INFORMÁTICA INDUSTRIAL

PROGRAMACIÓN BÁSICA C++ (II)

M. Abderrahim, A. Castro, J. C. Castillo
Departamento de Ingeniería de Sistemas y Automática

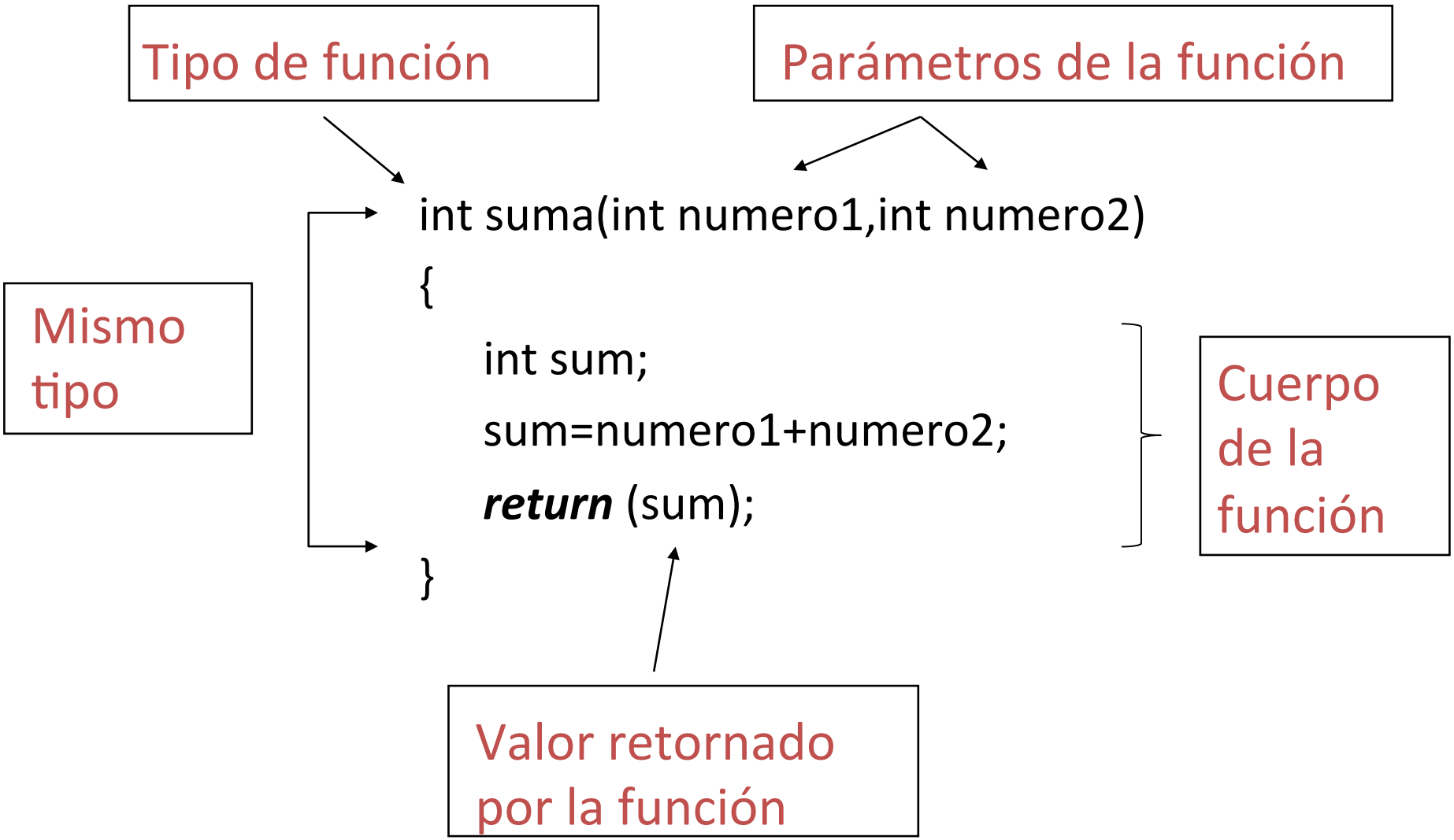
uc3m | Universidad Carlos III de Madrid

7. Funciones

Funciones

- El lenguaje C++ está basado en el uso de funciones:
  - Funciones de sistema
  - Funciones de dispositivo/equipo.
  - Diseñadas por el usuario.
- Ideas:
  - No reinventemos la rueda
  - No repetir innecesariamente
  - Siempre la mayor claridad posible

Funciones


Tipo de función
Parámetros de la función
int suma(int numero1,int numero2)
{
int sum;
sum=numero1+numero2;
return (sum);
}
Mismo tipo
Cuerpo de la función
Valor retornado por la función

Prototipos de las funciones

int suma(int numero1,int numero2);

Declaración de la función
Tipo nombre (parametros);

```
main()
{    int i, j=2;
     i=suma(j,6);
}
```

Llamada a la función: el número y tipo de los parámetros son verificados

```
int suma(int numero1,int numero2)
{
     int sum;
     sum=numero1+numero2;
     return(sum);
}
```

Implementación o definición de la función

Variables locales y globales

- Las variables definidas en el cuerpo de la función son accesibles únicamente dentro de la función (variables **locales**).
- Las variables definidas fuera de todas las funciones son conocidas por todas las funciones (variables **globales**).

Variables locales y globales

```
int i, j;

main()
{
    int a;
}

int funcion()
{
    int b;
    int a;
}
```

i,j variables globales .
Pueden ser utilizadas por *main* y *funcion*

a es una variable local de *main*.
funcion no la conoce

b es una variable local in funcion
Pero *main* no sabe que existe

Las variables locales con el mismo nombre pueden existir en muchas funciones.

