


PROGRAMES MULTIARXIU

Recordem...

- Recordeu que vam parlar que les funcions s'havien de declarar abans de fer-les servir?



```
#include <stdio.h>

int suma(int a, int b){
    return (a+b);
}

int main(){
    int r;
    r = suma(10,5);
    printf("El resultat es: %d\n",r);
    return 0;
}
```

Així és com hem estat escrivint els nostres codis fins ara

Recordem...

- Recordeu també que vam dir que si les volíem definir en algun altre lloc primer les havíem de declarar?

Declaració d'una funció

```
int suma(int a, int b);
```

- Declarar una funció equival a dir-li al compilador “existirà una funció que tindrà aquest nom, retornarà cert tipus”. També podem dir-li de quin tipus seran els seus paràmetres.
- Amb aquesta informació incompleta el compilador pot seguir compilant el programa encara que no conegui què fa la funció.

Recordem...

- Recordeu també que vam dir que si les volíem definir en algun altre lloc primer les havíem de declarar?

Declaració d'una funció

```
int suma(int a, int b);
```

- Declarar una funció equival a dir-li al compilador “existirà una funció que tindrà aquest nom, retornarà cert tipus”. També podem dir-li de quin tipus seran els seus paràmetres.
- Amb aquesta informació incompleta el compilador pot seguir compilant el programa encara que no conegui què fa la funció.

Definició d'una funció

```
int suma(int a, int b){  
    return (a+b);  
}
```

- Per definir una funció necessitem escriure'n tot el cos i proporcionar tota la informació necessària perquè la funció funcioni.
- Si només definim una funció, també compta com una declaració, però es pot declarar i definir per separat.

