Proiect la PCLP1 – testul nr 3.

Să se realizeze un modul care implementează o stivă ale cărei elemente sunt de tip *int*.

Prin *stivă* înțelegem o mulțime ordonată de elemente la care se are acces în conformitate cu principiul LIFO (Last In First OUT).

Cel mai simplu procedeu de implementare a unei stive este păstrarea elementelor ei într-un tablou unidimensional (vector).

În zona de memorie alocată stiveise pot păstra elementele ei, unul după altul. De asemenea, ele se pot scoate din zona de memorie respectivă, unul câte unul, în ordine inversă păstrării lor.

Ultimul element pus pe stivă se spune că este *vârful* stivei. Primul element pus pe stivă se află la *baza* stivei.

Vârful stivei se modifică atunci când se pune un element pe stivă sau când se scoate de pe stivă. În primul caz lungimea stivei (numărul elementelor din stivă) crește, iar în cel de al doilea caz descrește.

Într-adevăr, când se pune un element pe stivă, atunci acesta se pune după cel aflat în vârful stivei, și prin aceasta vârful stivei se modifică așa încât elementul nou pus pe stivă devine vârful stivei. Când se ia un element de pe stivă, atunci acesta este cel aflat în vârful stivei și apoi vârful stivei este se modifică așa încât, elementul care a post pus pe stivă înaintea celui scos să devine vârful stivei.

În figura de mai jos se explică grafic punerea / scoaterea unui element de pe stivă.

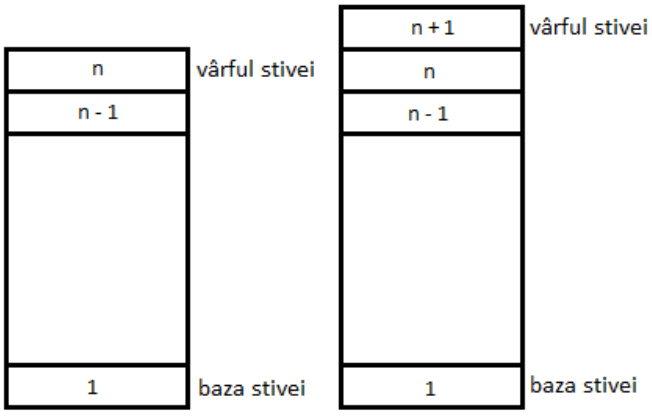

Fig. 1: Punerea unui element pe stivă: a) înainte de a pune un element pe stivă, b) după ce s-a pus un element pe stivă.

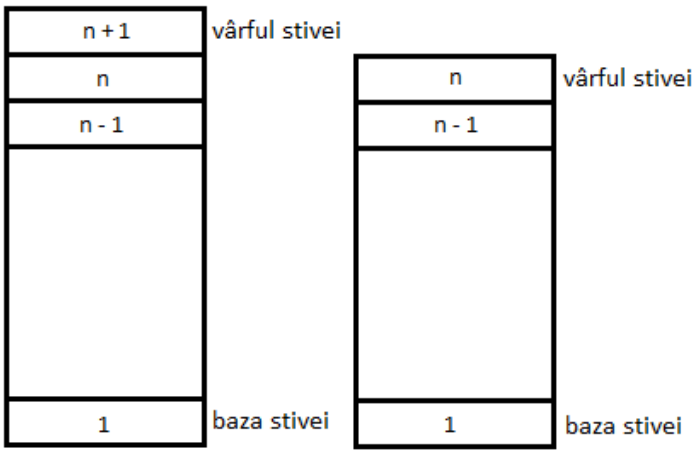

Fig. 2: Scoaterea unui element de pe stivă: a) înainte de a scoate un element de pe stivă, b) după ce s-a scos un element de pe stivă.

Se observă că totdeauna, ultimul element pus pe stivă este primul care se scoate din stivă. De aici provine afirmația că o stivă se gestionează conform principiului LIFO.

Pentru a implementa o stivă vor fi necesare două funcții:

- Una care să pună un element pe stivă - *push*,
- Una care să scoată un element din stivă - *pop*.

Aceste funcții au denumiri deja consacrate în literatura de specialitate și anume, funcția care pune un element pe stivă se numește *push*, iar cealaltă *pop*.
Vom păstra și noi aceste denumiri.

Stiva, în această lucrare, va fi implementată folosind un vector de tip int, ce va fi numit *stack*. *stack*[0] este elementul de la baza stivei.

Funcțiile *push* și *pop* gestionează vârful stivei și de aceea ele au nevoie de o variabilă care să definească indicele elementului din vârful stivei. De obicei, se păstrează indicele primului element liber, adică indicele elementului tabloului *stack* care urmează imediat după elementul aflat în vârful stivei.

Numim *next* variabila a cărei valoare curentă este primul element liber al tabloului *stack*. Evident, inițial *next* = 0, deoarece la început nefiind nici un element pe stivă, stack[0] este primul element liber.

