

LABORATORIS

PROGRAMACIÓ CIENTÍFICA

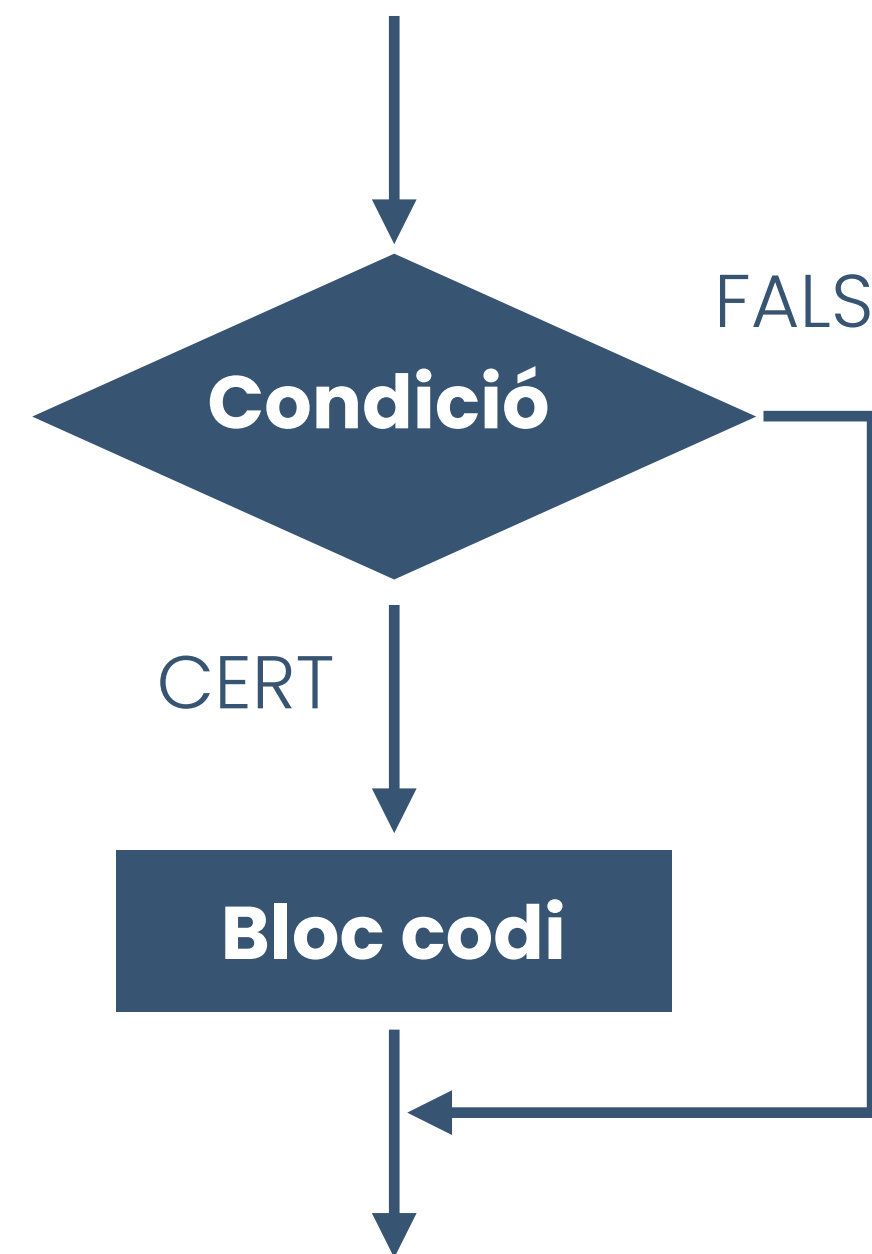
Sessió 3: Conditionals i bucles

ESTRUCTURES

CONDICIONALS

Recordem...

- El **condicional** o selecció és una estructura de control que permet l'execució d'instruccions segons si es compleixen o no unes condicions.



- Tipus:
 - Condicional simple
 - Condicional doble
 - Condicionals anidats
 - Condicional múltiple

Estructura condicional

Condicional simple

A teoria:

```
...  
si (a < b) llavors  
    a := a + b;  
fsi  
...
```

En C: if

- Es fa servir la paraula reservada **if**, i entre claus **{ }** posarem el codi que s'executarà en cas que es compleixi la condició

```
...  
if (a < b){  
    a = a + b;  
}  
...
```

Exemple

```
...  
#include <stdio.h>  
  
int main(){  
  
    int a, b;  
    scanf("%d", &a);  
    scanf("%d", &b);  
  
    if (a < b){  
        printf("A és menor que B\n");  
    }  
  
    return 0;  
}
```

- Què imprimeix si a = 3 i b = 4 ?
- Què imprimeix si a = 10 i b = 2?
- Què imprimeix si a = 5 i b = 5?

