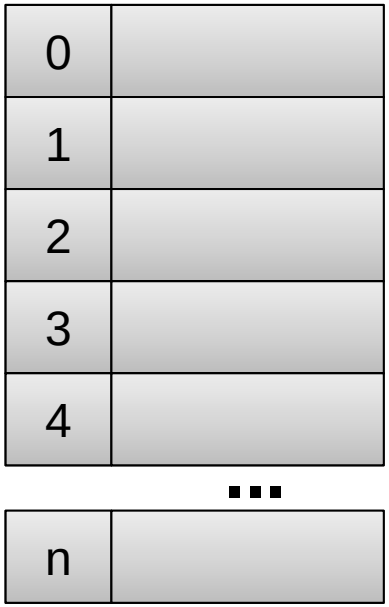
# Pràctiques de Programació

## La memòria

# La memòria

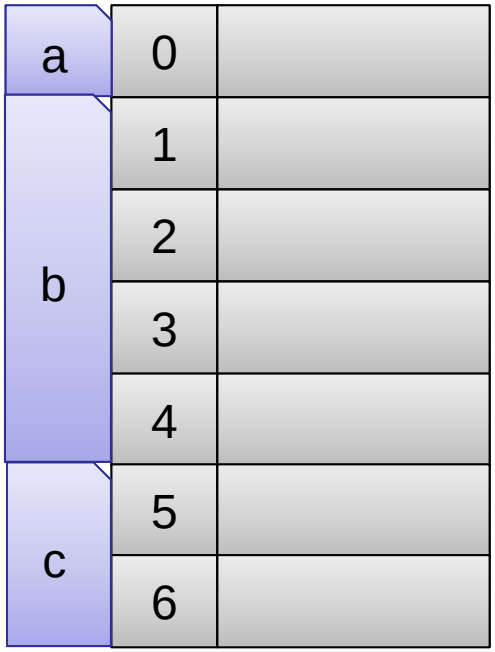
- La memòria és el recurs de la màquina que ens permet emmagatzemar informació de forma **temporal.** És un recurs escàs, i cal gestionar-lo correctament.
- Generalment, la memòria es representa en forma de vector.
  - Les posicions s'assumeixen continues.
  - Les posicions s'enumeren, obtenint les adreces, com les cases dins d'un carrer. Les adreces es solen mostrar en format Hexadecimal. L'adreça d'una variable s'accedeix amb l'operador **&**.



# La memòria

- Cada posició pot emmagatzemar un byte. Per tant, una variable pot ocupar més d'una posició, segons el seu tipus, la màquina i el sistema operatiu que estiguem utilitzant.
- A continuació es mostra un exemple de la evolució de la memòria en un codi simple.

```
char a;
int b;
char c[2]:

a='i'
b=4;
c[0]='h';
c[1]=a;
```

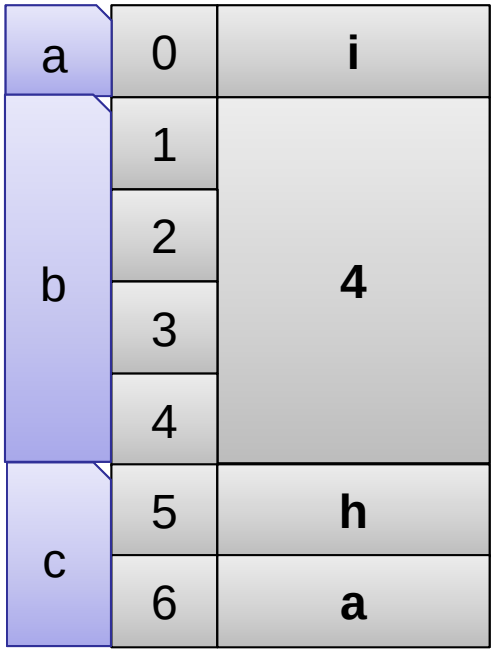



S'assignen posicions de memòria lliures a cada variable. Assumim char (1 byte) i int=float (4 bytes)

# La memòria

- Cada posició pot emmagatzemar un byte. Per tant, una variable pot ocupar més d'una posició, segons el seu tipus, la màquina i el sistema operatiu que estiguem utilitzant.
- A continuació es mostra un exemple de la evolució de la memòria en un codi simple.

```
char a;
int b;
char c[2]:

a='i'
b=4;
c[0]='h';
c[1]=a;
```

