



# LABORATORIS

## PROGRAMACIÓ CIENTÍFICA

---

Sessió 4: Taules i cadenes



# TAULES

# Taules en C

## A teoria:

**const**

MAX := 100;

N := 10;

**fconst**

**var**

v: taula[MAX] d'enters;

m: taula[MAX][N] de reals;

text: taula[MAX] de caràcter;

**fvar**

# Taules en C

## A teoria:

**const**

MAX := 100;

N := 10;

**fconst**

**var**


v: taula[MAX] d'enters;

m: taula[MAX][N] de reals;

text: taula[MAX] de caràcter;

**fvar**

## En C:



```
#define MAX 100
#define N 10

...
int v[MAX];
float m[MAX][N];
char text[MAX];

...
```

# Taules en C

## A teoria:

### const

```
MAX := 100;
```

```
N := 10;
```

### fconst

### var

```
v: taula[MAX] d'enters;
```

```
m: taula[MAX][N] de reals;
```

```
text: taula[MAX] de caràcter;
```

### fvar

## En C:

```
#define MAX 100
```

```
#define N 10
```

```
...
```

```
int v[MAX];
```

```
float m[MAX][N];
```

```
char text[MAX];
```

```
...
```

Dimensions

Tipus de la taula

Nom de la variable

# Taules en C

Què passa quan declarem una taula?

# Taules en C

Què passa quan declarem una taula?

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     printf("La mida d'un int es %d\n", sizeof(int));
8     printf("Adreça de variable v[0]: %x, dec=%u\n", &v[0], &v[0]);
9     printf("Adreça de variable v[1]: %x, dec=%u\n", &v[1], &v[1]);
10    printf("Adreça de variable v[2]: %x, dec=%u\n", &v[2], &v[2]);
11    printf("Adreça de variable v[3]: %x, dec=%u\n", &v[3], &v[3]);
12    printf("Adreça de variable v[4]: %x, dec=%u\n", &v[4], &v[4]);
13    return 0;
14 }
```

# Taules en C

Què passa quan declarem una taula?

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     printf("La mida d'un int es %d\n", sizeof(int));
8     printf("Adreça de variable v[0]: %x, dec=%u\n", &v[0], &v[0]);
9     printf("Adreça de variable v[1]: %x, dec=%u\n", &v[1], &v[1]);
10    printf("Adreça de variable v[2]: %x, dec=%u\n", &v[2], &v[2]);
11    printf("Adreça de variable v[3]: %x, dec=%u\n", &v[3], &v[3]);
12    printf("Adreça de variable v[4]: %x, dec=%u\n", &v[4], &v[4]);
13    return 0;
14 }
```

```
La mida d'un int es 4
Adreça de variable v[0]: ecc7aa50, dec=3972508240
Adreça de variable v[1]: ecc7aa54, dec=3972508244
Adreça de variable v[2]: ecc7aa58, dec=3972508248
Adreça de variable v[3]: ecc7aa5c, dec=3972508252
Adreça de variable v[4]: ecc7aa60, dec=3972508256
```



# Taules en C

Què passa quan declarem una taula?