Condicional doble

A teoria:

```
...  
si (a < b) llavors  
    a := a + b;  
sinó  
    a := 1;  
fsi
```

En C: if-else

- Farem servir la clàusula if-else. El cos de l'if s'executarà si la condició és certa, i si no, s'executarà el cos de l'else

```
1 ...  
2  if (a < b){  
3      a = a + b;  
4  }  
5  else {  
6      a = 1;  
7  }  
8  ...
```

Exemple

```
#include <stdio.h>  
#define LIMIT_APR 5  
int main(){  
  
    int nota;  
    scanf("%d", &nota);  
  
    if (nota >= LIMIT_APR){  
        printf("Estas aprovat.\n");  
    }  
    else {  
        printf("Estas suspès! Uy uy uyyy\n");  
    }  
  
    return 0;  
}
```

- Què imprimeix si nota = 4?
- Què imprimeix si nota = 6?
- Què imprimeix si nota = 5?
- Què imprimeix si nota = 352?

Condicional anidat

A teoria:

```
...  
si (a < b) llavors  
    si (a % b = 0) llavors  
        a := a + b;  
    fsi  
sinó  
    a := 1;  
fsi
```

En C:

- Podem incloure condicions dins de condicions, tant dins del cos de l'**if** com de l'**else**

```
1 ...  
2  if (a < b){  
3      if (a % b == 0){  
4          a = a + b;  
5      }  
6  }  
7  else {  
8      a = 1;  
9  }  
10 ...
```

Exemple

```
#include <stdio.h>  
#define LIMIT_APR 5  
int main(){  
  
    int nota;  
    scanf("%d", &nota);  
  
    if (nota < 0 || nota > 10){  
        printf("Error en l'escala!\n");  
    }  
    else{  
        if (nota >= LIMIT_APR){  
            printf("Estas aprovat.\n");  
        }  
        else {  
            printf("Estas suspès! Uy uy uyyy\n");  
        }  
    }  
  
    return 0;  
}
```

- Què imprimeix si nota = -3?
- Què imprimeix si nota = 352?
- Què imprimeix si nota = 3?

Condicional múltiple: opció 1

OPCIÓ 1: Amb **if-elseif - else**

- Si necessitem incloure més condicions es pot fer servir if - else if - else

```
if(cond_1) {  
    // S'executa si cond_1 és certa  
} else if(cond_2) {  
    // S'executa si cond_2 és certa  
} else if(cond_3) {  
    // S'executa si cond_3 és certa  
} else {  
    // S'executa si cap de les condicions  
    // anteriors és certa  
}
```

Exemple

```
1 #include <stdio.h>  
2  
3 int main(){  
4  
5     int nota;  
6     scanf("%d", &nota);  
7  
8     if (nota < 0 || nota > 10){  
9         printf("Error en l'escala!\n");  
10    }  
11    else{  
12        if (nota < 5){  
13            printf("SUSPÈS\n");  
14        }  
15        else if (nota < 7) {  
16            printf("APROVAT\n");  
17        }  
18        else if (nota < 9){  
19            printf("NOTABLE\n");  
20        }  
21        else {  
22            printf("EXCEL·LENT\n");  
23        }  
24    }  
25    return 0;  
26 }
```


Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

- El **switch** ens permet comparar els possibles valors d'una variable i executar codi en funció dels seus valors.
- Cada valor diferent que pugui prendre la nostra variable s'haurà de contemplar en un '**case**' diferent.
- Quan la variable a la qual fem switch sigui igual a un dels casos, s'executa aquell bloc de codi fins que trobem una instrucció **break**.
- Quan arribem a un break, sortim del switch i s'executa la següent línia de codi després del switch

```
1 int numero_dia;
2 scanf("%d",&numero_dia);
3
4 switch (numero_dia) {
5     case 1 :
6         printf("Dilluns \n");
7         break;
8     case 2:
9         printf("Dimarts \n");
10        break;
11    //...
12    case 7 :
13        printf("Diumenge \n");
14        break;
15
16
17 }
```

- Què imprimeix si numero_dia = 1 ?
- Què imprimeix si numero_dia = 9?

Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

- El **switch** ens permet comparar els possibles valors d'una variable i executar codi en funció dels seus valors.
- Cada valor diferent que pugui prendre la nostra variable s'haurà de contemplar en un **'case'** diferent.
- Quan la variable a la qual fem switch sigui igual a un dels casos, s'executa aquell bloc de codi fins que trobem una instrucció **break**.
- Quan arribem a un break, sortim del switch i s'executa la següent línia de codi després del switch
- En un switch, opcionalment podem tenir un cas per defecte, el **default** case. Aquest ha d'aparèixer al final del switch i el seu cos s'executarà si cap dels casos anteriors és cert. El default no cal que tingui break però millor posar-lo.

```
1 int numero_dia;
2 scanf("%d",&numero_dia);
3
4 switch (numero_dia) {
5     case 1 :
6         printf("Dilluns \n");
7         break;
8     case 2:
9         printf("Dimarts \n");
10        break;
11    //...
12    case 7 :
13        printf("Diumenge \n");
14        break;
15
16
17 }
```

- Què imprimeix ara si numero_dia = 9?

Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

Sobre els breaks

- No tots els casos necessiten tenir un break. Si no hi ha break, el flux d'execució seguirà a través del següent 'case' fins que trobi un break.

```
1 int numero_dia;  
2 scanf("%d",&numero_dia);  
3  
4 switch (numero_dia) {  
5     case 1 :  
6         printf("Dilluns \n");  
7         break;  
8     case 2:  
9         printf("Dimarts \n");  
10        break;  
11        //...  
12        case 6 :  
13        case 7 :  
14            printf("Cap de setmana \n");  
15            break;  
16        default :  
17            printf ("Valor invàlid\n");  
18 }
```

- Què imprimeix si numero_dia = 6 ?

Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

Sobre els breaks

- No tots els casos necessiten tenir un break. Si no hi ha break, el flux d'execució seguirà a través del següent 'case' fins que trobi un break.

Error més comú

- Deixar-se un break sense voler.

```
1 int numero_dia;
2 scanf("%d",&numero_dia);
3
4 switch (numero_dia) {
5     case 1 :
6         printf("Dilluns \n");
7     case 2:
8         printf("Dimarts \n");
9         break;
10    //...
11    case 6 :
12    case 7 :
13        printf("Cap de setmana \n");
14        break;
15    default :
16        printf ("Valor invàlid\n");
17 }
```

- Què imprimeix si numero_dia = 1 ?

Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

Limitació important !

- L'expressió d'un case ha de tenir el mateix tipus de dades que la variable del switch, i ha de ser una constant o un literal !!!

```
1 int numero_dia;  
2 scanf("%d",&numero_dia);  
3  
4 switch (numero_dia) {  
5     case 1:  
6         printf("Dilluns \n");  
7     case 2:  
8         printf("Dimarts \n");  
9         break;  
10    // ...  
11    case 6:  
12    case 7:  
13        printf("Cap de setmana \n");  
14        break;  
15    default :  
16        printf ("Valor invàlid\n");  
17 }
```

Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

Limitació important !

- L'expressió d'un case ha de tenir el mateix tipus de dades que la variable del switch, i ha de ser una constant o un literal !!!
- Això ens restringeix a usar switch amb variables de tipus enter, char, enumerat.

```
char lletra;  
scanf(" %c",&lletra);  
  
switch (lletra) {  
    case 'A':  
        printf("Es la A \n");  
        break;  
    case 'B':  
        printf("Es la B \n");  
        break;  
    //...  
    default :  
        printf ("No sé la lletra\n");  
}
```

Codi: OK

*Les lletres han d'anar entre cometes simples perquè
les tracti com a literals i no com a variables*


Condicional múltiple: opció 2

OPCIÓ 2: Amb switch

Limitació important !

- L'expressió d'un case ha de tenir el mateix tipus de dades que la variable del switch, i ha de ser una constant o un literal !!!
- Això ens restringeix a usar switch amb variables de tipus enter, char, enumerat.
- Això vol dir que no podem fer servir una expressió condicional!
- Aneu molt en compte amb el switch. Fer-lo servir només si ens estalviem codi respecte fer servir "if" o si resulta molt més llegible.

```
1 float nota;  
2 scanf("%f",&nota);  
3  
4 switch (nota) {  
5     case <5 :  
6         printf("Suspe\n");  
7         break;  
8     case <7 :  
9         printf("Aprovat \n");  
10        break;  
11        //...  
12    default :  
13        printf ("Nota invàlida\n");  
14 }
```



