

S01 – SDD

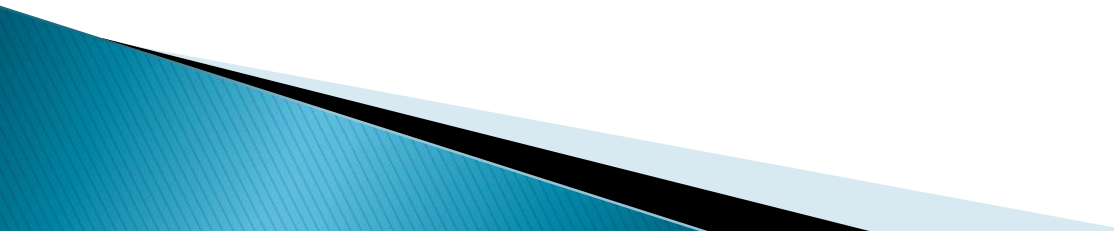
Alin Zamfiroiu

alin.zamfiroiu@csie.ase.ro

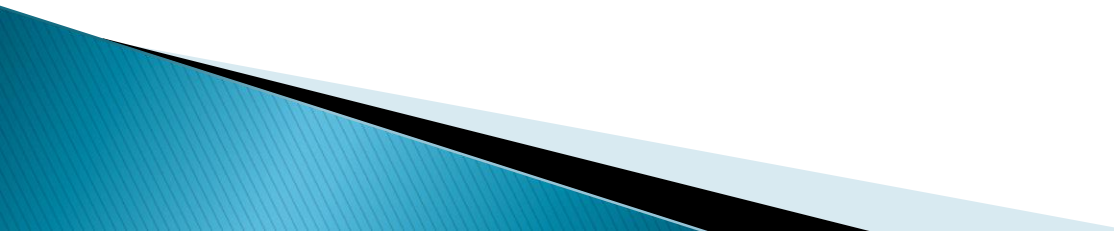
Consultații

- ▶ În fiecare zi de **marți** la sala 2215:
 - 18:00 – 19:30
- ▶ **Condiție:** Dat mail cel târziu luni seara cu ora la care se ajunge și problema care se dorește a fi discutată.
- ▶ În mail spunem cine suntem;
- ▶ Pe rețele de socializare spunem cine suntem.

Evaluare

- ▶ Examen – 60%
 - ▶ Lucrare – 16 aprilie – 20%
 - ▶ Proiect – 10%
 - ▶ Teme pe platforma – 10%
- 

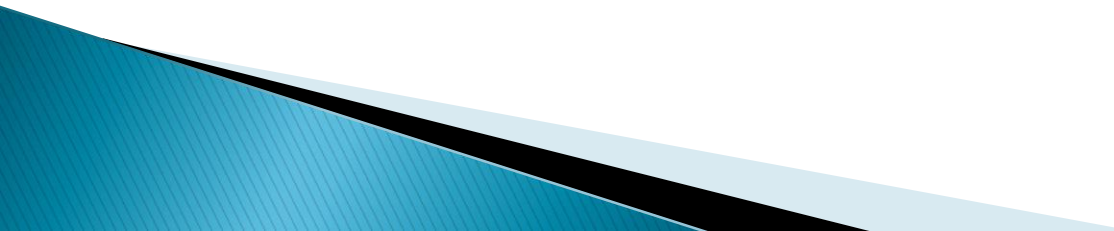
Proiect

- ▶ Sa se realizeze o aplicatie care citeste dintr-un fisier date cu privire la interactiunea utilizatorilor cu o aplicatie web, din punct de vedere al numarului de pagini deschise, numarul de click-uri, pozitia mouse-ului pe ecran si altele considerate de voi importante.
 - ▶ Alegeti minim 4 structuri adecvate in care sa memoratii informatiile din fisier.
 - ▶ Structura fisierului precum si datele stocate despre comportamentul utilizatorilor este la alegerea fiecarui student.
 - ▶ Aplicatia permite afisarea anumitor informatii dupa filtre sau pe utilizator.
 - ▶ Proiectul se prezinta la o data stabilita la a doua intalnire.
- 

Cuprins

- ▶ Vectori
 - ▶ Matrice
 - ▶ Liste simple
 - ▶ Liste duble
 - ▶ Arbori binari
-
- ▶ Tabele de dispersie
 - ▶ Heap

S01 – Continut

- ▶ Articol/structura;
 - ▶ Creare articol;
 - ▶ Tablou unidimensional static – vector;
 - ▶ Tablou unidimensional dinamic – vector;
 - Alocare spatiu;
 - Initializare cu valori;
 - Parcurgere;
 - Dealocare.
 - ▶ Tablou bidimensional static – matrice;
 - ▶ Tablou bidimensional dinamic – matrice;
 - Alocare spatiu;
 - Initializare cu valori;
 - Parcurgere;
 - Dealocare.
- 

S01 – Articole/Structuri

```
struct fruct{  
    char * denumire;  
    float gramaj;  
    int nrZile;  
    bool areSambure;  
};
```



S01 – Articol

```
void main()  
{  
    fruct mar;  
    mar.denumire=(char*)malloc((strlen("mar")+1)*sizeof(char));  
    strcpy(mar.denumire,"mar");  
    mar.gramaj=120.4;  
    mar.nrZile=1;  
    mar.areSambure=false;  
}
```


S01 – Articol

```
mar.denumire=new char[strlen("mar")+1];
```

The diagram illustrates the mapping of C++ code to C code. Four arrows point from the C++ code to the C code: from 'new' to 'malloc', from 'char' to 'sizeof(char)', from 'strlen("mar")+1' to 'strlen("mar")+1', and from the closing bracket of the array declaration to the closing bracket of the malloc call.

```
mar.denumire=(char*)malloc(sizeof(char)*(strlen("mar")+1));
```

S01 – Articol

```
printf("%s %5.2f %d %d", mar.denumire, mar.gramaj, mar.nrZile, mar.areSambure);
```

```
free(mar.denumire);
```



S01 – Tablou unidimensional



- ▶ rezervare – static
- ▶ comparare – dinamic
- ▶ un vector de 10 fructe

S01 – Vector alocat static

```
const int nrFructe=10;

fruct fructe[nrFructe];

for(int i=0;i<nrFructe;i++)
{
    fructe[i].denumire=(char*)malloc(sizeof(char)*(strlen("portocala")+1));
    strcpy(fructe[i].denumire,"portocala");
    fructe[i].gramaj=200.0;
    fructe[i].nrZile=2;
    fructe[i].areSambure=false;
}

afisareVectorFructe(fructe,nrFructe);
```

S01 – Metoda afisare vector

```
void afisareVectorFructe(fruct *fructe, int nrFructe)
{
    for(int i=0;i<nrFructe;i++)
    {
        printf("%s %5.2f %d %d \n", fructe[i].denumire,
            fructe[i].gramaj,fructe[i].nrZile,fructe[i].areSambure);
    }
}
```

S01 – Vector alocat dinamic

```
const int nrFructe=3;  
fruct *fructeDinamic;  
  
fructeDinamic=citireVectorFructe(nrFructe);  
  
afisareVectorFructe(fructeDinamic,nrFructe);
```

S01 – Metoda citire vector

```
fruct *citireVectorFructe(int nrFructe)
{
    fruct *fructe;
    fructe=(fruct*)malloc(nrFructe*sizeof(fruct));

    for(int i=0;i<nrFructe;i++)
    {
        printf("Denumire:");
        char aux[20];
        scanf("%s",&aux);
        fructe[i].denumire=(char*)malloc(sizeof(char)*(strlen(aux)+1));
        strcpy(fructe[i].denumire,aux);
        printf("Gramaj:");
        scanf("%f",&fructe[i].gramaj);
        printf("Acum cate zile a fost cules:");
        scanf("%d",&fructe[i].nrZile);
        printf("Are sambure? (0 - nu; 1 - da):");
        scanf("%d",&fructe[i].areSambure);
    }
    return fructe;
}
```

S01 – Alocarea dinamica

```
fructe=(fruct*)malloc(sizeof(fruct)*nrFructe);
```

The diagram illustrates the mapping between the C `malloc` function and the C++ `new` operator. Arrows point from the corresponding parts of the two code snippets below:

- `malloc` maps to `new`
- `sizeof(fruct)` maps to `fruct`
- `*nrFructe` maps to `[nrFructe]`

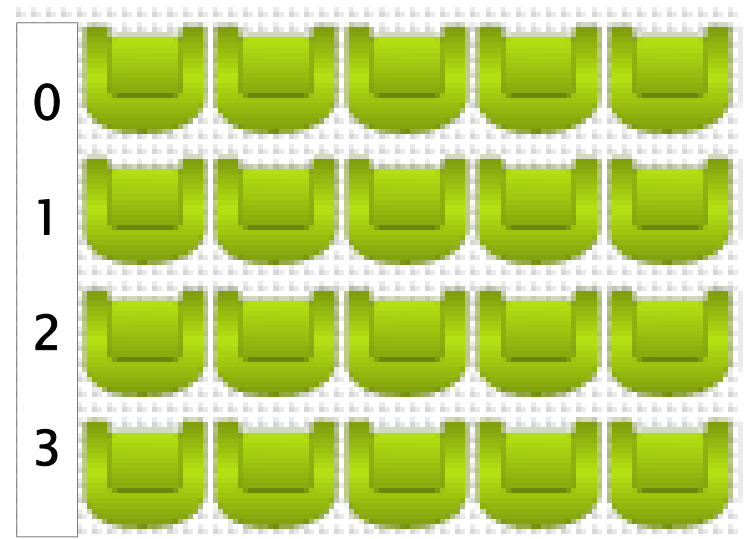
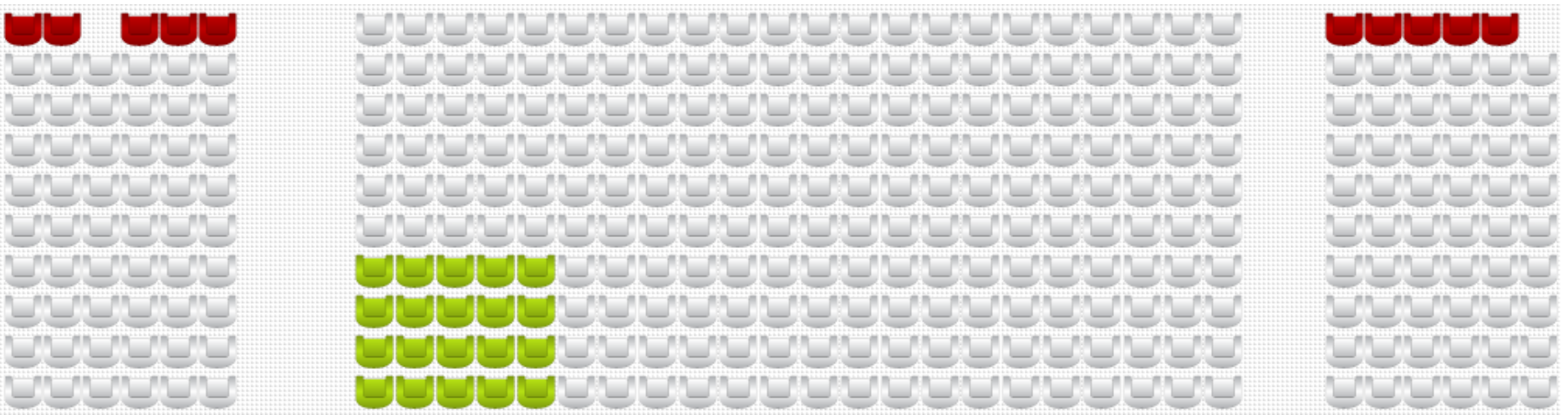
```
fructe = new fruct[nrFructe];
```


