S03 – DS

Alin Zamfiroiu
alin.zamfiroiu@csie.ase.ro

S03 – Content

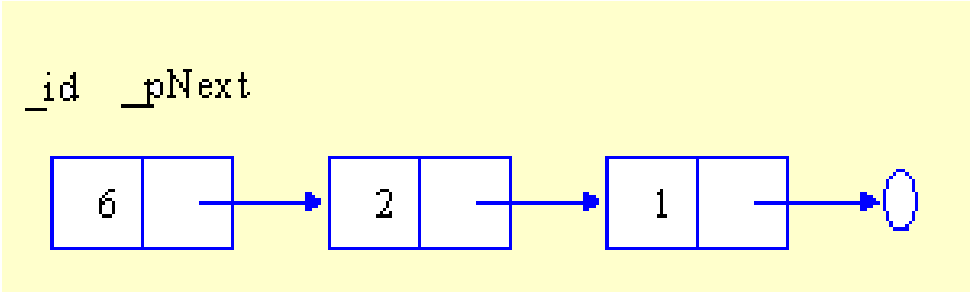
- Book
- Linear list
- Add element to the begin of the list
- Crossing the list
- Add element to the end of the list
- Add element in the middle of the list
- Delete an element
- Delete the list

S03 – Linear list

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

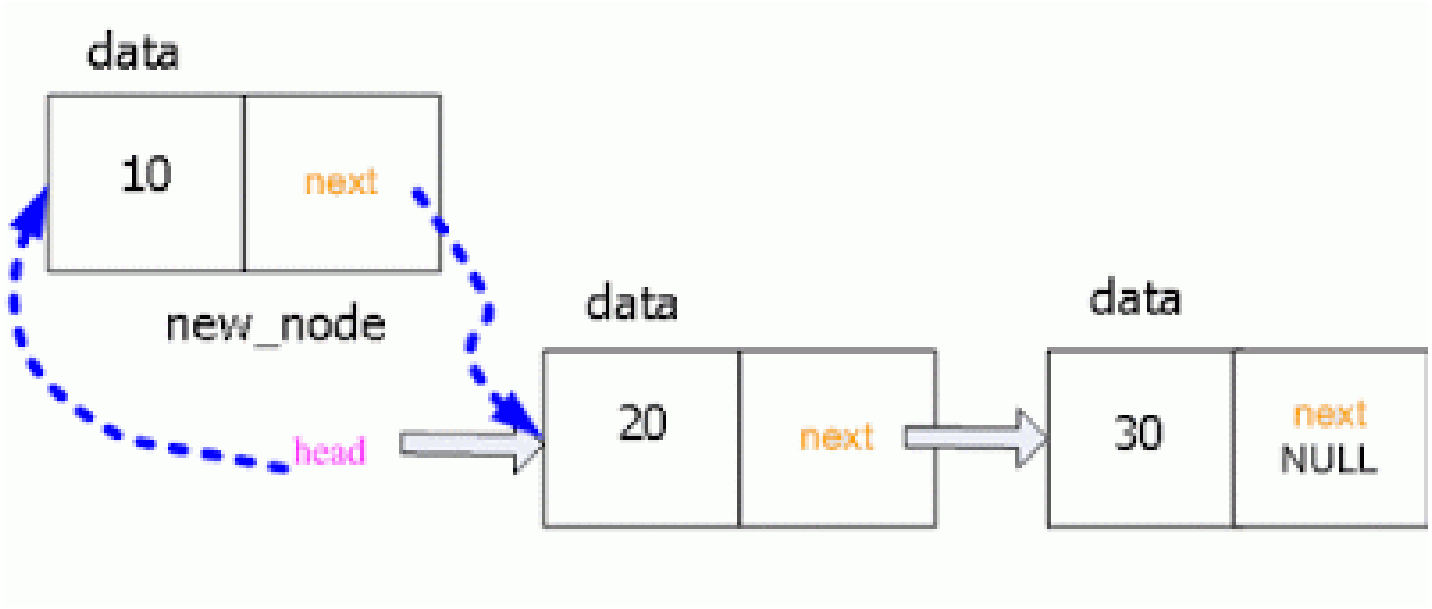
http://www.artline.ro/

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



www.relisoft.com