Tipus de declaracions

- N'hi ha de dos tipus:
paramètriques i no
paramètriques

**Paramètriques
(Prototype)**

**Declaració
d'una funció**



- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

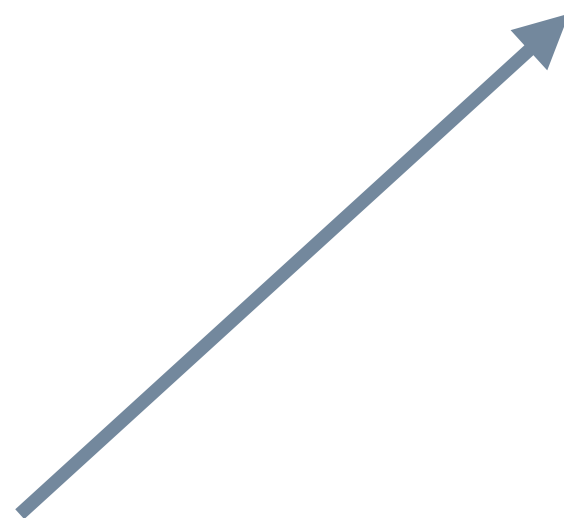
```
int suma(int a, int b);
```

Tipus de declaracions

- N'hi ha de dos tipus:
paramètriques i no
paramètriques

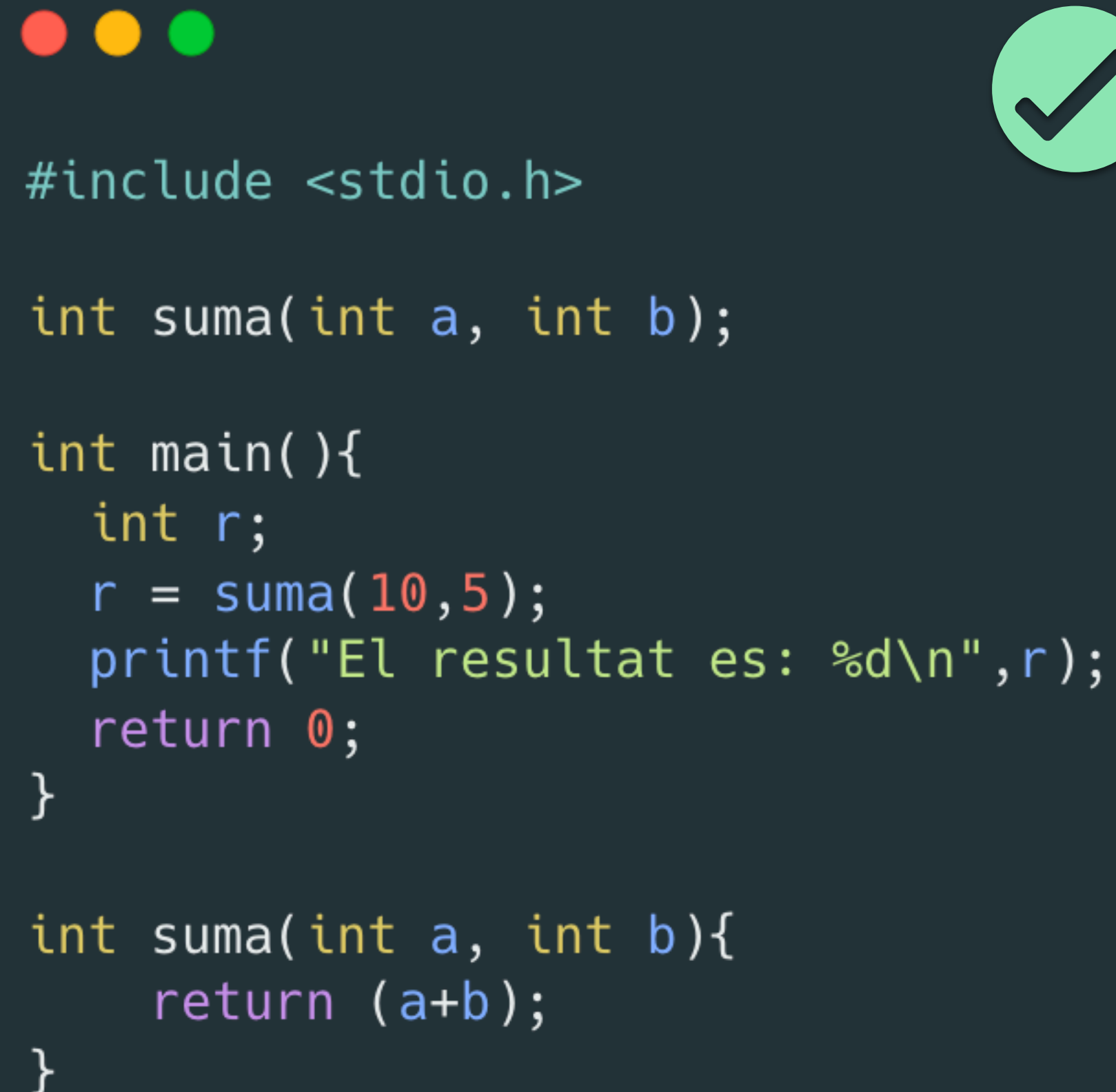
**Declaració
d'una funció**

**Paramètriques
(Prototype)**



- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int suma(int a, int b);
```



```
#include <stdio.h>

int suma(int a, int b);

int main(){
    int r;
    r = suma(10,5);
    printf("El resultat es: %d\n",r);
    return 0;
}


int suma(int a, int b){
    return (a+b);
}
```

Tipus de declaracions

Declaració
d'una funció

Paramètriques
(Prototype)

```
main.c:12:5: error: conflicting types for 'suma'
    12 |   int suma(int a, int b, int c){
        |         ^~~~
main.c:3:5: note: previous declaration of 'suma' was here
     3 |   int suma(int a, int b);
        |         ^~~~
```



```
#include <stdio.h>

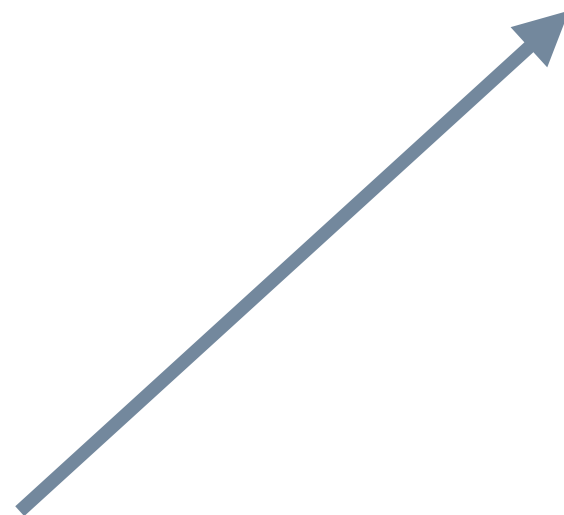
int suma(int a, int b);

int main(){
    int r;
    r = suma(10,5);
    printf("El resultat es: %d\n",r);
    return 0;
}

