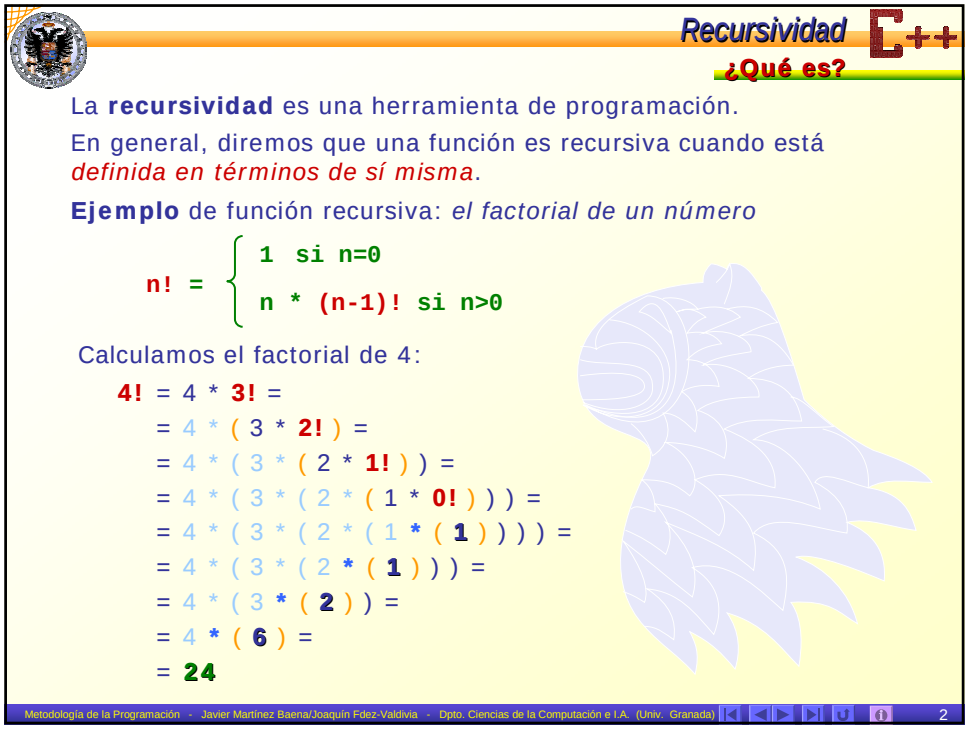
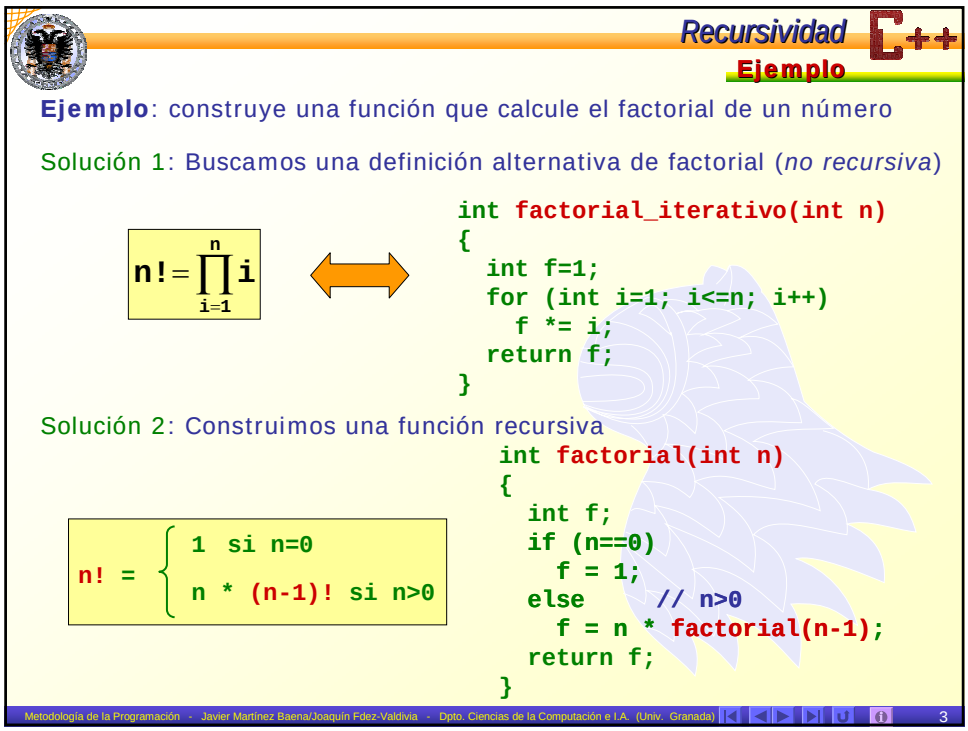