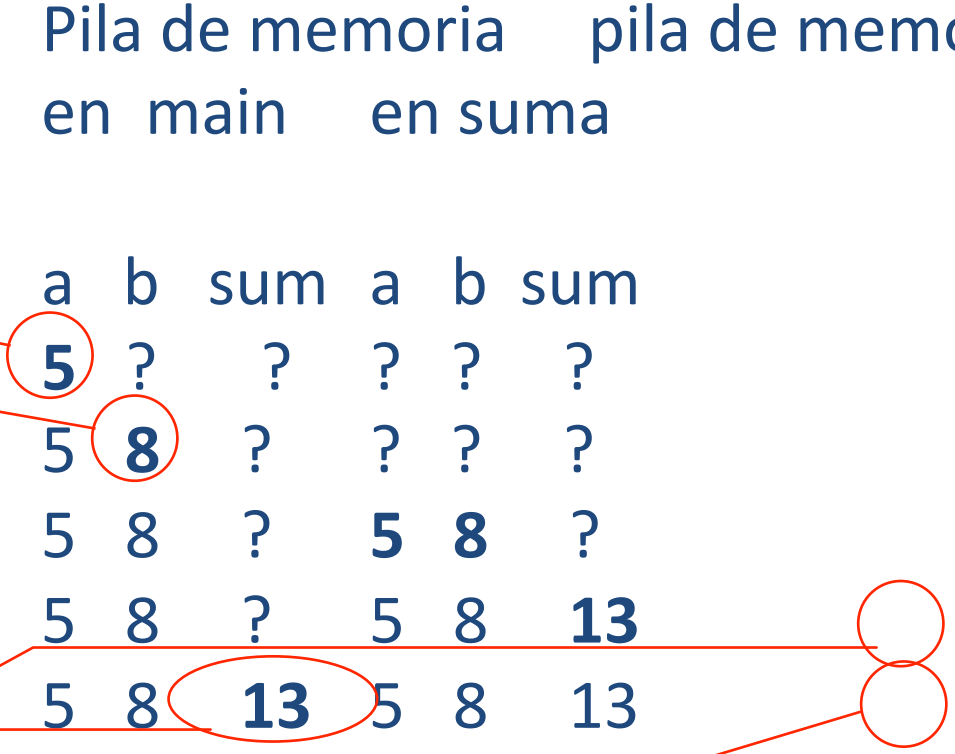
Argumentos de la función

- Una función es ejecutada mediante la invocación de su nombre y argumentos.
- Los argumentos son pasados mediante parámetros con su valor correspondiente
  - Cada argumento es evaluado y su valor es utilizado de forma local.
  - Los parámentros formales no son pasados a las funciones (la variable)

Argumentos y llamadas a función

```
main()
{
    int a,b,sum;
    a=5;
    b=8;
    sum=suma(a,b);
}

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



Argumentos y llamadas a función

```
main()
{
    int a,b,c;
    a=5;
    b=8;
    c=funcion(a,b);
}
int funcion(int a, int b)
{
    int sum;
    ++a;++b;
    sum=a+b;
    return(sum);
}
```

Pila de memoria en main

a	b	c
5	?	?
5	**8**	?
5	8	?
5	***8***	?
5	8	?
5	8	**15**

pila de memoria en function

a	b	sum
?	?	?
?	?	?
5	**8**	?
6	**9**	?
6	9	**15**
6	9	15

Paso de argumentos por referencia

- C++ dispone de un tipo especial de variable, llamada “referencia”, que puede ser utilizada como parámetro de la función que referencia al valor de la variable original.

- Una “referencia” es un “alias” de otra variable.

- Cualquier cambio realizado a través de la referencia se actualiza en la variable referenciada.

- Para declarar una referencia a una variable, se inserta el símbolo “&” delante del nombre de la variable a referenciar.

Paso de argumentos por referencia

```
#include <iostream>
using namespace std;

void main( )
{

      int count = 1;
      Int  &alias_for_count = count;
      cout << count << alias_for_count << endl;

      alias_for_count++;
      cout << count << alias_for_count << endl;

      count++;
      cout << count << alias_for_count << endl;
}
```

1	1
2	2
3	3

Paso de argumentos por referencia

- Cuando se pasa un argumento por referencia, el argumento debe ser una variable.

- Cuando se pasa el argumento por valor, el argumento puede ser un literal, una variable, una expresión, o incluso el valor de retorno de otra función

Referencias constantes como parámetros

```
#include <iostream>
using namespace std;

void max(const int &num1, const int &num2)
{
    if (num1 > num2)
        cout << num1 << endl;
    else
        cout << num1 << endl;
}

int main( )
{
    int x = 1;
    int y = 2;
    max(x, y);

    return 0;
}
```

Argumentos por defecto

- C++ permite declarar funciones con valores de argumentos por defecto

- Se pasan valores por defecto como parámetros de la función cuando dicha función es invocada sin argumentos

Argumentos por defecto

```
1   #include <iostream>
2   using namespace std;
3
4   void printArea(double radius = 1)
5   {
6      double area = radius * radius * 3.14159;
7      cout << area << endl;
8   }
9
10  int main( )
11  {
12      printArea( );
13      printArea(4);
14
15  return 0;
16  }
```

```
area is 3.14159
area is 50.2654
```

Arrays como argumento de funciones

- Tres formas:
  - Declarando el array con sus dimensiones
    - function(int t[10])
  - Declarando el array sin dimensiones
    - function(int t[])
  - A través de punteros
    - function(int *t)

uc3m | Universidad Carlos III de Madrid