S01 – Dezalocare spatiu

```
//stregere vector  
for(int i=0;i<nrFructe;i++)  
    free(fructeDinamic[i].denumire);  
free(fructeDinamic);|
```



S01 – Matrice



S01 – Matrice alocată static

```
int n=4;
int m=5;
int matrice[10][10];

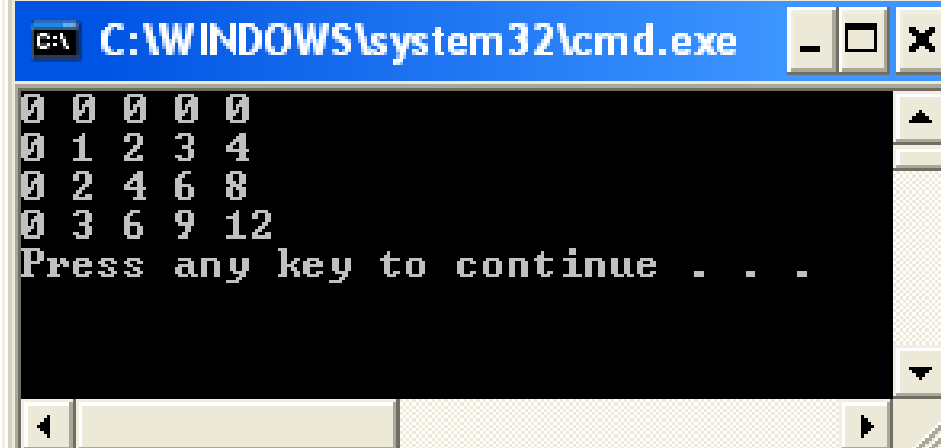
for(int i=0;i<n;i++)
{
    for(int j=0;j<m;j++)
    {
        matrice[i][j]=i*j;
    }
}

afisareMatriceInt(matrice,n,m);
```



S01 – Afişare matrice

```
void afisareMatriceInt(int matrice[][10], int n, int m)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            printf("%d ",matrice[i][j]);
        }
        printf("\n");
    }
}
```



A screenshot of a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The window displays the output of the matrix printing function, showing a 4x5 matrix of integers. The first row contains five zeros. The subsequent three rows contain the values 1, 2, 3, 4; 2, 4, 6, 8; and 3, 6, 9, 12 respectively. The prompt "Press any key to continue . . ." is visible at the bottom of the window.

```
C:\WINDOWS\system32\cmd.exe
0 0 0 0 0
0 1 2 3 4
0 2 4 6 8
0 3 6 9 12
Press any key to continue . . .
```

S01 – Matrice alocată dinamic

```
fruct ** matriceFructe;  
int n;  
int m;  
matriceFructe=CitireMatriceFructe(&n,&m);  
afisareMatriceFructe(matriceFructe, n, m);
```

- ▶ alocare linii;
- ▶ alocare elemente.



S01 – Citire matrice fructe

Pas 1.

```
fruct **CitireMatriceFructe(int *n,int *m)
{
    printf("Numarul de tipuri de fructe");
    scanf("%d",n);
    printf("Numarul de fructe pe tip");
    scanf("%d",m);

    fruct ** matriceFructe;
    matriceFructe=(fruct**)malloc(sizeof(fruct*)>(*n));
```

Pas 2.

```
for(int i=0;i<*n;i++)
{
    matriceFructe[i]=(fruct*)malloc(sizeof(fruct)*(*m));
    printf("Grupa %d de fructe:\n",(i+1));
```

S01 – Citire matrice fructe

Pas 3.

```
for(int j=0;j<*m;j++)
{
    printf("Denumire:");
    char aux[20];
    scanf("%s",&aux);
    matriceFructe[i][j].denumire=(char*)malloc(sizeof(char));
    strcpy(matriceFructe[i][j].denumire,aux);
    printf("Gramaj:");
    scanf("%f",&matriceFructe[i][j].gramaj);
    printf("Acum cate zile a fost cules:");
    scanf("%d",&matriceFructe[i][j].nrZile);
    printf("Are sambure? (0 - nu; 1 - da):");
    scanf("%d",&matriceFructe[i][j].areSambure);
}
```

Pas 4.

```
return matriceFructe;
```

S01 – Afişare matrice

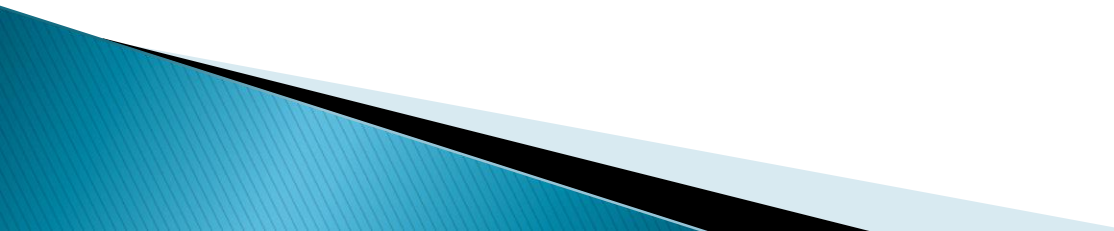
```
void afisareMatriceFructe(fruct **matrice, int n, int m)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            printf("%s %5.2f %d %d \t\t",
                matrice[i][j].denumire,
                matrice[i][j].gramaj,
                matrice[i][j].nrZile,
                matrice[i][j].areSambure);
        }
        printf("\n");
    }
}
```


S01 – Stergere spatiu matrice

```
for(int i=0;i<n;i++)
{
    for(int j=0;j<m;j++)
    {
        free(matriceFructe[i][j].denumire);
    }
    free(matriceFructe[i]);
}
free(matriceFructe);
```



S02 – Continut

- ▶ Articol televizor
 - ▶ Citire articol
 - ▶ Citire vector
 - ▶ Citire matrice
 - ▶ Matrice in zig zag
 - ▶ Tablou n-dimensional
 - ▶ Sortare vector
- 

S02 – Articol televizor

```
struct televizor  
{  
    char*marca;  
    int diagonala;  
};
```



S02 – Citire articol

```
void main()
{
    televizor t;
    t.marca=(char*)malloc((strlen("Anonim")+1)*sizeof(char));
    strcpy(t.marca,"Anonim");
    t.diagonala=10;
    citireArticol(t);
    printf("%s %d",t.marca,t.diagonala);
}
```



```
void citireArticol(televizor t)
{
    char aux[20];
    printf("Introduceti marca televizorului:");
    scanf("%s",&aux);
    t.marca=(char*)malloc((strlen(aux)+1)*sizeof(char));
    strcpy(t.marca,aux);
    printf("Introduceti diagonala televizorului:");
    scanf("%d", &t.diagonala);
}
```

S02 – Citire articol

```
void citireArticol(televizor* t)
{
    char aux[20];
    printf("Introduceti marca televizorului:");
    scanf("%s",&aux);
    t->marca=(char*)malloc((strlen(aux)+1)*sizeof(char));
    strcpy(t->marca,aux);
    printf("Introduceti diagonala televizorului:");
    scanf("%d", &t->diagonala);
}
```

```
televizor t;
citireArticol(&t);
```

***If you think you can
make it better...***

Do it.

S02 – Citire articol

```
void citireArticol(televizor& t)
{
    char aux[20];
    printf("Introduceti marca televizorului:");
    scanf("%s",&aux);
    t.marca=(char*)malloc((strlen(aux)+1)*sizeof(char));
    strcpy(t.marca,aux);
    printf("Introduceti diagonala televizorului:");
    scanf("%d", &t.diagonala);
}
```

```
void main()
{
    televizor t;
    citireArticol(t);
    printf("%s %d",t.marca,t.diagonala);
}
```

S02 – Citire vector

```
televizor* vector;  
int lungime;  
citireVector(&vector,&lungime);  
afisareVector(vector,lungime);
```



S02 – Citire vector

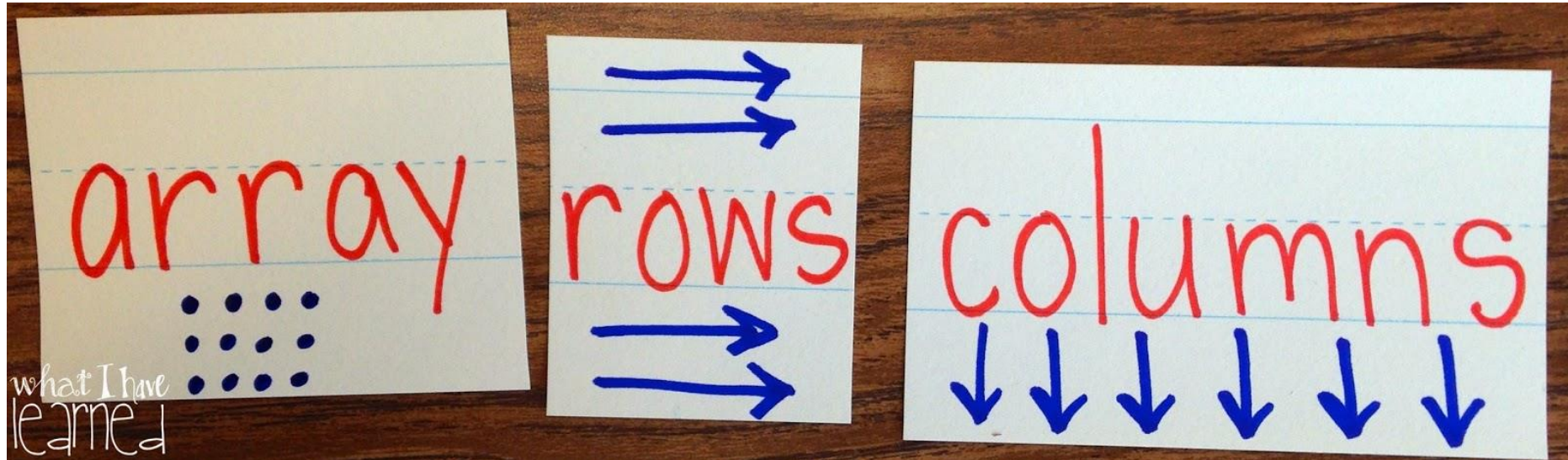
```
void citireVector(televizor**v,int*n)
{
    printf("Introduceti numarul de televizoare");
    scanf("%d",n);
    (*v)=(televizor*)malloc(sizeof(televizor)*(*n));
    for(int i=0;i<*n;i++)
    {
        printf("Introduceti marca televizorului:");
        char aux[20];
        scanf("%s",&aux);
        ((*v)[i]).marca=(char*)malloc(sizeof(char)*(strlen(aux)+1));
        strcpy((*v)[i].marca,aux);
        printf("Introduceti Diagonala televizorului:");
        scanf("%d",&(*v)[i].diagonala);
    }
}
```

```
void afisareVector(televizor*vector,int n)
{
    for(int i=0;i<n;i++)
        printf("%s %d\n",vector[i].marca,vector[i].diagonala);
}
```

S02 – Citire vector

```
void citireVector2(televizor* &v, int &n)
{
    printf("Introduceti numarul de televizoare");
    scanf("%d", &n);
    v = (televizor*) malloc(sizeof(televizor)*(n));
    for(int i=0; i<n; i++)
    {
        printf("Introduceti marca televizorului:");
        char aux[20];
        scanf("%s", &aux);
        (v[i]).marca = (char*) malloc(sizeof(char)*(strlen(aux)+1));
        strcpy(v[i].marca, aux);
        printf("Introduceti Diagonala televizorului:");
        scanf("%d", &v[i].diagonala);
    }
}
```

