

6. Subprogramas

Fundamentos de Informática

Especialidad de Electrónica – 2013-2014

Ismael Etxeberria Agiriano



Escuela Universitaria
de Ingeniería
Vitoria-Gasteiz

Ingeniaritzako
Unibertsitate Eskola
Vitoria-Gasteiz



Universidad
del País Vasco

eman ta zabal zazu
Euskal Herriko
Unibertsitatea

Índice

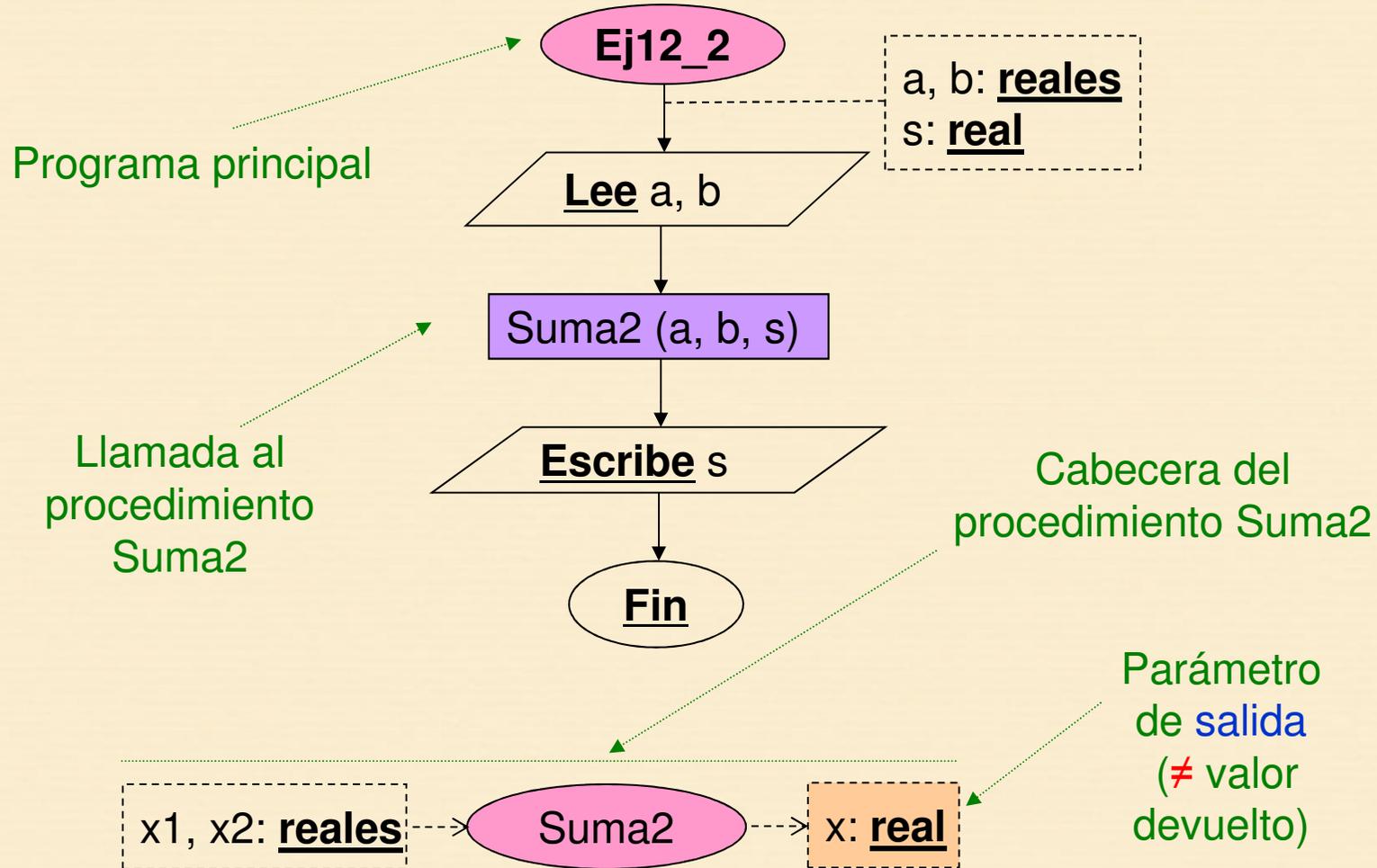
6. Subprogramas

1. Ej12: Suma
2. Ej13: Coseno
3. Ej14: Ecuación 2º grado

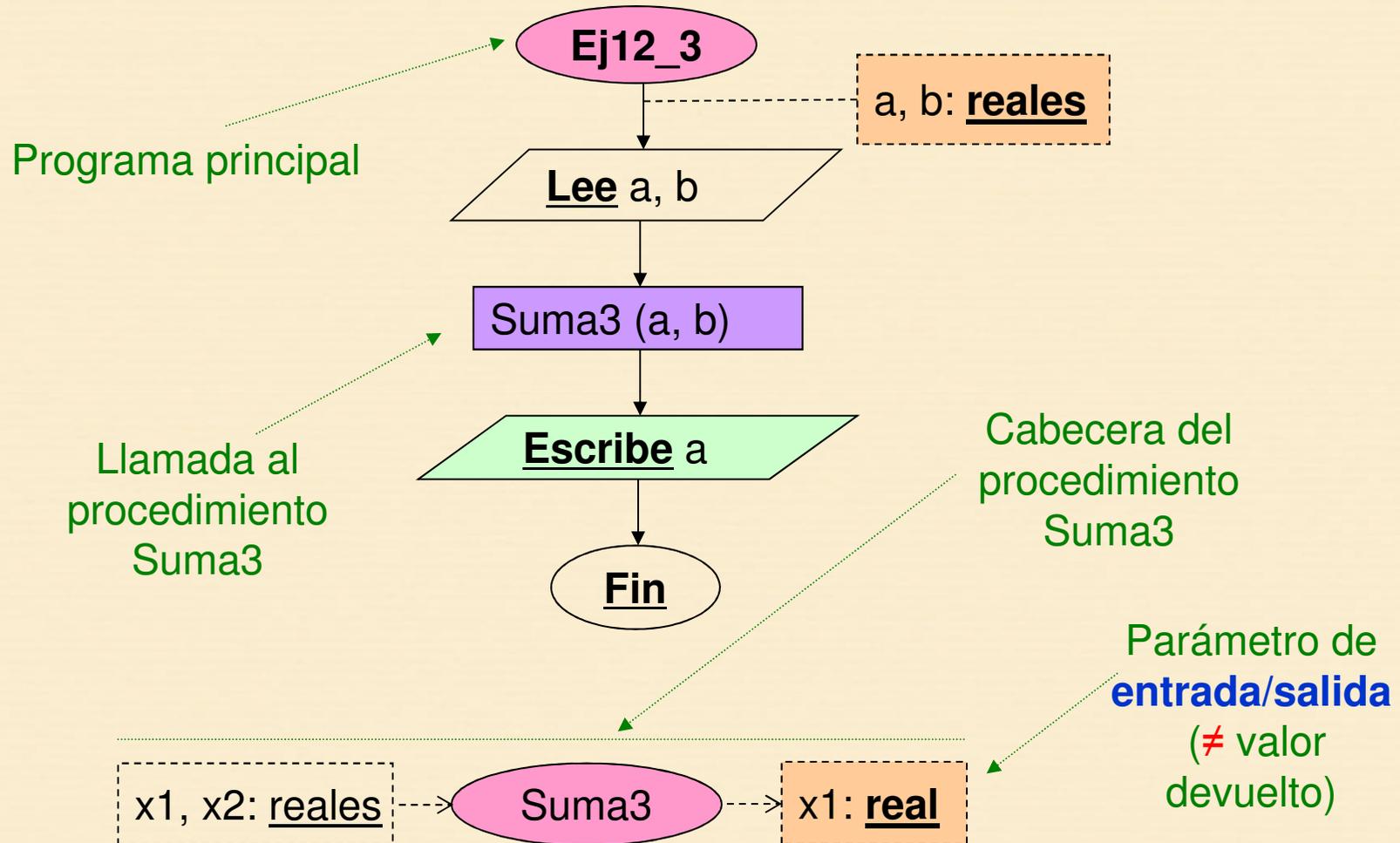
1. Ejemplo 12

- **Título:**
 - Subprograma de suma
- **Nombre**
 - Ej12
- **Descripción**
 - Escribir un **subprograma** que calcule la suma de dos números
 - V1: **función** con dos parámetros de entrada y devuelve el resultado
 - V2: **procedimiento** con dos parámetros de entrada y uno de salida
 - V3: **procedimiento** con un parámetro de entrada y otro de entrada/salida
- **Observaciones**
 - Ejemplo ilustrativo con cuerpo de cálculo evidente
 - Paso de parámetros
 - Los procedimientos no “devuelven” nada (usan parámetros de salida)