Utilizatorul stivei nu trebuie să aibă acces direct la elementele stivei și de aceea tabloul *stack*, cât și variabila *next* trebuie ”ascunse” în modul. În felul acesta utilizatorul are posibilitatea să pună un element pe stivă în vârful ei apelând funcția *push* și să scoată elementul din vârful stivei apelând funcția *pop*. Rezultă că aceste două funcții nu se ”ascund” în modul.

Se vor defini si următoarele funcții:

- *clear* – pentru a vida stiva;
- *empty* – stabilește dacă stiva este vidă sau nu,
- *full* – stabilește dacă stiva este plină sau nu,
- *top* – permite acces la elementul din vârful, stivei fără a-l scoate din stivă.

Toate aceste funcții trebuie să aibă acces atât la *stack* cât și la variabila *next*. De aceea *stack* și *next* se vor declara statice în afara corpurilor lor.

Cele șase funcții amintite mai sus vor avea un caracter global, ele putând fi apelate din orice modul al programului.

Funcția *push* are prototipul:

```
void push(int x);
```

În principiu, ea pune pe stivă valoarea lui x, deci trebuie să facă atribuirea:

```
stack[next] = x;
```

și apoi să incrementeze pe *next*, pentru ca acesta să definească locul liber curent. De aceea, atribuirea de mai sus se poate realiza împreună cu incrementarea lui next:

```
stack[next++] = x;
```

`stack[next] = x;` `next ++;`	echivalent cu :	`stack[next++] = x;`

Funcția trebuie să testeze, în prealabil, dacă există loc liber pentru a păstra pe x și numai în acest caz se va executa atribuirea de mai sus. În caz că estiva este plină (depășită), se dă un mesaj de eroare.

Funcția *pop* are prototipul:

```
void pop(int x);
```

Ea returnează valoarea din vârful stivei. În acest scop se va decrementa next și apoi se va returna valoarea elementului.

```
stack[next];
```

Aceste două acțiuni se pot realiza cu ajutorul insctrucțiunii:

```
return stack[--next];
```

`next --;` `return stack[next;`	echivalent cu :	`return stack[next--];`

Se observă că prin aceasta vârful stivei coboară cu un element, deoarece *next,* care exprimă primul element liber, este chiar indicele elementului eliminat de pe stivă.

Înainte de a executa instrucțiunea de mai sus, funcția *pop* va testa dacă stiva nu cumva este vidă. În cazul în c are stiva este vidă, funcția *pop* va afișa un mesaj de eroare și va returna valoarea *zero*.

Funcția *top* este ca și *pop*, cu deosebirea să nu se elimină elementul din vârful stivei, ci numai se returnează valoarea lui.

Funcția *clear* are prototipul:

```
void clear(void);
```

Ea videază stiva, ceea ce se realizează simplu, prin atribuirea valorii zero variabilei next. În felul acesta tot tabloul devine liber.

Funcția *empty* are prototipul:

```
void empty(void);
```

Ea returnează valoarea 1 dacă stiva este vidă și zero în caz contrar.

Funcția *full* are prototipul:

`void full(void);`

Ea returnează valoarea 1 dacă stiva este plină și zero în caz contrar.

Aceste funcții, împreună cu declarațiile lui stack și next se păstrează într-un fișier care formează modulul ce implementează stiva stack prin intermediul unui tablou de tip int.

Modulul stack.cpp

```
#define MAX 1000
static int stack[MAX];
static int next = 0;
void push (int x) /*pune pe x pe stiva*/
{
    if (next < MAX) stack[next++]=x;
    else printf("stiva este plina\n");
}
int pop() /*scoate din stiv[ elementul din varful ei */
{
    if(next > 0) return stack[--next];
    else printf("stiva vida\n");
}
int top() /*returneaza elementul din varful stivei */
{
    if(next>0) return stack[next-1];
    else printf ("stiva vida\n");
}
void clear() /*videaza stiva*/
{
    next = 0;
}
void empty() /*returneaza 1 daca stiva este vida si 0 altfel*/
{
    return !next;
}
void full() /*returneaza 1 daca stiva este plina si 0 altfel*/
{
    return next==MAX;
}
```

Punctaj:

Nota 6	Se realizează modulul de mai sus, cu biblioteca de funcții definite și o aplicație de tip meniu, care permite crearea unei stive de ”n” elemente întregi și selectarea funcțiilor aferente.
Nota 7	Să se scrie programul pentru implementarea unei ”cozi”. https://olidej.wikispaces.com/file/view/t1-Stiva+si+coada.pdf http://ase.softmentor.ro/StructuriDeDate/Fisiere/05_CoziStive.pdf http://infoscience.3x.ro/c++/coada.htm
Nota 8	Să se realizeze o vizualizare grafică a gradului de implementare a stivei / cozii
Nota 10	Să realizeze un proiect, care să cuprindă cele două aplicații de mai sus

Termen de predare: 20-01-2017, pe serverul ftp din laborator sau pe e-mail: sorinpopa[at]ub.ro.

Proiectul valorează 50% din nota la laborator.