S02 – Citire matrice



S02 – Citire matrice

```
void citireMatrice(televizor** &matrice, int &linii, int &coloane)
{
    printf("numarul de linii:");
    scanf("%d",&linii);
    printf("numarul de coloane:");
    scanf("%d",&coloane);

    matrice=(televizor**)malloc(sizeof(televizor*)*linii);
    for(int i=0;i<linii;i++)
    {
        matrice[i]=(televizor*)malloc(sizeof(televizor)*coloane);
    }

    for(int i=0;i<linii;i++)
    {
        for(int j=0;j<coloane;j++)
        {
            char aux[20];
            printf("Introduceti marca televizorului:");
            scanf("%s",&aux);
            matrice[i][j].marca=(char*)malloc((strlen(aux)+1)*sizeof(char));
            strcpy(matrice[i][j].marca,aux);
            printf("Introduceti diagonala televizorului:");
            scanf("%d", &matrice[i][j].diagonala);
        }
    }
}
```

S02 – Matrice in Zig Zag

	4	5			
	3	2	6	3	
	2	6	4		
	6	1			

S02 – Matrice in Zig Zag

```
void citireMatriceZigZag(televizor** &matrice, int &linii, int* &coloane)
{
    printf("numarul de linii:");
    scanf("%d",&linii);
    matrice=(televizor**)malloc(sizeof(televizor*)*linii);
    coloane=(int*)malloc(sizeof(int)*linii);

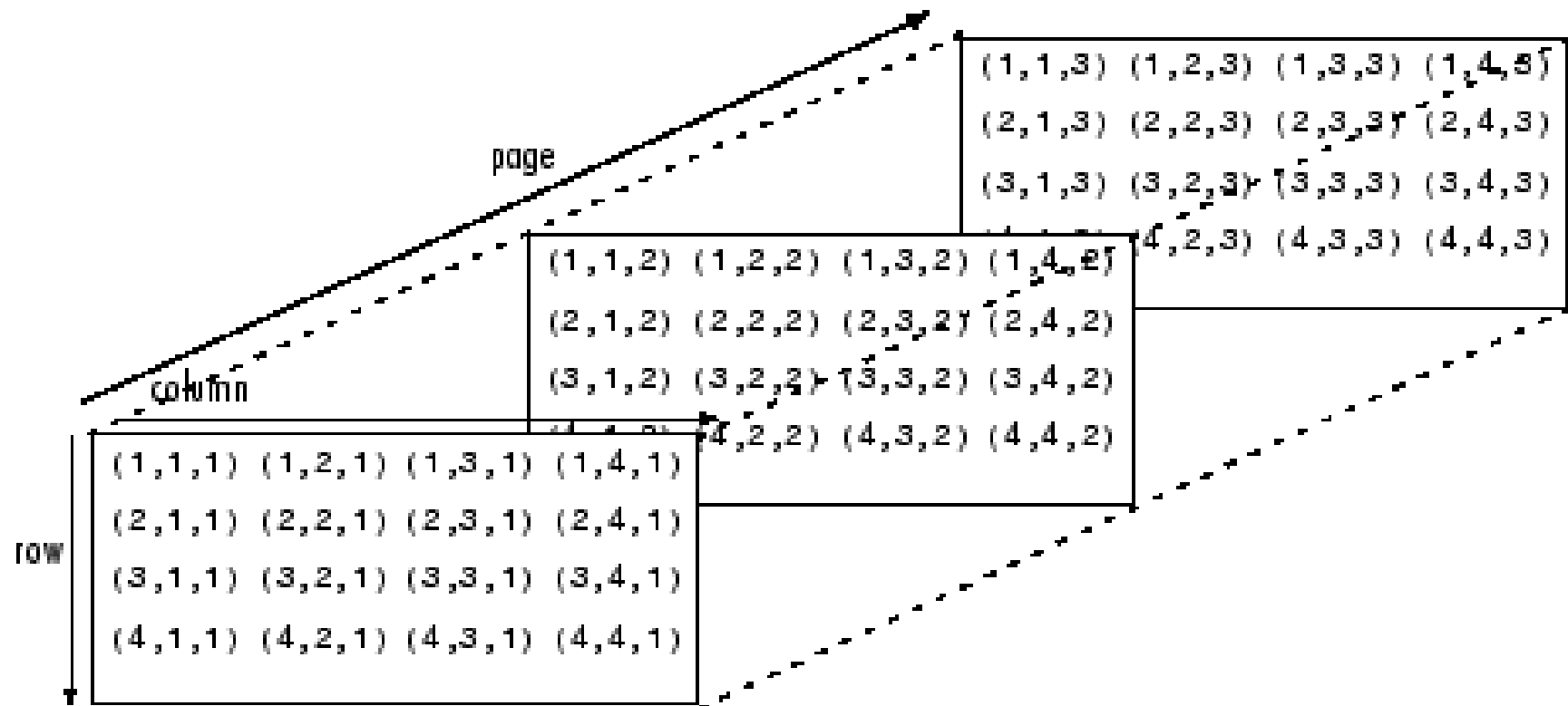
    for(int i=0;i<linii;i++)
    {
        printf("numarul de coloane pentru linia %d:",(i+1));
        scanf("%d",&coloane[i]);
        matrice[i]=(televizor*)malloc(sizeof(televizor)*coloane[i]);

        for(int j=0;j<coloane[i];j++)
        {
            char aux[20];
            printf("Introduceti marca televizorului:");
            scanf("%s",&aux);
            matrice[i][j].marca=(char*)malloc((strlen(aux)+1)*sizeof(char));
            strcpy(matrice[i][j].marca,aux);
            printf("Introduceti diagonala televizorului:");
            scanf("%d", &matrice[i][j].diagonala);
        }
    }
}
```

S02 – Matrice in Zig Zag

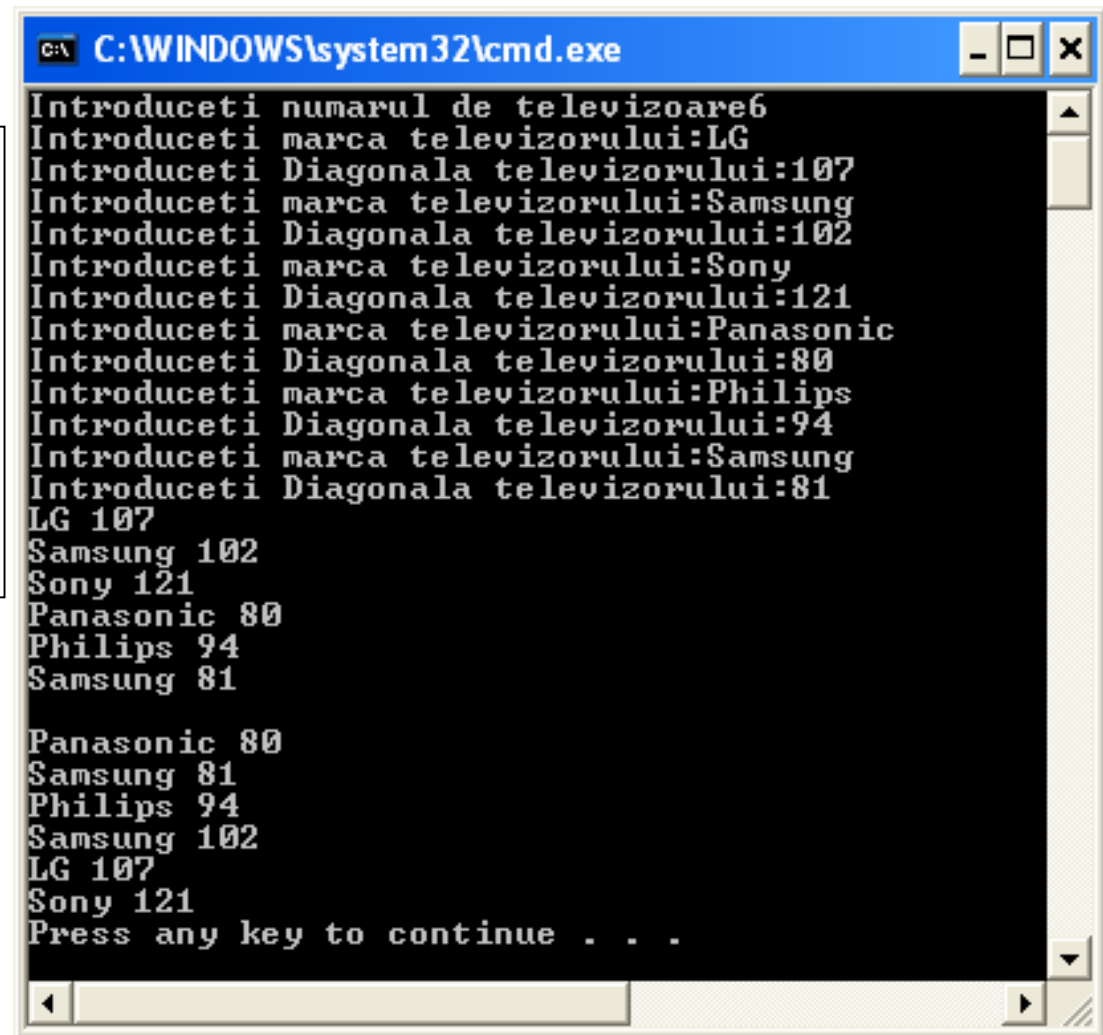
```
void afisareMatriceZigZag(televizor **matrice, int n, int *m)
{
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m[i];j++)
        {
            printf("%s %d \t\t", matrice[i][j].marca, matrice[i][j].diagonala);
        }
        printf("\n");
    }
}
```

S02 – Tablou n-dimensional



S02 – Sortare vector

```
televizor* vector;  
int lungime;  
citireVector2(vector, lungime);  
afisareVector(vector, lungime);  
printf("\n");  
sortareVector(vector, lungime);  
afisareVector(vector, lungime);
```



C:\WINDOWS\system32\cmd.exe


```
Introduceti numarul de televizoare6  
Introduceti marca televizorului:LG  
Introduceti Diagonala televizorului:107  
Introduceti marca televizorului:Samsung  
Introduceti Diagonala televizorului:102  
Introduceti marca televizorului:Sony  
Introduceti Diagonala televizorului:121  
Introduceti marca televizorului:Panasonic  
Introduceti Diagonala televizorului:80  
Introduceti marca televizorului:Philips  
Introduceti Diagonala televizorului:94  
Introduceti marca televizorului:Samsung  
Introduceti Diagonala televizorului:81  
LG 107  
Samsung 102  
Sony 121  
Panasonic 80  
Philips 94  
Samsung 81  
  
Panasonic 80  
Samsung 81  
Philips 94  
Samsung 102  
LG 107  
Sony 121  
Press any key to continue . . .
```

S02 – Sortare vector

```
void sortareVector(televizor* &v, int n)
{
    for(int i=0; i<n-1; i++)
    {
        for(int j=i+1; j<n; j++)
        {
            if(v[i].diagonala > v[j].diagonala)
            {
                char aux[20];
                strcpy(aux, v[i].marca);
                free(v[i].marca);
                v[i].marca = (char*)malloc(sizeof(char)*(strlen(v[j].marca)+1));
                strcpy(v[i].marca, v[j].marca);
                free(v[j].marca);
                v[j].marca = (char*)malloc(sizeof(char)*(strlen(aux)+1));
                strcpy(v[j].marca, aux);
                int temp = v[i].diagonala;
                v[i].diagonala = v[j].diagonala;
                v[j].diagonala = temp;
            }
        }
    }
}
```