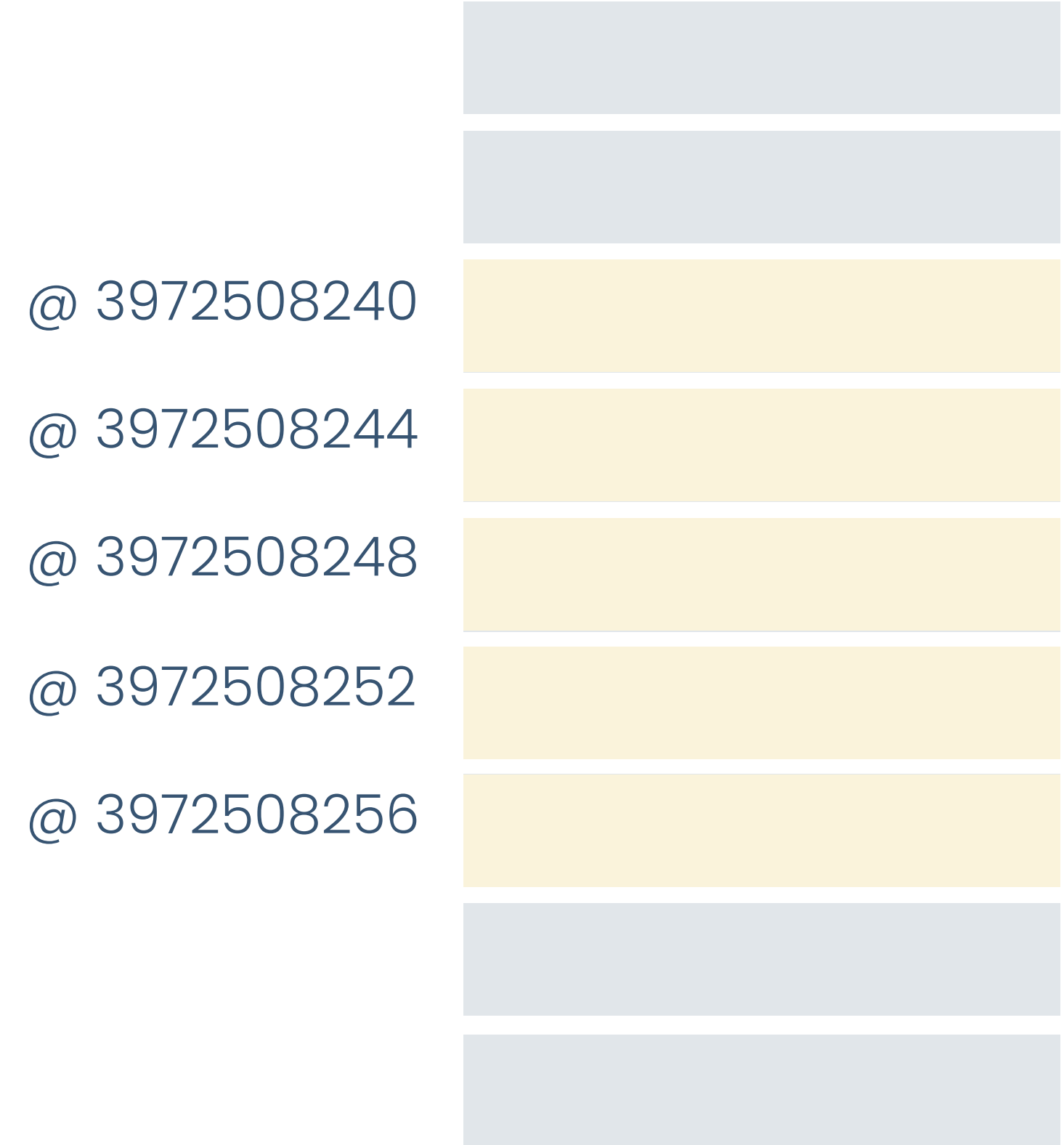
*Memòria de 32 bits*

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     printf("La mida d'un int es %d\n", sizeof(int));
8     printf("Adreça de variable v[0]: %x, dec=%u\n", &v[0], &v[0]);
9     printf("Adreça de variable v[1]: %x, dec=%u\n", &v[1], &v[1]);
10    printf("Adreça de variable v[2]: %x, dec=%u\n", &v[2], &v[2]);
11    printf("Adreça de variable v[3]: %x, dec=%u\n", &v[3], &v[3]);
12    printf("Adreça de variable v[4]: %x, dec=%u\n", &v[4], &v[4]);
13    return 0;
14 }
```

La mida d'un int es 4  
Adreça de variable v[0]: ecc7aa50, dec=3972508240  
Adreça de variable v[1]: ecc7aa54, dec=3972508244  
Adreça de variable v[2]: ecc7aa58, dec=3972508248  
Adreça de variable v[3]: ecc7aa5c, dec=3972508252  
Adreça de variable v[4]: ecc7aa60, dec=3972508256

# Taules en C

Què passa quan declarem una taula?



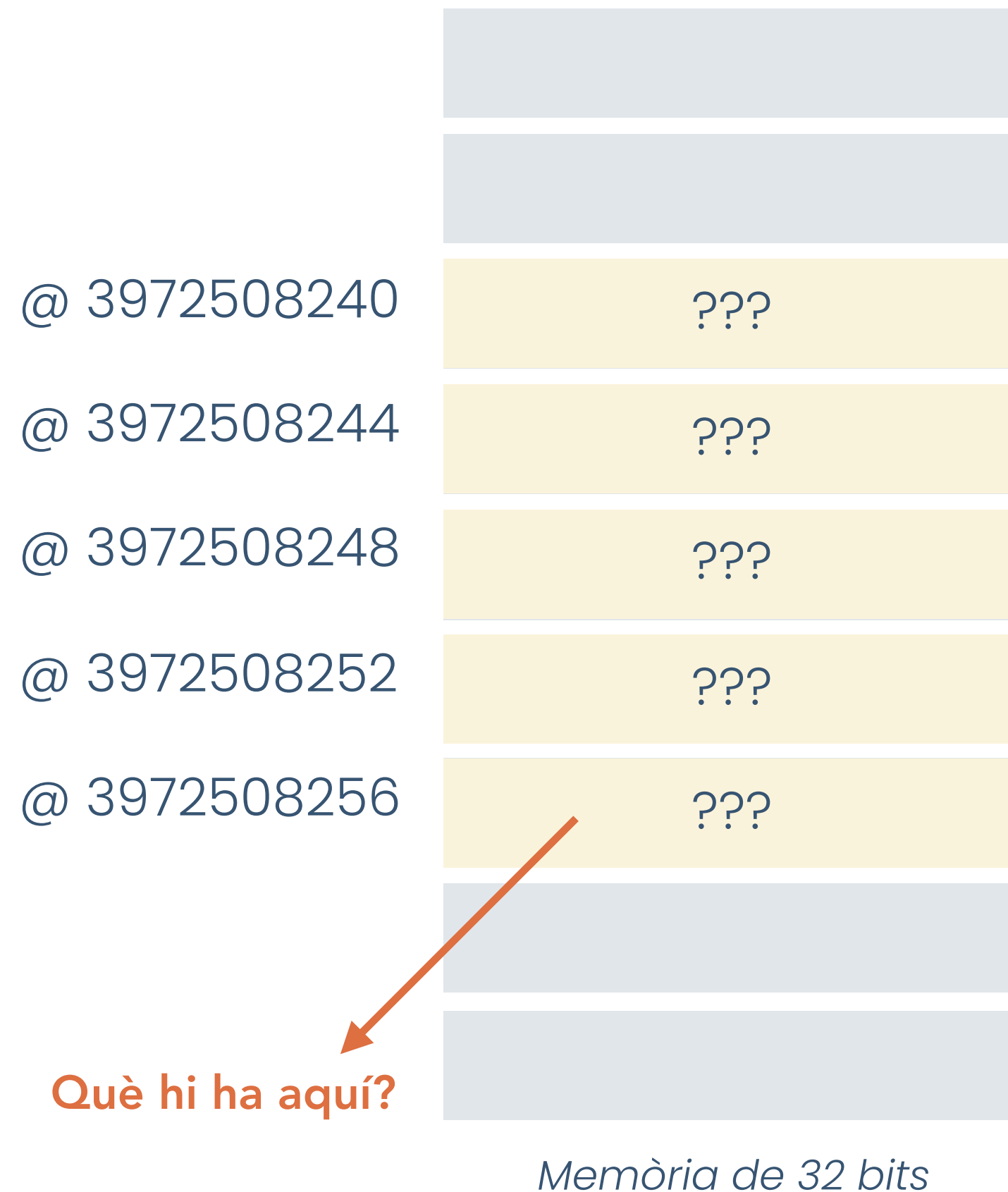
Memòria de 32 bits

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     printf("La mida d'un int es %d\n", sizeof(int));
8     printf("Adreça de variable v[0]: %x, dec=%u\n", &v[0], &v[0]);
9     printf("Adreça de variable v[1]: %x, dec=%u\n", &v[1], &v[1]);
10    printf("Adreça de variable v[2]: %x, dec=%u\n", &v[2], &v[2]);
11    printf("Adreça de variable v[3]: %x, dec=%u\n", &v[3], &v[3]);
12    printf("Adreça de variable v[4]: %x, dec=%u\n", &v[4], &v[4]);
13    return 0;
14 }
```

La mida d'un int es 4  
Adreça de variable v[0]: ecc7aa50, dec=3972508240  
Adreça de variable v[1]: ecc7aa54, dec=3972508244  
Adreça de variable v[2]: ecc7aa58, dec=3972508248  
Adreça de variable v[3]: ecc7aa5c, dec=3972508252  
Adreça de variable v[4]: ecc7aa60, dec=3972508256

# Taules en C

Què passa quan declarem una taula?



```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     printf("La mida d'un int es %d\n", sizeof(int));
8     printf("Adreça de variable v[0]: %x, dec=%u\n", &v[0], &v[0]);
9     printf("Adreça de variable v[1]: %x, dec=%u\n", &v[1], &v[1]);
10    printf("Adreça de variable v[2]: %x, dec=%u\n", &v[2], &v[2]);
11    printf("Adreça de variable v[3]: %x, dec=%u\n", &v[3], &v[3]);
12    printf("Adreça de variable v[4]: %x, dec=%u\n", &v[4], &v[4]);
13    return 0;
14 }
```

La mida d'un int es 4  
Adreça de variable v[0]: ecc7aa50, dec=3972508240  
Adreça de variable v[1]: ecc7aa54, dec=3972508244  
Adreça de variable v[2]: ecc7aa58, dec=3972508248  
Adreça de variable v[3]: ecc7aa5c, dec=3972508252  
Adreça de variable v[4]: ecc7aa60, dec=3972508256



# Taules en C

## Inicialització de taules

- Abans d'usar una taula s'ha d'inicialitzar, perquè declarar una taula només ens assegura que reservem una zona de memòria, no què hi ha dins.

@ 3972508240

@ 3972508244

@ 3972508248

@ 3972508252

@ 3972508256

Memòria de 32 bits

# Taules en C

## Inicialització de taules

- Abans d'usar una taula s'ha d'inicialitzar, perquè declarar una taula només ens assegura que reservem una zona de memòria, no què hi ha dins.

