STIVA-IMPLEMENTARE DINAMICĂ

CURS 5- 30.03.2021

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

Universitatea Petrol-Gaze din Ploieşti

OBIECTIVE:

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

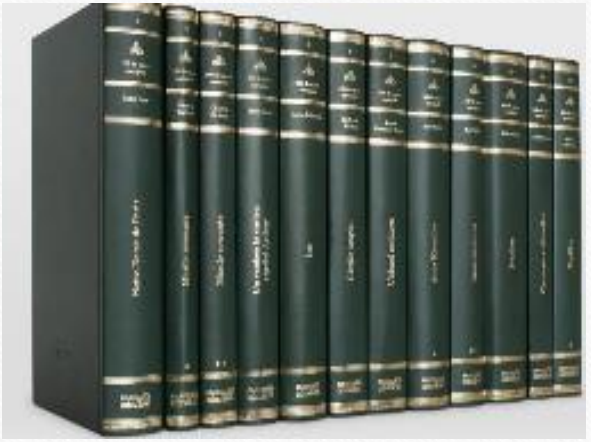
- ❑ Implementare statică;
- ❑ Implementare dinamică;

- ❑ Listă simplu înlănțuită de tip STIVĂ, conform LIFO;

- ❑ Operații de bază;
- ❑ Alte tipuri de operații ⇨ curs6;

NOȚIUNEA DE LISTĂ

- **Listă ⇨ <u>colecție omogenă</u>, secvențială de elemente (noduri);**
- **Listă ⇨ secvență finită L=(l1, l2,..., ln) de elemente (date elementare, structuri de date);**
- **Exemple de liste: lista studenților dintr-o grupă, lista cărților dintr-o bibliotecă, etc.**



MODURI IMPLEMENTARE STIVĂ

Se utilizează?

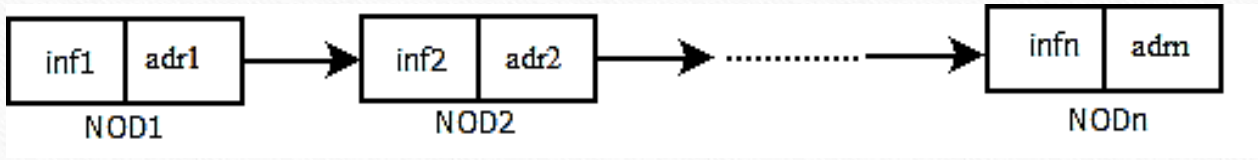
- **Implementare statică** ⇔ **implementare secvențială;**
- **Implementare statică** ⇨ **elemente vecine sunt memorate automat la adrese consecutive de memorie;**
- **Implementare statică;**
  - ❑ Avantaj;
  - ❑ Dezavantaj;

MODURI IMPLEMENTARE STIVĂ

Se utilizează? Stiva-listă simplu înlănţuită!

- **Implementare dinamică** ⇨ **implementare înlănţuită;**
- **Implementare dinamică** ⇨ **elemente vecine nu sunt memorate automat la adrese consecutive, memorarea fiecărui element se realizează în diferite zone de memorie;**

⇨ **fiecare nod al listei prezintă două câmpuri;**



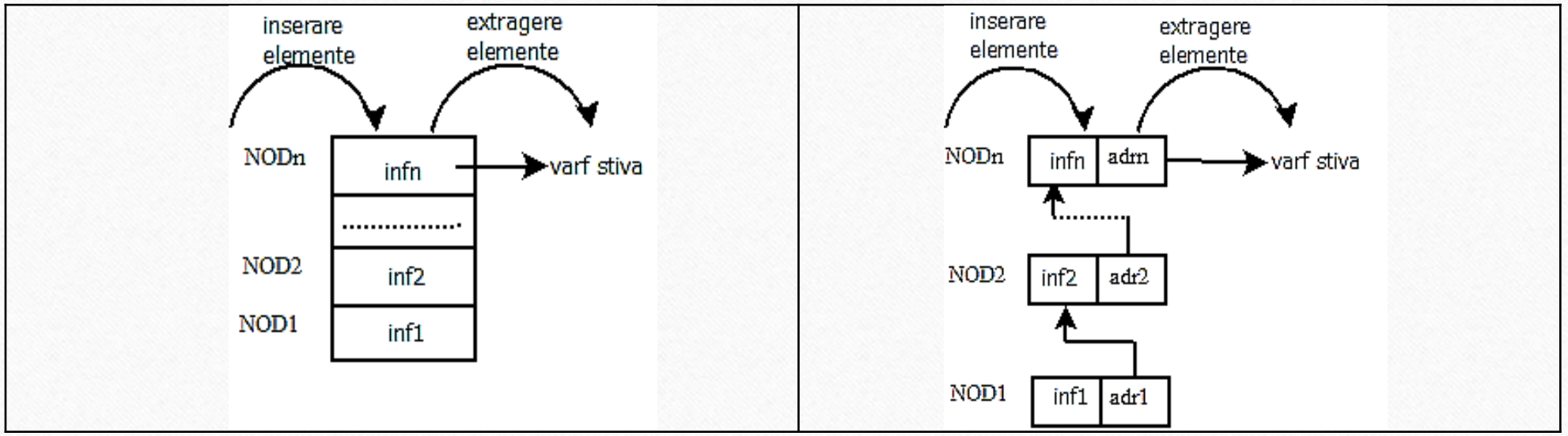
- **Implementare dinamică**
  - ❑ Avantaj;
  - ❑ Dezavantaj;