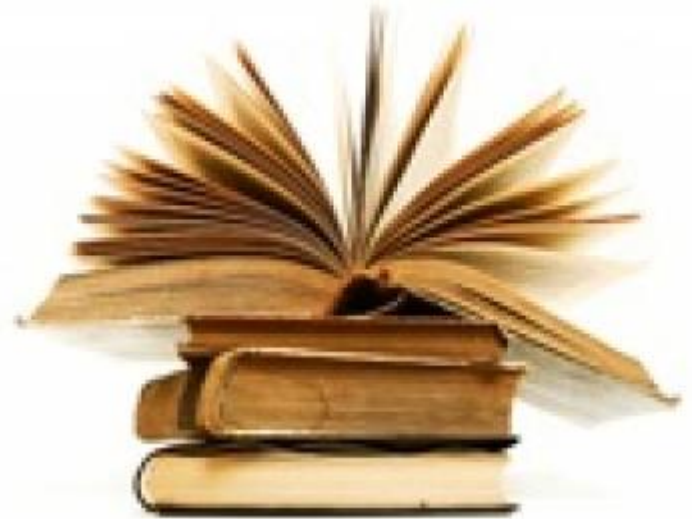
S03 – Continut

- ▶ Articol carte
 - ▶ Lista
 - ▶ Inserare la inceput
 - ▶ Parcurgere
 - ▶ Adaugare la sfarsit
 - ▶ Inserare la mijloc
 - ▶ Stergere element
 - ▶ Stergere lista
 - ▶ Liste duble
- 

S03 – Lista liniara

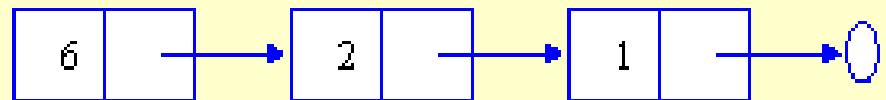
```
struct carte{  
    char*titlu;  
    float pret;  
};
```

```
struct nod{  
    carte info;  
    nod*next;  
};
```



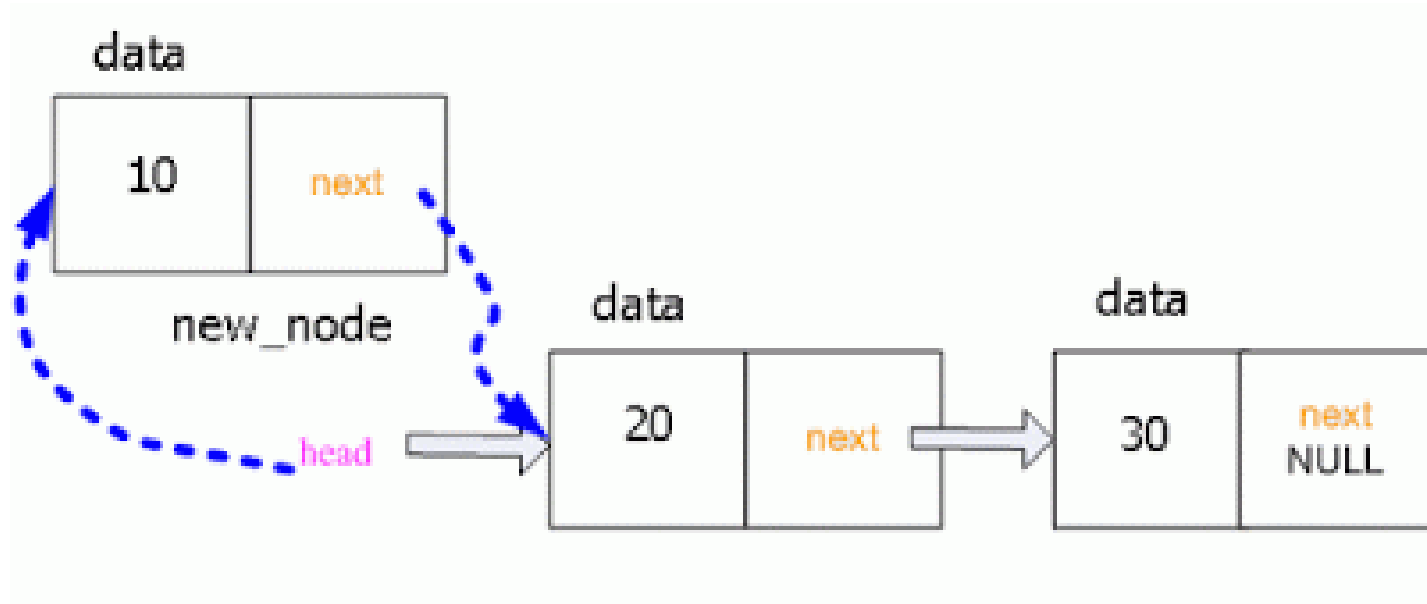
<http://www.artline.ro/>

_id _pNext



www.relisoft.com

S03 – Inserare la inceput



<http://www.c4learn.com/>

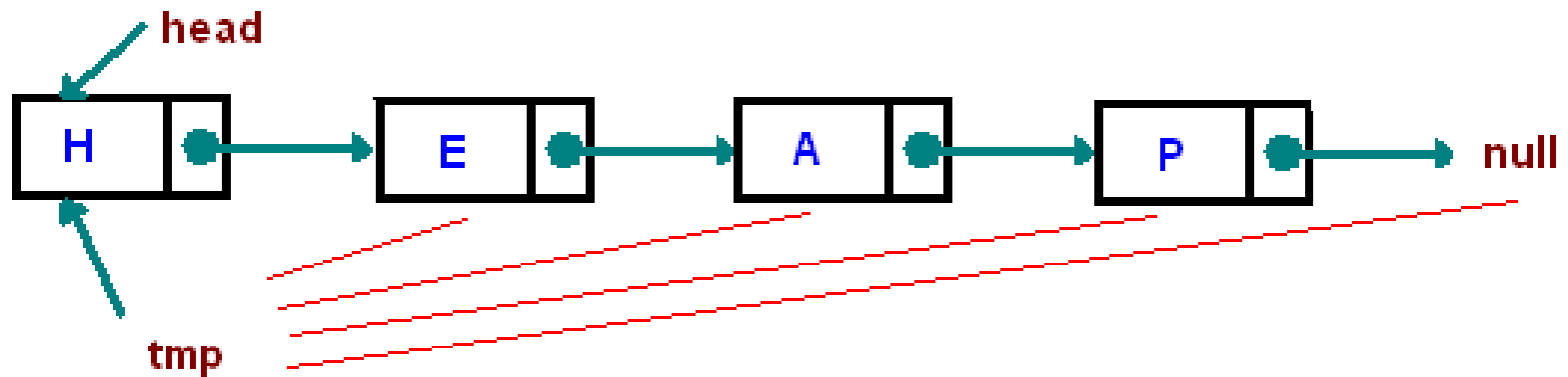
S03 – Inserare la inceput

```
nod*inserare_inceput(nod*cap, carte info)
{
    nod *aux=(nod*)malloc(sizeof(nod));
    aux->info=info;
    aux->next=cap;
    return aux;
}
```

Apelati functia din programul principal pentru inserarea a cel puțin 3 carti intr-o lista initializata cu NULL.

```
nod*inserare_inceput(nod*cap, carte info)
{
    //creez un nou nod si aloc spatiu pentru el
    nod *aux=(nod*)malloc(sizeof(nod));
    //setez informatia cu parametrul primit;
    aux->info=info;
    //fac legatura catre primul nod al listei
    aux->next=cap;
    //returnez noul ca si cap sau prim nod al listei
    return aux;
}
```

S03 – Parcurgere

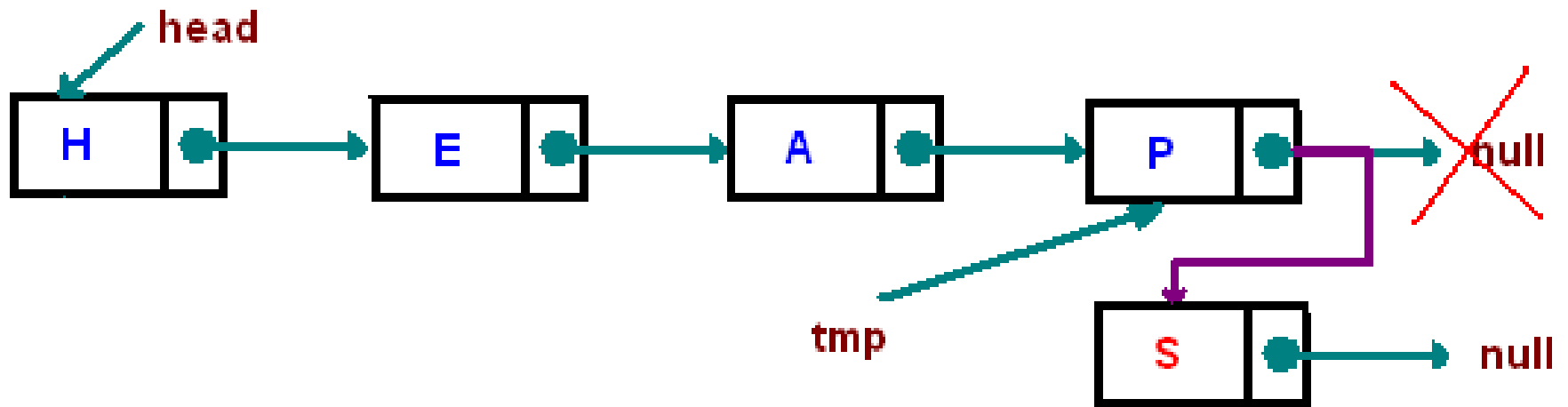


<http://www.cs.cmu.edu/>

S03 – Parcurgere

```
void afisareLista(nod*cap)
{
    nod*p=cap;
    while(p)
    {
        printf("%s costa %5.2f\n", p->info.titlu,p->info.pret);
        p=p->next;
    }
}
```