@ 3972508240	0
@ 3972508244	1107296256
@ 3972508248	0
@ 3972508252	1
@ 3972508256	-378987904
Memòria de 32 bits	

```
v[0]=0
v[1]=1107296256
v[2]=0
v[3]=1
v[4]=-378987904
```

# Taules en C

## Inicialització de taules

- Abans d'usar una taula s'ha d'inicialitzar, perquè declarar una taula només ens assegura que reservem una zona de memòria, no què hi ha dins.
- Imagineu-vos que hagués volgut fer un bucle que sumés valors, o alguna cosa, hauria fet servir aquests valors d'inici!
- Per inicialitzar una taula s'ha de fer un recorregut des de la primera fins la última posició i donar-li un valor segur. Per exemple, 0. (Depenent de què vulguem fer, serà un altre valor)

@ 3972508240	0
@ 3972508244	1107296256
@ 3972508248	0
@ 3972508252	1
@ 3972508256	-378987904

Memòria de 32 bits

```
v[0]=0
v[1]=1107296256
v[2]=0
v[3]=1
v[4]=-378987904
```



# Taules en C

## Inicialització de taules

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     // Inicialització
8     for(int i=0; i<MAX; i++){
9         v[i] = 0;
10    }
11    // A partir d'aquí ja puc fer coses
12    // Coses ...
13    return 0;
14 }
```

@ 3972508240	0
@ 3972508244	1107296256
@ 3972508248	0
@ 3972508252	1
@ 3972508256	-378987904
Memòria de 32 bits	

# Taules en C

## Inicialització de taules

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     // Inicialització
8     for(int i=0; i<MAX; i++){
9         v[i] = 0;
10    }
11    // A partir d'aquí ja puc fer coses
12    // Coses ...
13    return 0;
14 }
```

@ 3972508240	0
@ 3972508244	0
@ 3972508248	0
@ 3972508252	0
@ 3972508256	0
Memòria de 32 bits	

# Taules en C

## Inicialització de taules

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     // Inicialització
8     for(int i=0; i<MAX; i++){
9         v[i] = 0;
10    }
11    // A partir d'aquí ja puc fer coses
12    // Coses ...
13    return 0;
14 }
```

- Una manera alternativa d'inicialitzar (poc freqüent a la vida real)

*S'ha de fer en el moment de la declaració i és molt manual*

```
int v2[MAX] = {0, 0, 0, 0, 0};
```

@ 3972508240

0

@ 3972508244

0

@ 3972508248

0

@ 3972508252

0

@ 3972508256

0

*Memòria de 32 bits*



# Taules en C

## Accés a les taules

```
1 #include <stdio.h>
2
3 #define MAX 5
4 int main(){
5
6     int v[MAX];
7     // Inicialització
8     for(int i=0; i<MAX; i++){
9         v[i] = 0;
10    }
11    // A partir d'aquí ja puc fer coses
12    // Coses ...
13    return 0;
14 }
```

Accés

@ 3972508240

0

@ 3972508244

0

@ 3972508248

0

@ 3972508252

0

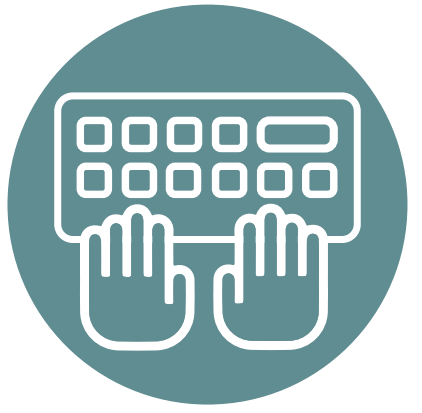
@ 3972508256

0

Memòria de 32 bits

- Es fa servir el nom de la taula i l'índex entre **[ i ]**
- L'índex comença a **0** i acaba a **N-1**
- És obligació del programador controlar que l'índex no se surti de rang

# Taules en C



Exemple pas a pas: Declarar una taula d'enters de N elements, inicialitzar-la a tot zeros, i imprimir-la

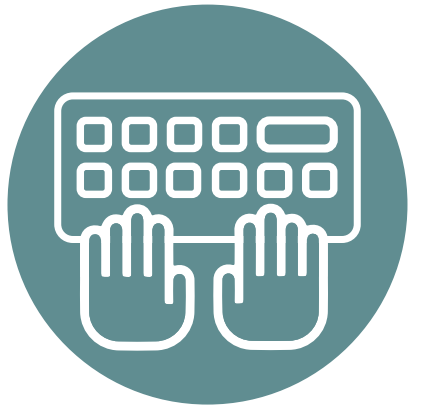
```
#include <stdio.h>
#define N 10

int main(){
    int v[N];
    /* Omplir vector */
    for(int i=0; i < N; i++){
        v[i] = 0;
    }
    /* Imprimir vector */
    for(int i=0; i < N; i++){
        printf("v[%d]=%d\n",i,v[i]);
    }
    return 0;
}
```

Sortida esperada:

```
v[0]=0
v[1]=0
v[2]=0
v[3]=0
v[4]=0
v[5]=0
v[6]=0
v[7]=0
v[8]=0
v[9]=0
```

# Taules en C



Exemple pas a pas: Declarar una taula d'enters de N elements, inicialitzar-la a 100, 200, 300..., i imprimir-la

```
#include <stdio.h>
#define N 10

int main(){
    int v[N];
    /* Omplir vector */
    for(int i=0; i < N; i++){
        v[i] = (i+1)*100;
    }
    /* Imprimir vector */
    for(int i=0; i < N; i++){
        printf("v[%d]=%d\n",i,v[i]);
    }
    return 0;
}
```

Sortida esperada:

```
v[0]=100
v[1]=200
v[2]=300
v[3]=400
v[4]=500
v[5]=600
v[6]=700
v[7]=800
v[8]=900
v[9]=1000
```



# Taules en C

## Taules multi-dimensionals

```
1 #include <stdio.h>
2
3 #define FIL 10
4 #define COL 5
5
6 int main(void)
7 {
8     int taula[FIL][COL];
9
10    for(int i=0; i<FIL; i++){
11        for (int j=0; j<COL; j++){
12            taula[i][j] = 0;
13        }
14    }
15 }
```

*Declaració i inicialització d'una taula **2D***

# Taules en C

## Taules multi-dimensionals

```
1 #include <stdio.h>
2
3 #define FIL 10
4 #define COL 5
5
6 int main(void)
7 {
8     int taula[FIL][COL];
9
10    for(int i=0; i<FIL; i++){
11        for (int j=0; j<COL; j++){
12            taula[i][j] = 0;
13        }
14    }
15 }
```

Declaració i inicialització d'una taula **2D**

- Accés a una taula bi-dimensional:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[ 0 ][ 0 ]	a[ 0 ][ 1 ]	a[ 0 ][ 2 ]	a[ 0 ][ 3 ]
Row 1	a[ 1 ][ 0 ]	a[ 1 ][ 1 ]	a[ 1 ][ 2 ]	a[ 1 ][ 3 ]
Row 2	a[ 2 ][ 0 ]	a[ 2 ][ 1 ]	a[ 2 ][ 2 ]	a[ 2 ][ 3 ]

↓  
Fila

↓  
Columna

# Taules en C

## Taules multi-dimensionals

```
1 #include <stdio.h>
2
3 #define FIL 10
4 #define COL 5
5
6 int main(void)
7 {
8     int taula[FIL][COL];
9
10    for(int i=0; i<FIL; i++){
11        for (int j=0; j<COL; j++){
12            taula[i][j] = 0;
13        }
14    }
15 }
```

*Declaració i inicialització d'una taula **2D***

# Taules en C

## Taules multi-dimensionals

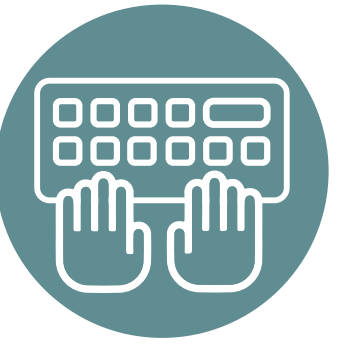
```
1 #include <stdio.h>
2
3 #define FIL 10
4 #define COL 5
5
6 int main(void)
7 {
8     int taula[FIL][COL];
9
10    for(int i=0; i<FIL; i++){
11        for (int j=0; j<COL; j++){
12            taula[i][j] = 0;
13        }
14    }
15 }
```

Declaració i inicialització d'una taula **2D**