Ej12-2: Diagramas de Flujo (v2: procedimiento)



Ej12-3: Diagramas de Flujo (v3: procedimiento)



Ej12: Cabeceras y prototiposx1, x2: reales

Suma1

real

```
double Suma1 (double x1, double x2);
```

Función que
devuelve
el resultado

x1, x2: reales

Suma2

x: real

```
void Suma2 (double x1, double x2,  
            double *x);
```

Procedimiento con
parámetro de
salida, x

x1, x2: reales

Suma3

x1: real

```
void Suma3 (double *x1, double x2);
```

Procedimiento con
parámetro de
entrada/salida, x1



Ej12: Cabeceras, prototipos y llamadasx1, x2: reales

Suma1

real

s ← Suma1 (a, b)

`double Suma1 (double x1, double x2);``s = Suma1 (5, 7);`x1, x2: reales

Suma2

x: real

Suma2 (a, b, s)

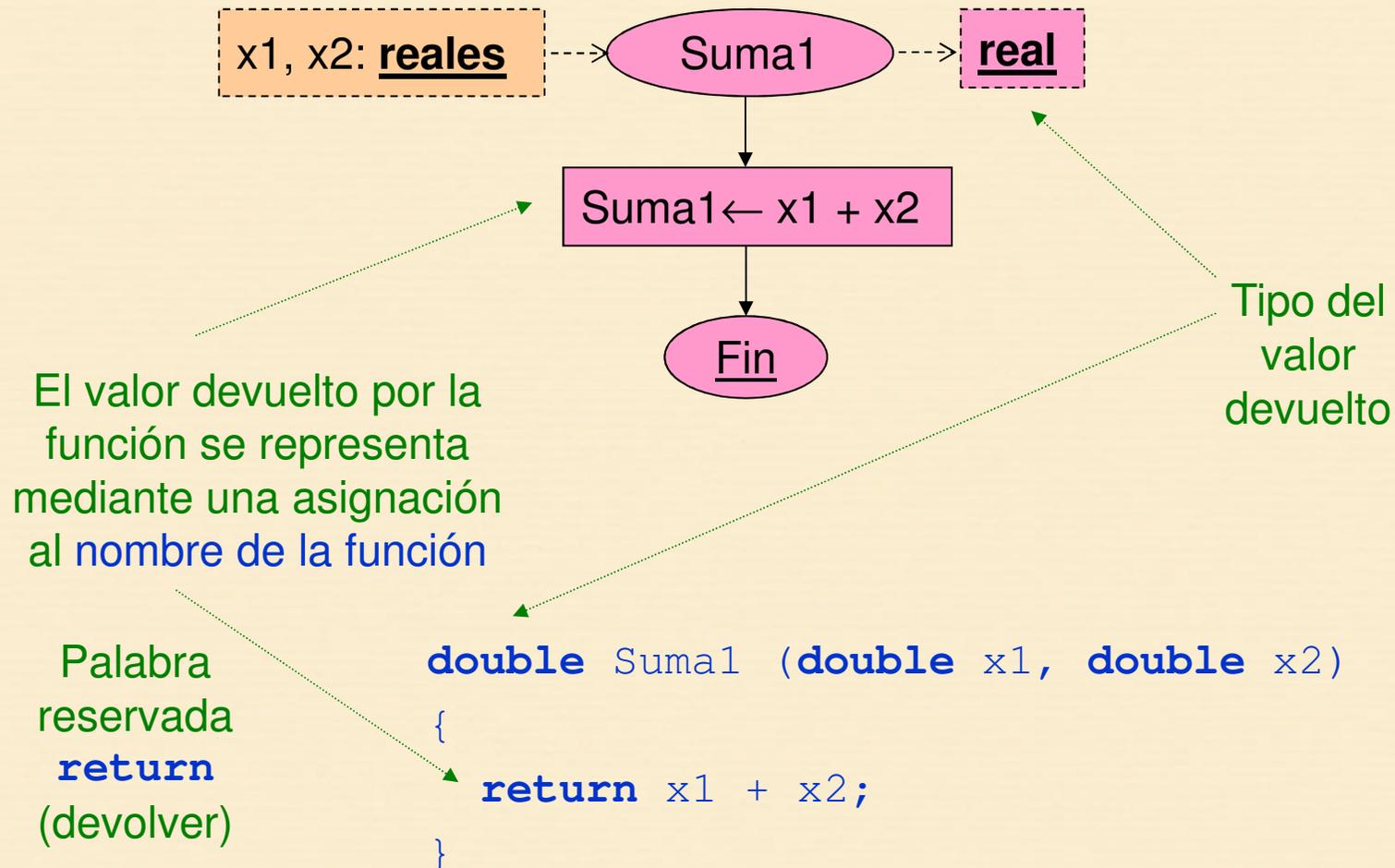
`void Suma2 (double x1, double x2,
double *x);``Suma2 (5, 7, &s);`x1, x2: reales

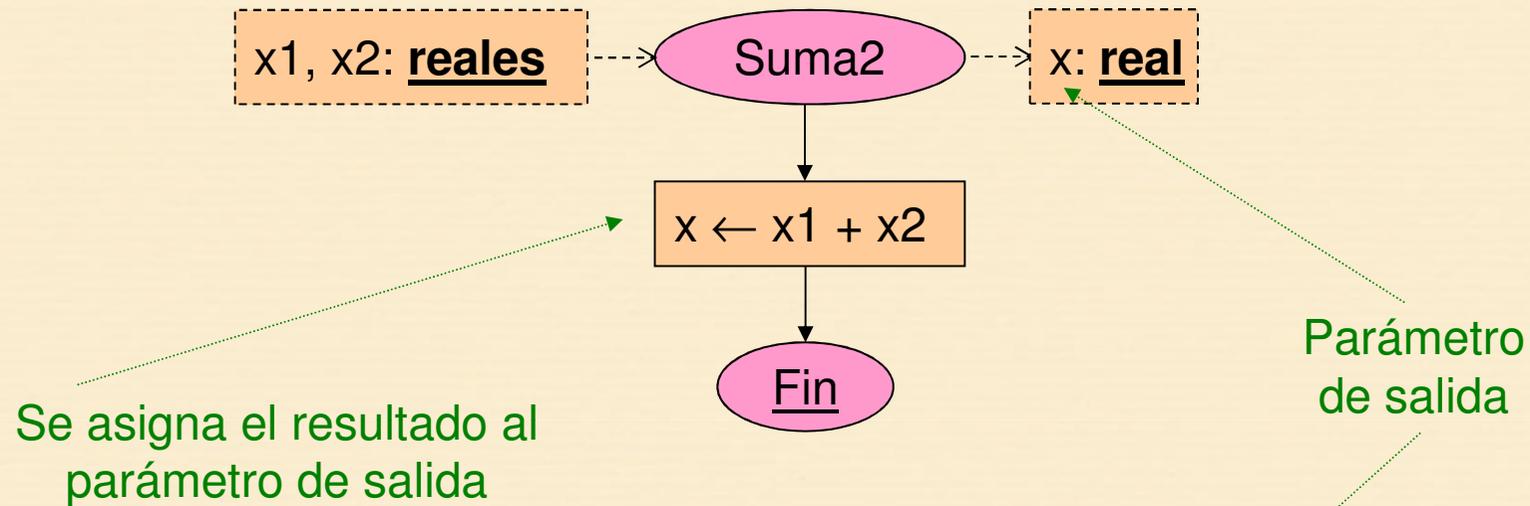
Suma3

x1: real

Suma3 (a, b)