S03 – Adaugare la sfarsit



<http://www.cs.cmu.edu/>

S03 – Adaugare la sfarsit

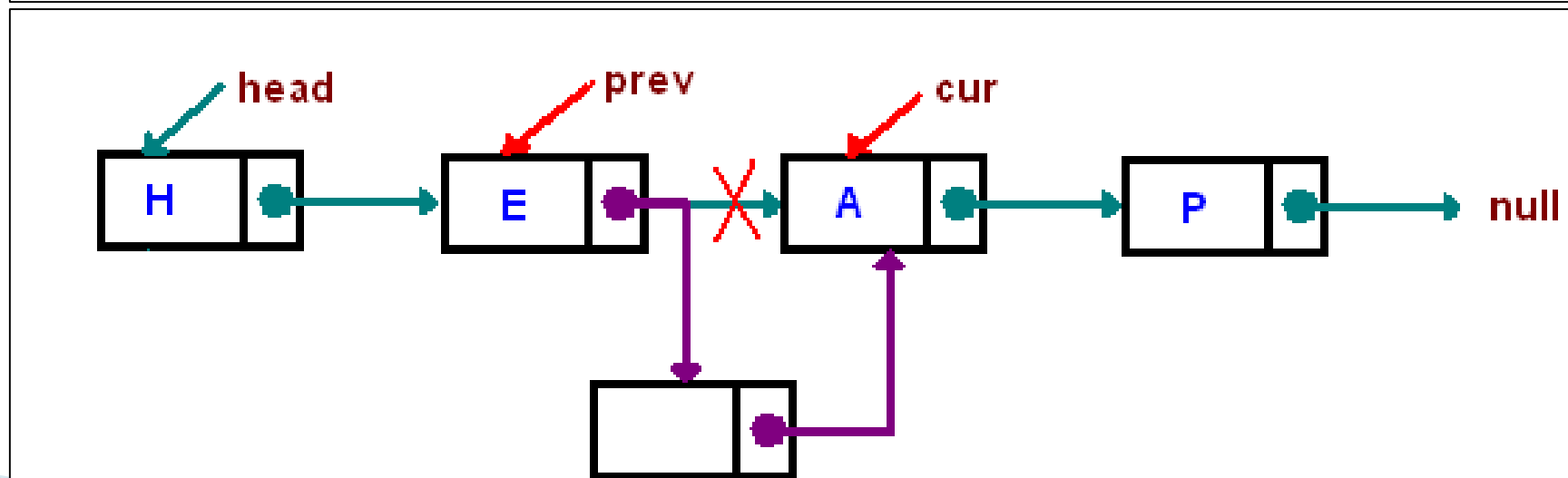
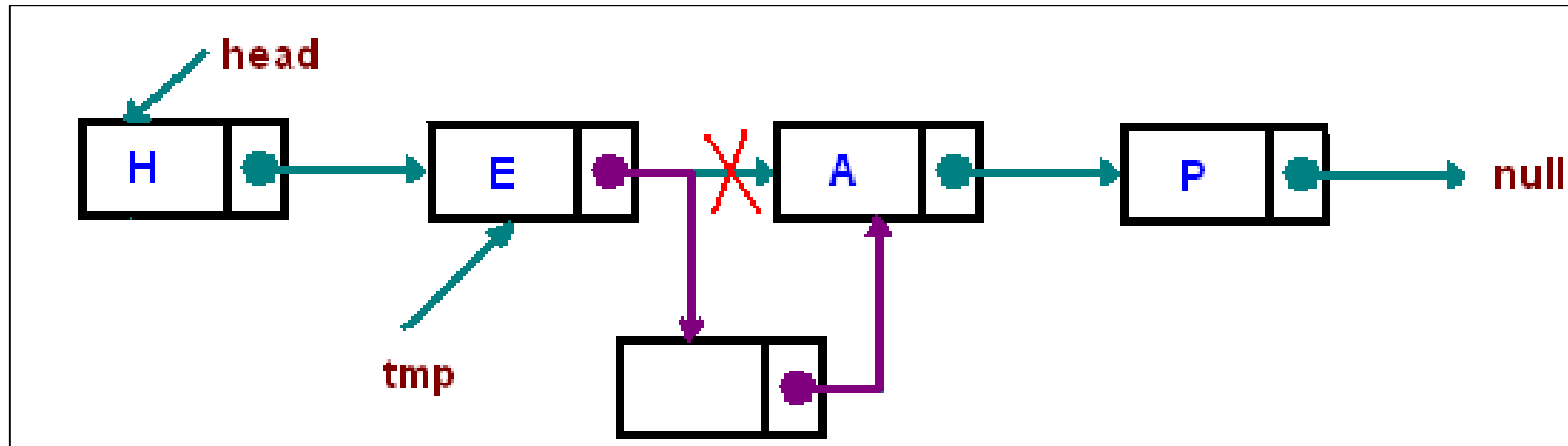
Apelati functia din programul principal pentru inserarea a cel puțin 3 carti intr-o lista initializata cu NULL.

Dupa inserarea celor 3 carti la sfarsitul listei, inserati alte 3 carti la inceputul listei.

```
nod* inserare_sfarsit(nod*cap, carte info)
{
    nod* nou=(nod*)malloc(sizeof(nod));
    nou->info=info;
    nou->next=NULL;
    if(cap)
    {
        nod*p=cap;
        while(p->next)
        {
            p=p->next;
        }
        p->next=nou;
    }
    else
    {
        cap=nou;
    }

    return cap;
}
```

S03 – Inserare la mijloc



S03 – Inserare la mijloc

```
nod*inserare_crescatoare(nod*cap, carte info)
{
    nod*nou=(nod*)malloc(sizeof(nod));
    nou->info=info;
    if(cap)
    {
        if(cap->info.pret>info.pret)
        {
            cap=inserare_inceput(cap, info);
            return cap;
        }
    }
}
```

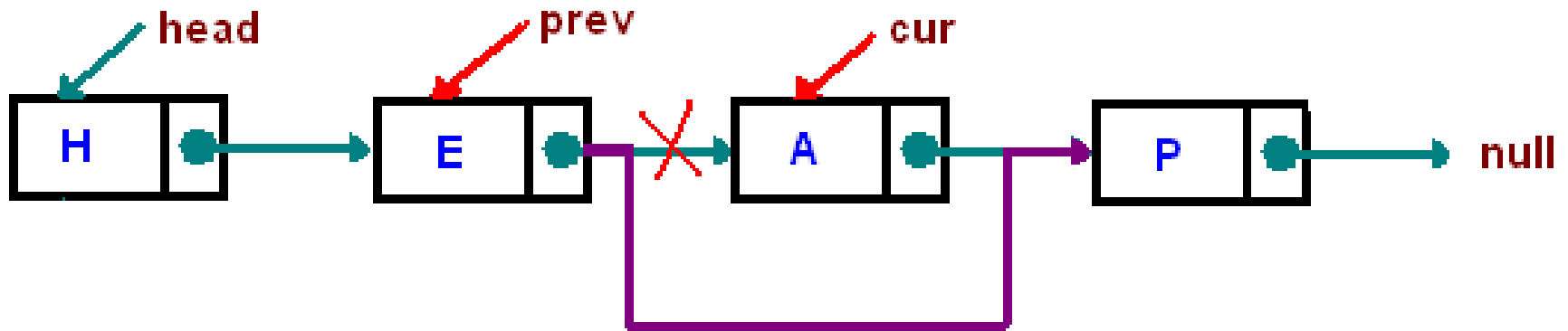


S03 – Inserare la mijloc

```
    else
    {
        nod *p=cap;
        while(p->next && p->next->info.pret<info.pret)
            p=p->next;
        nou->next=p->next;
        p->next=nou;
        return cap;
    }
else
{
    return nou;
}
}
```



S03 – Stergere element



S03 – Stergere element

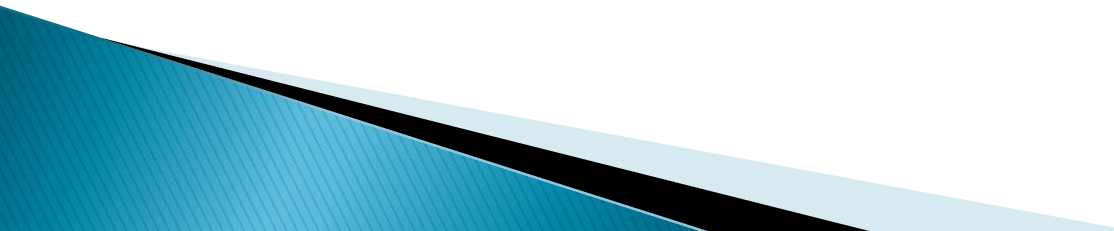
```
nod*stergere(nod*cap,float pret)
{
    if(cap)
    {
        if(cap->info.pret==pret)
        {
            nod*p=cap;
            cap=cap->next;
            free(p->info.titlu);
            free(p);
        }
        else
        {
            nod*p=cap;
            while(p->next&& p->next->info.pret!=pret)
            {
                p=p->next;
            }
            nod*aux=p->next;
            p->next=p->next->next;
            free(aux->info.titlu);
            free(aux);
        }
    }
    return cap;
}
```


S03 – Stergere lista

```
nod*stergere_lista(nod*cap)
{
    while(cap)
    {
        nod*p=cap;
        cap=cap->next;
        free(p->info.titlu);
        free(p);
    }
    return NULL;
}
```



S04 – Continut

- ▶ Articol cladiare
 - ▶ Lista dubla: noduri si lista
 - ▶ Inserare la inceput
 - ▶ Parcurgere
 - ▶ Adaugare la sfarsit
 - ▶ Inserare la mijloc
 - ▶ Extragere element
 - ▶ Stive si cozi
- 

S04 – Lista dubla



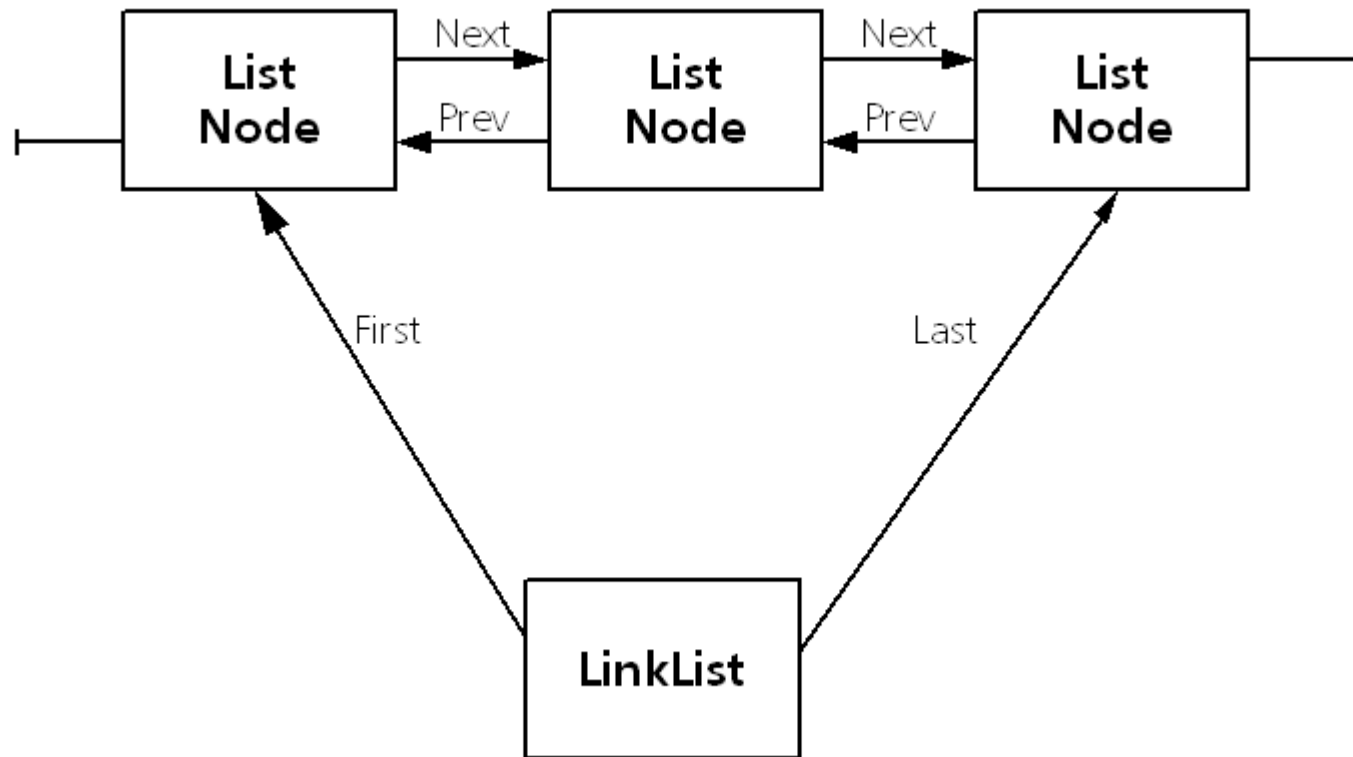
<http://proiectantidestructuri.ro/>

```
struct cladire
{
    int nr_etaje;
    int nr_camere;
};

struct nodLDI{
    cladire info;
    nodLDI *next;
    nodLDI *prev;
};

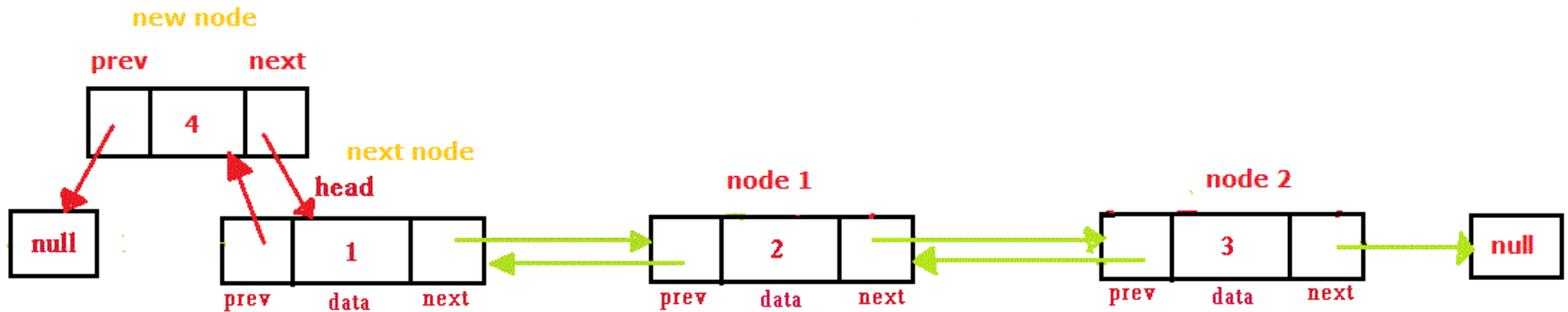
struct LDI
{
    nodLDI *prim;
    nodLDI *ult;
};
```