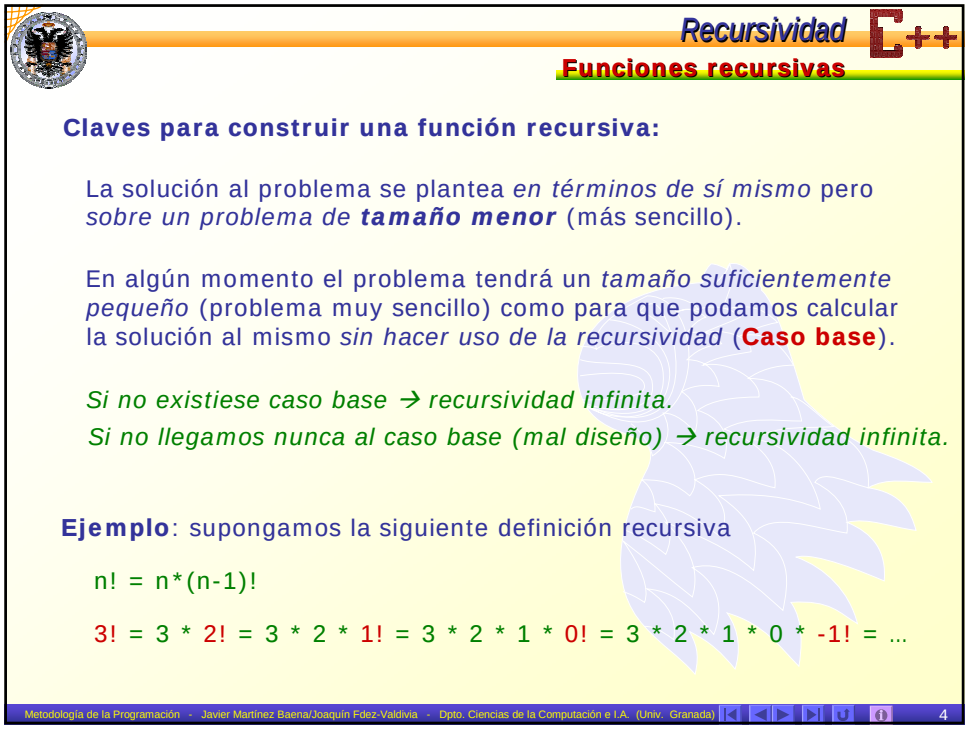
