este punctată și încearcă să vă facă o oarecare idee a tipurilor de întrebări pe care le puteți întâlni la un job interview (internship, part-time, full-time, etc.) din materia prezentată în cadrul laboratorului.

1. Implementați, folosind un singur vector, 3 stive

2. Scrieți un program cu ajutorul căruia să sortați o stivă. Nu aveți acces decât la operațiile push(), pop(), top() și isEmpty()
3. Descrieți cum este folosită stiva sistemului în cazul transmiterii parametrilor apelului unei funcții

Resurse

[1] C++ Reference [http://www.cplusplus.com]

[2] Array Stack Visualization [http://www.cs.usfca.edu/~galles/visualization/StackArray.html]

[3] Linked List Stack Visualziation [http://www.cs.usfca.edu/~galles/visualization/StackLL.html]

sd-ca/laboratoare/laborator-03.txt · Last modified: 2013/03/03 18:10 by matei.pavaluca