S04 – Lista dubla



<http://destructor.de/linklist/index.htm>

S04 – Inserare la inceput



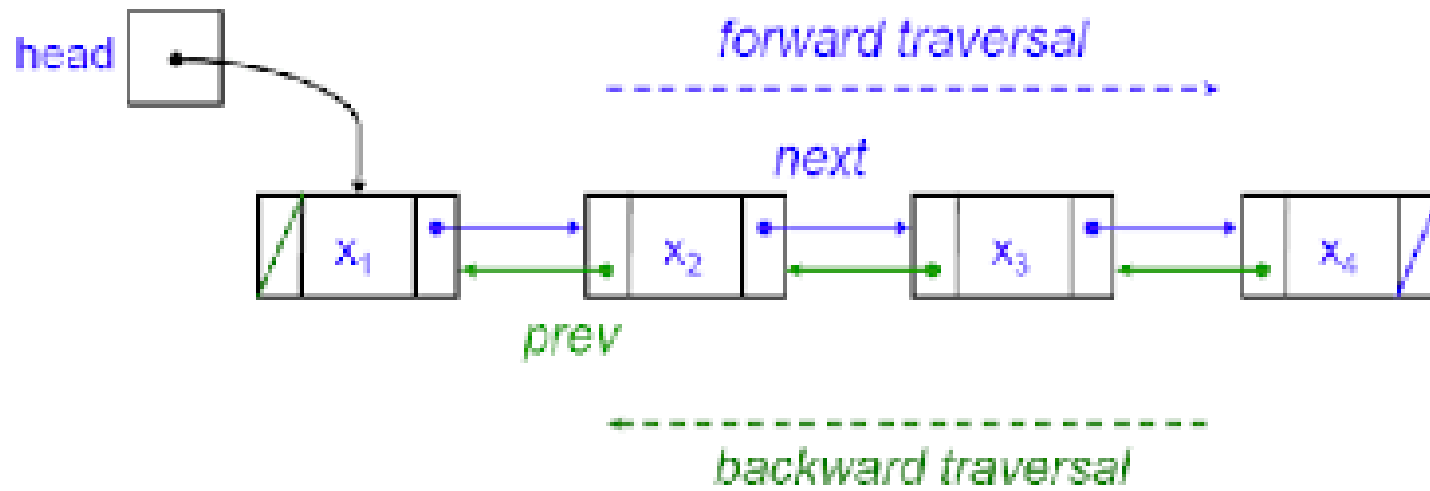
<http://www.mybodhizone.com/>

S04 – Inserare la inceput

```
- void inserare_inceput(LDI*lista,cladire c){  
    nodLDI* nou=(nodLDI*)malloc(sizeof(nodLDI));  
    nou->info=c;  
    nou->next=nou->prev=NULL;  
    if(lista->prim==NULL){  
        lista->prim=lista->ult=nou;  
    }  
    else  
    {  
        nou->next=lista->prim;  
        lista->prim->prev=nou;  
        lista->prim=nou;  
    }  
}
```

Apelati functia din programul principal pentru inserarea a cel puțin 3 cladiri într-o lista dubla.

S04 – Parcurgere



https://wiki.cs.auckland.ac.nz/compsci105ss/index.php/Linked_Lists

S04 – Parcurgere

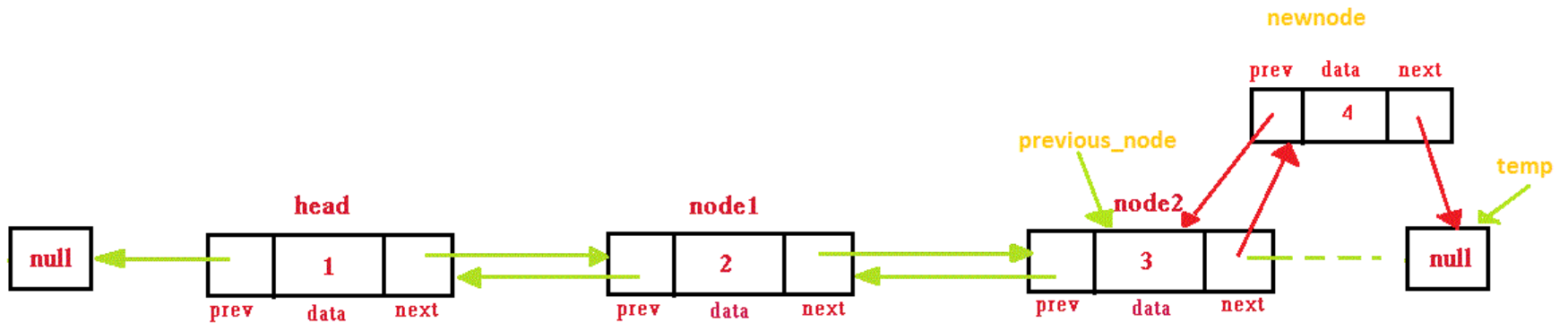
```
void parcurgereLista(LDI*lista)
{
    nodLDI* p=lista->prim;
    while(p){
        printf("Cladirea are %d etaje si %d camere",p->info.nr_etaje,p->info.nr_camere);
        p=p->next;
    }
}
```

Cum facem parcurgerea inversa?

S04 – Parcurgere

```
void parcurgereInversa(LDI*lista)
{
    nodLDI* p=lista->ult;
    while(p){
        printf("Cladirea are %d etaje si %d camere",p->info.nr_etaje,p->info.nr_camere);
        p=p->prev;
    }
}
```

S04 – Adaugare la sfarsit



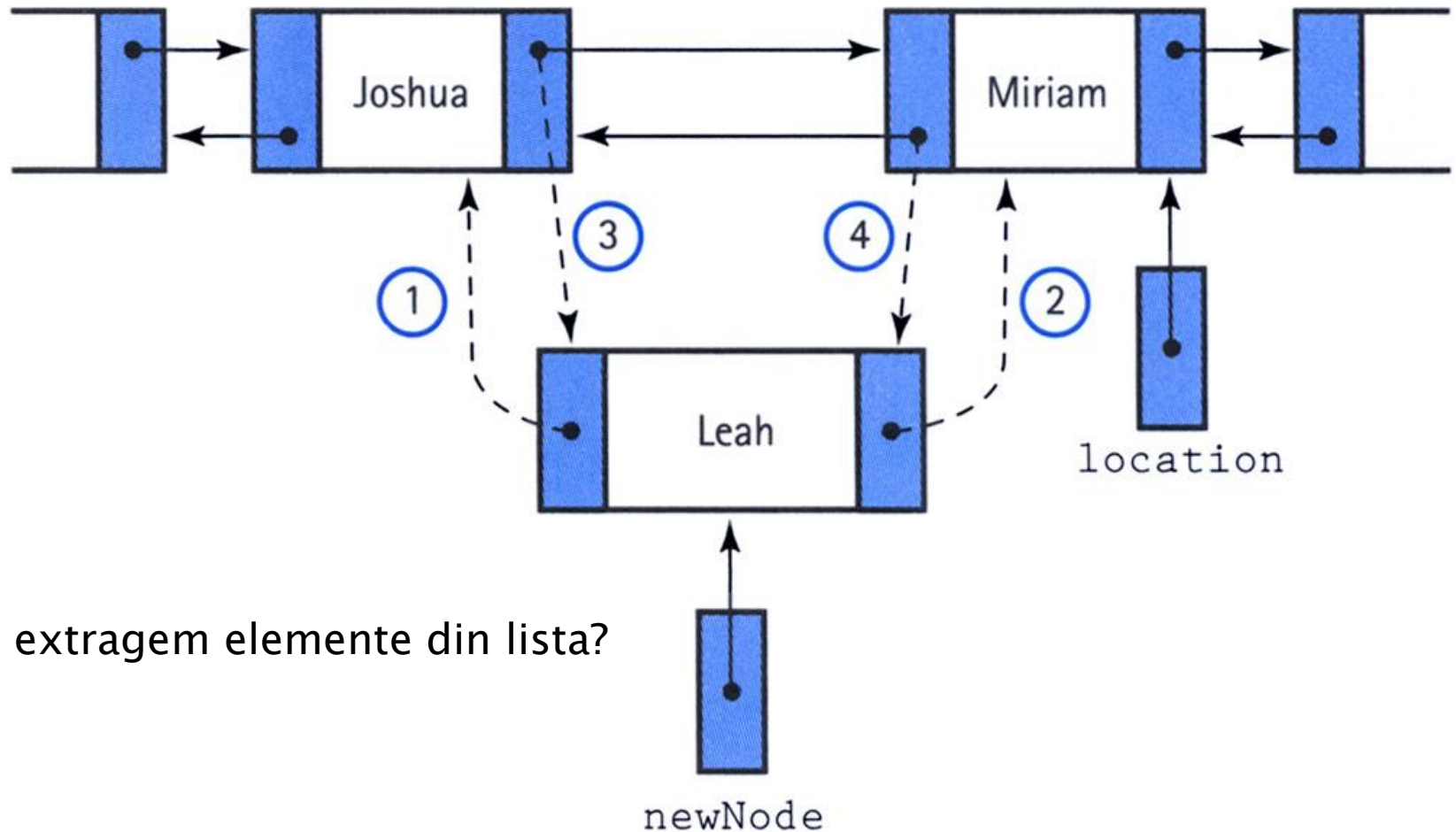
<http://www.mybodhizone.com/>

S04 – Adaugare la sfarsit

Apelati functia din programul principal pentru inserarea a cel putin 3 cladiri intr-o lista dubla.

```
void inserare_sfarsit(LDI*lista,cladire c){
    nodLDI* nou=(nodLDI*)malloc(sizeof(nodLDI));
    nou->info=c;
    nou->next=nou->prev=NULL;
    if(lista->prim==NULL){
        lista->prim=lista->ult=nou;
    }
    else
    {
        nou->prev=lista->ult;
        lista->ult->next=nou;
        lista->ult=nou;
    }
}
```