```
1 #define FIL 10
2 #define COL 5
3 #define Z 3
4
5 int main(void)
6 {
7     int taula[FIL][COL][Z];
8
9     for(int i=0; i<FIL; i++){
10         for (int j=0; j<COL; j++){
11             for(int k=0; k<Z; k++){
12                 taula[i][j][k] = 0;
13             }
14         }
15     }
16 }
```

Declaració i inicialització d'una taula **3D**

# Taules en C



**Exercici: escriu un codi que donada una taula, l'imprimeixi per pantalla amb bon format (així la farem servir després)**

E.g. si la nostra taula és:

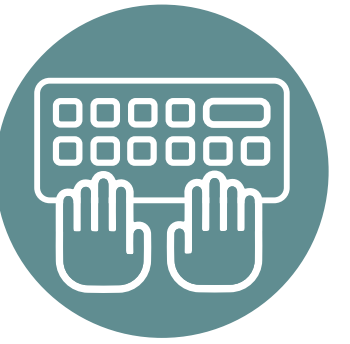
```
int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
```

Volem que la sortida sigui:

1	2	3
4	5	6



# Taules en C



**Exercici: escriu un codi que donada una taula, l'imprimeixi per pantalla amb bon format (així la farem servir després)**

E.g. si la nostra taula és:

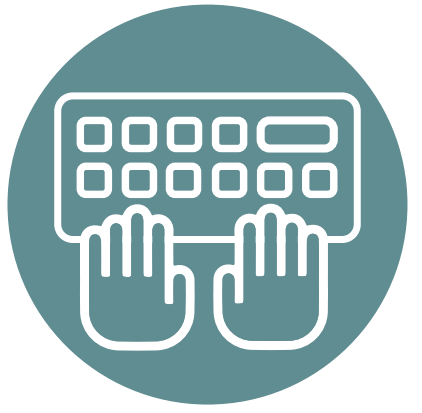
```
int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
```

Volem que la sortida sigui:

1	2	3
4	5	6

```
1 #include <stdio.h>
2 #define FILES 2
3 #define COLS 3
4
5 int main(){
6     /* Taula d'exemple */
7     int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
8     /* Codi per imprimir amb bon format */
9     for(int i = 0; i<FILES; i++){
10         for(int j = 0; j<COLS; j++){
11             printf("%d ", dades[i][j]);
12         }
13         printf("\n");
14     }
15     return 0;
16 }
```

# Taules en C



**Exercici: trobar el mínim d'una taula 2D d'enters. Que a més, ens retorni la posició on es troba el mínim.**

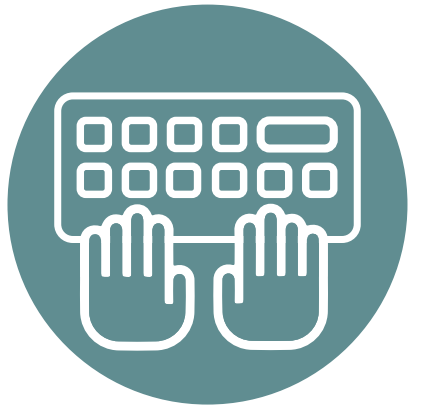
E.g. si la nostra taula és:

```
int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
```

Volem que la sortida sigui:

```
El mínim es 1 i es troba a la  
posició (0,0)
```

# Taules en C



**Exercici: trobar el mínim d'una taula 2D d'enters. Que a més, ens retorni la posició on es troba el mínim.**

E.g. si la nostra taula és:

```
int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
```

Volem que la sortida sigui:

El mínim es 1 i es troba a la posició (0,0)

```
1 #include <stdio.h>
2 #define FILES 2
3 #define COLS 3
4
5 int main(){
6     int dades[FILES][COLS] = {{1, 2, 3}, {4, 5, 6}};
7     int min = dades[0][0];
8     int i_saved, j_saved;
9     for(int i = 0; i<FILES; i++){
10         for(int j = 0; j<COLS; j++){
11             if (dades[i][j]<min){
12                 min = dades[i][j];
13                 i_saved = i;
14                 j_saved = j;
15             }
16         }
17     }
18     printf("El mínim es: %d i es troba a la posició (%d, %d)\n",
19           min, i_saved, j_saved);
20     return 0;
21 }
```


# **CADENES DE CARÀCTERS**

## **AKA STRINGS**

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** ('**\0**').



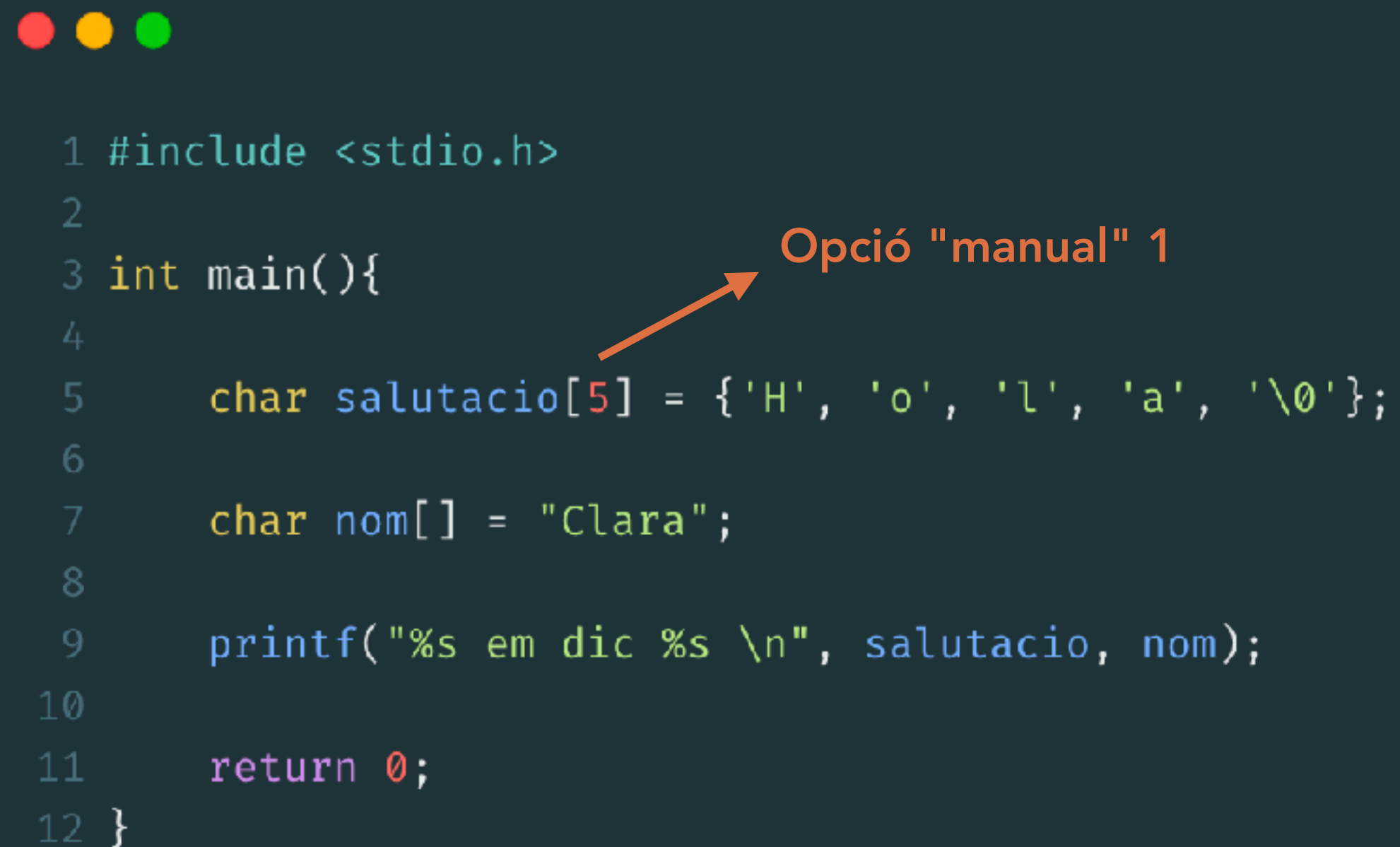
```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```



# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** (`'\0'`).



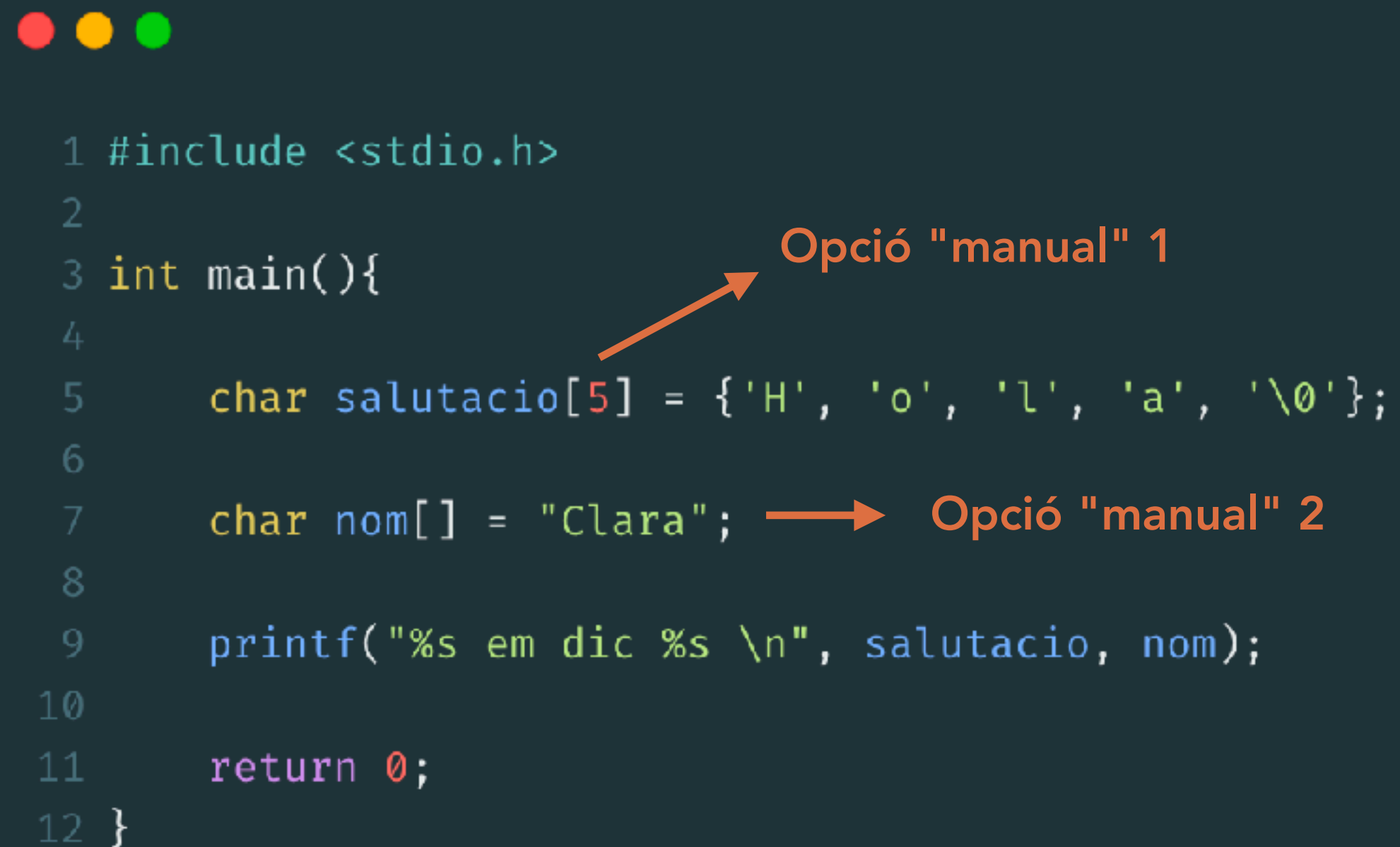
```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** ('**\0**').