NOȚIUNEA DE STIVĂ (STACK)

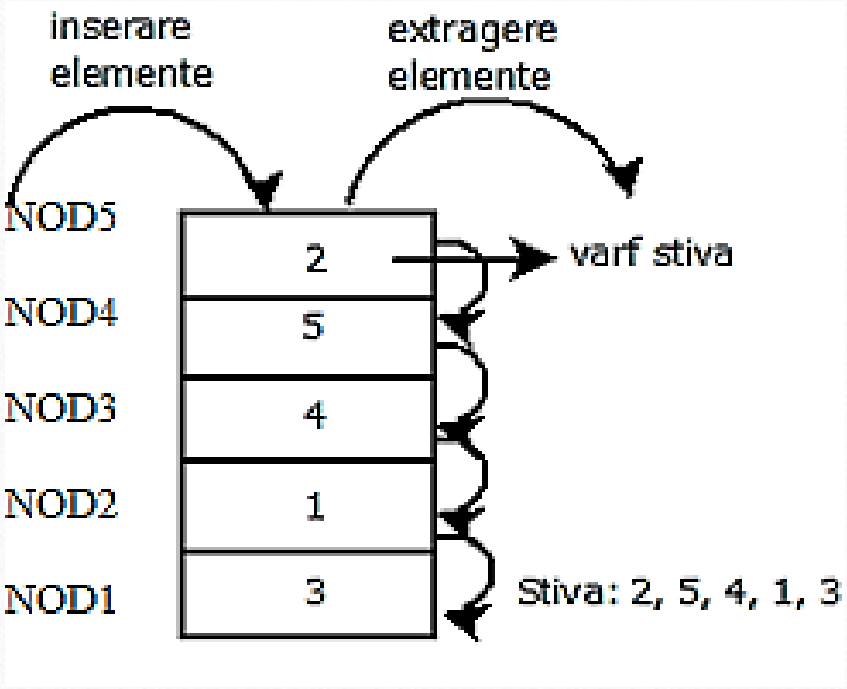
Stiva-tip particular de listă!

- Stiva ⇨ caz particular de listă simplu înlănțuită;
- Stiva ⇨ toate operațiile de adăugare și ștergere de elemente se realizează la un singur capăt al listei ⇨ vârf stivă;
- Exemple de stive: stiva de cărți, stiva de farfurii, stiva de CD-uri, stiva de documente, etc.
- Principiu ⇨ LIFO (Last-In, First-Out);
- TDA Stiva.

PRINCIPIUL LIFO



MOD DE LUCRU L.S.Î DE TIP STIVĂ CONFORM LIFO



DECLARARE STIVĂ

```
typedef  struct lista
{int inf;
 struct lista*leg;
}STIVA;
```