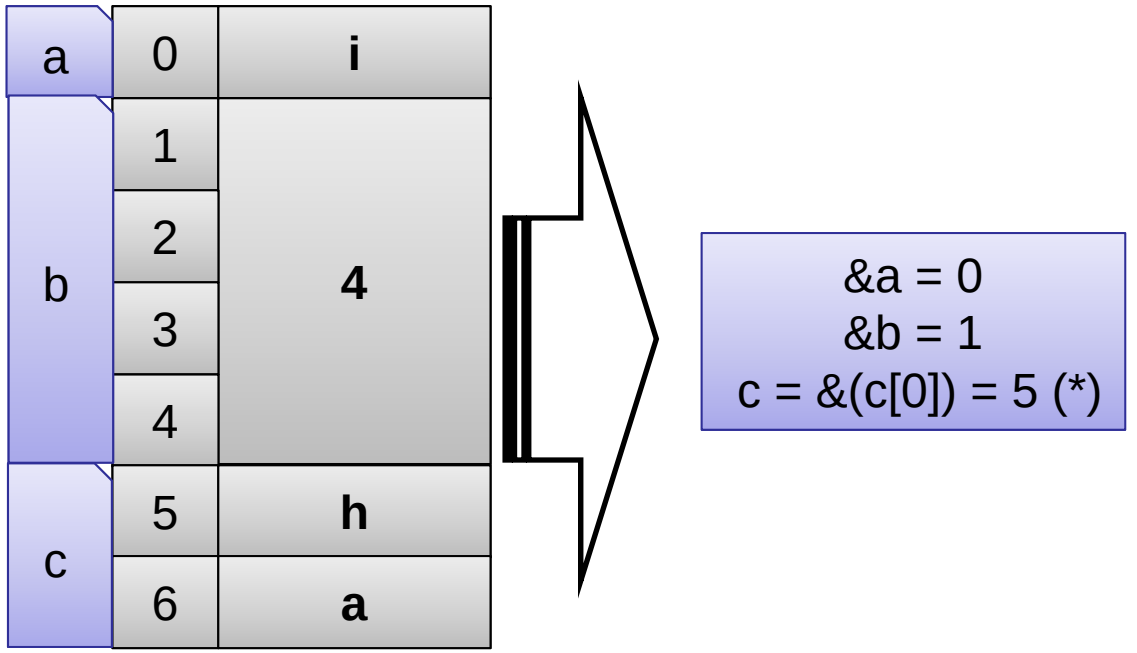



Es canvia el valor del contingut de les posicions de memòria associats a cada variable.

# La memòria

- Quan declarem una variable, el nom d'aquesta variable s'assigna a l'adreça de memòria de la seva primer posició. Per exemple, en l'exemple anterior, les adreces corresponents a les variables declarades, obtingudes amb l'operador &, serien:

```
char a;
int b;
char c[2]:

a='i'
b=4;
c[0]='h';
c[1]=a;
```