```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

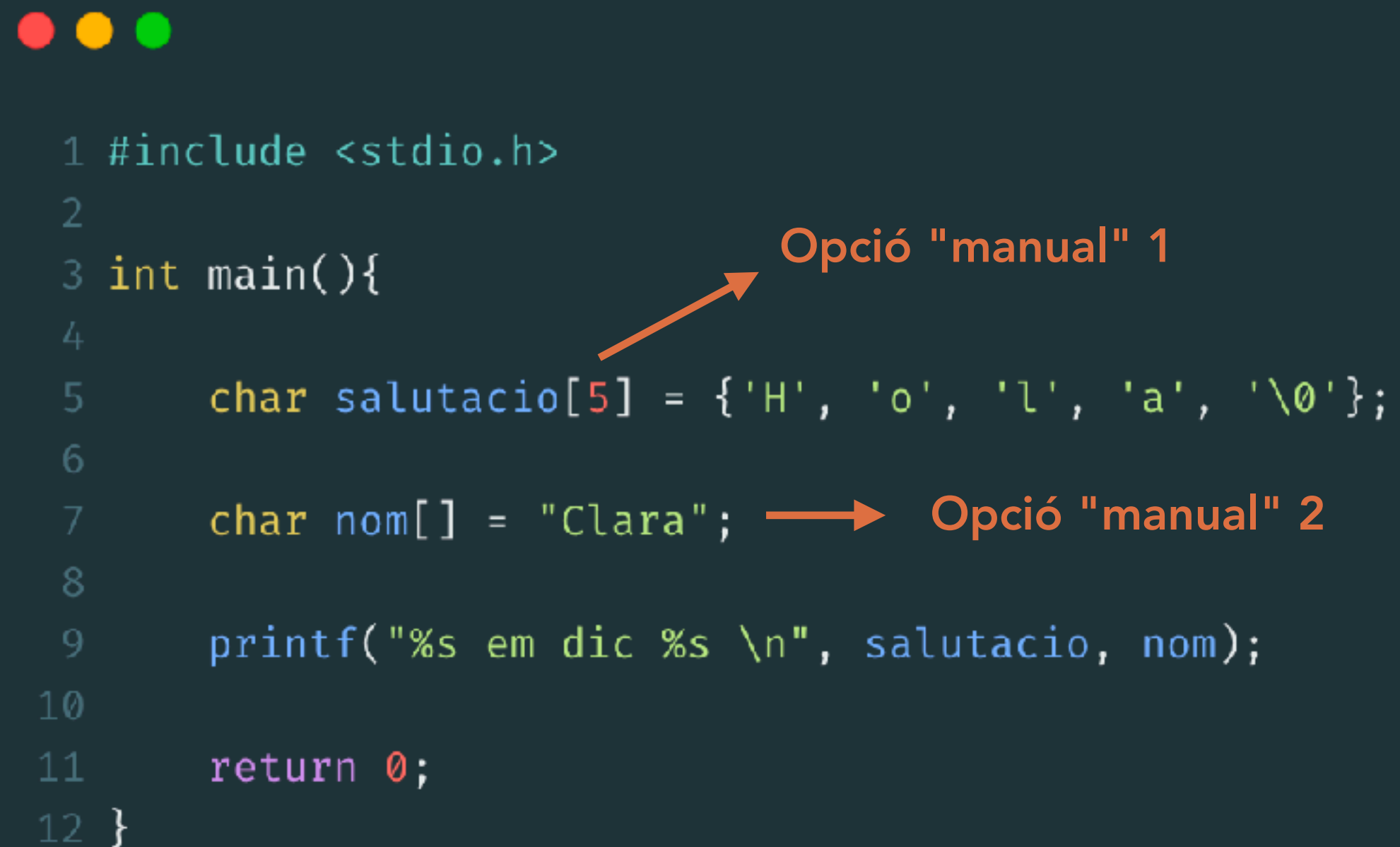
Opció "manual" 1

Opció "manual" 2

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** (`'\0'`).



```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

Opció "manual" 2

The image shows a code editor window with a dark background. It contains a C program that declares two character arrays: 'salutacio' and 'nom'. The 'salutacio' array is explicitly initialized with the characters 'H', 'o', 'l', 'a', and a null terminator '\0'. The 'nom' array is initialized with the string "Clara". The program uses printf to output the strings. Two orange arrows point from text labels to the array declarations. The first arrow points to 'salutacio[5]' and is labeled 'Opció "manual" 1'. The second arrow points to 'nom[]' and is labeled 'Opció "manual" 2'.

Hola em dic Clara

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** ('**\0**').

```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

Opció "manual" 2

```
#include <stdio.h>

int main(){

    char nom[];

    nom = "Clara";

    printf("Em dic %s\n", nom);

    return 0;

}
```

Hola em dic Clara

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** (`'\0'`).

```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

Opció "manual" 2

Hola em dic Clara

```
#include <stdio.h>

int main(){

    char nom[];

    nom = "Clara";

    printf("Em dic %s\n", nom);

    return 0;
}
```

**error:** definition of variable with array type needs an explicit size or an initializer  
char nom[];



# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** (`'\0'`).

```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

Opció "manual" 2

Hola em dic Clara

```
#include <stdio.h>

int main(){

    char nom[];
    nom = "Clara";
    printf("Em dic %s\n", nom);
    return 0;
}
```

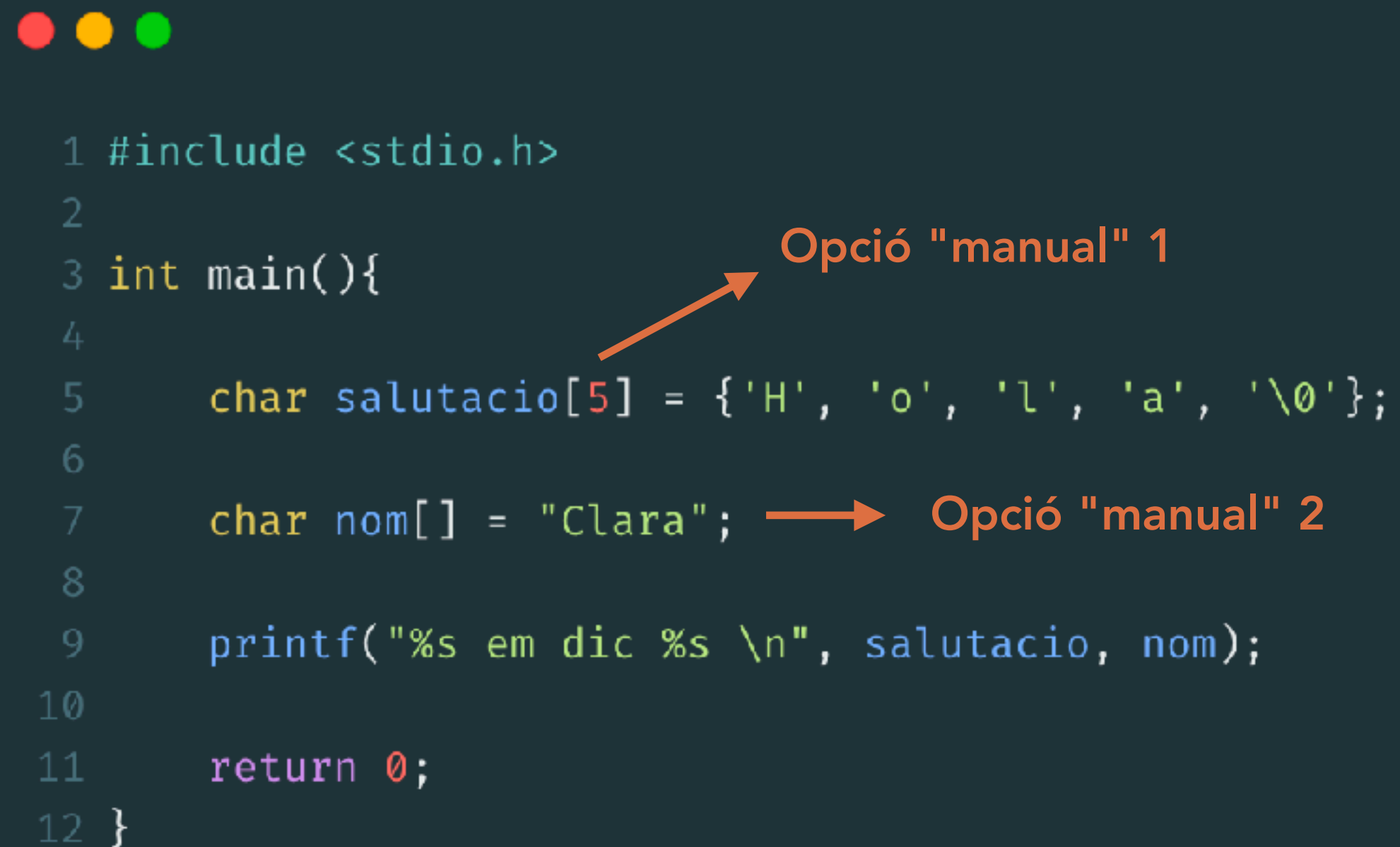
No sap quant espai reservar!