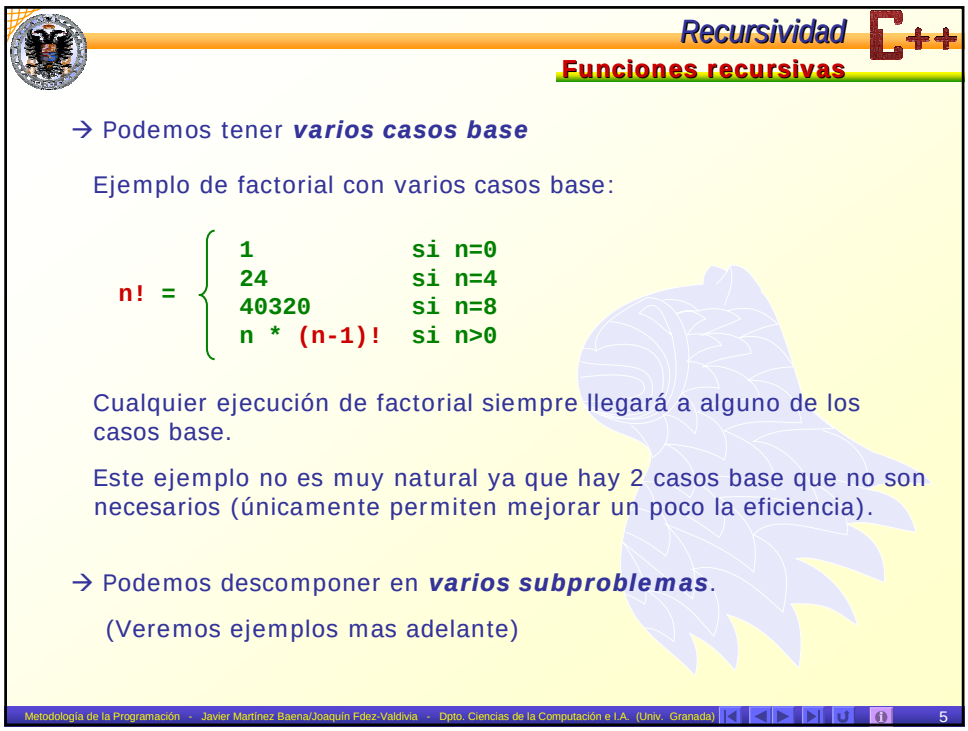
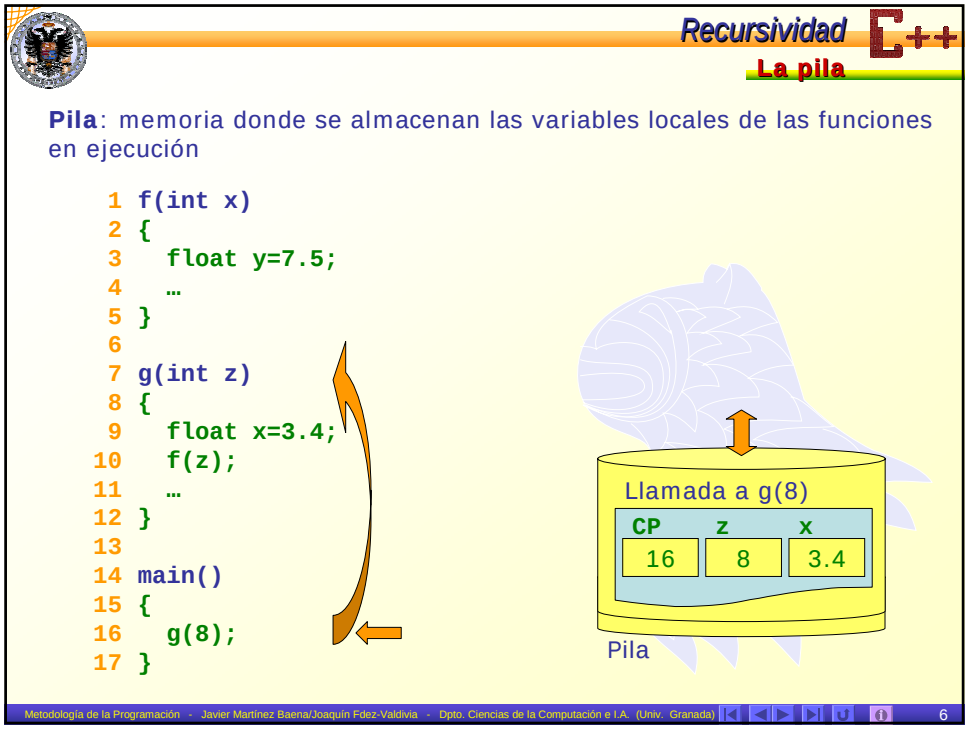