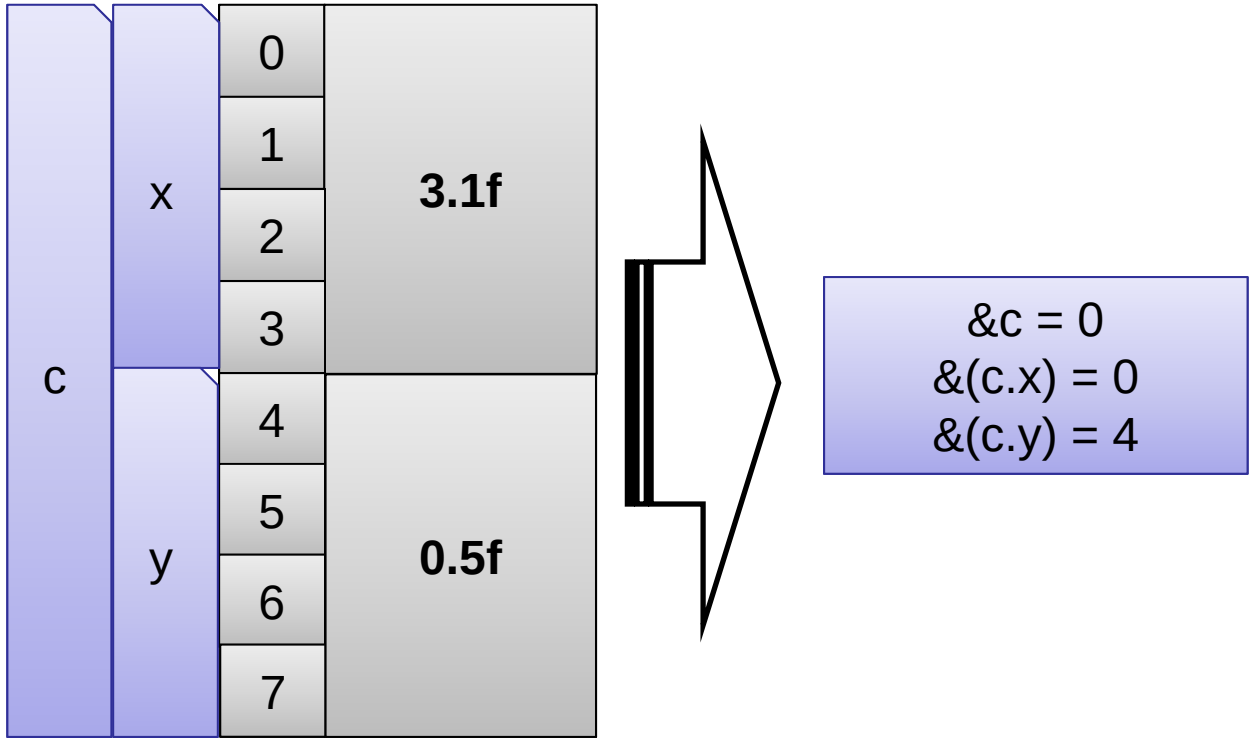



(*) El cas de la variable **c** és lleugerament diferent, ho veurem més endavant, quan parlem de punters.

# La memòria

- Quan treballem amb tuples, la memòria funciona exactament de la mateixa manera, essent la mida de la tupla la suma de mides dels seus camps(*). Per exemple, la tupla *coordenades*, quedaria d'aquesta manera:

```
struct {
    float x,y;
} coordenades;

coordenades c;
c.x=3.1f;
c.y=0.5f;
```




(*) Alguns sistemes ajusten les mides a múltiples de n bytes, i per tant la mida de la tupla pot ser lleugerament major. Això s'anomena "*Alineació de memòria*".

# La memòria

- Per conèixer la mida exacta en bytes d'un tipus de dades, s'utilitza ***sizeof***. Aquesta funció ens retorna el valor exacte, tenint en compte el sistema en que ens trobem i el compilador utilitzat.
- A continuació es mostren uns quants exemples que podeu provar. Intenteu entendre els resultats i d'on surt cada mida. Podeu intentar fer el dibuix de la memòria per a cada cas.

```
char c[26];
struct {
    int nCaracters;
    char caracters[26];
} linia1;
struct {
    int nCaracters;
    char caracters[30];
} linia2;
struct {
    char caracters[13];
    int nCaracters;
    char caracters2[13];
} linia3;
printf("a) La mida d'un tipus <%s> es %d bytes\n","char",sizeof(char));
printf("b) La mida d'un tipus <%s> es %d bytes\n","int",sizeof(int));
printf("c) La mida d'un tipus <%s> es %d bytes\n","float",sizeof(float));
printf("d) La mida d'un tipus <%s> es %d bytes\n","char[26]",sizeof(c));
printf("e) La mida d'un tipus <%s> es %d bytes\n","linia1 => [ int + char[26] ]",sizeof(linia1));
printf("f) La mida d'un tipus <%s> es %d bytes\n","linia2 => [ int + char[30] ]",sizeof(linia2));
printf("g) La mida d'un tipus <%s> es %d bytes\n","linia3 => [ char[13] + int + char[13] ]",sizeof(linia3));
```

# La memòria

***Autoavaluació:***

1. Dibuixa la memòria per al següent codi:

```
int b;
char a;

a='A'
b=(int)a;
```

2. Dibuixa la memòria per al següent codi:

```
float a;

a=1.23f;
```

3. Dibuixa la memòria per al següent codi:

```
float a[2];

a[0]=1.23f;
a[1]=2.0f;
```

# La memòria

***Autoavaluació:***

4. Dibuixa la memòria per al següent codi:

```
coordenada c[2];
c[0].x=10.0f;
c[0].y=20.0f;
c[1].x=30.0f;
c[1].y=40.0f;
```