Aquest codi és totalment incorrecte i no compila

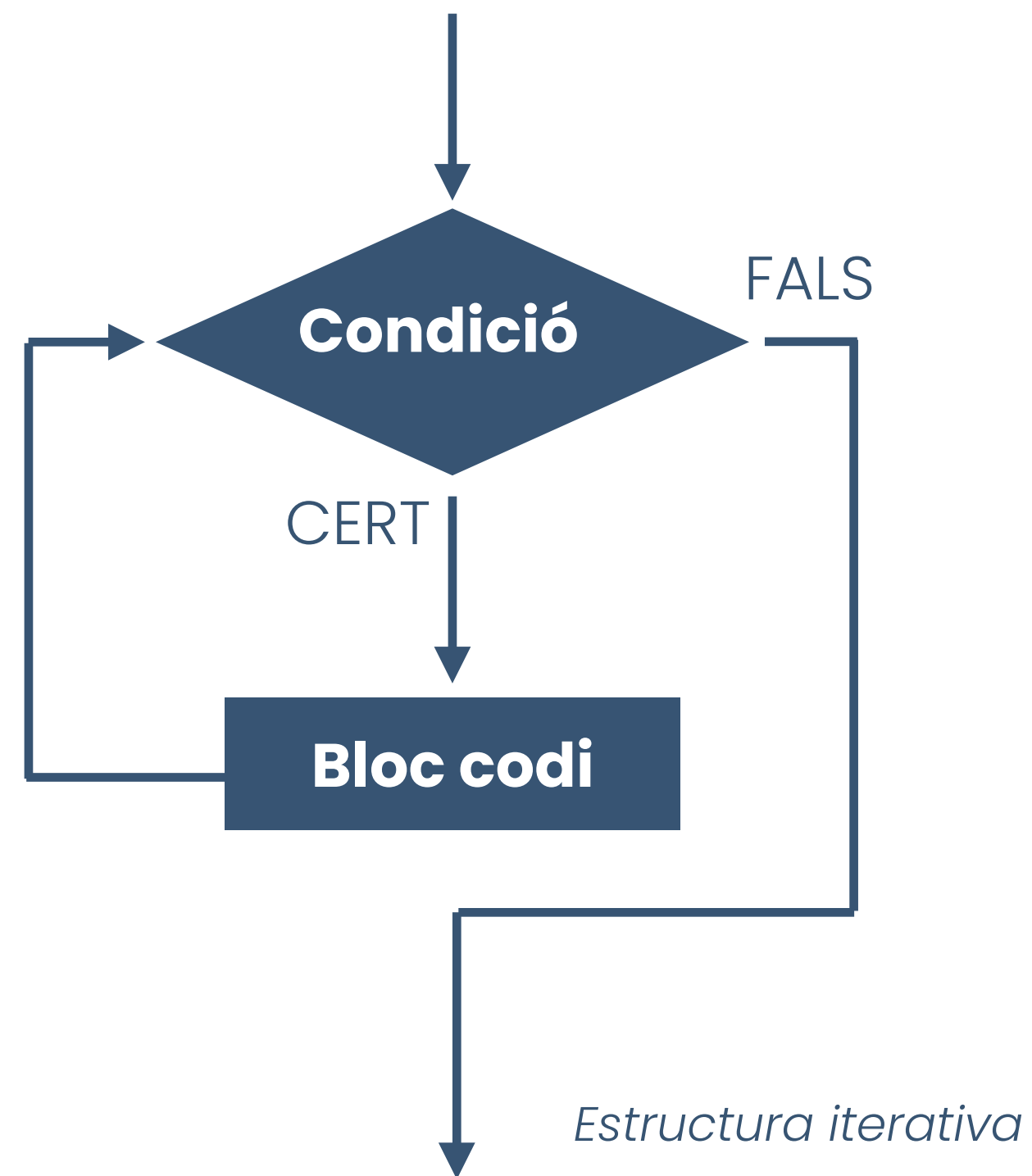
error: expression is not an integer constant expression

BUCL

(LOOPS)

Recordem...

- El **bucle** (o iteració o loop) permet executar instruccions un determinat nombre de vegades o mentre es compleixi una condició.