`void Suma3 (double *x1, double x2);``Suma3 (&a, 7);`

Ej12: DdF y codificación función

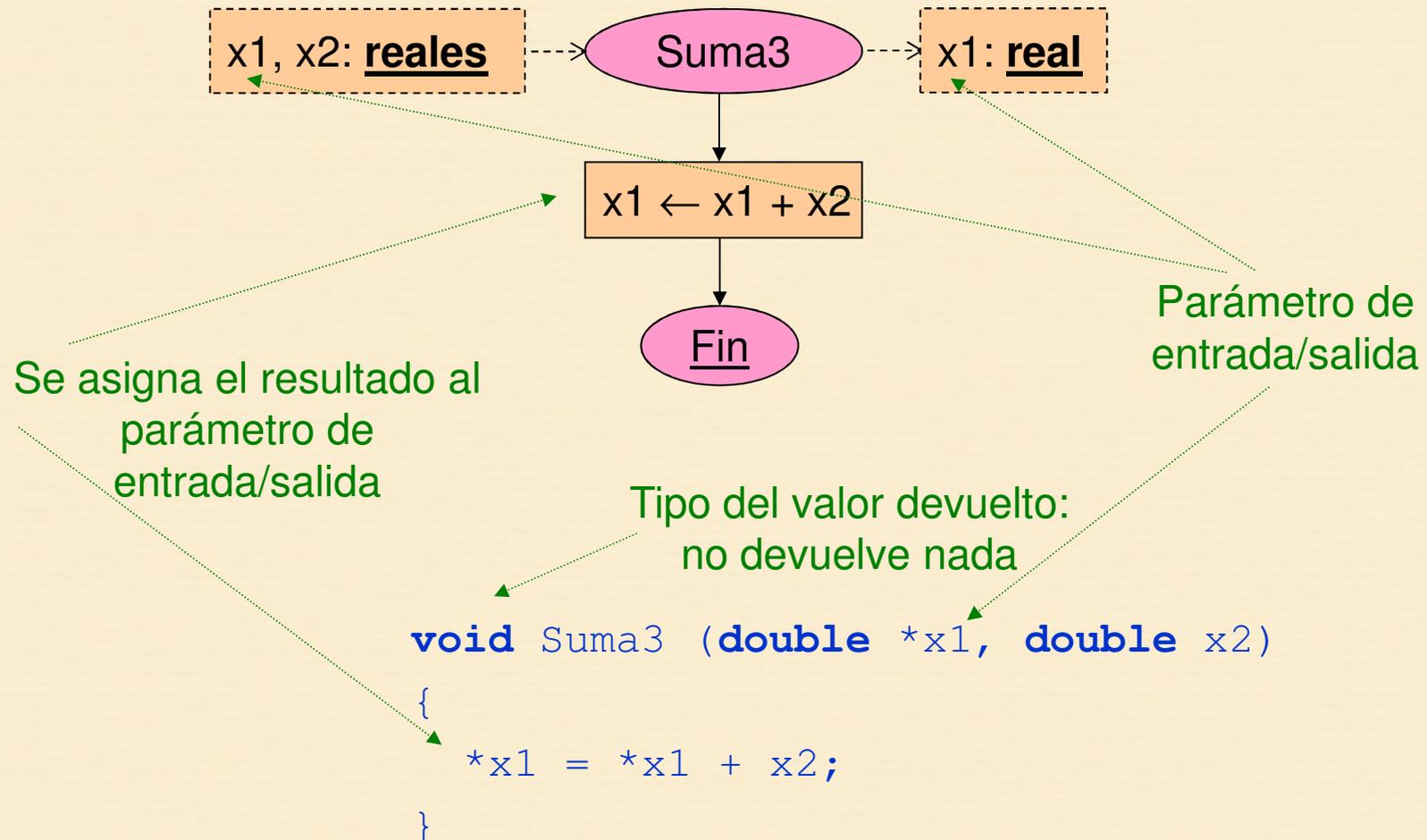
Ej12: Procedimiento con parámetro de salida

Tipo del valor devuelto:
no devuelve nada

```
void Suma2 (double x1, double x2,
            double *x)
```

```
{
    *x = x1 + x2;
}
```



Ej12: Procedimiento con parámetro de entrada/salida

Ej12: Comparación

```
double Suma1 (double x1, double x2)
```

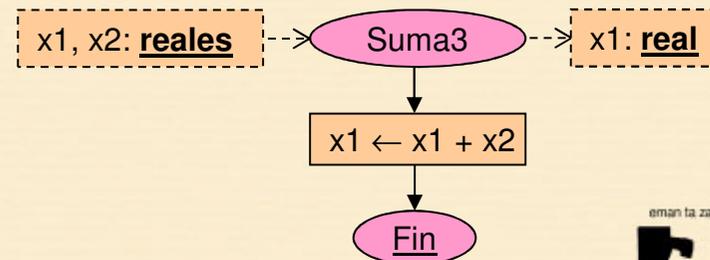
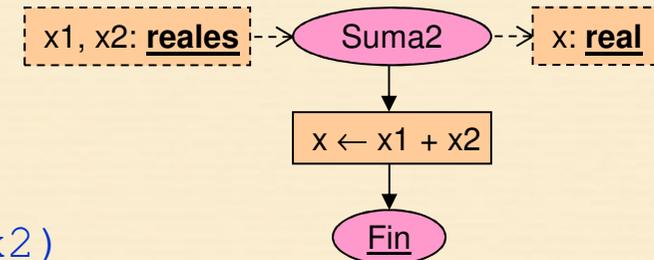
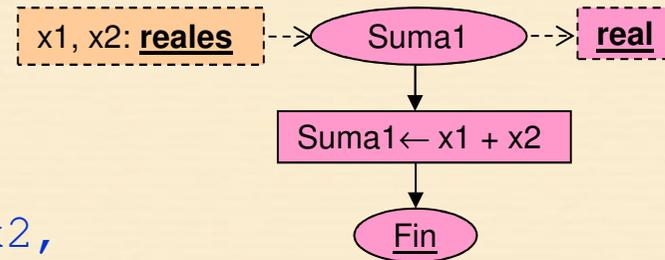
```
{
    return x1 + x2;
}
```

```
void Suma2 (double x1, double x2,
            double *x)
```

```
{
    *x = x1 + x2;
}
```

```
void Suma3 (double *x1, double x2)
```

```
{
    *x1 = *x1 + x2;
}
```