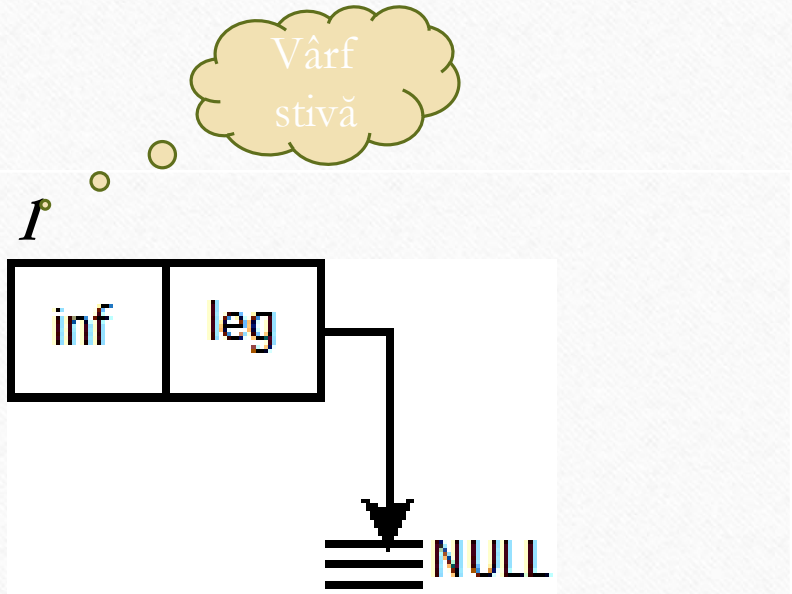
TIPURI DE OPERAȚII

- **Operații de bază:**
  - **Inițializarea stivei;**
  - **Crearea stivei;**
  - **Parcurgerea elementelor stivei;**
- **Alte tipuri de operații** ⇨ **curs 6;**

OPERAȚII DE BAZĂ

- **Inițializarea stivei** ⇨ **crearea unei stive vide/inițializarea vârfului stivei cu NULL;**

```
STIVA *init (STIVA *l)
{l=NULL;
 return l;
}
```



OPERAȚII DE BAZĂ

push

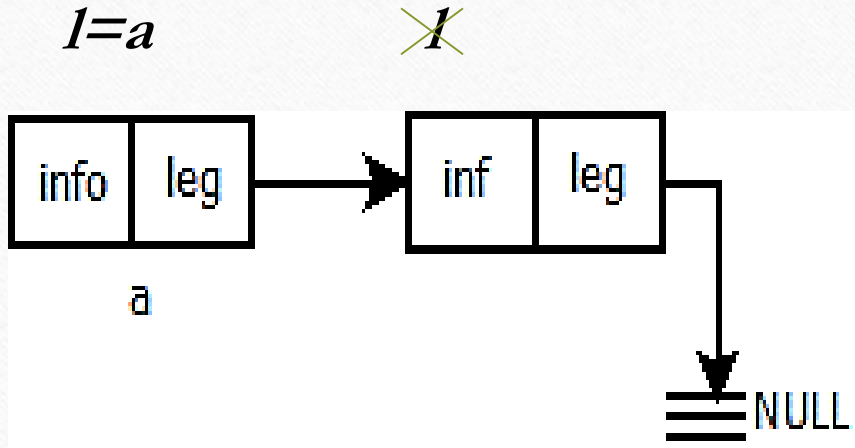
- **Crearea stivei** ⇨ **adăugarea treptată de elemente, respecând LIFO, prin citirea informației utile de la tastatură;**
- **Pași:**
  - se declară nodul ce se dorește a se adăuga;
  - se alocă memorie pentru noul nod adăugat;
  - se completează câmpul informație al noului nod;
  - se realizează legarea noului element adăugat de restul elementelor stivei.

malloc();

OPERAȚII DE BAZĂ

- **Crearea stivei;**

```
STIVA *creare(STIVA*l, int info)
{STIVA *a;
a=(STIVA *)malloc(sizeof(STIVA));
a→inf=info;
a→leg=l;
l=a;
return l;
}
```



