STIVA-IMPLEMENTARE DINAMICĂ

CURS 6- 06.04.2021

Titular: Şef. Lucr. Dr. Mat. Cărbureanu Mădălina

Universitatea Petrol-Gaze din Ploieşti

OBIECTIVE:

- **NOȚIUNEA DE STIVĂ(STACK);**
- **MODURI IMPLEMENTARE STIVĂ;**
- **PRINCIPIUL LIFO;**
- **MOD DE LUCRU;**
- **DECLARARE STIVĂ;**
- **TIPURI DE OPERAȚII;**
- **APLICAȚII PROPUSE.**

- ❑ Implementare statică;
- ❑ Implementare dinamică;

- ❑ STIVĂ- tip particular de l.s.î, (principiul LIFO);
- ❑ STIVĂ- l.s.î generală.

- ❑ Operații de bază ⇨ Curs 5;
- ❑ **Alte tipuri de operații** ⇨ **Curs 6;**

ALTE TIPURI DE OPERAȚII

Stiva- tip particular de l.s.î;
Stiva- l.s.î. generală;

- Diferite **tipuri de ștergeri** ale unui element dintr-o stivă:
  - Ștergerea primului element (vârfului stivei), **conform LIFO**;
  - Ștergerea unui element cu o anumită informație utilă din interiorul stivei;
  - Ștergerea ultimului element din stivă;
  - Ștergerea unui element de pe o anumită poziție din stivă.

pop

ALTE TIPURI DE OPERAȚII

Stiva- tip particular de l.s.î;
Stiva- l.s.î. generală;

- Diferite **tipuri de adăugări** ale unui element dintr-o stivă:

push

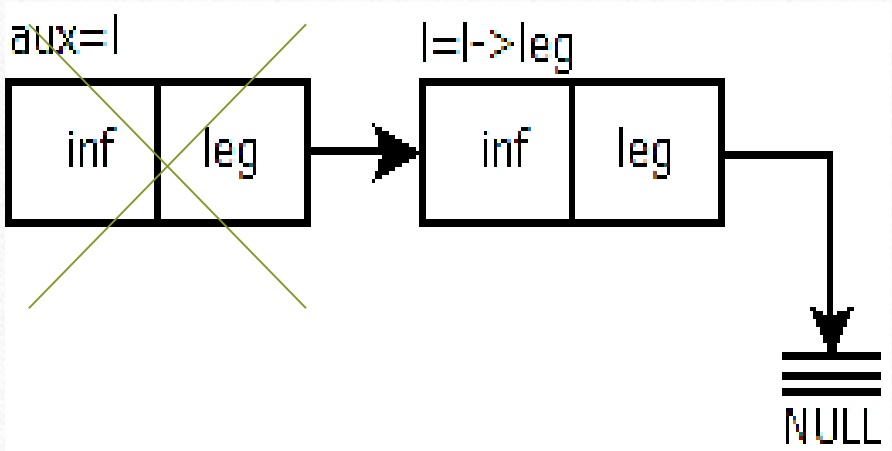
  - Adăugarea unui element desupra elementului aflat în vârful stivei (**conform LIFO**);
  - Adăugarea unui element cu informația utilă *q* după un element cu informația utilă *p* (adăugarea unui element în interior stivă folosind informația utilă);
  - Adăugarea unui element pe o anumită poziție (adăugarea unui element în stivă funcție de o poziție citită de la tastatură).

TIPURI DE ȘTERGERI ⟹ Ștergerea primului element (vârfului stivei), **conform LIFO**

- **Pași**:
  - declararea unui nou nod *aux* de tip pointer către STIVA;
  - noul nod *aux* preia adresa vârfului stivei;
  - mutarea vârfului stivei (*l*) pe următorul element din stivă;
  - ștergerea primului element din stivă;
  - returnarea stivei fără primul element.