int suma(int a, int b, int c){
    return (a+b+c);
}
```

Tipus de declaracions

**Declaració
d'una funció**



**Paramètriques
(Prototype)**

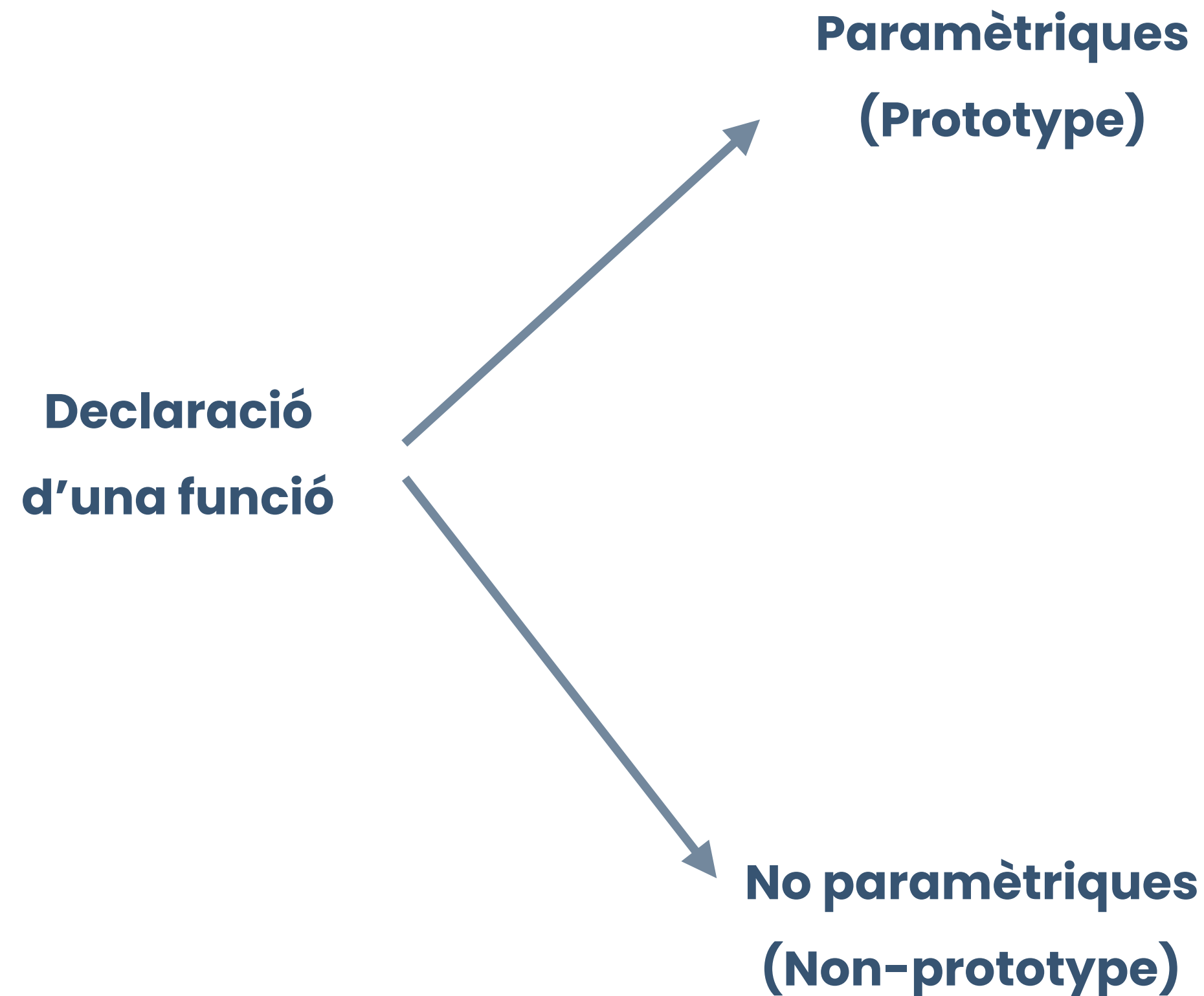
- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int suma(int a, int b);
```

- Del nom dels paràmetres el compilador no en fa res, de manera que ens els podem estalviar (i així estalviar-nos d'haver de recordar quins noms havíem fet servir per la declaració).

```
int suma(int, int);
```