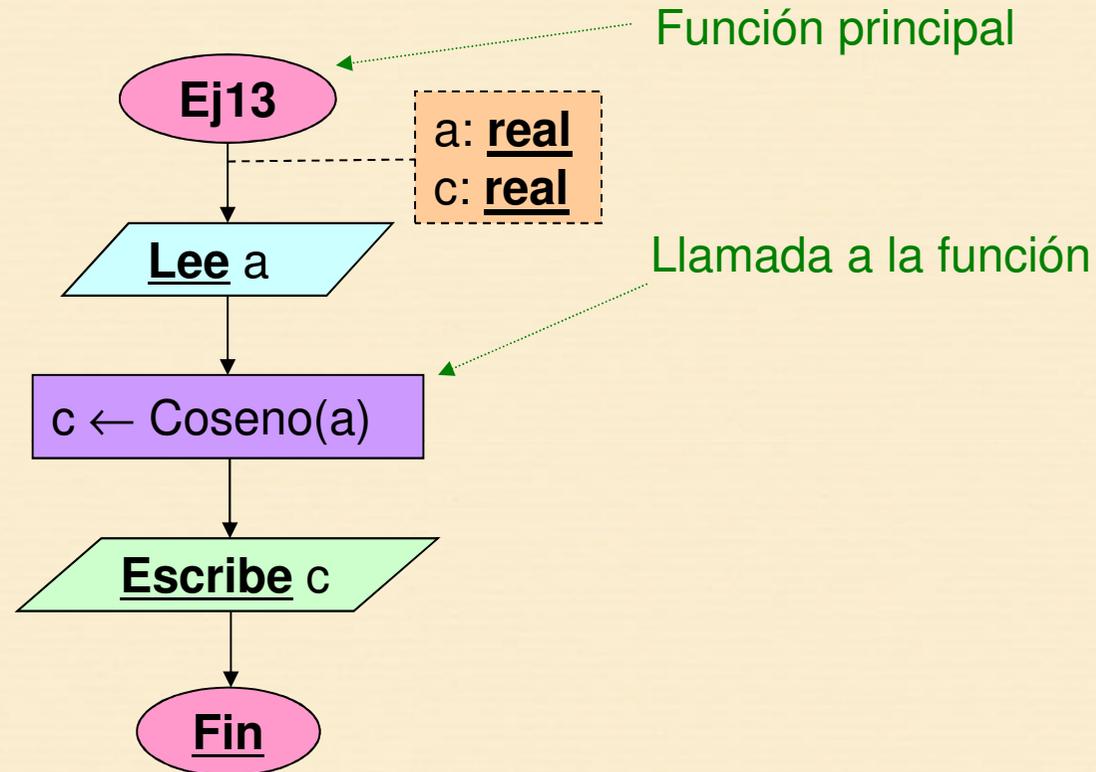


2. Ejemplo 13

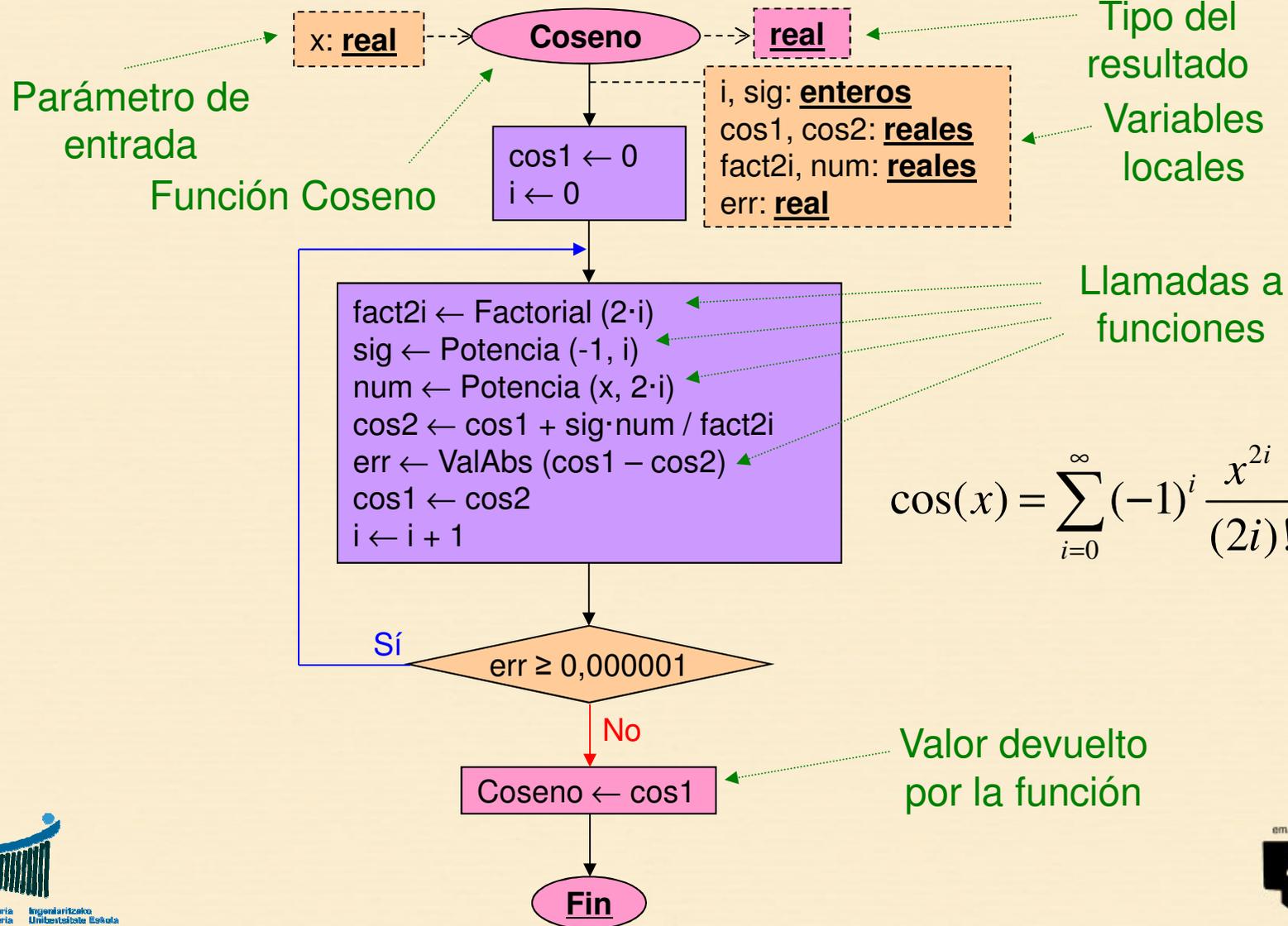
- **Título:**
 - Coseno
- **Nombre**
 - Ej13
- **Descripción**
 - Calcular el coseno de un ángulo dado en radianes utilizando series de Taylor-MacLaurin
- **Observaciones**
 - Descomposición en funciones

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

Ej13: Diagramas de Flujo (1)

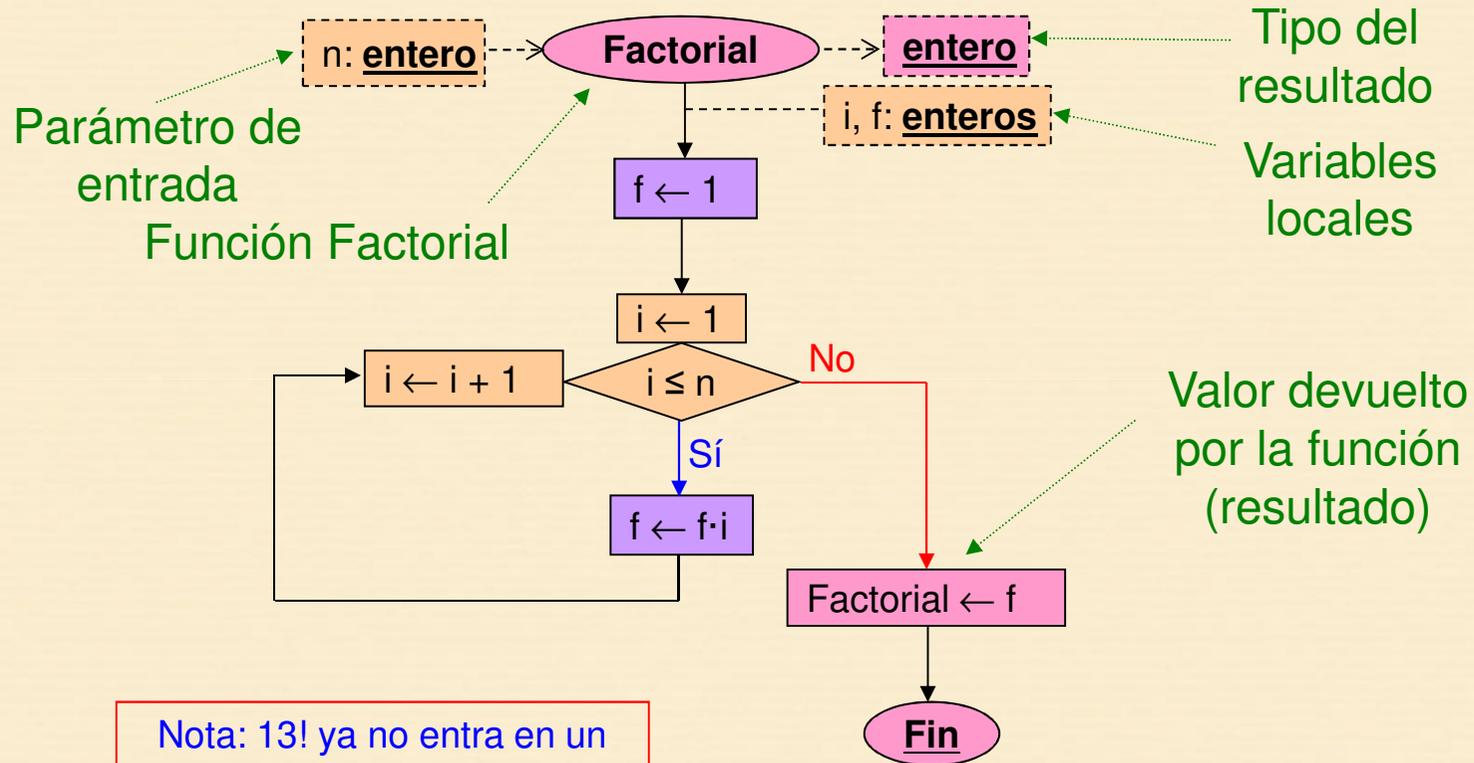


Ej13: Diagramas de Flujo (2)