TIPURI DE ȘTERGERI ⇨ Ștergerea primului element (vârfului stivei), **conform LIFO**

```
STIVA*elim_primul(STIVA*l)
{STIVA*aux;
  aux=l;
  l=l→leg;
  free(aux);
  return l;
}
```

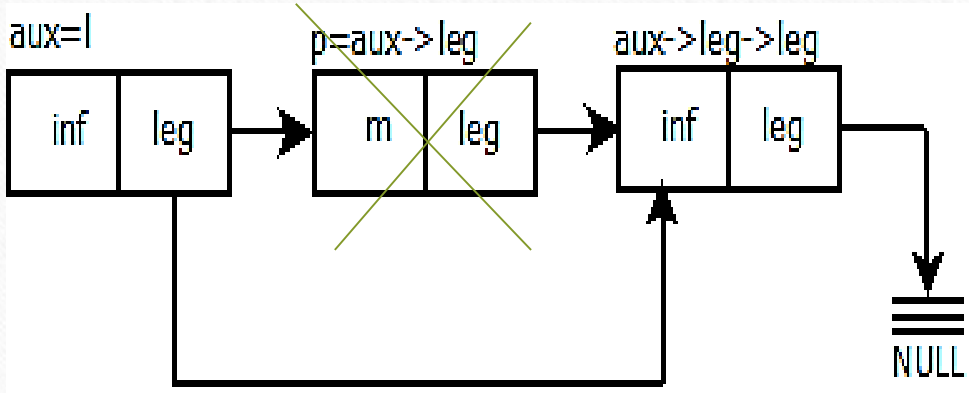


TIPURI DE ȘTERGERI ⇨ Ștergerea unui element cu o anumită informație utilă din interiorul stivei

- **Pași**:
  - declararea unui nou nod *aux* de tip pointer către STIVĂ;
  - declararea unui mod *p* de tip pointer către STIVĂ, care reține adresa elementului cu informația utilă căutată (*m)*;
  - noul nod *aux* preia adresa vârfului stivei;
  - se evaluează condiția "*atâta timp cât nu am ajuns la elementul cu informația utilă căutată*":
    - dacă condiția este îndeplină, *aux* devine următorul element al stivei;
    - în caz contrar (s-a identificat elementul cu informația utilă căutată):
      - nodul *p* primește adresa elementului cu informația utilă căutată;
      - se realizează legătura dintre predecesorul și succesorul elementului căutat;
      - se șterge elementul căutat*;*
  - se returnează stiva fără elementul șters.

TIPURI DE ȘTERGERI ⇨ Ștergerea unui element cu o anumită informație utilă din interiorul stivei

```
STIVA*elim_int_info(STIVA*l, int m)
{STIVA*aux,*p;
aux=l;
while(aux→leg→inf!=m)
{aux=aux→leg;
}
p=aux→leg;
aux→leg=aux→leg→leg;
free(p);
return l;
}
```

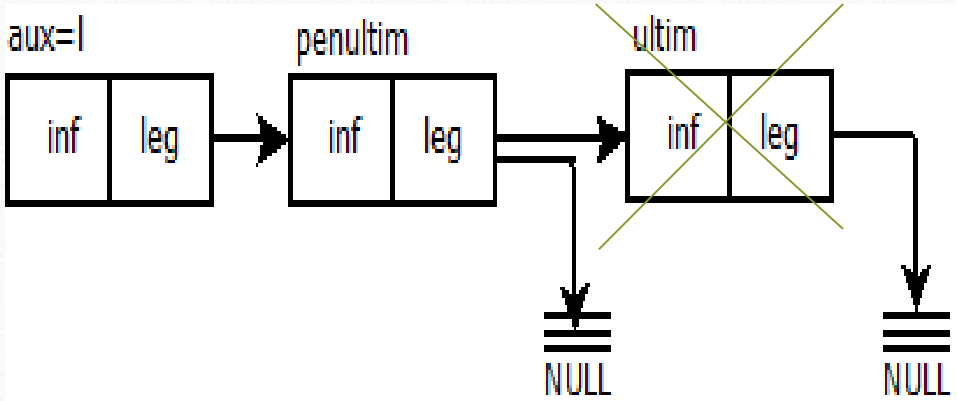


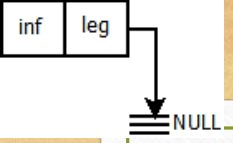
TIPURI DE ȘTERGERI ⟹ Ștergerea ultimului element din stivă

- **Pași:**
  - declararea unui nou nod *aux* de tip pointer către STIVA*;*
  - noul nod *aux* preia adresa vârfului stivei*;*
  - se evaluează condiția "*atâta timp cât nu am ajuns la ultimul element din stivă*" *(*sunt pe penultimul element):
    - dacă condiția este îndeplină, *aux* devine următorul element al stivei;
    - în caz contrar (s-a identificat ultimul element al stivei)*:*
      - acest ultim element din stivă se șterge*;*
      - se realizează legătura la pointerul *NULL* pentru elementul care a devenit ultimul element din stivă;
  - se returnzează stiva fără ultimul element șters.