Tipus de declaracions



- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int suma(int a, int b);
```

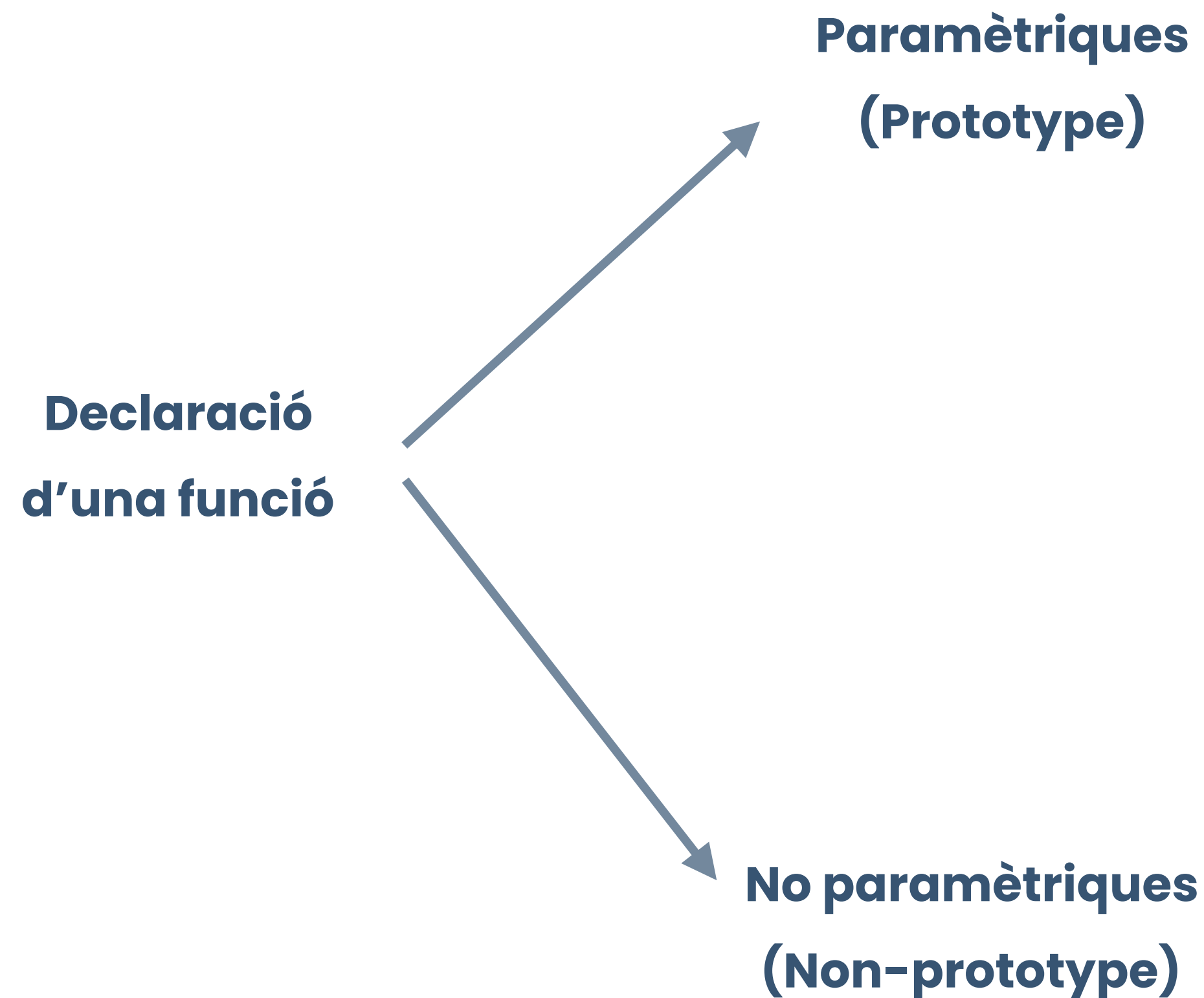
- Del nom dels paràmetres el compilador no en fa res, de manera que ens els podem estalviar (i així estalviar-nos d'haver de recordar quins noms havíem fet servir per la declaració).

```
int suma(int, int);
```

- També podem no dir-li res dels paràmetres: s'admeten declaracions sense aquesta informació (llavors no es diuen prototips)

```
int suma( );
```

Tipus de declaracions



- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int suma(int a, int b);
```

- Del nom dels paràmetres el compilador no en fa res, de manera que ens els podem estalviar (i així estalviar-nos d'haver de recordar quins noms havíem fet servir per la declaració).

```
int suma(int, int);
```

- També podem no dir-li res dels paràmetres: s'admeten declaracions sense aquesta informació (llavors no es diuen prototips)

```
int suma();
```

Tot i que aquestes declaracions s'accepten, a partir de la versió ANSI C es van introduir les declaracions paramètriques, i tot i que les no-paramètriques encara funcionen, **es recomana que no es facin servir.**