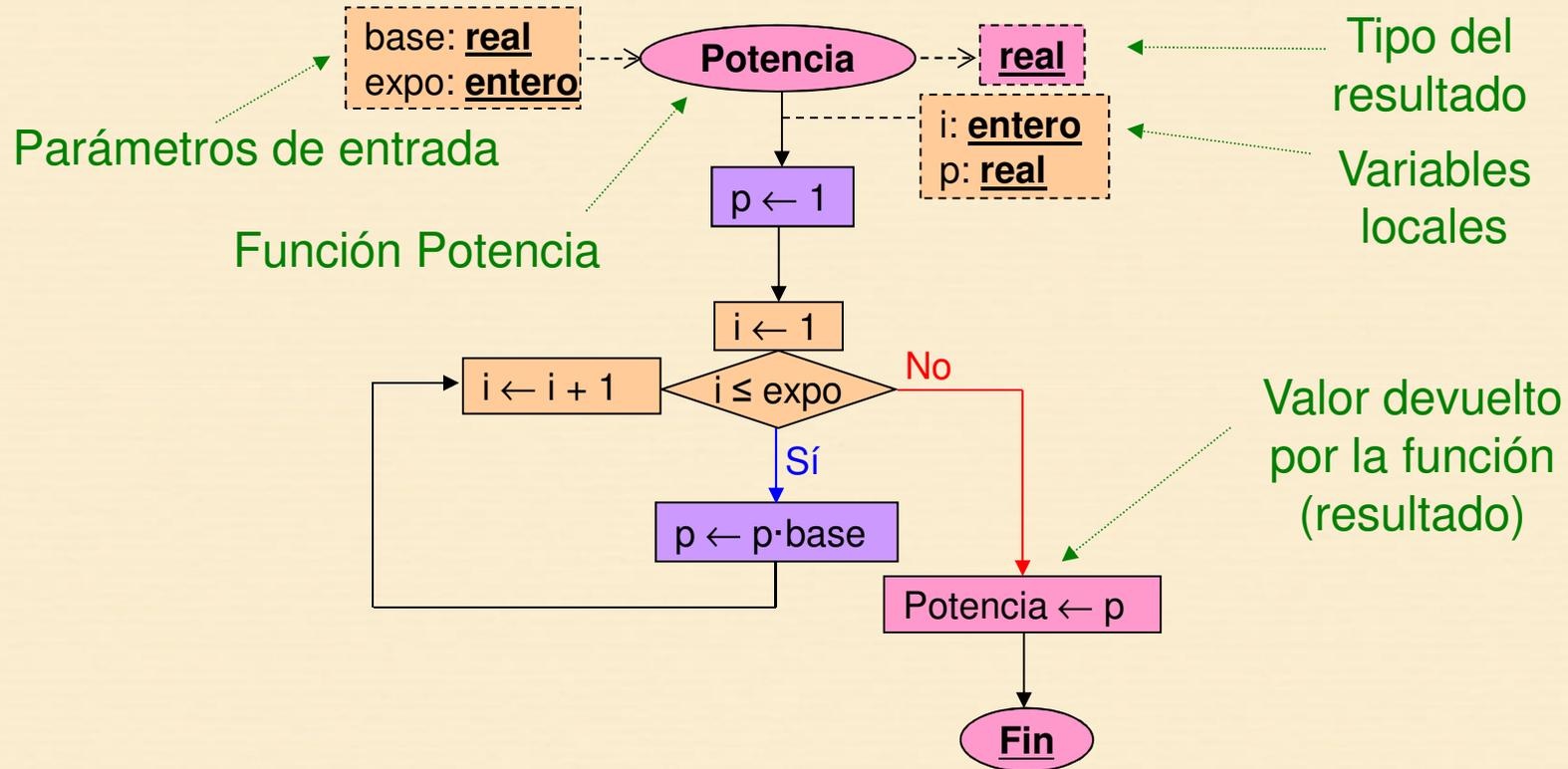
$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

Ej13: Diagramas de Flujo (3)



Nota: 13! ya no entra en un entero de 32 bits por lo que sería mejor utilizar reales dobles para f y el resultado

Ej13: Diagrama de Flujo (4)



Ej13: Codificación C (1)

```

/* Ej13 */
#include <stdio.h>

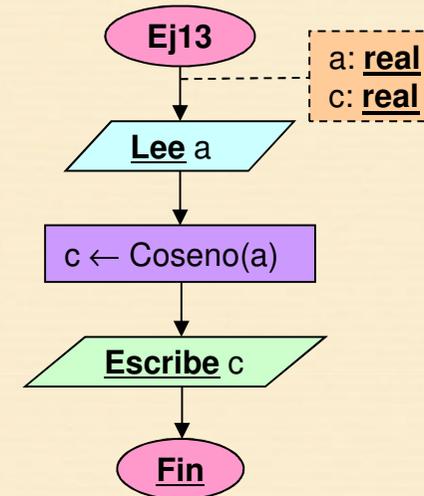
double Coseno (double a);

void main (void)
{
    double a, c;
    printf ("Introduce un ángulo: ");
    scanf ("%lf", &a);
    c = Coseno(a);
    printf ("Coseno de %lf: %lf\n", a, c);
}

```

Prototipo de la función

Llamada a la función



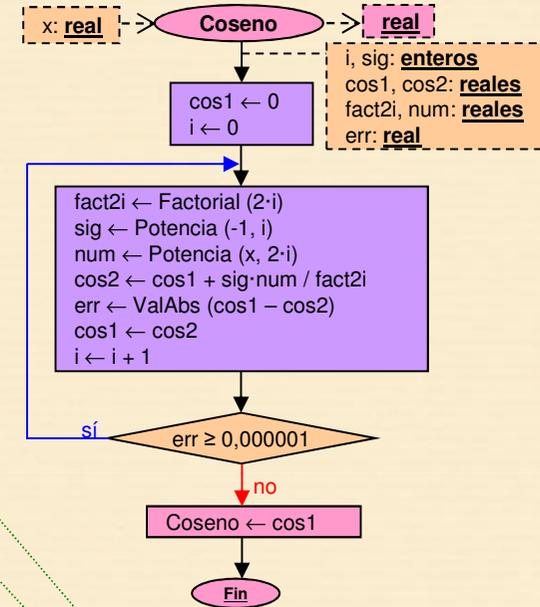
Ej13: Codificación C (2)

```

/* Ej13 continuación */
int Factorial (int n);
double Potencia (double base, int expo);
double ValAbs (double num);

double Coseno (double a)
{
    int i, sig; double cos1, cos2;
    double fact2i, num; double err;
    cos1 = 0;
    i = 0;
    do {
        fact2i = Factorial (2*i);
        sig = Potencia (-1, i);
        num = Potencia (a, 2*i);
        cos2 = cos1 + sig*num / fact2i;
        err = ValAbs (cos1 - cos2);
        cos1 = cos2;
        i++;
    } while (err >= 1e-6);
    return cos1;
}
    
```

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$



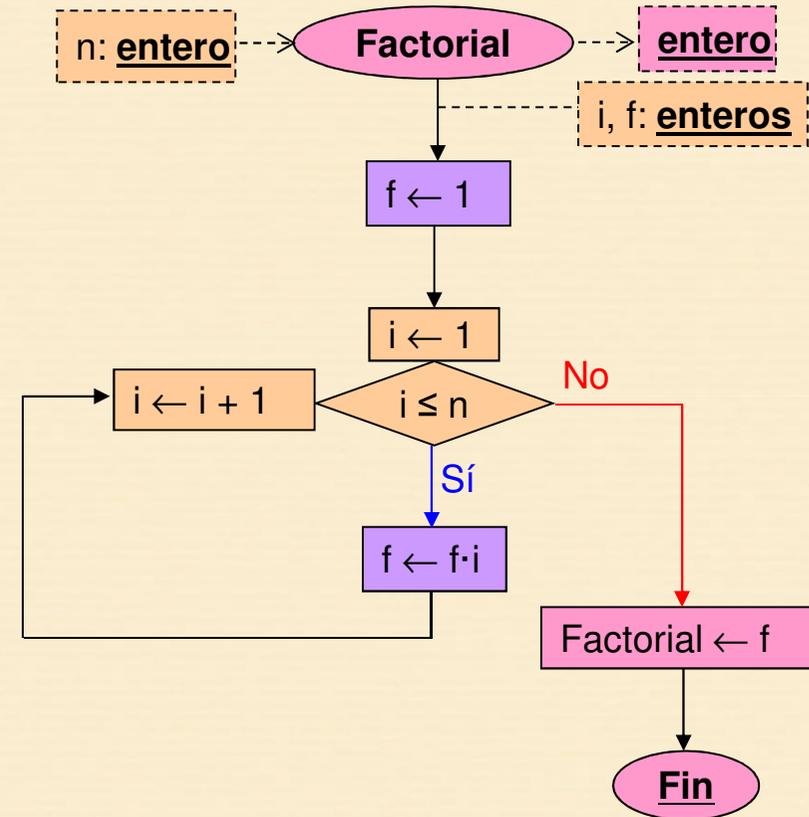
Prototipos de los funciones

Llamadas a las funciones

Valor devuelto por la función

Ej13: Codificación C (3)

```
/* Ej13 continuación */  
int Factorial (int n)  
{  
    int i, f;  
  
    f = 1;  
    for (i = 1; i <= n; i++)  
        f *= i;  
    return f;  
}
```