Metodología de la Programación
Tema 0.- Recursividad
¿Qué es?
Funciones recursivas
La pila
La pila y la recursividad
ugr Universidad de Granada
DECSAI Universidad de Granada
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 1

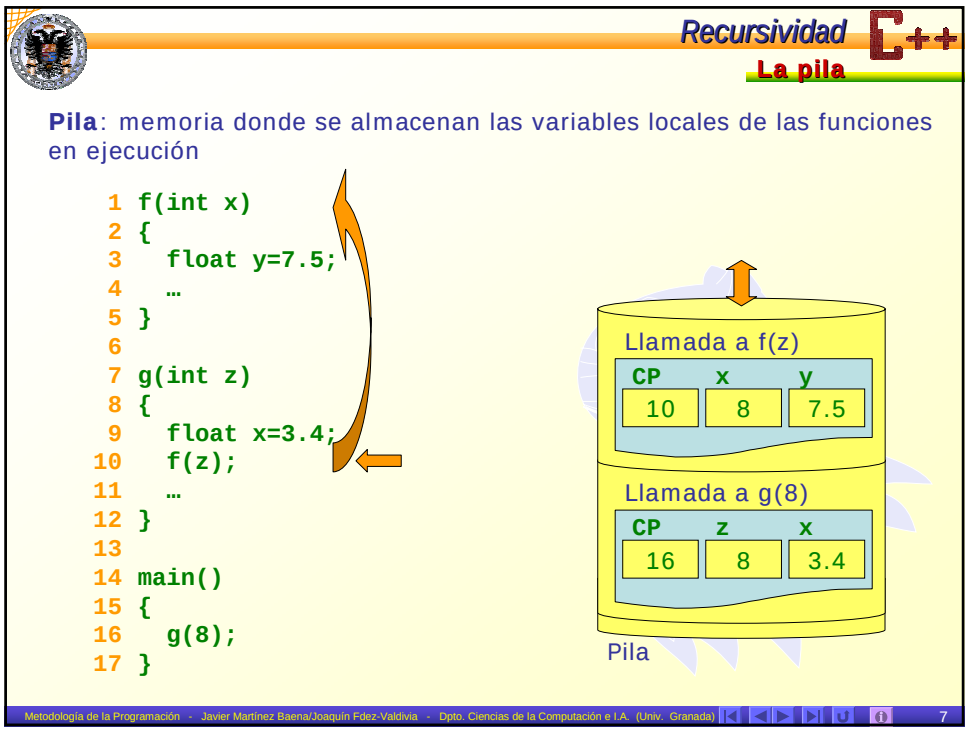

Recursividad
¿Qué es?
La recursividad es una herramienta de programación.
En general, diremos que una función es recursiva cuando está definida en términos de sí misma.
Ejemplo de función recursiva: el factorial de un número
n! = { 1 si n=0 ; n * (n-1)! si n>0
Calculamos el factorial de 4:
4! = 4 * 3! =
= 4 * (3 * 2!) =
= 4 * (3 * (2 * 1!)) =
= 4 * (3 * (2 * (1 * 0!))) =
= 4 * (3 * (2 * (1 * (1)))) =
= 4 * (3 * (2 * (1))) =
= 4 * (3 * (2)) =
= 4 * (6) =
= 24
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 2