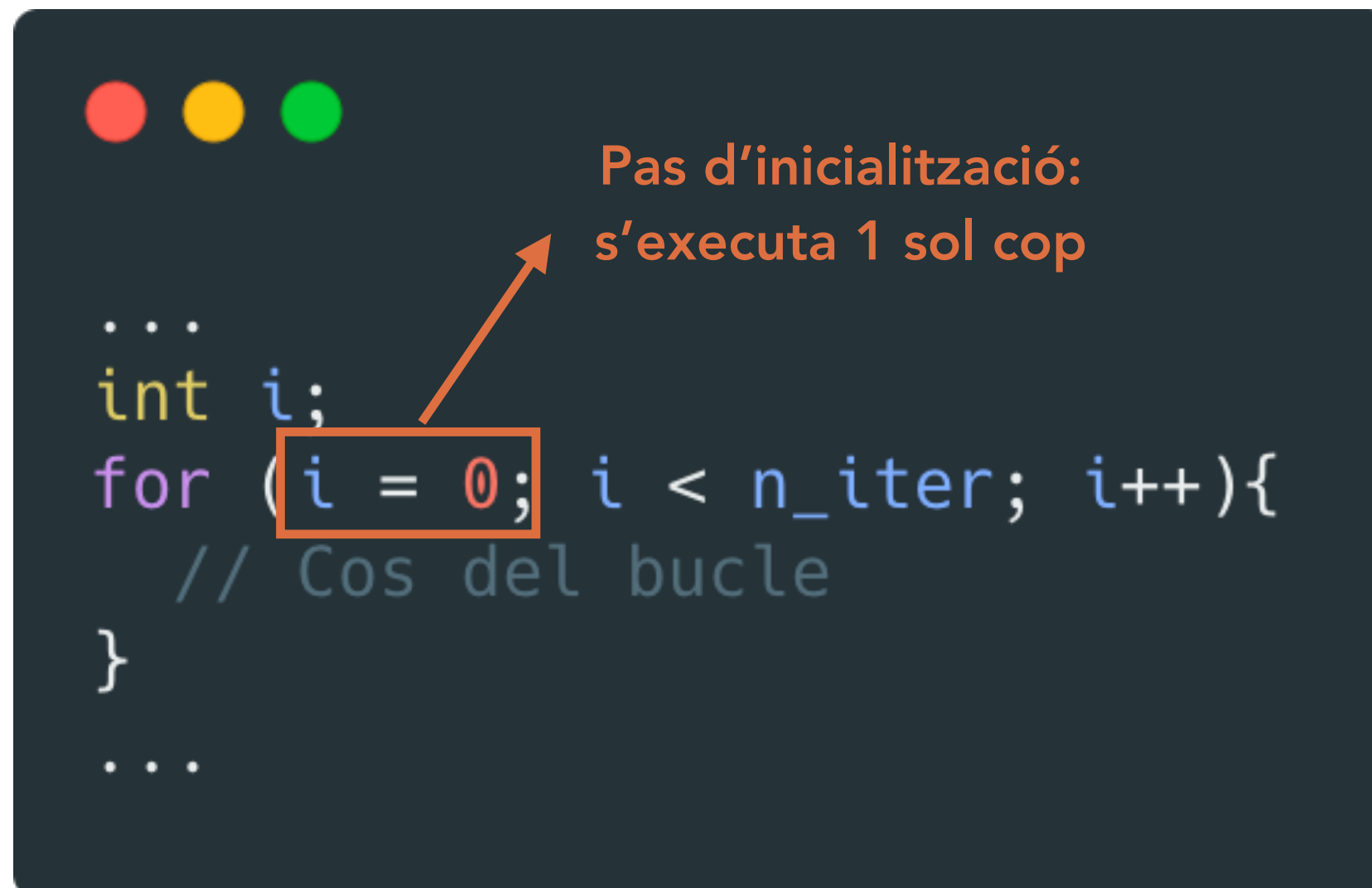
Bucle 'for'

A teoria:

```
per (i:= 0; i<n_iteracions; i:= i+1) fer  
    Cos del bucle  
fper
```

- Recordeu: El seu ús es restringeix a **quan coneixem el nombre d'iteracions**

En C: for



The diagram shows a code editor window with a dark background. At the top left, there are three colored circles (red, yellow, green) representing window controls. The code is as follows:

```
...  
int i;  
for (i = 0; i < n_iter; i++){  
    // Cos del bucle  
}  
...
```

An orange arrow points from the text "Pas d'inicialització: s'executa 1 sol cop" to the initialization part of the for loop, "i = 0". The "i = 0;" is enclosed in an orange rectangular box.