Tipus de declaracions

- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int power();
```

No parameter list was permitted, so the compiler could not readily check that `power` was being called correctly. Indeed, since by default `power` would have been assumed to return an `int`, the entire declaration might well have been omitted.

The new syntax of function prototypes makes it much easier for a compiler to detect errors in the number of arguments or their types. The old style of declaration and definition still works in ANSI C, at least for a transition period, but we strongly recommend that you use the new form when you have a compiler that supports it.

*Extracte del llibre de Kernighan & Ritchie "The C Programming Language"
(Segona edició, de l'any 1988)*

Declaració
d'una funció

encara

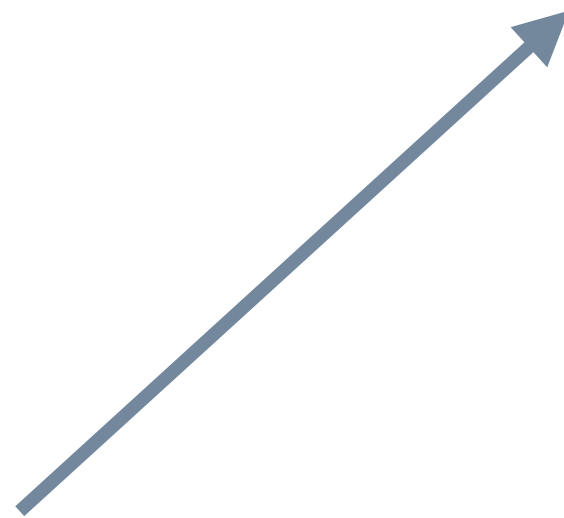
declaracions

versió ANSI C

es van introduir les declaracions paramètriques, i tot i que les no-paramètriques encara funcionen, **es recomana que no es facin servir.**

Tipus de declaracions

**Declaració
d'una funció**



**Paramètriques
(Prototype)**

- Donem informació dels paràmetres que rebrà: n'especifiquem el tipus i el nom.
- Si a l'hora de definir la funció els tipus no concorden, tindrem un error.

```
int suma(int a, int b);
```

- Del nom dels paràmetres el compilador no en fa res, de manera que ens els podem estalviar (i així estalviar-nos d'haver de recordar quins noms havíem fet servir per la declaració).

```
int suma(int, int);
```

Pregunta:

- I com escriuríem la declaració paramètrica d'una funció que no rep cap argument? Així?

```
int suma( );
```

- No, l'anterior seria una declaració no paramètrica. La manera correcta de fer-ho és així:

```
int suma(void);
```

Programes multiarxiu

Per què programes multiarxiu?

- Què passa si tenim moltes funcions d'un tema concret o bé una funció molt llarga: l'hem de definir obligatòriament al mateix arxiu? Ens quedaria un arxiu molt llarg! Seria genial poder definir les funcions en un arxiu i tenir el meu programa principal en un altre.
- Per altra banda, què passa si creo una funció que vull fer servir des de diversos programes?

Programes multiarxiu

Per què programes multiarxiu?

- Què passa si tenim moltes funcions d'un tema concret o bé una funció molt llarga: l'hem de definir obligatòriament al mateix arxiu? Ens quedaria un arxiu molt llarg! Seria genial poder definir les funcions en un arxiu i tenir el meu programa principal en un altre.
- Per altra banda, què passa si creo una funció que vull fer servir des de diversos programes?

```
int ll_string(char s[]){  
    ...  
}  
...  
l = ll_string(...);  
...
```

programa1.c

```
int ll_string(char s[]){  
    ...  
}  
...  
len = ll_string(...);  
...
```

programa2.c

Programes multiarxiu

Per què programes multiarxiu?

- Què passa si tenim moltes funcions d'un tema concret o bé una funció molt llarga: l'hem de definir obligatòriament al mateix arxiu? Ens quedaria un arxiu molt llarg! Seria genial poder definir les funcions en un arxiu i tenir el meu programa principal en un altre.
- Per altra banda, què passa si creo una funció que vull fer servir des de diversos programes?