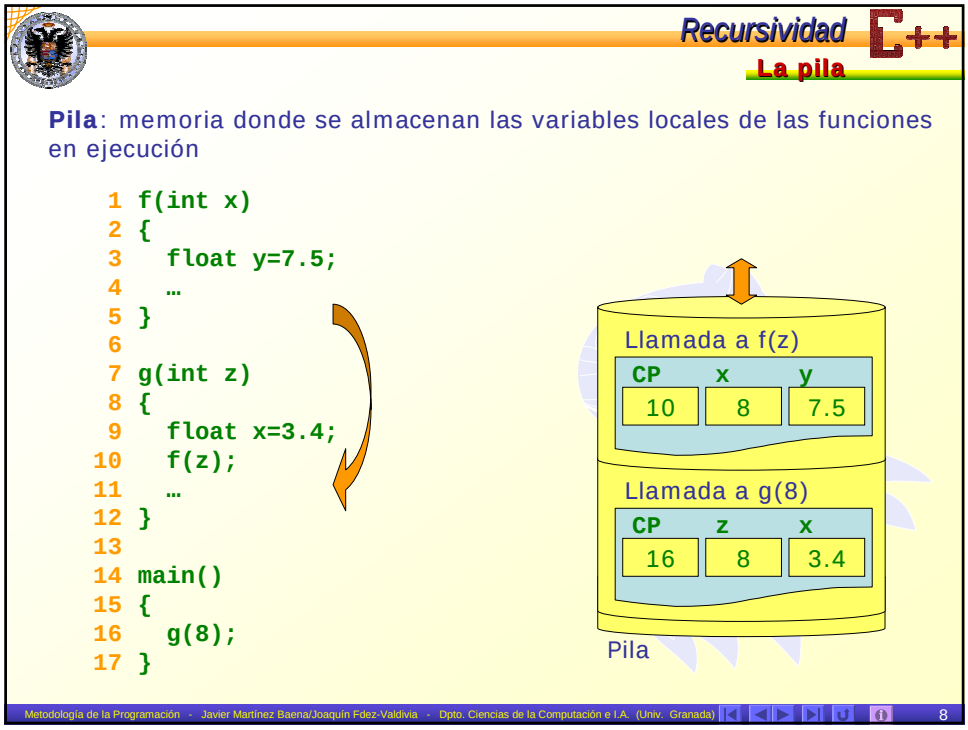

Recursividad
Ejemplo
Ejemplo: construye una función que calcule el factorial de un número
Solución 1: Buscamos una definición alternativa de factorial (no recursiva)
$n! = \prod_{i=1}^{n} i$
int factorial_iterativo(int n)
{
int f=1;
for (int i=1; i<=n; i++)
f *= i;
return f;
}
Solución 2: Construimos una función recursiva
n! = { 1 si n=0 ; n * (n-1)! si n>0
int factorial(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f = n * factorial(n-1);
return f;
}
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
3


Recursividad
Funciones recursivas
Claves para construir una función recursiva:
La solución al problema se plantea en términos de sí mismo pero sobre un problema de tamaño menor (más sencillo).
En algún momento el problema tendrá un tamaño suficientemente pequeño (problema muy sencillo) como para que podamos calcular la solución al mismo sin hacer uso de la recursividad (Caso base).
Si no existiese caso base → recursividad infinita.
Si no llegamos nunca al caso base (mal diseño) → recursividad infinita.
Ejemplo: supongamos la siguiente definición recursiva
n! = n*(n-1)!
3! = 3 * 2! = 3 * 2 * 1! = 3 * 2 * 1 * 0! = 3 * 2 * 1 * 0 * -1! = ...
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
4


Recursividad
Funciones recursivas
→ Podemos tener varios casos base
Ejemplo de factorial con varios casos base:
n! = { 1 si n=0; 24 si n=4; 40320 si n=8; n * (n-1)! si n>0
Cualquier ejecución de factorial siempre llegará a alguno de los casos base.
Este ejemplo no es muy natural ya que hay 2 casos base que no son necesarios (únicamente permiten mejorar un poco la eficiencia).
→ Podemos descomponer en varios subproblemas.
(Veremos ejemplos mas adelante)
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
5


Recursividad
La pila
Pila: memoria donde se almacenan las variables locales de las funciones en ejecución
1 f(int x)
2 {
3 float y=7.5;
4 …
5 }
6
7 g(int z)
8 {
9 float x=3.4;
10 f(z);
11 …
12 }
13
14 main()
15 {
16 g(8);
17 }
Llamada a g(8)
CP z x
16 8 3.4
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
6

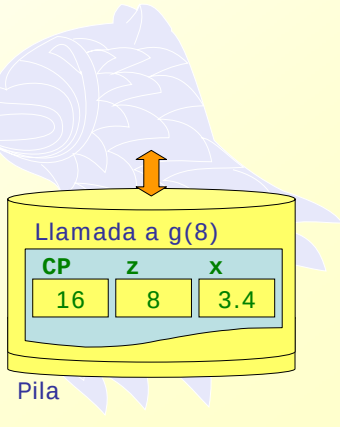

Recursividad
La pila
Pila: memoria donde se almacenan las variables locales de las funciones en ejecución
1 f(int x)
2 {
3 float y=7.5;
4 …
5 }
6
7 g(int z)
8 {
9 float x=3.4;
10 f(z);
11 …
12 }
13
14 main()
15 {
16 g(8);
17 }
Llamada a f(z)
CP x y
10 8 7.5
Llamada a g(8)
CP z x
16 8 3.4
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 7


Recursividad
La pila
Pila: memoria donde se almacenan las variables locales de las funciones en ejecución
1 f(int x)
2 {
3 float y=7.5;
4 …
5 }
6
7 g(int z)
8 {
9 float x=3.4;
10 f(z);
11 …
12 }
13
14 main()
15 {
16 g(8);
17 }
Llamada a f(z)
CP x y
10 8 7.5
Llamada a g(8)
CP z x
16 8 3.4
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 8

Recursividad
La pila

Pila: memoria donde se almacenan las variables locales de las funciones en ejecución

```
 1 f(int x)
 2 {
 3   float y=7.5;
 4   …
 5 }
 6
 7 g(int z)
 8 {
 9   float x=3.4;
10   f(z);
11   …
12 }
13
14 main()
15 {
16   g(8);
17 }
```



Recursividad
La pila y la recursividad

¿Qué ocurre con la pila en las llamadas recursivas?