**error:** definition of variable with array type needs an explicit size or an initializer  
char nom[];

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null** (**\0**).



```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

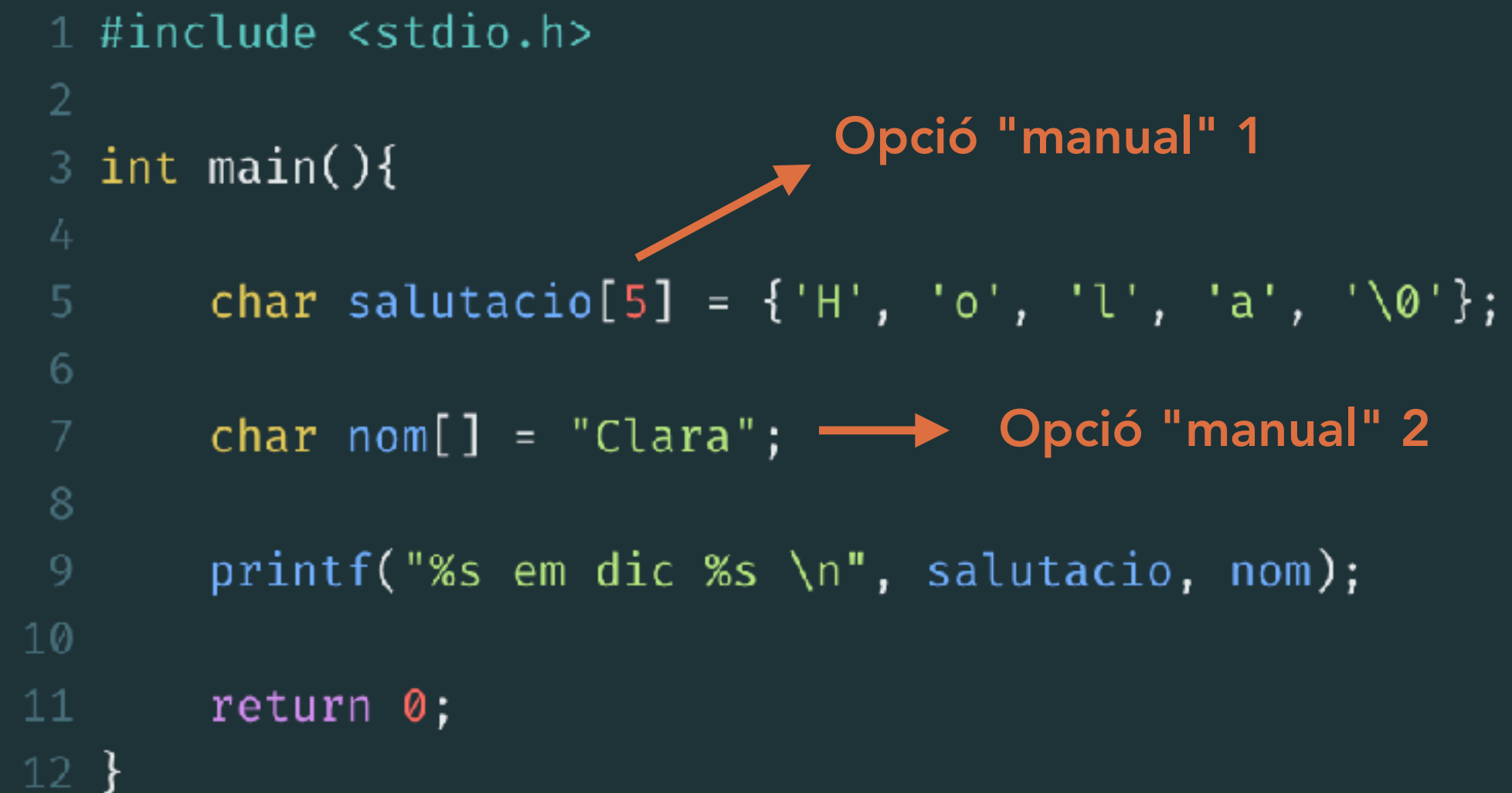
Opció "manual" 1

Opció "manual" 2

# Strings

## Declaració

- Les cadenes de caràcters (strings) no són res més que taules de tipus caràcter que acaben amb un caràcter **null (\0)**.



```
1 #include <stdio.h>
2
3 int main(){
4
5     char salutacio[5] = {'H', 'o', 'l', 'a', '\0'};
6
7     char nom[] = "Clara";
8
9     printf("%s em dic %s \n", salutacio, nom);
10
11     return 0;
12 }
```

Opció "manual" 1

Opció "manual" 2

### Per provar a casa: Què passa si...?

- No afegeixo el caràcter final en l'opció 1?
- En l'opció 2, puc especificar la mida també. Què passa si especifico una mida més petita? I més gran?

# Strings

## Tractament com taules

- Com que són taules, es poden recórrer com taules.

```
1 #include <stdio.h>
2
3 int main(){
4
5     char nom[] = "Programacio";
6
7     for(int i=0; i<11; i++){
8         printf("%c\n", nom[i]);
9     }
10
11     return 0;
12 }
```

P  
r  
o  
g  
r  
a  
m  
a  
c  
i  
o

# Strings

## Tractament com taules

- Com que són taules, es poden recórrer com taules.

```
1 #include <stdio.h>
2
3 int main(){
4
5     char nom[] = "Programacio";
6
7     for(int i=0; i<11; i++){
8         printf("%c\n", nom[i]);
9     }
10
11     return 0;
12 }
```

Disculpeu

P  
r  
o  
g  
r  
a  
m  
a  
c  
i  
o



# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?

```
#include <stdio.h>
#include <stdbool.h>

#define N 11
int main(){

    char s1[N] = "Programacio";
    char s2[N] = "programacio";

    //Puc fer això??
    if (s1 == s2) {
        printf("S1 i S2 son iguals\n");
    }

    return 0;
}
```

# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?

```
#include <stdio.h>
#include <stdbool.h>

#define N 11
int main(){

    char s1[N] = "Programacio";
    char s2[N] = "programacio";

    //Puc fer això??
    if (s1 == s2) {
        printf("S1 i S2 son iguals\n");
    }

    printf("Què val S1 = %u\n", s1);
    printf("Què val S2 = %u\n", s2);

    return 0;
}
```

# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?

```
#include <stdio.h>
#include <stdbool.h>

#define N 11
int main(){

    char s1[N] = "Programacio";
    char s2[N] = "programacio";

    //Puc fer això??
    if (s1 == s2) {
        printf("S1 i S2 son iguals\n");
    }

    printf("Què val S1 = %u\n", s1);
    printf("Què val S2 = %u\n", s2);

    return 0;
}
```

```
Què val S1 = 3860523360
Què val S2 = 3860523344
```

# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?
- No perquè estic comparant adreces de memòria.

```
#include <stdio.h>
#include <stdbool.h>

#define N 11
int main(){

    char s1[N] = "Programacio";
    char s2[N] = "programacio";

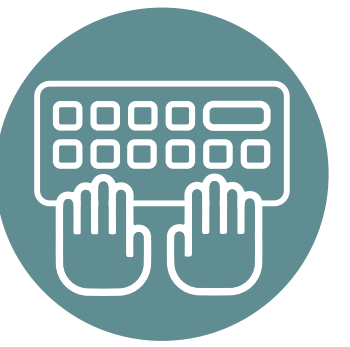
    //Puc fer això??
    if (s1 == s2) {
        printf("S1 i S2 son iguals\n");
    }

    printf("Què val S1 = %u\n", s1);
    printf("Què val S2 = %u\n", s2);

    return 0;
}
```

```
Què val S1 = 3860523360
Què val S2 = 3860523344
```

# Strings



## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?
- No perquè estic comparant adreces de memòria.
- Dos strings són iguals si cadascun dels seus caràcters són iguals.

# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?
- No perquè estic comparant adreces de memòria.
- Dos strings són iguals si cadascun dels seus caràcters són iguals.



