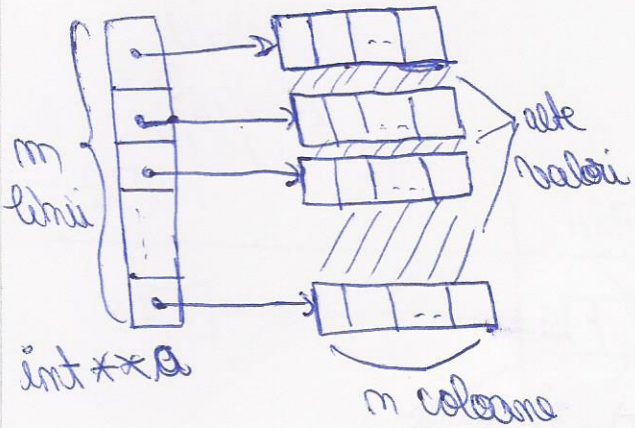
PROGRAMARE PROCEDURALĂ (CURS 6 | 14 noiembrie 2015)

Alocarea dinamică a tablourilor bidimensionale

a) alocare necompactă




```
int **a;
a=(int **)malloc(m*sizeof(int*));
for(i=0;i<m;i++)
    a[i]=(int*)malloc(n*sizeof(int));
```

- alocare

```
for(i=0;i<m;i++)
    free(a[i]);
free(a);
```

- dealocare

*dacă nu facem free()
⇒ zone ocupate de memorie

b) alocare compactă; m*n elemente


```
int **a;
a=(int **)malloc(m*sizeof(int*));
a[0]=(int *)malloc(m*n*sizeof(int));
for(i=1;i<m;i++)
    a[i]=a[0]+i*n;
```

- alocare

~~for(i=1;i<m;i++)~~
~~a[i]=a[0]+i*m;~~

dezalocare { free(a[0]);
free(a);

c) alocarea unei matrici triunghiulare

1
1 2
1 2 3
~~1 2 3 ...~~

⇒ linia i are i+1 elemente

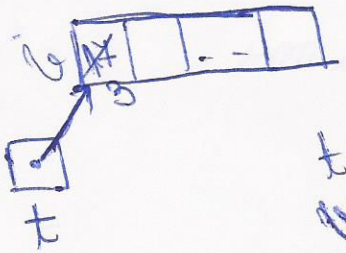
1 2 3 ... m

```
int **a;
a=(int **) malloc(m*sizeof(int *));
for(i=0;i<m;i++)
    a[i]=(int *) malloc((i+1)*sizeof(int));
```

Grecli:

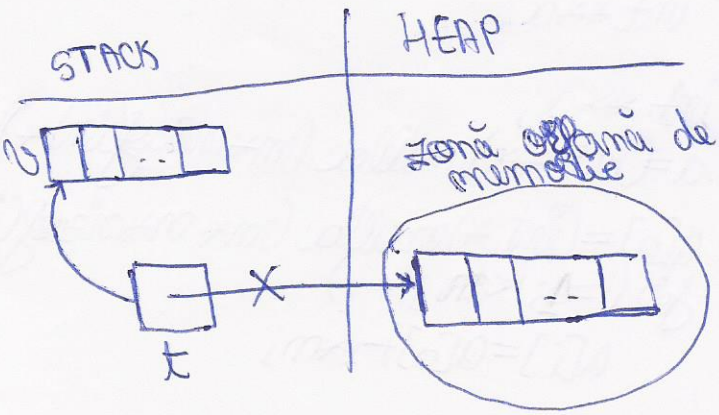
a) `int v[100],*t;` → `t=(int *) malloc(n*sizeof(int));`
```
for(i=0;i<n;i++)
    t[i]=v[i];
```

b) `int v[100],*t;`



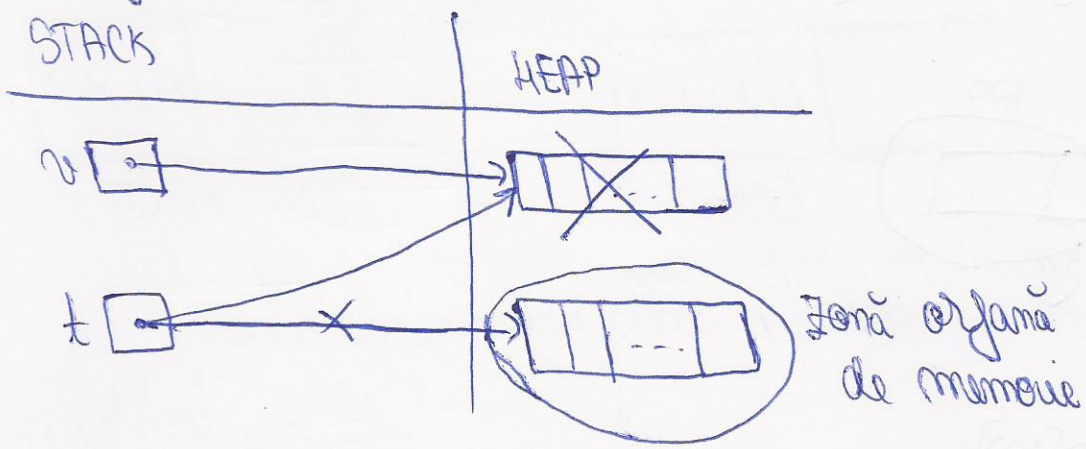

```
t=0;
v[0]=17;
t[0]=3;
```

c) `int v[100],*t;`
`t=(int*)malloc(n*sizeof(int));`
`t=v;`



~~int~~) int *v,*t;

```
v=(int*)malloc(...);
t=(int*)malloc(...);
t=v;
free(v);
```