```
int ll_string(char s[]){  
    ...  
}  
...  
l = ll_string(...);  
...
```

programa1.c

```
int ll_string(char s[]){  
    ...  
}  
...  
len = ll_string(...);  
...
```

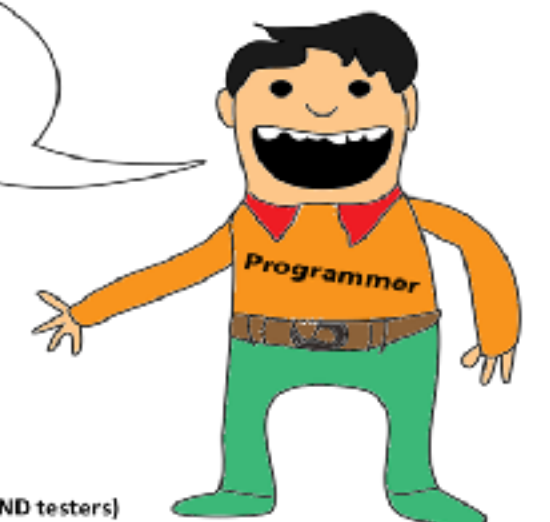
programa2.c

*Què faig? La copio als
dos programes? I si
llavors l'he
d'actualitzar, què?*

D.R.Y.

**Don't
Repeat
Yourself**

(Good advice for programmers AND testers)



Programes multiarxiu

Per què programes multiarxiu?

- Què passa si tenim moltes funcions d'un tema concret o bé una funció molt llarga: l'hem de definir obligatòriament al mateix arxiu? Ens quedaria un arxiu molt llarg! Seria genial poder definir les funcions en un arxiu i tenir el meu programa principal en un altre.
- Per altra banda, què passa si creo una funció que vull fer servir des de diversos programes?

```
...  
l = ll_string(...);  
...
```

programa1.c

```
...  
len = ll_string(...);  
...
```

programa2.c

```
int ll_string(char s[]){  
    ...  
}
```

funcions_strings.c

Així sí !!!



Programes multiarxiu

```
1 #include <stdio.h>
2
3 int main(){
4
5     int a = 10;
6     int b = 20;
7     printf("La suma es %d\n", suma(a,b));
8     printf("La resta es %d\n", resta(a,b));
9     printf("La mult es %d\n", multiplicacio(a,b));
10    printf("La divi es %d\n", divisio(a,b));
11 }
```

principal.c

El programa principal fa servir les funcions suma, resta, multiplicació i divisió però la implementació està en algun altre lloc

```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Implementació de les funcions anteriors

Programes multiarxiu

- Amb els dos programes per separat, el programa principal.c no coneix les definicions de les funcions suma, resta, multiplicació i divisió, perquè estan a un altre arxiu. Com els vinculem?

```
1 #include <stdio.h>
2
3 int main(){
4
5     int a = 10;
6     int b = 20;
7     printf("La suma es %d\n", suma(a,b));
8     printf("La resta es %d\n", resta(a,b));
9     printf("La mult es %d\n", multiplicacio(a,b));
10    printf("La divi es %d\n", divisio(a,b));
11 }
```

principal.c

El programa principal fa servir les funcions suma, resta, multiplicació i divisió però la implementació està en algun altre lloc



```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Implementació de les funcions anteriors

Programes multiarxiu

- Per vincular els dos programes, necessitem:
 - **Afegir al programa principal la capçalera de les funcions externes**

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

```
int suma(int a, int b);  
int resta(int a, int b);  
int multiplicacio(int a, int b);  
int divisio(int a, int b);
```

```
int main(){
```

```
    int a = 10;  
    int b = 20;
```

```
    printf("La suma es %d\n", suma(a,b));  
    printf("La resta es %d\n", resta(a,b));  
    printf("La mult es %d\n", multiplicacio(a,b));  
    printf("La divi es %d\n", divisio(a,b));
```

```
}
```

principal.c

```
1 int suma(int a, int b){  
2     return a+b;  
3 }  
4  
5 int resta(int a, int b){  
6     return a-b;  
7 }  
8  
9 int multiplicacio(int a, int b){  
10    return a*b;  
11 }  
12  
13 int divisio(int a, int b){  
14    return a/b;  
15 }
```

funcions_matematiques.c

Programes multiarxiu

- Per vincular els dos programes, necessitem:
 - Afegir al programa principal la capçalera de les funcions externes
 - **Compilar els dos programes junts**

○ ○ ○

```
gcc principal.c funcions_matematiques.c -o principal
```

Els dos codis han de ser a la mateixa carpeta

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

```
int suma(int a, int b);  
int resta(int a, int b);  
int multiplicacio(int a, int b);  
int divisio(int a, int b);
```