```
#include <stdio.h>
#include <stdbool.h>
```

```
#define N 11
int main() {
```

```
    char s1[N] = "Programacio";
    char s2[N] = "programacio";
    bool iguals = true;
    int i = 0;
```

**Quin codi va aquí?**

```
    if (iguals) {
        printf("Els strings son iguals\n");
    }
    else {
        printf("Els strings son diferents\n");
    }
    return 0;
}
```



# Strings

## Comparació de strings

- Puc fer una comparació directa amb l'operador `==` ?
- No perquè estic comparant adreces de memòria.
- Dos strings són iguals si cadascun dels seus caràcters són iguals.

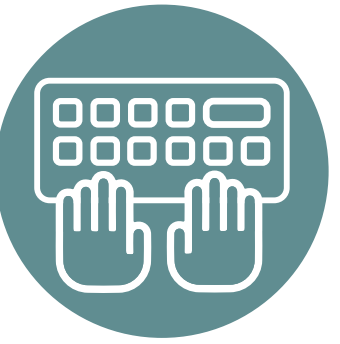
```
#include <stdio.h>
#include <stdbool.h>

#define N 11
int main(){

    char s1[N] = "Programacio";
    char s2[N] = "programacio";
    bool iguals = true;
    int i = 0;

    while (i<N && iguals){
        if (s1[i] != s2[i]){
            iguals = false;
        }
        i++;
    }

    if (iguals){
        printf("Els strings son iguals\n");
    }
    else {
        printf("Els strings son diferents\n");
    }
    return 0;
}
```



# Strings

**Llibreria <string.h> —> *No us confongueu amb la llibreria <strings.h>***

- A la llibreria <string.h> tenim algunes funcions que ens faciliten la vida amb els strings.

[https://www.tutorialspoint.com/c\\_standard\\_library/string\\_h.htm](https://www.tutorialspoint.com/c_standard_library/string_h.htm)

# Strings

**Llibreria <string.h> → *No us confongueu amb la llibreria <strings.h>***

- A la llibreria <string.h> tenim algunes funcions que ens faciliten la vida amb els strings.  
[https://www.tutorialspoint.com/c\\_standard\\_library/string\\_h.htm](https://www.tutorialspoint.com/c_standard_library/string_h.htm)
- Longitud d'un string amb **strlen**:

```
size_t strlen(const char *str)
```

```
strlen(s1);
```

Retorna la longitud de l'string s1

# Strings

**Llibreria <string.h> → No us confongueu amb la llibreria <strings.h>**

- A la llibreria <string.h> tenim algunes funcions que ens faciliten la vida amb els strings.

[https://www.tutorialspoint.com/c\\_standard\\_library/string\\_h.htm](https://www.tutorialspoint.com/c_standard_library/string_h.htm)

- Longitud d'un string amb **strlen**:

```
size_t strlen(const char *str)
```

```
strlen(s1);
```

Retorna la longitud de l'string s1

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define N 100
5 int main () {
6
7     char str1[N] = "Hello";
8     int len;
9     len = strlen(str1);
10    printf("strlen(str1) :  %d\n", len );
11
12    return 0;
13 }
```

```
strlen(str1) : 5
```

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)

        strcpy(s1, s2);
```

Copia l'string src a l'string dest

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
    strcpy(s1, s2);
```

Copia l'string src a l'string dest

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11
12
13
14
15
16     return 0;
17 }
```



# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
    strcpy(s1, s2);
```

Copia l'string src a l'string dest

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11
12
13
14
15
16     return 0;
17 }
```

```
s1 = Hello
s2 = \0???
```

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
strcpy(s1, s2);
```

Copia l'string src a l'string dest

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11     /* Copiar s1 a s2 */
12     // Puc fer s2 = s1?
13     strcpy(s2, s1);
14     printf("s1 = %s\n", s1);
15     printf("s2 = %s\n", s2);
16     return 0;
17 }
```

```
s1 = Hello
s2 = Ъ???
```

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
    strcpy(s1, s2);
```

Copia l'string src a l'string dest

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11     /* Copiar s1 a s2 */
12     // Puc fer s2 = s1?
13     strcpy(s2, s1);
14     printf("s1 = %s\n", s1);
15     printf("s2 = %s\n", s2);
16     return 0;
17 }
```

```
s1 = Hello
s2 = \0\0\0
s1 = Hello
s2 = Hello
```

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
    strcpy(s1, s2);
```

Copia l'string src a l'string dest

- **Pregunta:** per què necessito una funció, no puc fer `s1 = s2`?

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11     /* Copiar s1 a s2 */
12     // Puc fer s2 = s1?
13     strcpy(s2, s1);
14     printf("s1 = %s\n", s1);
15     printf("s2 = %s\n", s2);
16     return 0;
17 }
```

```
s1 = Hello
s2 = \0\0\0
s1 = Hello
s2 = Hello
```

# Strings

## Llibreria <string.h>

- Copiar els continguts d'un string a un altre amb **strcpy**.

```
char *strcpy(char *dest, const char *src)
```

```
    strcpy(s1, s2);
```

Copia l'string src a l'string dest

- **Pregunta:** per què necessito una funció, no puc fer `s1 = s2`?
- **Resposta:** assignar taules no està permès pel compilador, i si compilés, què creieu que faria?
- Què són `s1` i `s2`?

```
1 #include <stdio.h>
2 #include <string.h>
3 #define N 100
4 int main () {
5
6     char s1[N] = "Hello";
7     char s2[N];
8     printf("s1 = %s\n", s1);
9     printf("s2 = %s\n", s2);
10
11     /* Copiar s1 a s2 */
12     // Puc fer s2 = s1?
13     strcpy(s2, s1);
14     printf("s1 = %s\n", s1);
15     printf("s2 = %s\n", s2);
16     return 0;
17 }
```

```
s1 = Hello
s2 = \0\0\0
s1 = Hello
s2 = Hello
```

# Strings

## Llibreria <string.h>

- Concatenar dos strings amb **strcat**

```
char *strcat(char *dest, const char *src)
```

```
strcat(s1, s2);
```

**OJO!** Afegeix l'string s2 al final de s1



# Strings

## Llibreria <string.h>

- Concatenar dos strings amb **strcat**

```
char *strcat(char *dest, const char *src)
```

```
strcat(s1, s2);
```

**OJO!** Afegeix l'string s2 al final de s1

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define N 100
5 int main () {
6
7     char s1[N] = "Hello";
8     char s2[N] = "World";
9
10    /* Concatenar s1 i s2 */
11    strcat( s1, s2);
12    printf("strcat( s1, s2):  %s\n", s1);
13    return 0;
14 }
```

# Strings

## Llibreria <string.h>

- Concatenar dos strings amb **strcat**

```
char *strcat(char *dest, const char *src)
```

```
strcat(s1, s2);
```

**OJO!** Afegeix l'string s2 al final de s1

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define N 100
5 int main () {
6
7     char s1[N] = "Hello";
8     char s2[N] = "World";
9
10    /* Concatenar s1 i s2 */
11    strcat( s1, s2);
12    printf("strcat( s1, s2):  %s\n", s1);
13    return 0;
14 }
```

```
strcat( s1, s2):  HelloWorld
```

# **A CASA...**

## **EXERCICIS L4 del Moodle**

Compte que el qüestionari tanca dimarts a la nit!

# **EL PROPER DIA...**

## **PROCEDIMENTS**