Fundamentos de la programación

2 Tipos e instrucciones I

Anexo: Introducción a funciones definidas por el programador

Grado en Ingeniería Electrónica y Comunicaciones

Pilar Sancho Thomas
Luis Hernández Yáñez
Facultad de Informática
Universidad Complutense

Funciones definidas por el programador

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Funciones en C++

Los programas pueden incluir otras funciones además de `main()`

Forma general de una función en C++:

```
tipo nombre(parámetros) // Cabecera
{
    // Cuerpo
}
```

- ✓ *Tipo* de dato que devuelve la función como resultado
- ✓ *Parámetros* para proporcionar datos a la función
  Declaraciones de variables separadas por comas
- ✓ *Cuerpo*: secuencia de declaraciones e instrucciones
  ¡Un bloque de código!

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Datos en las funciones

- ✓ Datos locales: declarados en el cuerpo de la función
  Datos auxiliares que utiliza la función (puede no haber)
- ✓ Parámetros: declarados en la cabecera de la función
  Datos de entrada de la función (puede no haber)

Ambos son de uso exclusivo de la función y no se conocen fuera

```
double f(int x, int y) {
   // Declaración de datos locales:
   double resultado;

   // Instrucciones:
   resultado = 2 * pow(x, 5) + sqrt(pow(x, 3)
                / pow(y, 2))  / abs(x * y) - cos(y);

   return resultado; // Devolución del resultado
}
```

$$f(x,y) = 2x^5 + \frac{\sqrt{\frac{x^3}{y^2}}}{|x \times y|} - \cos(y)$$

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Argumentos

Llamada a una función con parámetros

`Nombre(Argumentos)`

Al llamar a la función:

- Tantos argumentos entre los paréntesis como parámetros
- Seguir el orden de declaración de los parámetros
- Cada argumento: mismo tipo que su parámetro
- Cada argumento: expresión válida (se pasa el resultado)

Se copian los valores resultantes de las expresiones en los correspondientes parámetros

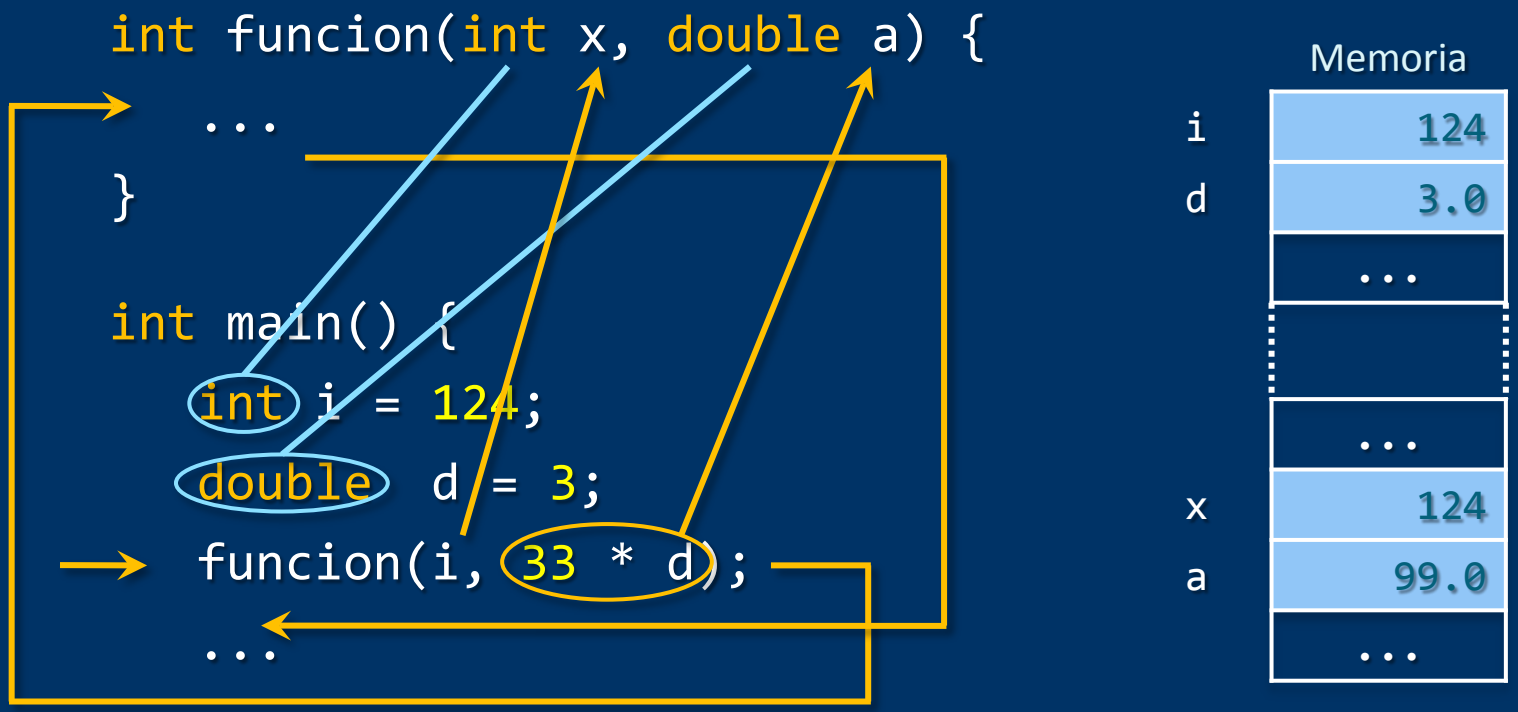
Llamadas a la función: en expresiones de otras funciones

```
int valor = f(2, 3);
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Paso de argumentos

Se copian los argumentos en los parámetros



Los argumentos no se modifican

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Paso de argumentos

Dadas las siguientes declaraciones:

```
int i;
double d;
int func(int x, double a);
```

Los argumentos y las variables se pueden llamar igual o no

¿Qué pasos de argumentos son correctos? ¿Por qué no?

`func(3, i, d);`	✘	Nº de argumentos ≠ Nº de parámetros
`func(i, d);`	✓	
`func(3 * i + 12, d);`	✓	
`func(d, i);`	✘	¡Argumento `double` para parámetro `int`!
`func(3.5, d);`	✘	¡Argumento `double` para parámetro `int`!
`func(i);`	✘	Nº de argumentos ≠ Nº de parámetros

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Resultado de la función

La función ha de devolver un resultado

La función termina su ejecución devolviendo un resultado

La instrucción `return` *(sólo una en cada función)*

- Devuelve el dato que se pone a continuación (tipo de la función)
- Termina la ejecución de la función

El dato devuelto sustituye a la llamada de la función:

```
int cuad(int x) {
    return x * x;
    x = x * x;
}
```

```
int main() {
    cout << 2 * cuad(16);

    return 0;
}
```

256

Esta instrucción
no se ejecutará nunca

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Prototipos de las funciones

¿Qué funciones hay en el programa?

Colocaremos las funciones después de `main()`

¿Son correctas las llamadas a funciones del programa?

- ¿Existe la función?
- ¿Concuerdan los argumentos con los parámetros?

→ Prototipos tras las inclusiones de bibliotecas

Prototipo de función: Cabecera de la función terminada en `;`

```
double f(int x, int y);
int funcion(int x, double a)
int cuad(int x);
...
```

`main()` es la única función que no hay que prototipar

Un programa con funciones

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

// Prototipos de las funciones (excepto main())
bool par(int num);
bool letra(char car);
int suma(int num);
double formula(double x, double y);

int main() {
    int numero;
    char caracter;
    double x = 3.5, y = 2.2;
    cout << "Entero: ";
    cin >> numero;
    //Se muestra si el número es par o impar
    if (par(numero)) {
        cout << "Par";
    }
    ...
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Un programa con funciones

```
    else {
        cout << "Impar";
    }
    cout << endl;
    //Si el número es mayor que 1 se hace el sumatorio de 1 a numero
    if (numero > 1) {
        cout << "Sumatorio de 1 a " << numero << ": "
             << suma(numero) << endl;
    }
    cout << "Carácter: ";
    cin >> caracter;
    //Se muestra si el carácter introducido es una letra
    if (!letra(caracter)) {
        cout << "no ";
    }
    cout << "es una letra" << endl;
    cout << "f(x, y) = " << formula(x, y) << endl;
    // Los argumentos pueden llamarse igual o no que los parámetros

    return 0;
} ...
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Un programa con funciones

```
// Implementación de las funciones propias

bool par(int num) {
    bool esPar;

    if (num % 2 == 0) {
        esPar = true;
    }
    else {
        esPar = false;
    }

    return esPar;
}
...
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Un programa con funciones

```
bool letra(char car) {
   bool esLetra;
   char carMayus = toupper(car);
   if (carMayus >= 'A') {
      if (carMayus <= 'Z')
        esLetra = true;
      else
        esLetra = false;
   }
   else {
      esLetra = false;
   }
   return esLetra;
}

...
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Un programa con funciones

funciones.cpp

```
int suma(int num) {
    int sum = 0, i = 1;
    while (i < num) {
        sum = sum + i;
        i++;
    }
    return sum;
}

double formula(double x, double y) {
    double f;

    f = 2 * pow(x, 5) + sqrt(pow(x, 3) / pow(y, 2))
          / abs(x * y) - cos(y);

    return f;
}
```

Luis Hernández Yáñez / Raquel Hervás Ballesteros

Acerca de *Creative Commons*

Licencia CC (Creative Commons)

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

- Reconocimiento (*Attribution*):
  En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
- No comercial (*Non commercial*):
  La explotación de la obra queda limitada a usos no comerciales.
- Compartir igual (*Share alike*):
  La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Material original elaborado por Luis Hernández Yáñez, con modificaciones de Raquel Hervás Ballesteros.