```
int main(){
```

```
    int a = 10;  
    int b = 20;
```

```
    printf("La suma es %d\n", suma(a,b));  
    printf("La resta es %d\n", resta(a,b));  
    printf("La mult es %d\n", multiplicacio(a,b));  
    printf("La divi es %d\n", divisio(a,b));
```

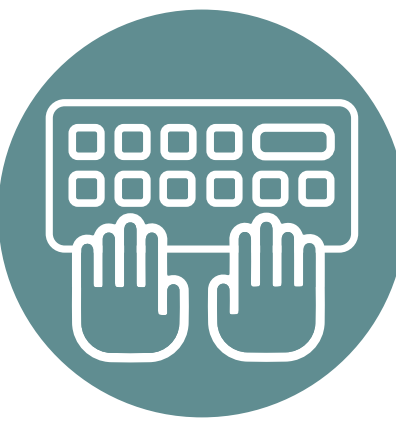
```
}
```

principal.c

```
1 int suma(int a, int b){  
2     return a+b;  
3 }  
4  
5 int resta(int a, int b){  
6     return a-b;  
7 }  
8  
9 int multiplicacio(int a, int b){  
10    return a*b;  
11 }  
12  
13 int divisio(int a, int b){  
14    return a/b;  
15 }
```

funcions_matematiques.c

Programes multiarxiu



```
gcc principal.c funciones_matematicas.c -o principal
```

Els dos codis han de ser a la mateixa carpeta



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

```
int suma(int a, int b);  
int resta(int a, int b);  
int multiplicacio(int a, int b);  
int divisio(int a, int b);
```

```
int main(){
```

```
    int a = 10;  
    int b = 20;
```

```
    printf("La suma es %d\n", suma(a,b));  
    printf("La resta es %d\n", resta(a,b));  
    printf("La mult es %d\n", multiplicacio(a,b));  
    printf("La divi es %d\n", divisio(a,b));
```

```
}
```

principal.c



```
1 int suma(int a, int b){  
2     return a+b;  
3 }  
4  
5 int resta(int a, int b){  
6     return a-b;  
7 }  
8  
9 int multiplicacio(int a, int b){  
10    return a*b;  
11 }  
12  
13 int divisio(int a, int b){  
14    return a/b;  
15 }
```

funciones_matematicas.c

- **Compileu i executeu** el programa i comproveu que sense capçaleres no funciona. (Excepte en estàndard ANSI C)
- **Modifiqueu** el programa principal.c perquè inclogui les declaracions de les funcions externes.
- **Compileu i executeu** el programa i comproveu que funciona

Programes multiarxiu

- Si hem de fer servir aquestes funcions des d'altres arxius, haurem d'escriure aquestes capçaleres a tots els arxius.
- Però això suposa una repetició de codi, i per tant, una font d'errors.
- Com podem estalviar-nos repetir l'escriptura de les capçaleres?

```
#include <stdio.h>

int suma(int a, int b);
int resta(int a, int b);
int multiplicacio(int a, int b);
int divisio(int a, int b);

int main(){

    int a = 10;
    int b = 20;

    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));
}
```

principal.c

```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Programes multiarxiu

Fitxers header

- Els fitxers de capçalera inclouen les declaracions de funcions i de constants que es fan servir a més d'un arxiu.

```
1 // Declaració de les funcions
2
3
4
5
```

funcions_matematiques.h

```
#include <stdio.h>

int suma(int a, int b);
int resta(int a, int b);
int multiplicacio(int a, int b);
int divisio(int a, int b);

int main(){

    int a = 10;
    int b = 20;

    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));
}
```

principal.c

```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Programes multiarxiu

Fitxers header

- Els fitxers de capçalera inclouen les declaracions de funcions i de constants que es fan servir a més d'un arxiu.

```
1 // Declaració de les funcions
2 int suma(int a, int b);
3 int resta(int a, int b);
4 int multiplicacio(int a, int b);
5 int divisio(int a, int b);
```

funcions_matematiques.h

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