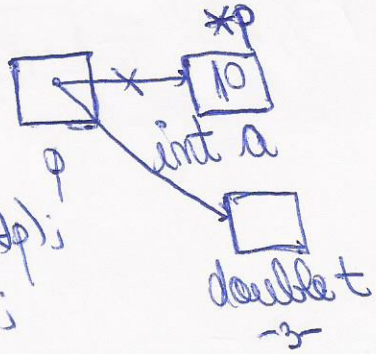


Funcții pentru manipularea memoriei (string.h)

Pointer generic = un pointer de tip void *

Pointer generic
- poate memora adrese de orice tip (fără conversie explicită)
- nu au aritmetică (necesită conversie explicită)

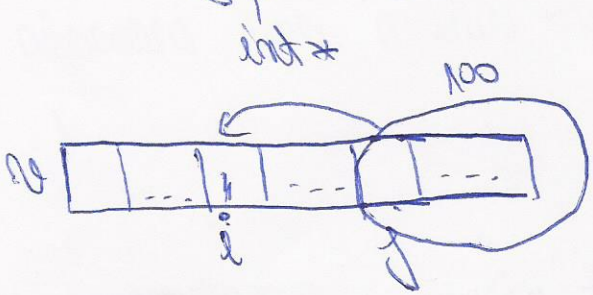
Ex:
```
int a=10;
double t=3.14;
void *p;
p=&a;
printf("%d", *(int*)p);
*(int*)p=7;
p=&t;
printf("%g", *(double*)p);
*(double*)p=100.5;
```



a) void *memcpy(void *dest, void *sursă, int nr_octeți)

- nu se utilizează dacă sursa și destinația sunt zone de memorie suprapuse

memcpy(v+i, v+j, 100*sizeof(int));

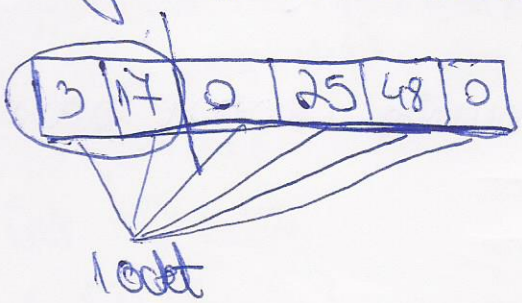


- corect

int a[100], b[100];
memcpy(a, b, n*sizeof(int));

b) void *memmove(void *dest, void *sursă, int nr_octeți)

- la fel ca memcpy, dar folosește un buffer intern pentru copiere
- se folosește când destinația și sursa se suprapun

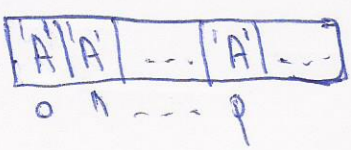


considerată ca octet fără semn (unsigned char)

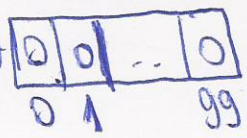
c) void *memset(void *adresă, int val, int nr_octeți)

- setează nr octeți începând de la adresa indicată la val indicată ca parametru

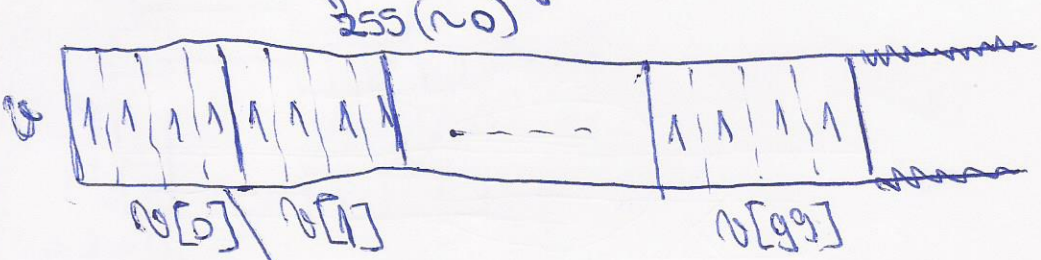
char v[100];
memset(v, 65, 10);



int v[100];
memset(v,0,100*sizeof(int));



int v[100];
!!! memset(v,1,100*sizeof(int));



$00000001\,00000001\,00000001\,00000001 = 2^0+2^8+2^{16}+2^{24}$

$= 1\cdot\frac{[(2^8)^4-1]}{2^8-1} = \frac{2^{32}-1}{2^8-1} = \dots$

d) void *memset(void *adresă, int val, int nrocteți);
↓
unsigned char

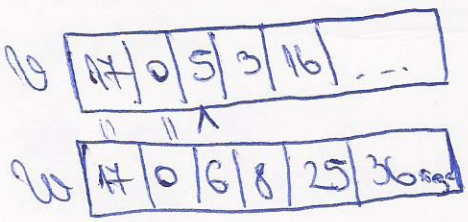
~~scrie primii octeți începând cu adresa dată~~

e) ~~void~~ int *memcmp(void *adresă_1, void *adresă_2, int nrocteți);

<0, dacă adresă$_1$ $\leq_L$ adresă$_2$ (lexicografic)

=0, dacă adresă$_1$ $=_L$ adresă$_2$

>0, dacă adresă$_1$ $\geq_L$ adresă$_2$



memcmp(v, v, 2) ⇒ 0
memcmp(v, v, 5) ⇒ <0