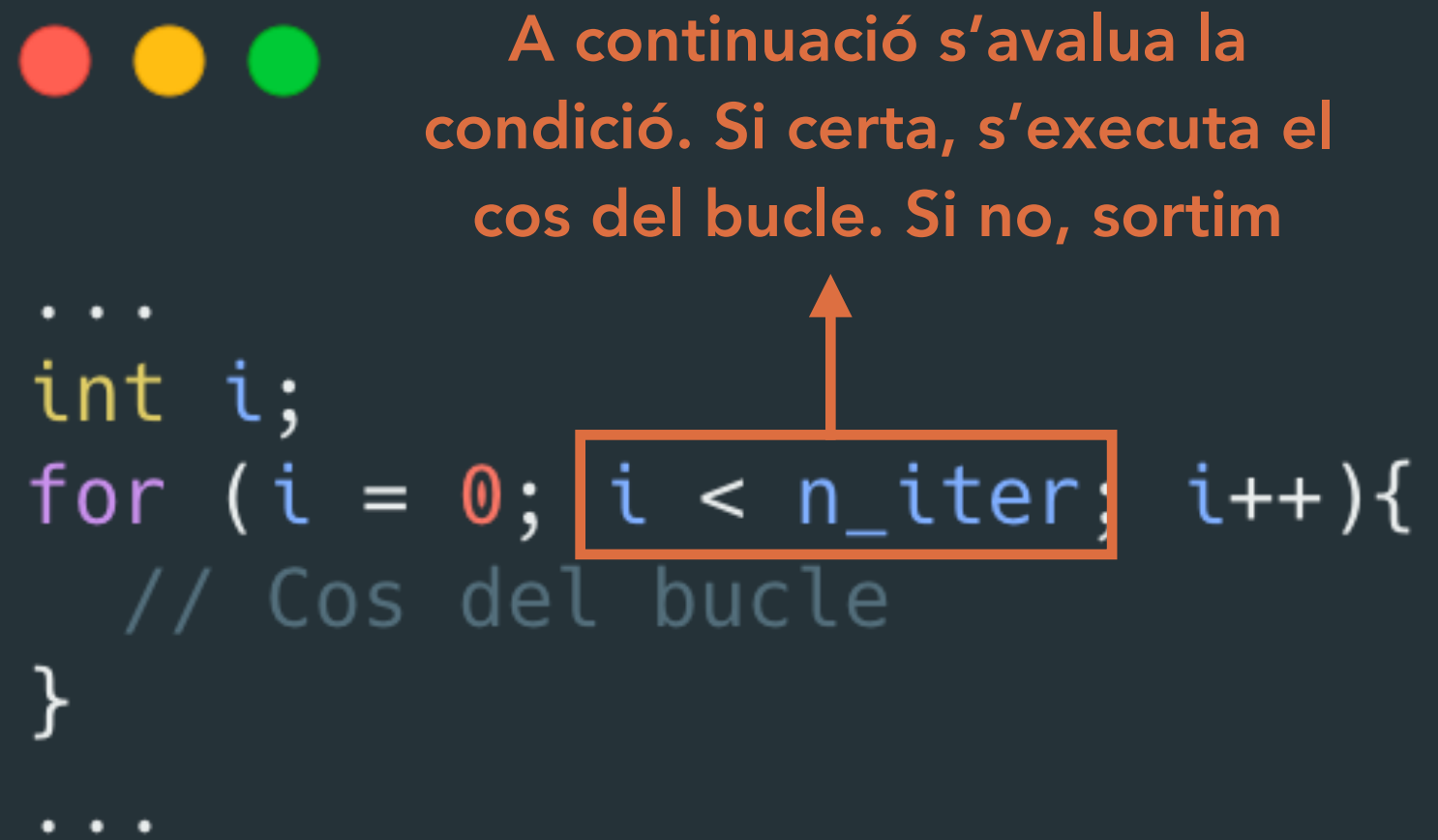
Bucle 'for'

A teoria:

```
per (i:= 0; i<n_iteracions; i:= i+1) fer  
    Cos del bucle  
fper
```

- Recordeu: El seu ús es restringeix a **quan coneixem el nombre d'iteracions**

En C: for



A continuació s'avalua la condició. Si certa, s'executa el cos del bucle. Si no, sortim

```
...  
int i;  
for (i = 0; i < n_iter; i++){  
    // Cos del bucle  
}  
...
```

The diagram shows a C code snippet for a for loop. An orange box highlights the condition 'i < n_iter' in the for loop header. An orange arrow points from this box to the explanatory text above it.