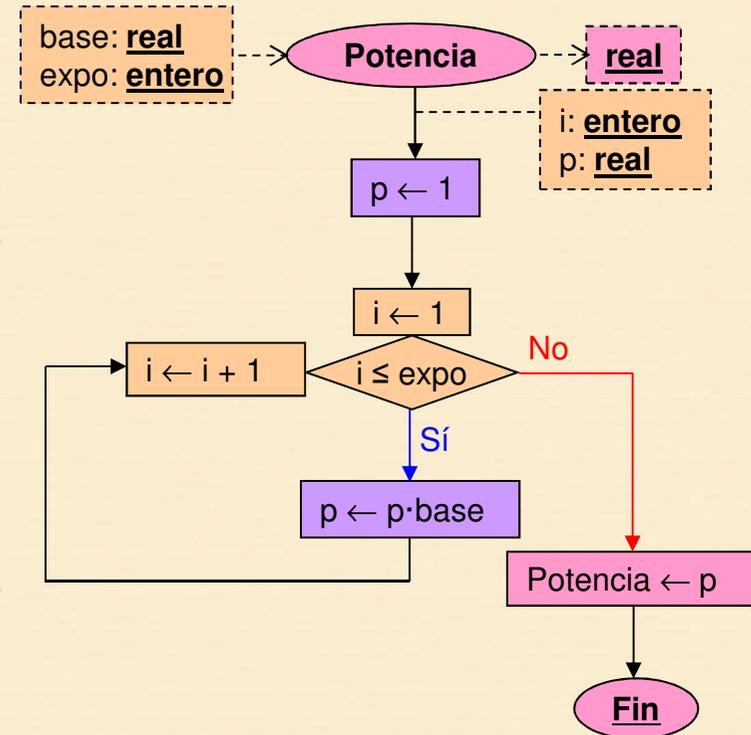
Ej13: Codificación C (4)

```

/* Ej13 continuación */
double Potencia (double base,
                 int expo)
{
    int i;
    double p;

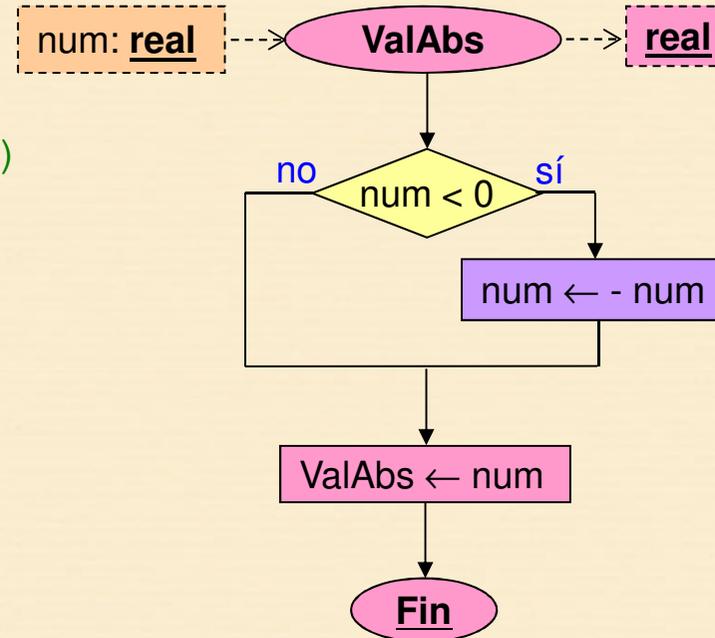
    p = 1;
    for (i = 1; i <= expo; i++)
        p *= base;
    return p;
}

```



Ej13: Codificación C (5)

```
/* Ej13 continuación */  
double ValAbs (double num)  
{  
    if (num < 0)  
        num = -num;  
    return num;  
}
```



3. Ejemplo 14

- **Título:**

- Ecuación de 2º grado

- **Nombre**

- Ej14

$$ax^2 + bx + c = 0$$

- **Descripción**

- Calcular las raíces de una ecuación de 2º grado

- Tipo 0: No es una ecuación
- Tipo 1: Ecuación lineal
- Tipo 2: Soluciones reales
- Tipo 3: Soluciones complejas

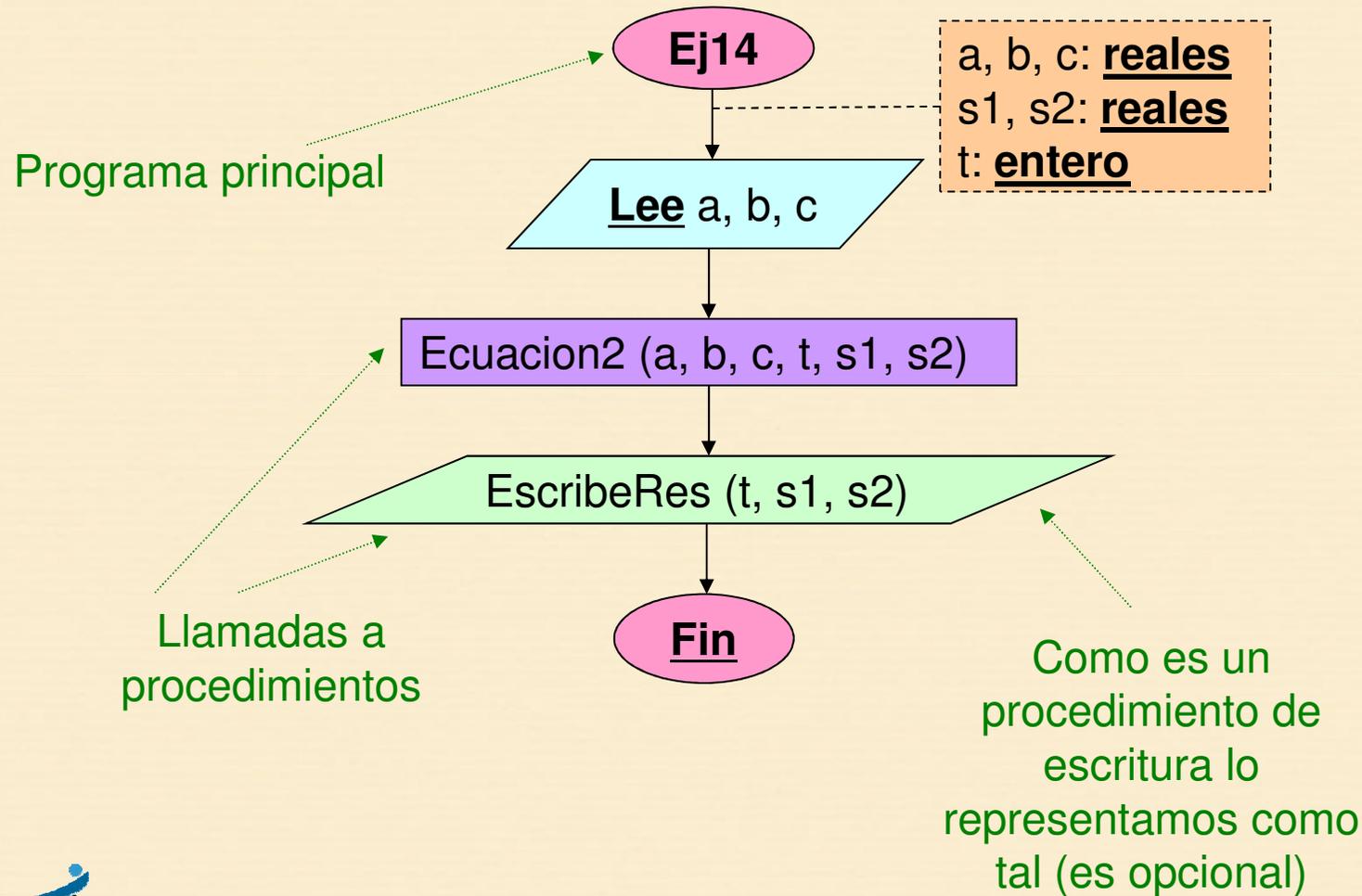
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- **Observaciones**