S03 – Add element to the begin of the list



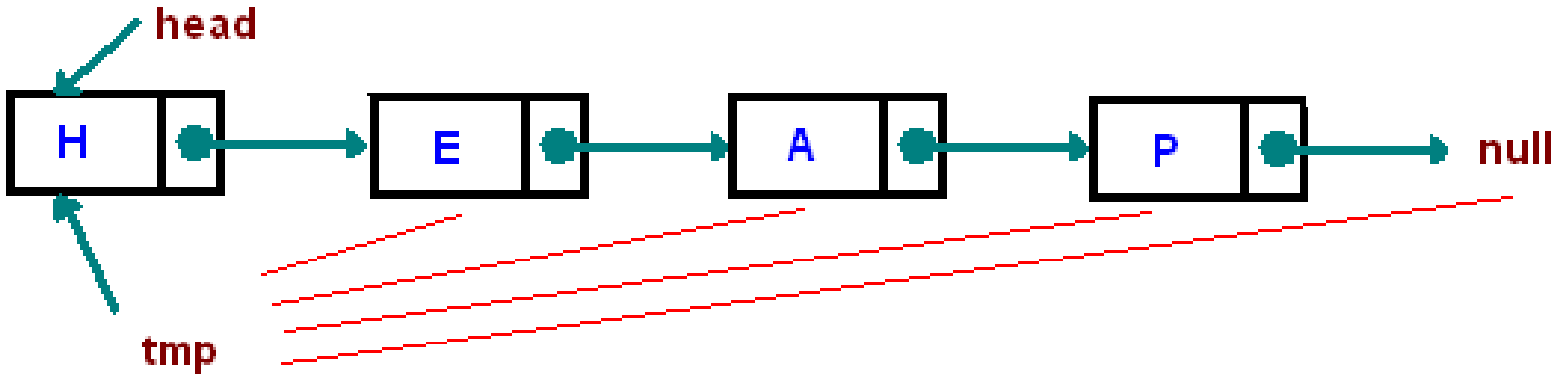
http://www.c4learn.com/

S03 – Add element to the begin of the list

```
nod*inserare_inceput(nod*cap, carte info)
{
    nod* aux = (nod*)malloc(sizeof(nod));
    aux->info.titlu = (char*)malloc(sizeof(char)*(strlen(info.titlu) + 1));
    strcpy(aux->info.titlu, info.titlu);
    aux->info.pret = info.pret;
    aux->next = cap;
    return aux;
}
```

Call this function to add more than 3 books in a linear list, initialized with NULL.