TIPURI DE ȘTERGERI ⟹ Ștergerea ultimului element din stivă

```
STIVA*sterg_ultim(STIVA*l)
{STIVA *aux;
aux=l;
while(aux→leg→leg!=NULL)
aux=aux→leg;
free(aux→leg);
aux→leg=NULL;
return l;
}
```





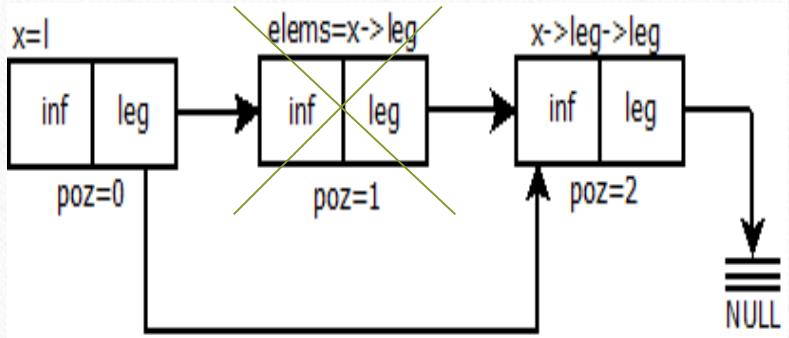
TIPURI DE ȘTERGERI ⇨ Ștergerea unui element de pe o anumită poziție din stivă

- **Pași:**
  - definirea unei funcții *caut_poz (STIVA*l, int poziție)* pentru identificarea poziției *pozitie* în stivă:
    - se parcurge stiva până la ultimul element sau până la identificarea poziției curente *k*;
    - dacă a fost identificată poziția curentă în cadrul stivei, atunci se returnează vârful stivei;
  - definirea unei funcții *sterg_poz (STIVA*l, int poz):*
    - dacă *poz=0* se apelează funcția pentru ștergerea primului element din stivă *(l=elim_primul(l));*
    - altfel, se apelează funcția *caut_poz(l,poz-1)* și este eliminat din stivă elementul care se află pe poziția *poz* (se realizează legătura între predecesorul și succesorul elementului ce se dorește a fi șters și apoi se șterge efectiv elementul de pe poziția dorită);
  - se returnează stiva fără elementul șters de pe poziția dorită.

TIPURI DE ȘTERGERI ⇨ Ștergerea unui element de pe o anumită poziție din stivă

```
STIVA*caut_poz
(STIVA*l, int pozitie)
{int k;
k=0;
 while((l!=NULL)&&
(k<poziție))
      {l=l→leg; k++;}
 if(k==pozitie) return
l;
 else return NULL;
}
```

```
STIVA*sterg_poz
(STIVA*l, int poz)
{if(poz==0) l=elim_primul(l);
else
{STIVA*x=caut_poz(l, poz-1);
STIVA*elems=x->leg;
x->leg=x->leg->leg;
free(elems);
}
return l;
}
```

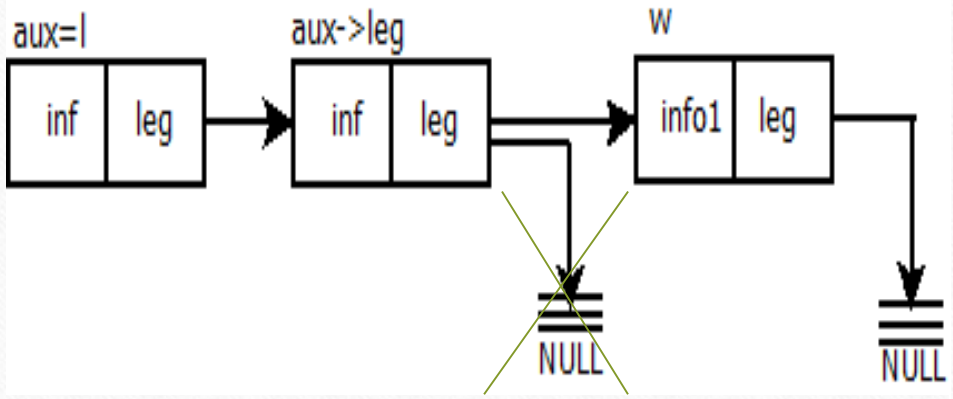


TIPURI DE ADĂUGĂRI ⇒ Adăugarea unui element desupra elementului aflat în vârful stivei (conform LIFO)