```
int main(){
```

```
    int a = 10;
    int b = 20;
```

```
    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));
```

```
}
```

principal.c

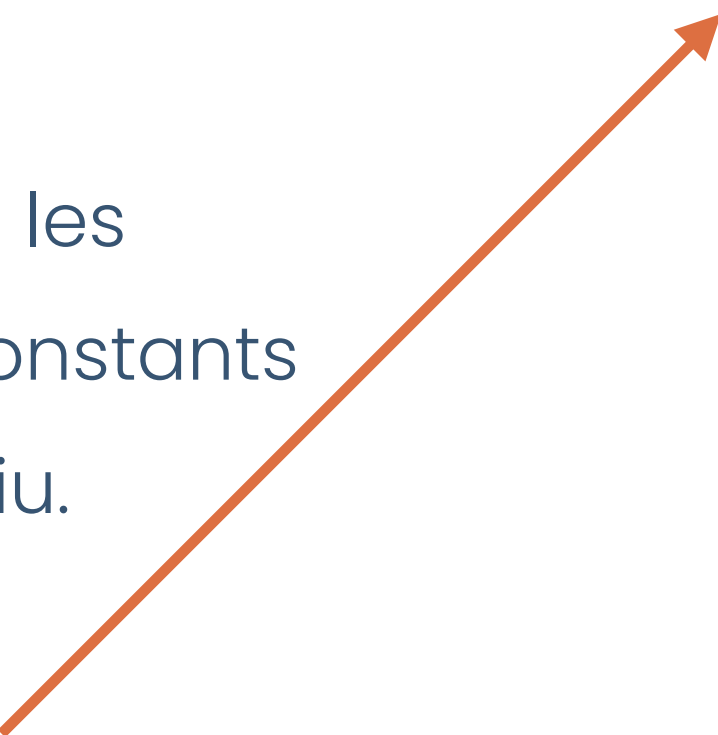
```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Programes multiarxiu

Fitxers header

- Els fitxers de capçalera inclouen les declaracions de funcions i de constants que es fan servir a més d'un arxiu.



```
1 // Declaració de les funcions
2 int suma(int a, int b);
3 int resta(int a, int b);
4 int multiplicacio(int a, int b);
5 int divisio(int a, int b);
```

funcions_matematiques.h

```
#include <stdio.h>
#include "funcions_matematiques.h"

int main(){

    int a = 10;
    int b = 20;

    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));
}
```

principal.c

```
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Programes multiarxiu

Fitxers header

- Els fitxers de capçalera inclouen les declaracions de funcions i de constants que es fan servir a més d'un arxiu.

```
1 // Declaració de les funcions
2 int suma(int a, int b);
3 int resta(int a, int b);
4 int multiplicacio(int a, int b);
5 int divisio(int a, int b);
```

funcions_matematiques.h

```
#include <stdio.h>
#include "funcions_matematiques.h"
```

```
int main(){

    int a = 10;
    int b = 20;

    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));

}
```

principal.c

Amb la comanda "include"
estem copiant els
continguts íntegres de
funcions_matemàtiques.h
als arxius
principal.c i
funcions_matemàtiques.c

Fem servir cometes "" i no <>
perquè ens estem referint a
un arxiu que no forma part
de la llibreria de C.

```
#include "funcions_matematiques.h"
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

Programes multiarxiu

Fitxers header

- Els fitxers de capçalera inclouen les declaracions de funcions i de constants que es fan servir a més d'un arxiu.

- Al fitxer de capçalera també hi inclourem les **definicions de les constants** que fem servir als diversos arxius, per evitar repeticions.
- També hi inclourem els tipus (typedef) que haguem de compartir.

```
#define PI 3.14
1 // Declaració de les funcions
2 int suma(int a, int b);
3 int resta(int a, int b);
4 int multiplicacio(int a, int b);
5 int divisio(int a, int b);
```

funcions_matematiques.h

```
#include <stdio.h>
#include "funcions_matematiques.h"

int main(){

    int a = 10;
    int b = 20;

    printf("La suma es %d\n", suma(a,b));
    printf("La resta es %d\n", resta(a,b));
    printf("La mult es %d\n", multiplicacio(a,b));
    printf("La divi es %d\n", divisio(a,b));
}
```

principal.c

```
#include "funcions_matematiques.h"
1 int suma(int a, int b){
2     return a+b;
3 }
4
5 int resta(int a, int b){
6     return a-b;
7 }
8
9 int multiplicacio(int a, int b){
10    return a*b;
11 }
12
13 int divisio(int a, int b){
14    return a/b;
15 }
```

funcions_matematiques.c

On posar els #includes?

- Els includes es poden posar tant en el codi font (llibreria_escriure.c) com al fitxer de capçalera (llibreria_escriure.h).
- Funciona en els dos casos.
- Convenció: posar-lo on es faci servir.

principal.c

```

#include "llibreria_escriure.h"
/* Programa principal. Aquí no he de fer servir res
de cap llibreria estàndard, només de la meva*/

int main(){

    escriure("Hola, què tal?");
    return 0;
}
```

llibreria_escriure.c

```

#include <stdio.h>
void escriure(char * missatge){
    printf("%s\n", missatge);
}
```

L'incloc aquí perquè és aquí on faig servir la funció "printf" de la llibreria "stdio"

llibreria_escriure.h

```

/* Funció que rep per paràmetre un missatge
(taula de caràcters) i l'escriu per la sortida
estàndard (pantalla)*/
void escriure(char * missatge);
```