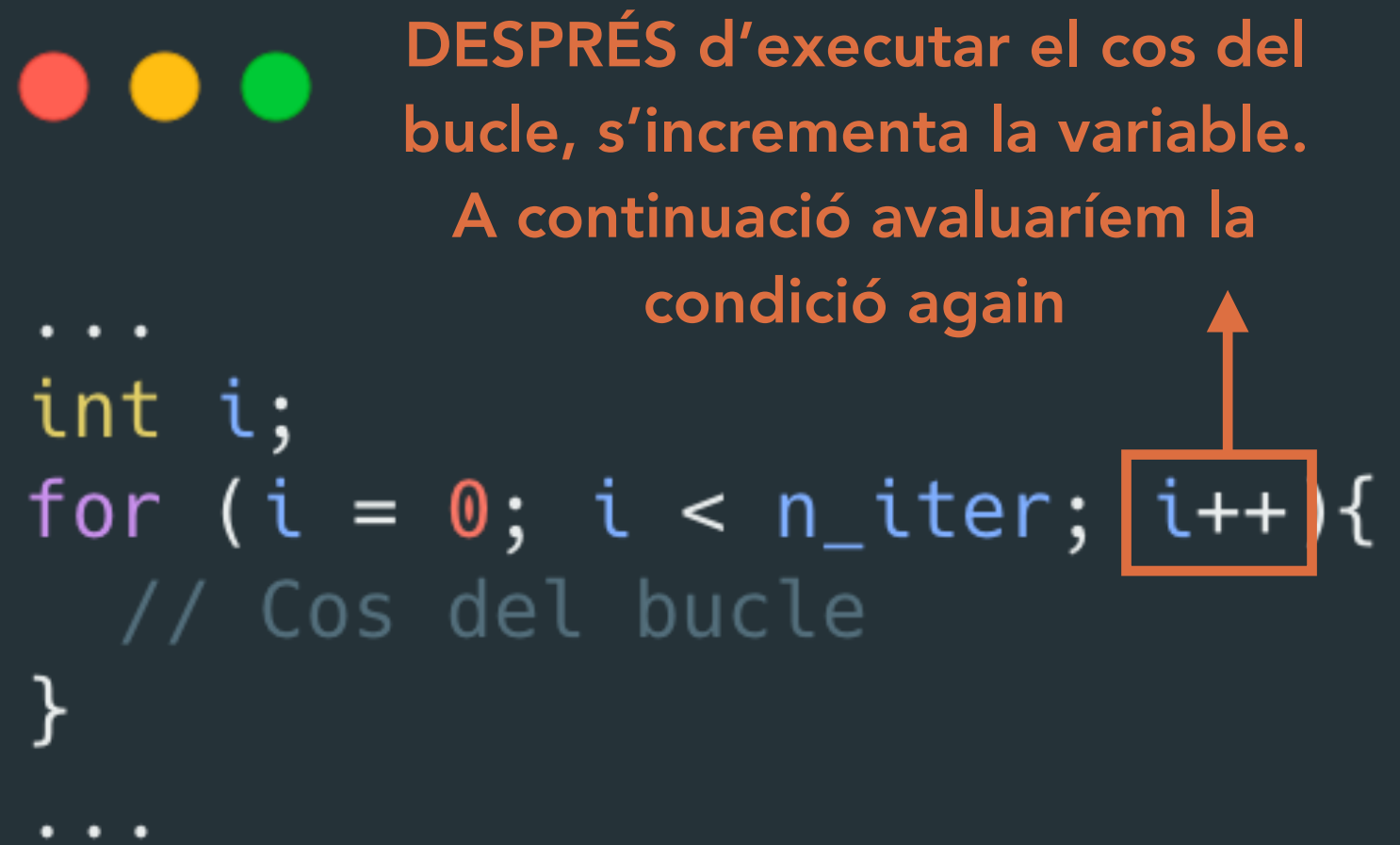
Bucle 'for'

A teoria:

```
per (i:= 0; i<n_iteracions; i:= i+1) fer  
    Cos del bucle  
fper
```

- Recordeu: El seu ús es restringeix a **quan coneixem el nombre d'iteracions**

En C: for



The diagram shows a C code snippet for a for loop. The code is: `...
int i;
for (i = 0; i < n_iter; i++){
 // Cos del bucle
}
...`. The `i++` part of the for loop is highlighted with an orange box. An orange arrow points from this box to the text: "DESPRÉS d'executar el cos del bucle, s'incrementa la variable. A continuació avaluaríem la condició again".

```
...  
int i;  
for (i = 0; i < n_iter; i++){  
    // Cos del bucle  
}  
...
```

Bucle 'for'

A teoria:

```
per (i:= 0; i<n_iteracions; i:= i+1) fer  
    Cos del bucle  
fper
```

- Recordeu: El seu ús es restringeix a **quan coneixem el nombre d'iteracions**

En C: for

```
...  
int i;  
for (i = 0; i < n_iter; i++){  
    // Cos del bucle  
}  
...
```

```
...  
for (int i = 0; i < n_iter; i++){  
    // Cos del bucle  
}  
...
```