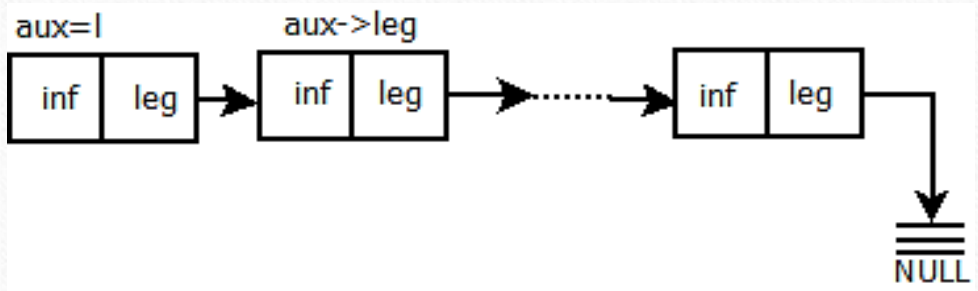
OPERAȚII DE BAZĂ

- **<u>Parcurgerea elementelor stivei</u> ⇨ parcurgerea fiecărui nod în parte, pornind de la vârful stivei până la baza acesteia și afișarea informației utile asociate fiecărui nod parcurs;**
- **<u>Pași:</u>**
  - se declară un 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 baza stivei*":
    - dacă condiția este îndeplină, se afișează informația utilă de la adresa *aux* ;
    - *aux* devine următorul element al stivei;
  - se repetă pașii anteriori până în momentul în care condiția nu mai este îndeplinită.

OPERAȚII DE BAZĂ

- **Parcurgerea elementelor stivei;**

```
void afisare(STIVA *l)
{STIVA *aux;
aux=l;
while(aux!=NULL)
{cout≪aux→inf;
aux=aux→leg;
}
}
```



EXEMPLU APLICAȚIE

- **Concatenarea a două l.s.î. de tip STIVĂ;**
- **Concatenare ⇨ operație de bază;**
- **Concatenare ⇨ parcurgerea primei stive până la sfârșitul acesteia și realizarea legăturii cu cea de-a doua stivă;**
- **Pași:**
  - **se declară un nou nod aux de tip pointer către STIVA;**
  - **stiva rezultat l primește adresa vârfului stivei l1 (l=l1);**
  - **noul nod aux preia adresa vârfului stivei (aux=l), respectiv adresa vârfului stivei l1;**
  - **se evaluează condiția "*atâta timp cât nu am ajuns la sfârșitul primei liste l1*":**
    - **dacă aceasta este îndeplinită, se trece la următorul element din lista l1;**
    - **în caz contrar, se realizează legătura cu vârful celei de-a doua liste, respectiv l2;**
  - **se returnează stiva l, respectiv stiva concatenată.**

CONCATENAREA A DOUĂ L.S.Î. DE TIP STIVĂ

```
#include<iostream.h>.....
#include<alloc.h>
typedef  struct lista
{int inf;
 struct lista*leg;
}STIVA;
STIVA *l1,*l2,*l;
STIVA*init(STIVA*l)
{l=NULL; return l;}
STIVA*creare(STIVA*l,int info)
{STIVA*a;
a=(STIVA*)malloc(sizeof(STIVA));
a→inf=info;
a→leg=l;
l=a; return l; }
```

CONCATENAREA A DOUĂ L.S.Î. DE TIP STIVĂ

```
void afisare(STIVA*l)
{STIVA*aux;
aux=l;
while(aux!=NULL)
{cout<<aux→inf;
 aux=aux→leg;}
}
STIVA* concatenare(STIVA*l1,STIVA*l2,STIVA*l)
{STIVA*aux;
l=l1;
aux=l;
while(aux→leg!=NULL)
  aux=aux→leg;
  aux→leg=l2;
  return l;}
```

CONCATENAREA A DOUĂ L.S.Î. DE TIP STIVĂ

```
void main()
{clrscr();
int i,x,y,n,m;
cout<<"Nr. elem stiva1:"; cin>>n;
l1=init(l1);
for(i=1;i<=n;i++)
{cout<<"info:"; cin>>x; l1=creare(l1,x);}
cout<<"stiva1 este:"; afisare(l1);
cout<<"\nNr. elem stiva2:"; cin>>m;
l2=init(l2);
for(i=1;i<=m;i++)
{cout<<"info:"; cin>>y; l2=creare(l2,y); }
cout<<"\nstiva2 este:"; afisare(l2);
cout<<"\nSTIVA CONCATENATA ESTE:";
l=concatenare(l1,l2,l); afisare(l);
getch();}
```

APLICAȚII PROPUSE

- **Testarea aplicațiilor 5.2.1, 5.2.2 și 5.2.3, pag. 86-92;**
- **Rezolvarea aplicațiilor 3, 4, 5, 6, 7, 8, 9, 10, pag. 93;**
- **Obs: se va utiliza suportul de curs și laborator " *Elemente de proiectarea algoritmilor. Ghid teoretic și practic*", Șef lucr. dr. mat. Mădălina Cărbureanu, Editura Universității Petrol-Gaze din Ploiești, 2021.**

Să vă fie de folos și spor la lucru!