S04 – Inserare la mijloc



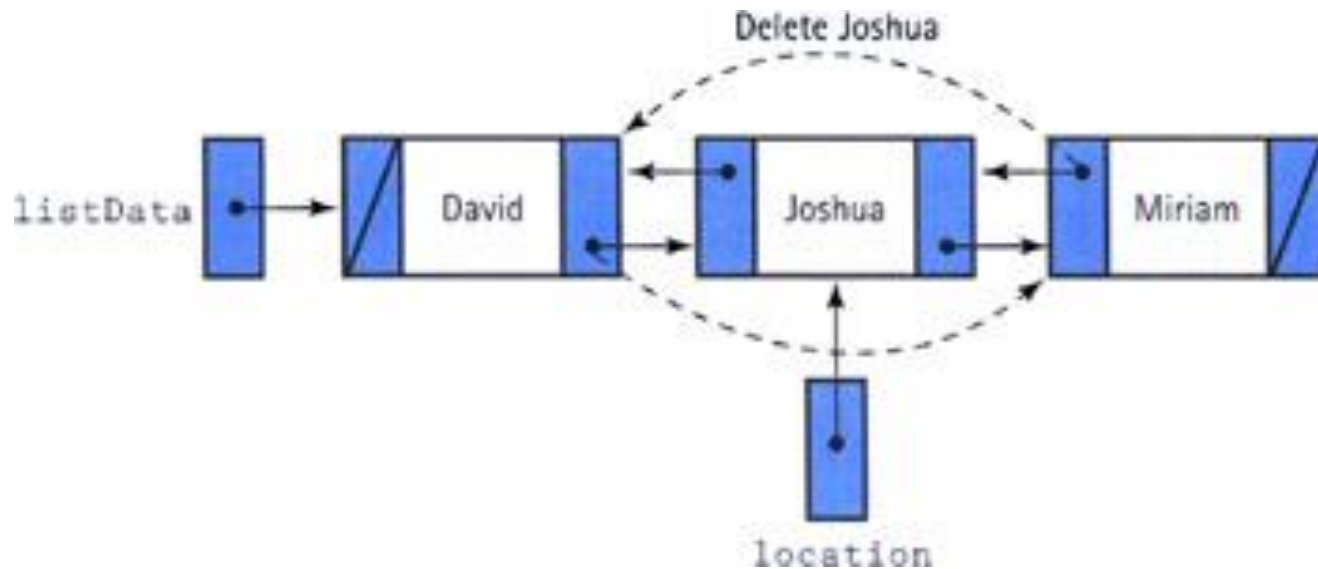
Cum extragem elemente din lista?

S04 – Extragere element

```
cladire extrageDeLaInceput(LDI* lista)
{
    if(lista->prim)
    {
        cladire c=lista->prim->info;
        nodLDI*p=lista->prim;
        lista->prim=lista->prim->next;
        if(lista->prim==NULL)
            lista->ult=NULL;
        free(p);
        return c;
    }
}
```

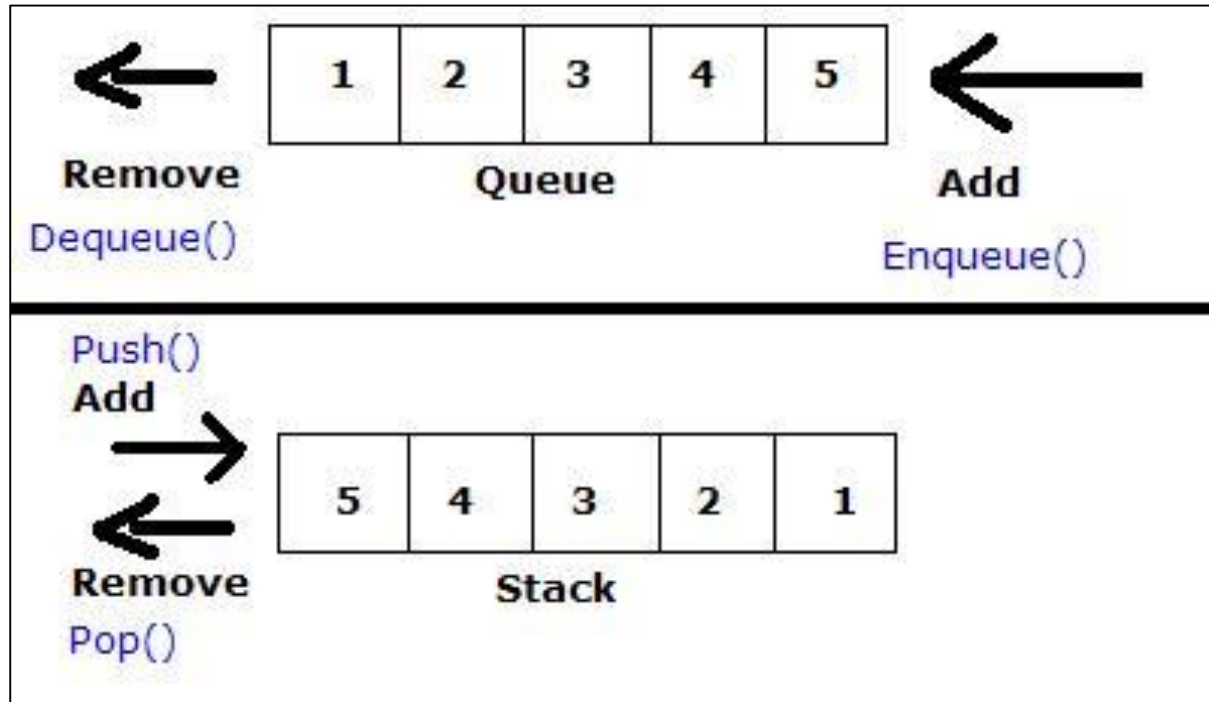
Cum extragem de la sfarsit?

S04 – Extragere element



<http://younginc.site11.com/source/5895/fos0052.html>

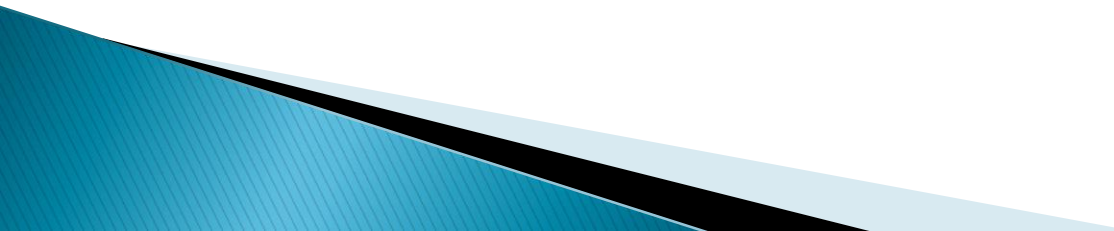
Stive si cozi



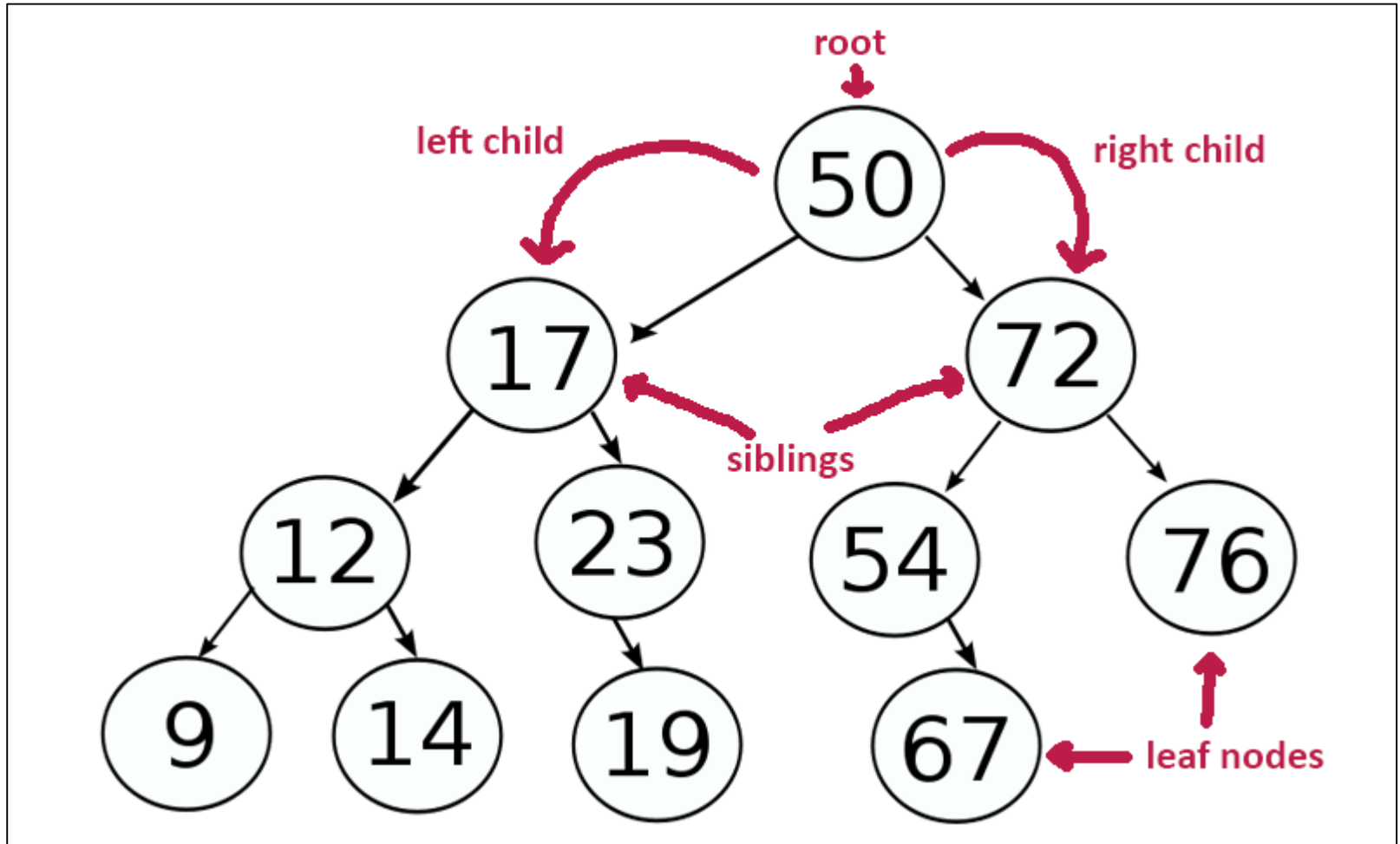
<http://www.c-sharpcorner.com/>



S06 – Arbori binari

- ▶ Structura tara si nod
 - ▶ Creare nod
 - ▶ Inserare
 - ▶ Parcurgere si afisare
 - ▶ Numarul de nivele
 - ▶ Afisare de pe un nivel
 - ▶ Cautare nod
 - ▶ Stergere
- 

S06 – Arbori binari



S06 – Structura Tara si nod

```
struct tara{  
    int id;  
    char* nume;  
    int nr_locuitori;  
    float suprafata;  
};
```

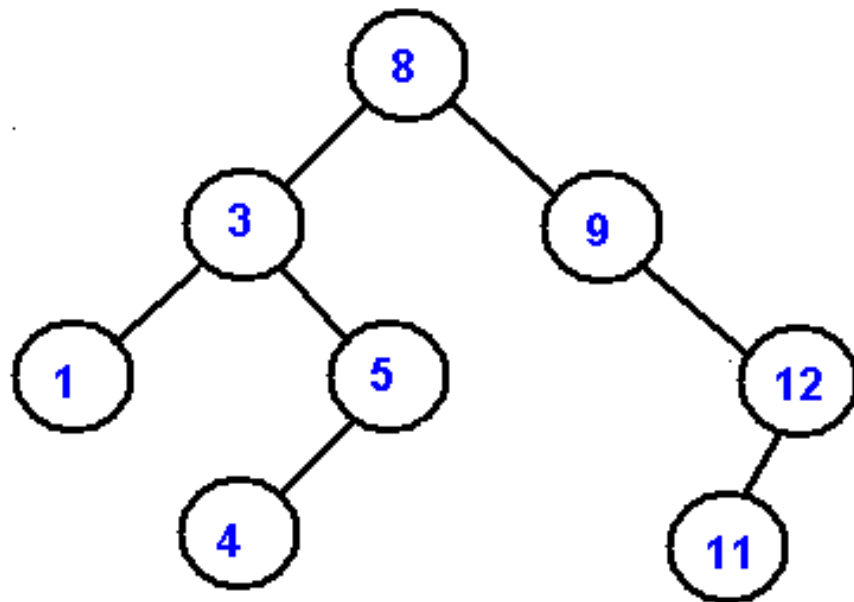
```
struct nod{  
    tara info;  
    nod* st;  
    nod* dr;  
};
```

- ▶ Pentru crearea unui nod se primesc cele trei informatii necesare: informatia, nodul din stanga si nodul din dreapta.