S03 – Crossing the list

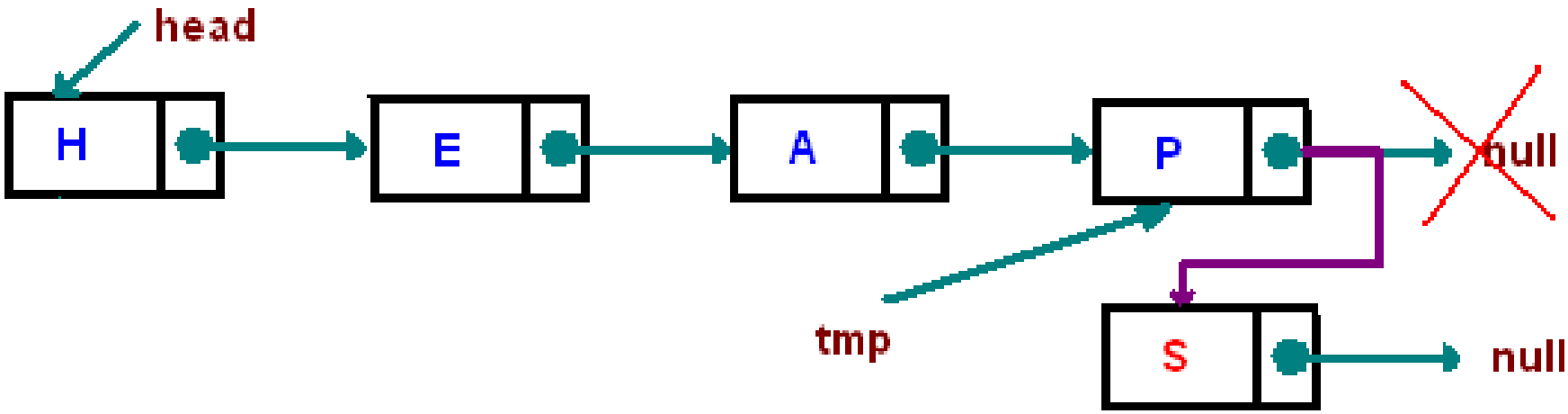


http://www.cs.cmu.edu/

S03 – Crossing the list

```
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 – Add element to the end of the list



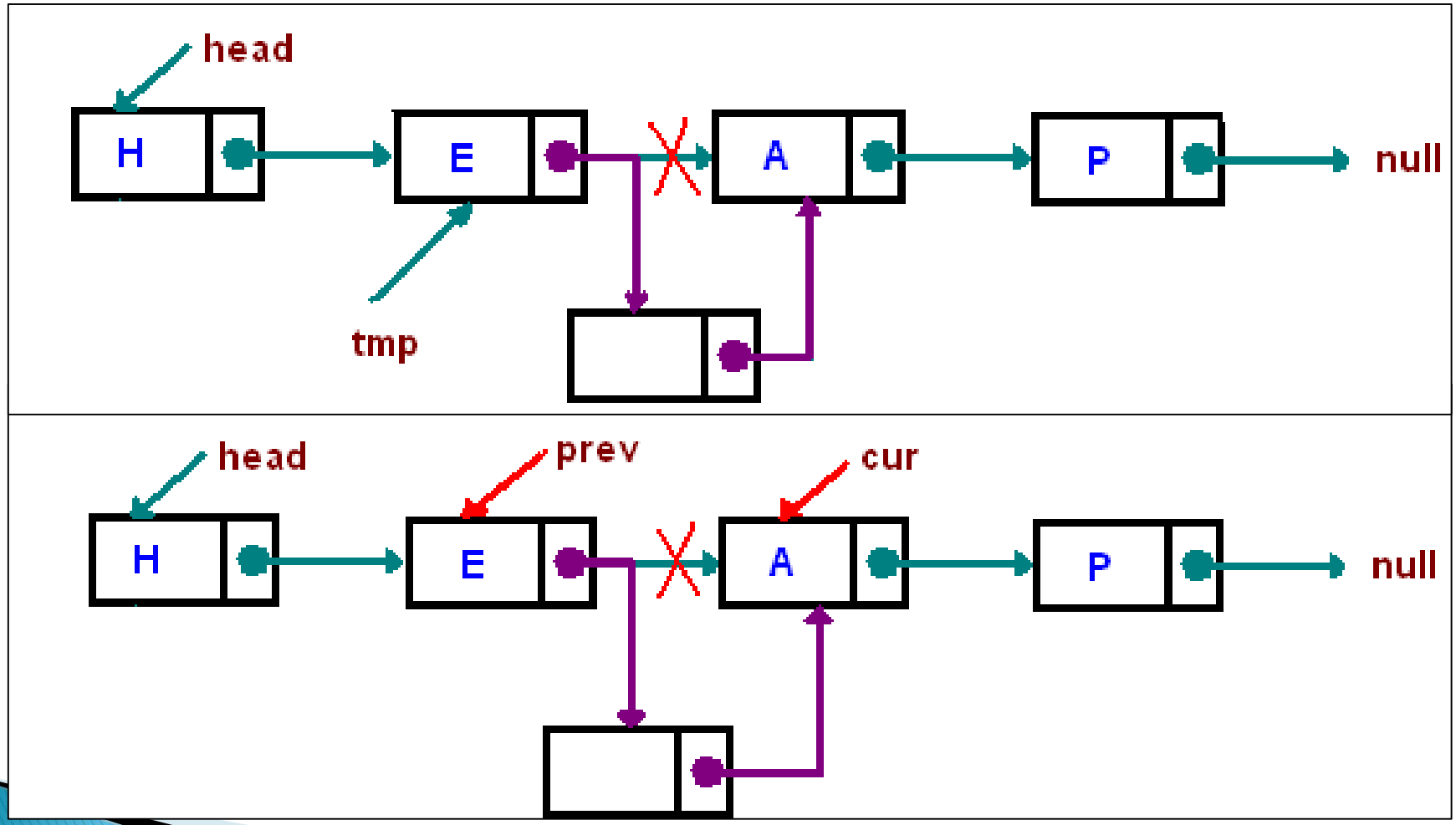
http://www.cs.cmu.edu/

S03 – Add element to the end of the list

Call this function to add more than 3 books in a linear list initialized with NULL, and after call the function of adding elements to the begin of the list.