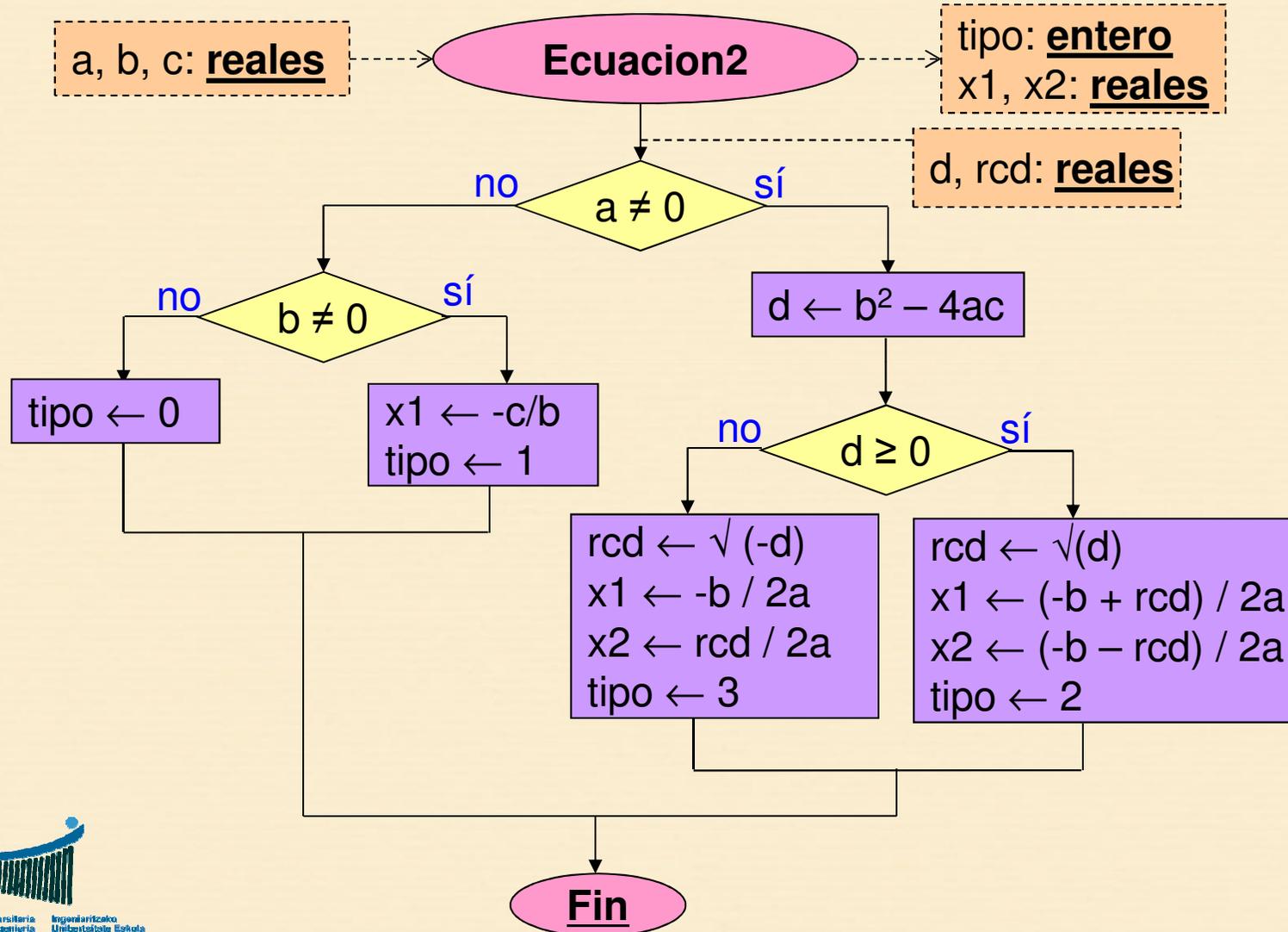
- Paso de parámetros por referencia



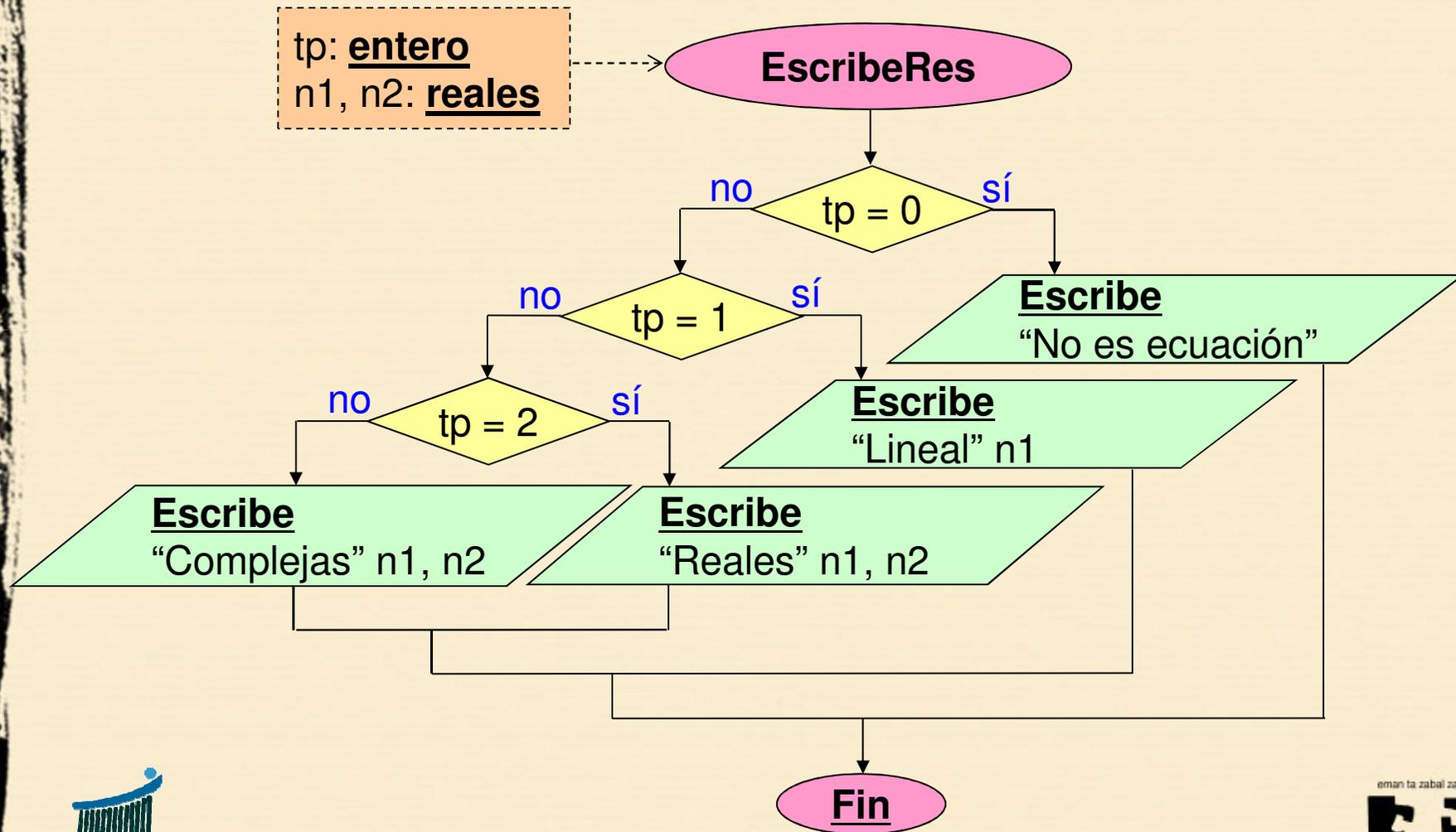
Ej14: Diagramas de Flujo (1)



Ej14: Diagramas de Flujo (2)



Ej14: Diagramas de Flujo (3)



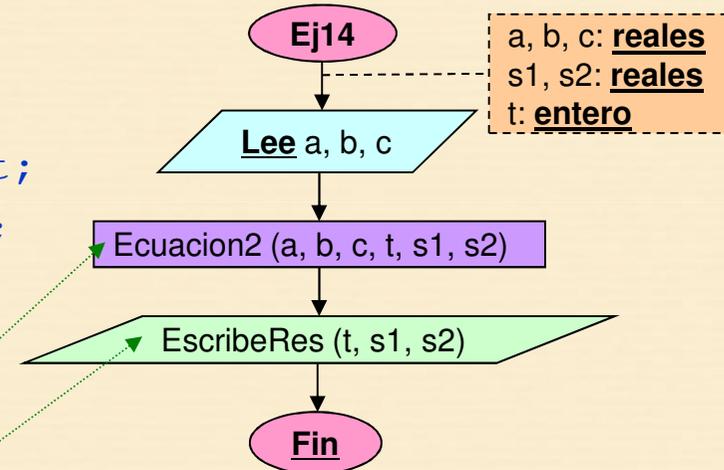
Ej14: Codificación C (1)

```

/* Ej14 */
void Ecuacion2 (double a, double b, double c,
               int *t, double *x1, double *x2);
void EscribeRes (int t, double n1, double n2);
void main (void)
{
    double a, b, c; double s1, s2; int t;
    printf ("Introduce coeficientes: ");
    scanf ("%lf%lf%lf", &a, &b, &c);
    Ecuacion2 (a, b, c, &t, &s1, &s2);
    EscribeRes (t, s1, s2);
}