- **Paşi:**
- declararea unui nou nod *aux* de tip pointer către STIVA;
- declararea unui pointer *w* către STIVA (noul element ce va fi adăugat deasupra elementului aflat în vârful stivei);
- noul nod *aux* preia adresa vârfului stivei;
- se alocă memorie pentru noul element *w;*
- se completează câmpul informaţie pentru noul element *w* ce va fi adăugat;
- se evaluează condiţia "*atâta timp cât nu am ajuns în vârful stivei*":
  - dacă condiţia este îndeplină, *aux* devine următorul element al stivei;
  - în caz contrar, se realizează legătura ultimului element (elementului din vârful stivei) cu noul element adăugat;
  - noul element adăugat devine noul vârf al stivei;
- se returnează stiva cu noul element adăugat.

TIPURI DE ADĂUGĂRI ⇨ Adăugarea unui element desupra elementului aflat în vârful stivei (conform LIFO)

```
STIVA* adaug_vf(STIVA*l,int info1)
{STIVA*aux,*w;
aux=l;
w=(STIVA*)malloc(sizeof(STIVA));
w→inf=info1;
while(aux→leg!=NULL)
   aux=aux→leg;
   aux→leg=w;
   w→leg=NULL;
   return l;
}
```



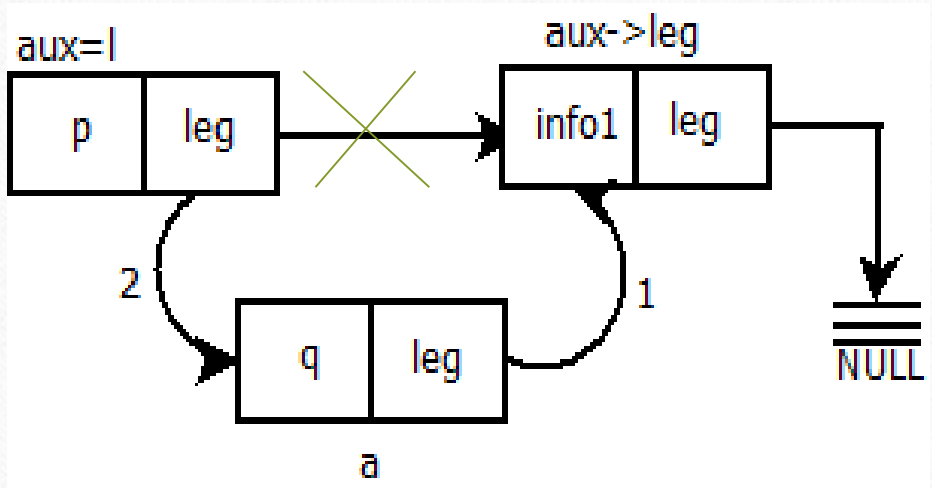
TIPURI DE ADĂUGĂRI ⇨ Adăugarea unui element cu informaţia utilă *q* după un element cu informaţia utilă *p*

- **Paşi:**

- declararea unui nou nod *aux* de tip pointer către STIVA;
- declararea unui pointer *a* către STIVA, care reprezintă noul element ce va fi adăugat ce prezintă informaţia utilă *q*;
- se alocă memorie pentru noul element *a;*
- se completează câmpul informaţie pentru noul element *a* ce va fi adăugat;
- noul nod *aux* preia adresa vârfului stivei;
- se evaluează condiţia ”*atâta timp cât informaţia elementului curent este diferită de informaţia elementului după care se realizează adăugarea”:*
  - dacă condiţia este îndeplină, *aux* devine următorul element al stivei;
  - în caz contrar, înformaţia elementului curent coincide cu informaţia elementului după care se realizează adăugarea (informaţia utilă *p*):
    - se realizează legătura noului element adăugat *a* cu următorul element din stivă;
    - se realizează legătura predecesorului cu noul element adăugat (nodul *a);*
- se returnează stiva cu noul element adăugat.