5. Dibuixa la memòria per al següent codi:

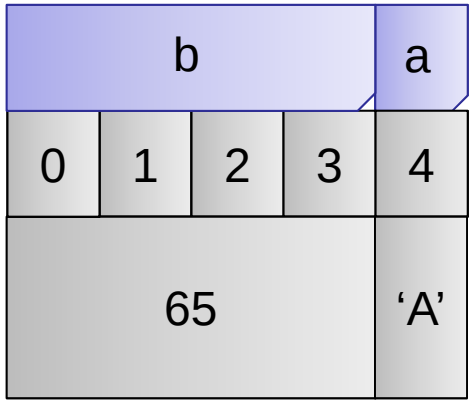
```
struct {
    coordenada p1,p2;
} vector;
vector  v1;
v1.p1.x=10.0f;
v1.p1.y=20.0f;
v1.p2.x=v1.p1.y+10.0f:
v1.p2.y=v1.p1.x+30.0f:
```

# La memòria

***Autoavaluació (Resposta):***

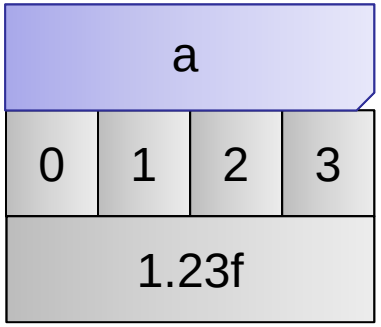
1. Dibuixa la memòria per al següent codi:

```
int b;
char a;

a='A'
b=(int)a;
```



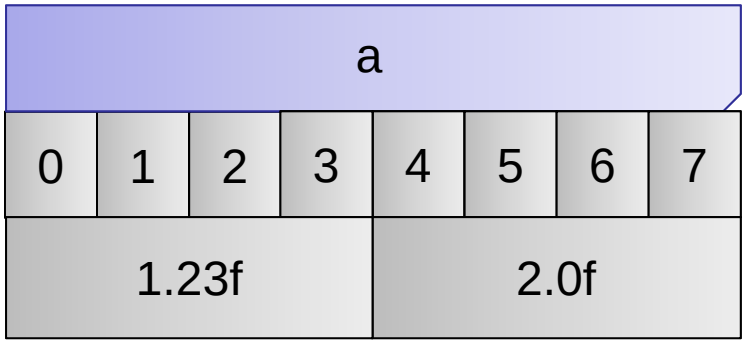

2. Dibuixa la memòria per al següent codi:

```
float a;

a=1.23f;
```




3. Dibuixa la memòria per al següent codi:

```
float a[2];

a[0]=1.23f;
a[1]=2.0f;
```

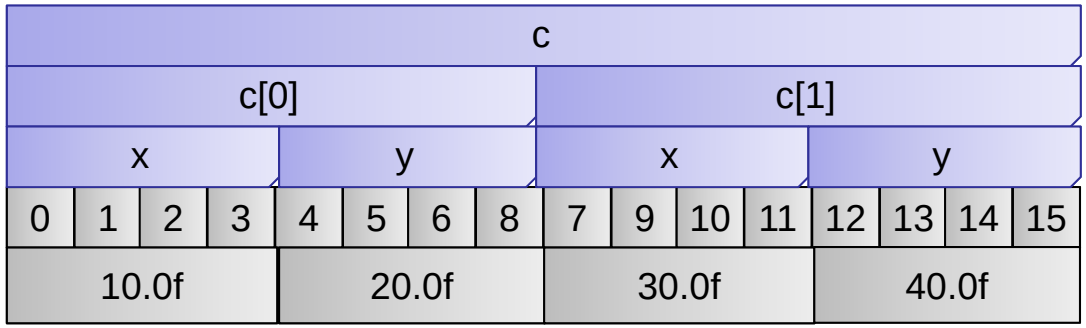


# La memòria

***Autoavaluació (Resposta):***

4. Dibuixa la memòria per al següent codi:

```
coordenada c[2];
c[0].x=10.0f;
c[0].y=20.0f;
c[1].x=30.0f;
c[1].y=40.0f;
```




5. Dibuixa la memòria per al següent codi:

```
struct {
        coordenada p1,p2;
} vector;
vector  v1;
v1.p1.x=10.0f;
v1.p1.y=20.0f;
v1.p2.x=v1.p1.y+10.0f:
v1.p2.y=v1.p1.x+30.0f:
```