```

Prototipos



Llamadas a la funciones



Ej14: Codificación C (2)

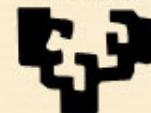
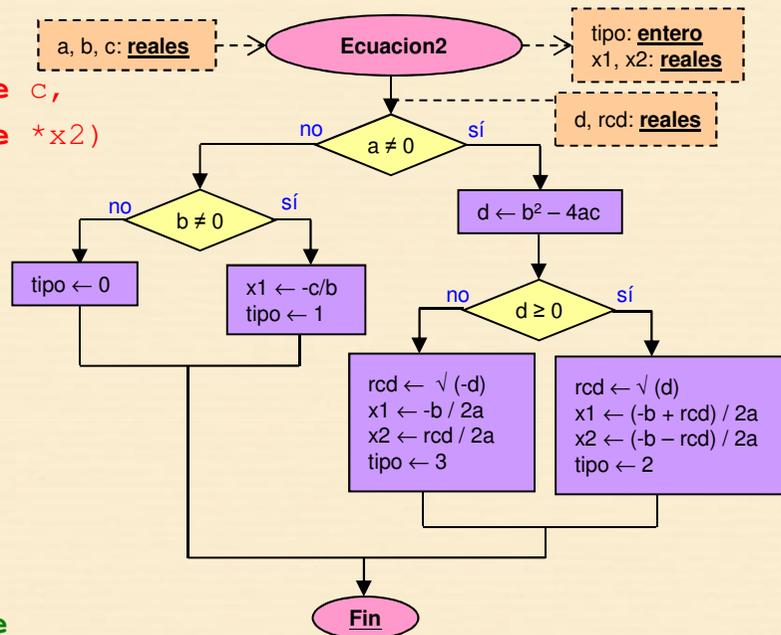
```
/* Ej14 continuación */
```

```
#include <math.h>
```

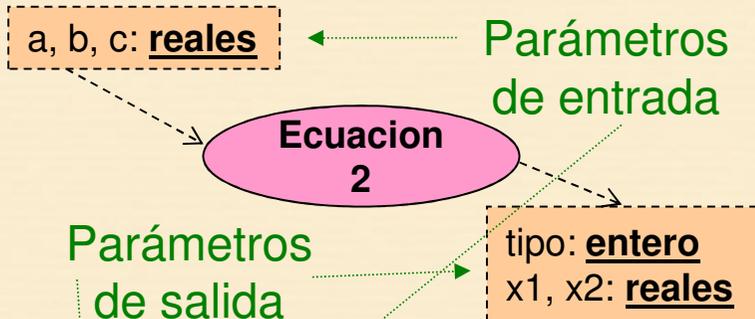
```
void Ecuacion2 (double a, double b, double c,
               int *tipo, double *x1, double *x2)
```

```
{
    double d, rcd;
    if (a != 0) {
        d = b*b - 4*a*c;
        if (d >= 0) {
            rcd = sqrt (d);
            *x1 = (-b + rcd)/(2*a);
            *x2 = (-b - rcd)/(2*a);
            *tipo = 2;
        }
        else {
            rcd = sqrt (-d);
            *x1 = -b / (2*a);
            *x2 = rcd / (2*a);
            *tipo = 3;
        }
    }
}
```

```
else
{
    if (b != 0) {
        *x1 = -c/b;
        *tipo = 1;
    }
    else
        *tipo = 0;
}
```



Ej14: Codificación C (2')



```

/* Ej14 (repetido)*/
#include <math.h>
void Ecuacion2 (
    double a, double b, double c,
    int *tipo,
    double *s1, double *s2)
{
    double d, rcd;
    ...
}

```

Variables
locales

```

if (a != 0) {
    d = b*b - 4*a*c;
    if (d >= 0) {
        rcd = sqrt (d);
        *x1 = (-b + rcd)/(2*a);
        *x2 = (-b - rcd)/(2*a);
        *tipo = 2;
    }
    else {
        rcd = sqrt (-d);
        *x1 = -b / (2*a);
        *x2 = rcd / (2*a);
        *tipo = 3;
    }
}
else
    if (b != 0) {
        *x1 = -c/b;
        *tipo = 1;
    }
    else
        *tipo = 0;
}

```

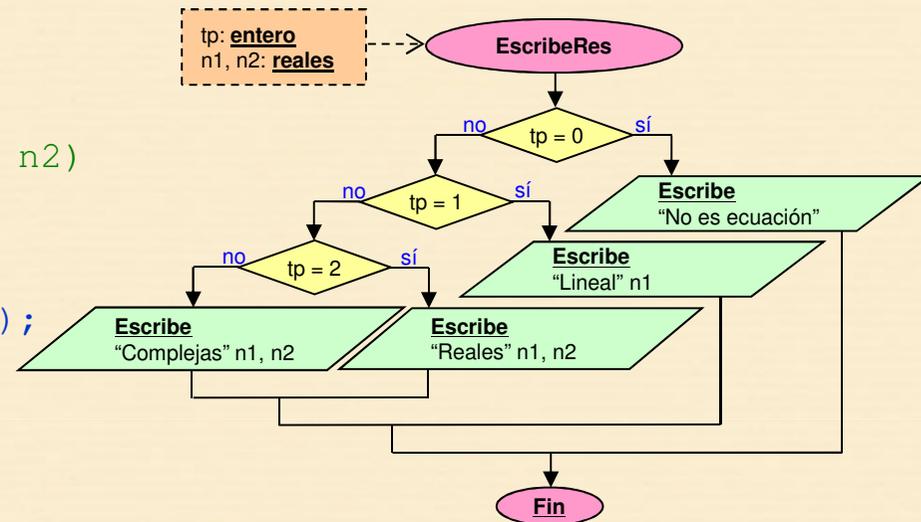


Ej14: Codificación C (3)

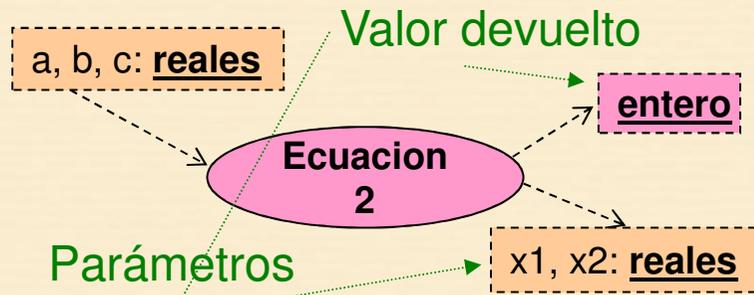
```

/* Ej14 continuación */
void EscribeRes (int tp,
                 double n1, double n2)
{
    if (tp == 0)
        printf ("No es una ecuación\n");
    else if (tp == 1)
        printf ("Ecuación lineal.\n"
                " x: %.2lf\n", n1);
    else if (tp == 2)
        printf ("Soluciones reales\n"
                " x1: %.2lf\nx2: %.2lf\n", n1, n2);
    else
        printf ("Soluciones complejas\n"
                " x1: %.2lf + %.2lfi\n"
                " x2: %.2lf - %.2lfi\n", n1, n2, n1, n2);
}

```



Ej14: Codificación alternativa



Parámetros
de salida

```
/* Ej14 (repetido)*/
```

```
#include <math.h>
```

```
int
```

```
Ecuacion2 (  
    double a, double b, double c,  
    double *s1, double *s2)
```

```
{  
    double d, rcd;  
    ...  
}
```

```
if (a != 0) {  
    d = b*b - 4*a*c;  
    if (d >= 0) {  
        rcd = sqrt (d);  
        *x1 = (-b + rcd)/(2*a);  
        *x2 = (-b - rcd)/(2*a);  
        return 2;  
    }  
    else {  
        rcd = sqrt (-d);  
        *x1 = -b / (2*a);  
        *x2 = rcd / (2*a);  
        return 3;  
    }  
}  
else  
    if (b != 0) {  
        *x1 = -c/b;  
        return 1;  
    }  
else  
    return 0;  
}
```

Valor devuelto



Escuela Universitaria de Ingeniería Vitoria-Gasteiz Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

eman ta zabal zazu



Universidad del País Vasco Euskal Herriko Unibertsitatea