A partir de l'estàndard C99
`gcc programa.c -std=c99 -o programa`

- Què passa si la variable **i** ja existia?

```
#include <stdio.h>  
/* PROVA VARIABLES LOCALS DEL FOR  
   Compilat amb -std=c99 */  
#define N 5  
  
int main(){  
    int i=666;  
    for(int i=0; i<N; i++){  
        printf("Valor: %d\n", i);  
    }  
    printf("Valor de i: %d\n", i);  
}
```

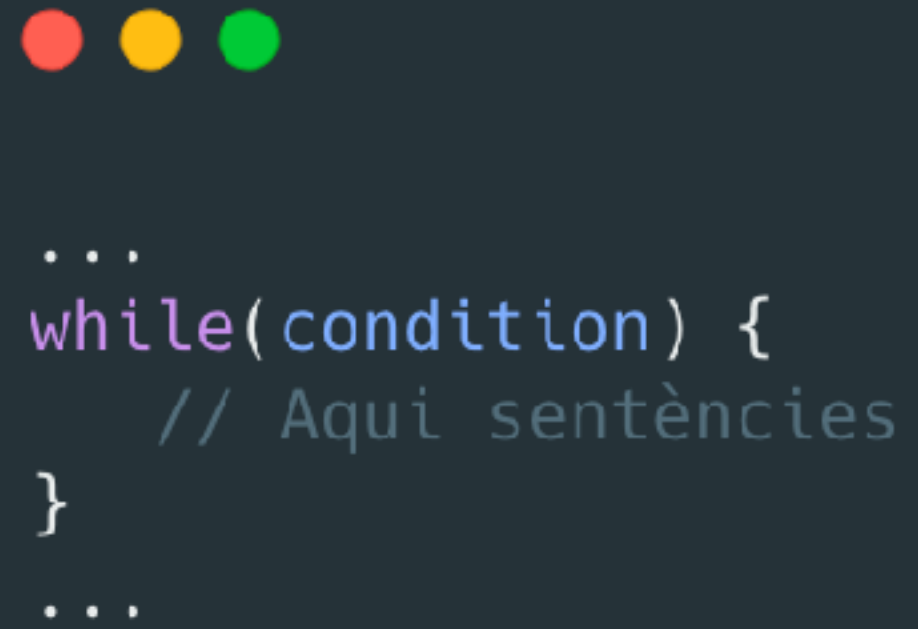
```
Valor: 1  
Valor: 2  
Valor: 3  
Valor: 4  
Valor de i: 666
```


Bucle 'while'

A teoria: mentre

```
Inicialització variable control
mentre (condició) fer
    Cos del bucle
    Actualització variable control
fmentre
```

En C: while



```
...
while(condition) {
    // Aquí sentències
}
...
```

Bucle 'while'

A teoria: mentre

```
Inicialització variable control
mentre (condició) fer
    Cos del bucle
    Actualització variable control
fmentre
```

En C: while

```
...
while(condition) {
    // Aquí sentències
}
...
```

Bucles infinits

- Si la condició de sortida és sempre certa, s'executarà infinitament.

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

int main(){
    while(true){
        printf("Soc un bucle infinit\n");
    }
    return 0;
}
```

Un bucle infinit "voluntari"

Bucle 'while'

A teoria: mentre

```
Inicialització variable control
mentre (condició) fer
    Cos del bucle
    Actualització variable control
fmentre
```

En C: while

```
...
while(condition) {
    // Aquí sentències
}
...
```

Bucles infinits

- Si la condició de sortida és sempre certa, s'executarà infinitament.

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

int main(){
    int i = 0;
    while (i < MAX){
        printf("Valor de i = %d\n", i);
    }
    return 0;
}
```

Un bucle infinit involuntari

Bucle 'do-while'

A teoria: fer mentre

```
Inicialització variable de control  
fer  
    Cos del bucle  
    Actualització variable de control  
mentre (condició)
```

- La única diferència és que el cos del bucle s'executarà com a mínim una vegada (si o sí), independentment de que la condició sigui certa o no.

En C: do-while

```
do {  
    // Sentències  
} while ( condició );
```

Cas paradigmàtic

```
#include <stdio.h>  
  
int main(){  
  
    int n;  
  
    do {  
        printf("---- TRIA UNA OPCIO -----\n");  
        printf("[1] Sumar\n");  
        printf("[2] Restar\n");  
        printf("[3] Multiplicar\n");  
        printf("[0] SORTIR\n");  
        scanf("%d",&n);  
        printf("Has triat %d\n",n);  
    } while(n != 0);  
  
    return 0;  
}
```


A CASA...

EXERCICIS L3 del Moodle

EL PROPER DIA...

PROCEDIMENTS