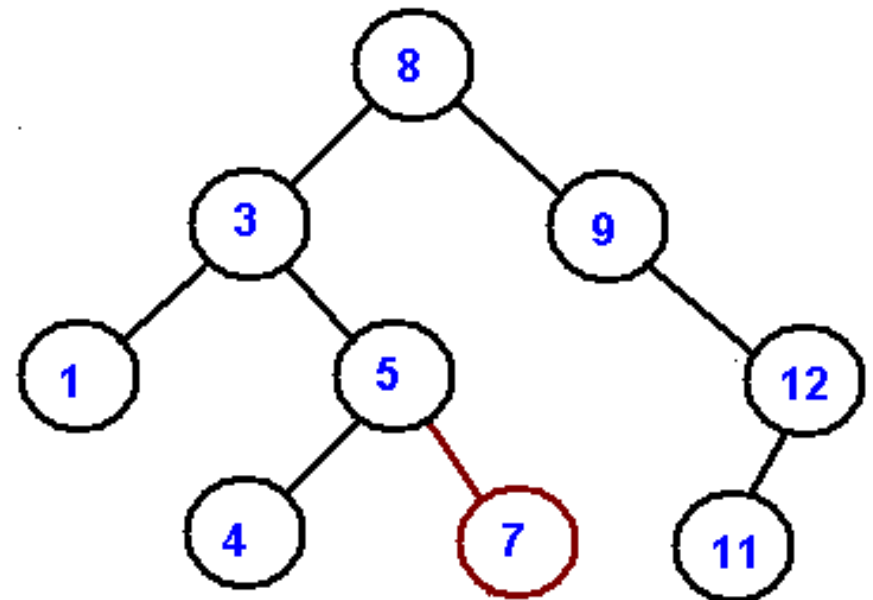
S06 – Creare nod

```
nod* creareNod(tara info, nod* st, nod* dr){  
    nod* temp=(nod*)malloc(sizeof(nod));  
    temp->info.id=info.id;  
    temp->info.num=(char*)malloc((strlen(info.num)+1)*sizeof(char));  
    strcpy(temp->info.num,info.num);  
    temp->info.nr_locuitori=info.nr_locuitori;  
    temp->info.suprafata=info.suprafata;  
  
    temp->dr=dr;  
    temp->st=st;  
  
    return temp;  
}
```

S06 – Inserare



before insertion



after insertion

S06 – Inserare

```
nod* inserare(nod*radacina, tara info){  
    if(radacina==NULL)  
    {  
        radacina=creareNod(info,NULL,NULL);  
        return radacina;  
    }  
    else  
    {  
        if(info.id < radacina->info.id)  
        {  
            radacina->st=inserare(radacina->st,info);  
        }  
        else  
        {  
            radacina->dr=inserare(radacina->dr,info);  
        }  
        return radacina;  
    }  
}
```

S06 – Parcurgere si afisare

- ▶ Preordine → RSD;
- ▶ Inordine → SRD;
- ▶ Postordine → SDR.

S06 – Parcurgere si afisare

```
void SRD(nod*rad)
{
    if(rad)
    {
        SRD(rad->st);
        printf("%d.%s are", rad->val, rad->num);
        SRD(rad->dr);
    }
}
```

S06 – Numarul de nivele

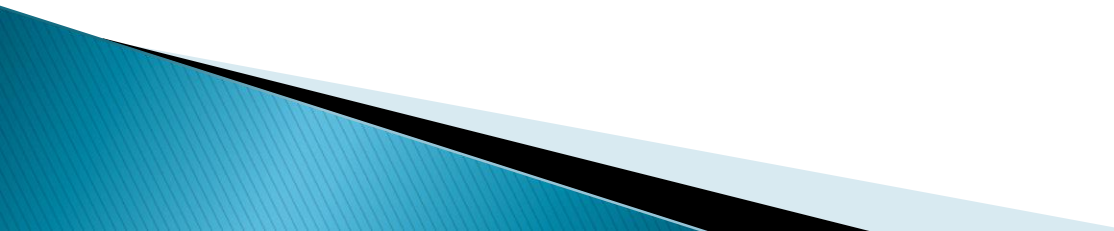
- ▶ Pentru determinarea numarului de nivele se calculeaza numarul de nivele pentru fiecare subarbore si se determina maximul dintre acestea, la care se adauga valoarea 1.

```
int max(int a,int b)
{
    if(a>b) return a;
    else
        return b;
}
```

S06 – Numarul de nivele

```
= int nivele(nod*rad)
{
    if(rad)
        return 1+max(nivele(rad->st),nivele(rad->dr));
    else
        return 0;
}
```

S06 – Afisarea de pe un nivel

- ▶ Se furnizeaza ca si parametrii: radacina arborelui, nivelul de pe care dorim sa afisam informatiile si nivelul aferent radacinii (nivel 1).
 - ▶ La fiecare apel atat pentru subarborele stang cat si pentru subarborele drept nivelul curent creste cu o unitate.
 - ▶ Atunci cand cele doua nivele sunt egale, se afiseaza informatiile din nodul respectiv.
- 

S06 – Afisarea de pe un nivel

```
void afisareNivel(nod*rad,int nivel,int nivelCurent)
{
    if(rad)
    {
        if(nivel==nivelCurent)
        {
            printf("%d.%s are %d locuitori si o suprafata\n",rad->val,rad->nume,nivelCurent);
        }
        else
        {
            afisareNivel(rad->st,nivel,nivelCurent+1);
            afisareNivel(rad->dr,nivel,nivelCurent+1);
        }
    }
}
```

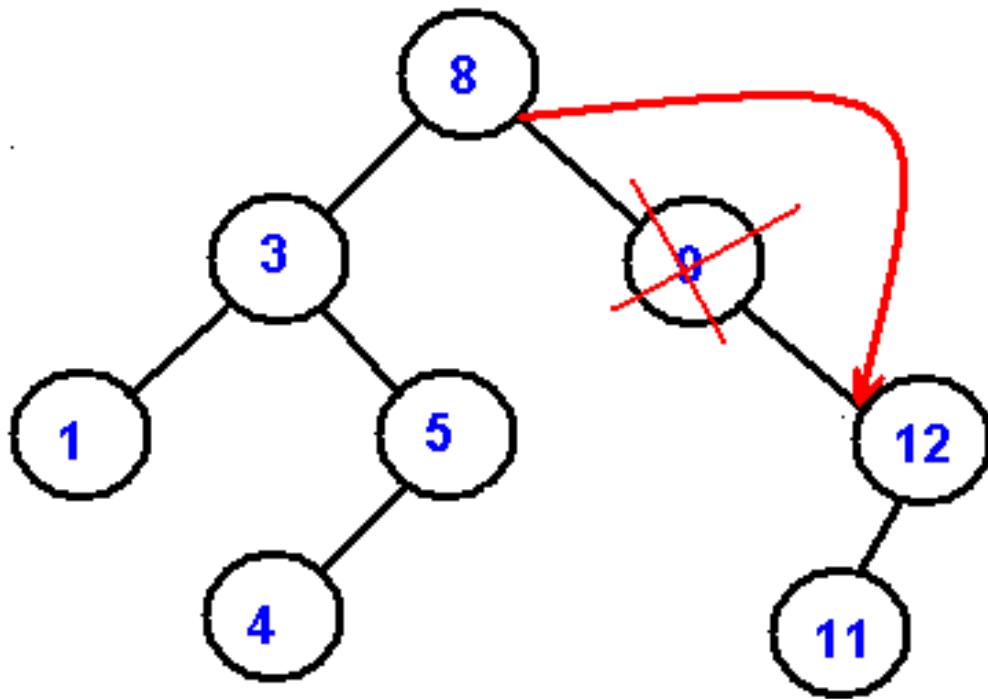
S06 – Cautare nod

- ▶ Pentru cautare se furnizeaza radacina arborelui si codul tarii de cautat sau tara ce este cautata.
- ▶ Cautarea se face dupa id-ul tarii deoarece arborele este sortat dupa acest id.

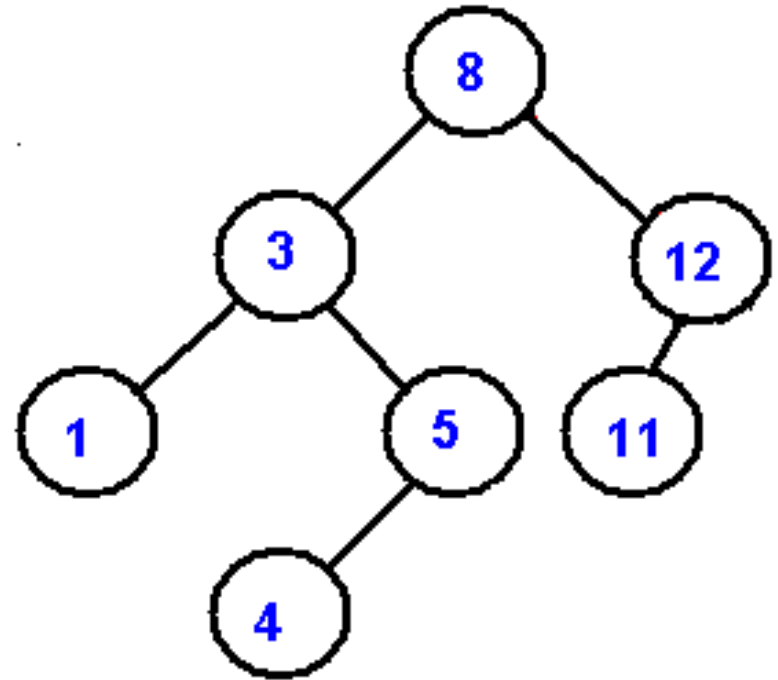
S06 – Cautare nod

```
nod* cautare(nod*rad, int cod)
{
    if(rad)
    {
        if(rad->info.id==cod)
        {
            return rad;
        }
        else
        {
            if(rad->info.id>cod)
                return cautare(rad->st,cod);
            else
                return cautare(rad->dr,cod);
        }
    }
}
```

S06 – Stergere nod (Tema)



before deletion



after deletion

S06 – Stergere nod (Tema)