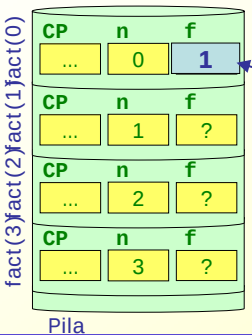
```
main()
{
  ...
  x=fact(3);
  ...
}
```

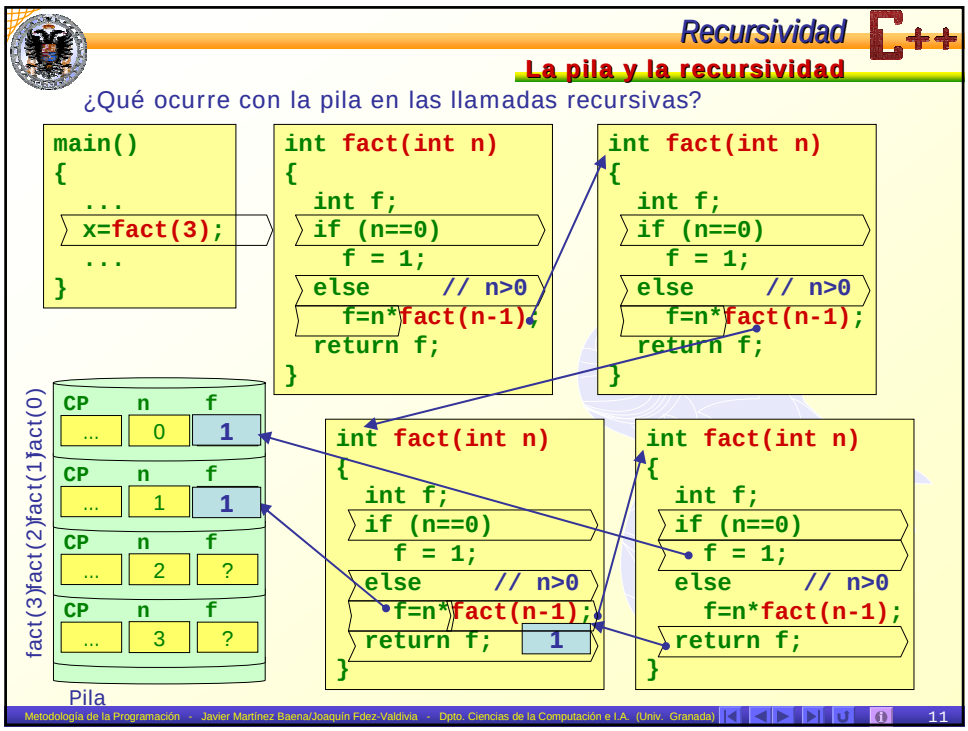
```
int fact(int n)
{
  int f;
  if (n==0)
    f = 1;
  else      // n>0
    f=n*fact(n-1);
  return f;
}
```