TIPURI DE ADĂUGĂRI⇨Adăugarea unui element cu informaţia utilă *q* după un element cu informaţia utilă *p*

```
STIVA*adaug_int(STIVA*l, int p, int q)
{STIVA*a,*aux;
a=(STIVA*)malloc(sizeof(STIVA));
a→inf=q;
aux=l;
while(aux→inf!=p)
aux=aux→leg;
a→leg=aux→leg;
aux→leg=a;
return l;
}
```



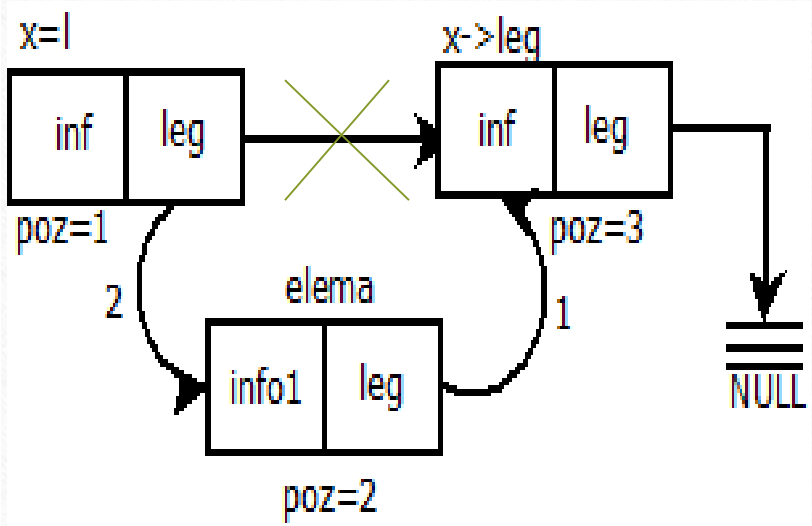
TIPURI DE ADĂUGĂRI⇨Adăugarea unui element pe o anumită poziție

- **Pași:**
  - definirea unei funcții *caut_poz (STIVA*l, int poziție)* pentru identificarea poziției *pozitie* în stivă:
    - se parcurge stiva până la ultimul element sau până la identificarea poziției curente *k*;
    - dacă a fost identificată poziția curentă în cadrul stivei, atunci se returnează vârful stivei;
  - definirea unei funcții *adaug_poz (STIVA*l, int poz, int info1)* ce apelează funcția *caut_poz (l, poz-1)* și este adăugat în stivă pe poziția *poz* elementul *elema* pentru care se alocă memorie, se completează câmpul informație și se realizează legăturile cu succesorul și predecesorul acestuia;
  - se returnează stiva cu elementul adăugat pe poziția dorită.

TIPURI DE ADĂUGĂRI ⇨ Adăugarea unui element pe o anumită poziție

```
STIVA*caut_poz(STIVA*l, int pozitie)
{int k;
k=0;
 while((l!=NULL)&&(k<poziție))
      {l=l→leg; k++;}
 if(k==pozitie) return l;
 else return NULL;
}
```

```
STIVA*adaug_poz(STIVA*l, int poz, int info1)
{ STIVA*x=caut_poz(l,poz-1);
STIVA*elema=(STIVA*)malloc(sizeof(STIVA));
        elema→inf=info1;
        elema→leg=x→leg;
        x→leg=elema;
return l;
}
```



APLICAȚII PROPUSE

- Testarea operaţiilor de ștergere și adăugare;
- Rezolvarea aplicaţiilor nr. 1, 2 și 4, pag. 94;
- Obs: 3, 2, 1 ⇨ 3, 2, 2, 1, 1;
- Obs: 3, 0, 2, 0, 1 ⇨ 3, 2, 1;
- Obs: pentru apl.4 ⇨ exemplu 5.2.3, pag. 91;
- Obs: se va utiliza cartea "*Elemente de proiectarea algoritmilor. Ghid teoretic și practic*", Șef. lucr. dr. mat. Cărbureanu Mădălina, Editura Universităţii Petrol-Gaze din Ploieşti, 2021.

Să vă fie de folos și spor la lucru!