```
nod* inserare_sfarsit(nod*cap, carte info)
{
    nod* nou = (nod*)malloc(sizeof(nod));
    nou->info.titlu = (char*)malloc(sizeof(char)*(strlen(info.titlu) + 1));
    strcpy(nou->info.titlu, info.titlu);
    nou->info.pret = info.pret;
    if (cap)
    {
        nod*p = cap;
        while (p->next)
        {
            p = p->next;
        }
        p->next = nou;
    }
    else
    {
        cap = nou;
    }
    return cap;
}
```

S03 – Add element in the middle of the list

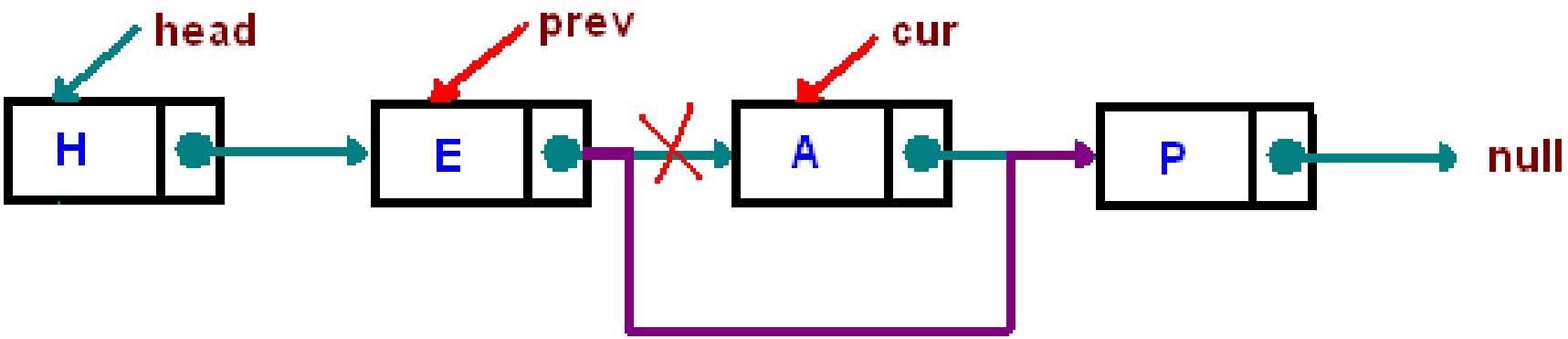


http://www.cs.cmu.edu/

S03 – Add element in the middle of the list

```
nod* inserare_crescatoare(nod*cap, carte info)
{
    nod* nou = (nod*)malloc(sizeof(nod));
    nou->info.titlu = (char*)malloc(sizeof(char)*(strlen(info.titlu) + 1));
    strcpy(nou->info.titlu, info.titlu);
    nou->info.pret = info.pret;
    nou->next = NULL;
    if (cap)
    {
        if (cap->info.pret > info.pret)
        {
            cap = inserare_inceput(cap, info);
            return cap;
        }
        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 – Delete an element



S03 – Delete an 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;
            }
            if (p->next)
            {
                nod* aux = p->next;
                p->next = p->next->next;
                free(aux->info.titlu);
                free(aux);
            }
        }
    }
    return cap;
}
```

S03 – Delete the list

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