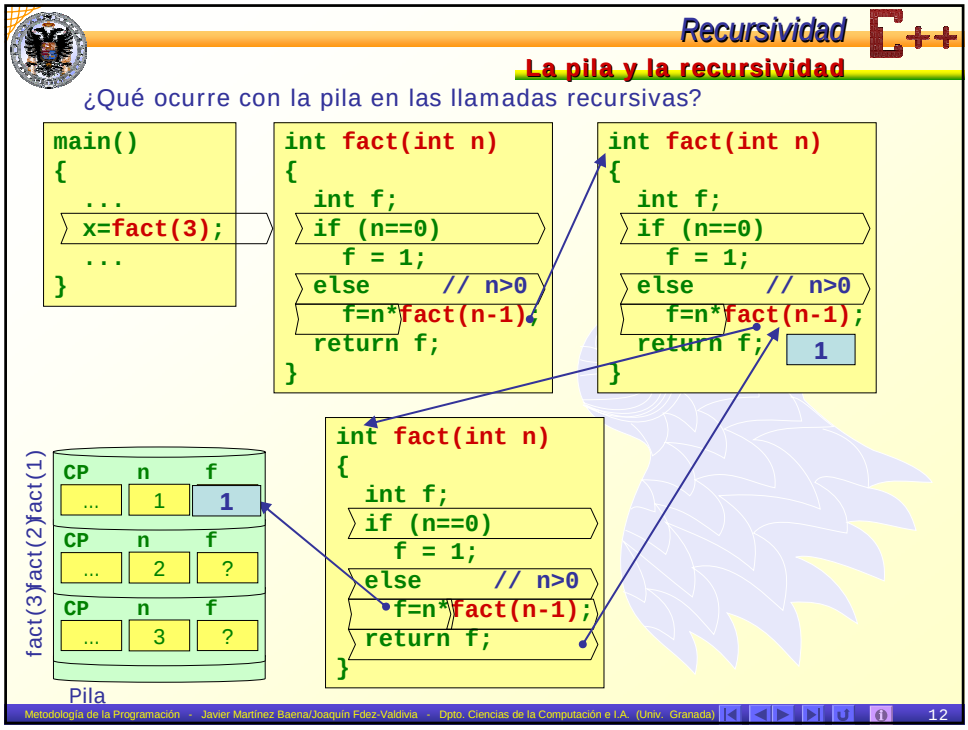
```
int fact(int n)
{
  int f;
  if (n==0)
    f = 1;
  else      // n>0
    f=n*fact(n-1);
  return f;
}
```

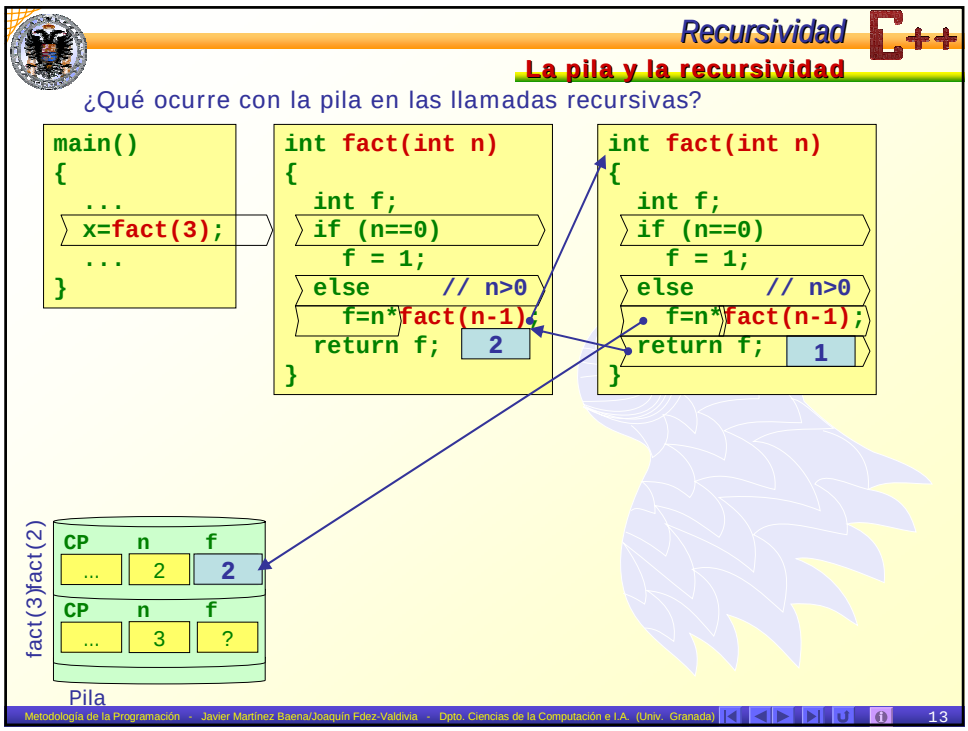
```
int fact(int n)
{
  int f;
  if (n==0)
    f = 1;
  else      // n>0
    f=n*fact(n-1);
  return f;
}
```

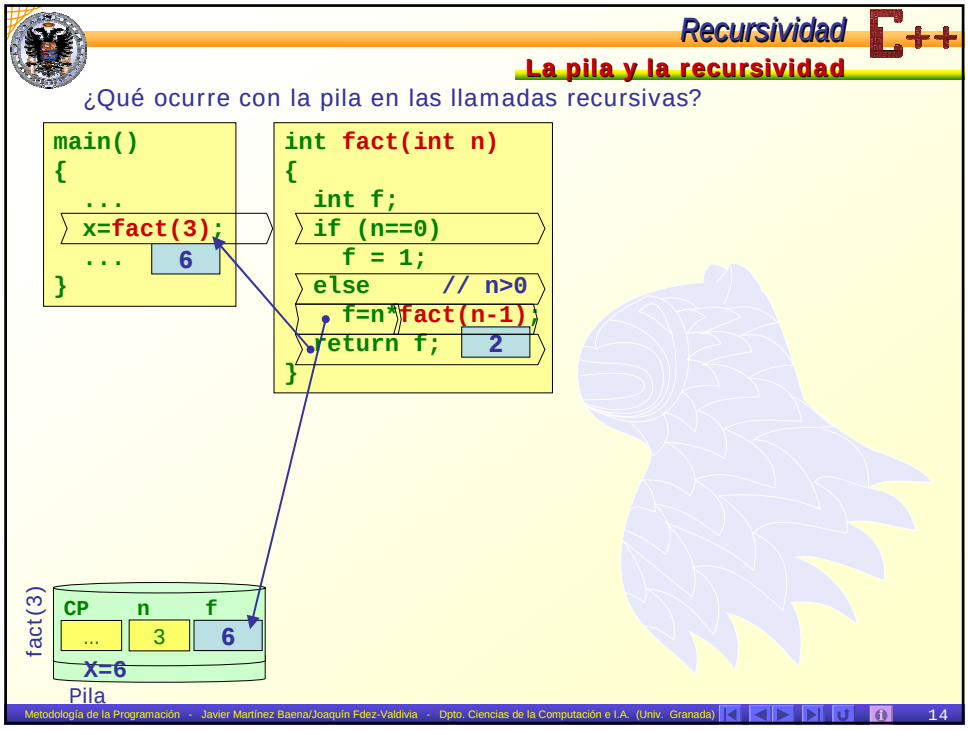
```
int fact(int n)
{
  int f;
  if (n==0)
    f = 1;
  else      // n>0
    f=n*fact(n-1);
  return f;
}
```




Recursividad
La pila y la recursividad
¿Qué ocurre con la pila en las llamadas recursivas?
main()
{
...
x=fact(3);
...
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f;
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f;
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f; 1
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f;
}
CP n f
... 0 1
CP n f
... 1 1
CP n f
... 2 ?
CP n f
... 3 ?
fact(3)fact(2)fact(1)fact(0)
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 11


Recursividad
La pila y la recursividad
¿Qué ocurre con la pila en las llamadas recursivas?
main()
{
...
x=fact(3);
...
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f;
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f; 1
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f;
}
CP n f
... 1 1
CP n f
... 2 ?
CP n f
... 3 ?
fact(3)fact(2)fact(1)
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada) 12


Recursividad
La pila y la recursividad
¿Qué ocurre con la pila en las llamadas recursivas?
main()
{
...
x=fact(3);
...
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f; 2
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f; 1
}
fact(3)fact(2)
CP n f
... 2 2
CP n f
... 3 ?
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
13


Recursividad
La pila y la recursividad
¿Qué ocurre con la pila en las llamadas recursivas?
main()
{
...
x=fact(3);
... 6
}
int fact(int n)
{
int f;
if (n==0)
f = 1;
else // n>0
f=n*fact(n-1);
return f; 2
}
fact(3)
CP n f
... 3 6
X=6
Pila
Metodología de la Programación - Javier Martínez Baena/Joaquín Fdez-Valdivia - Dpto. Ciencias de la Computación e I.A. (Univ. Granada)
14