Funciones en Lenguaje C

FACULTAD DE INGENIERÍA
CÁTEDRA: INFORMATICA

¿Qué necesitamos saber de las funciones?

1. Cómo se las declara.
2. Cómo y qué retornan cuando se las llama.
3. Cómo se establece la comunicación entre ellas .
4. Cómo y dónde se las define.

¿Qué es una función?

- En C una función es
  - Una sección de código autocontenida e independiente
  - Su objetivo es ejecutar una tarea específica, pudiendo opcionalmente regresar un valor al programa que la llama.
  - Las funciones ahorran espacio, reduciendo repeticiones y haciendo
  - más fácil la programación.

Forma general de una función

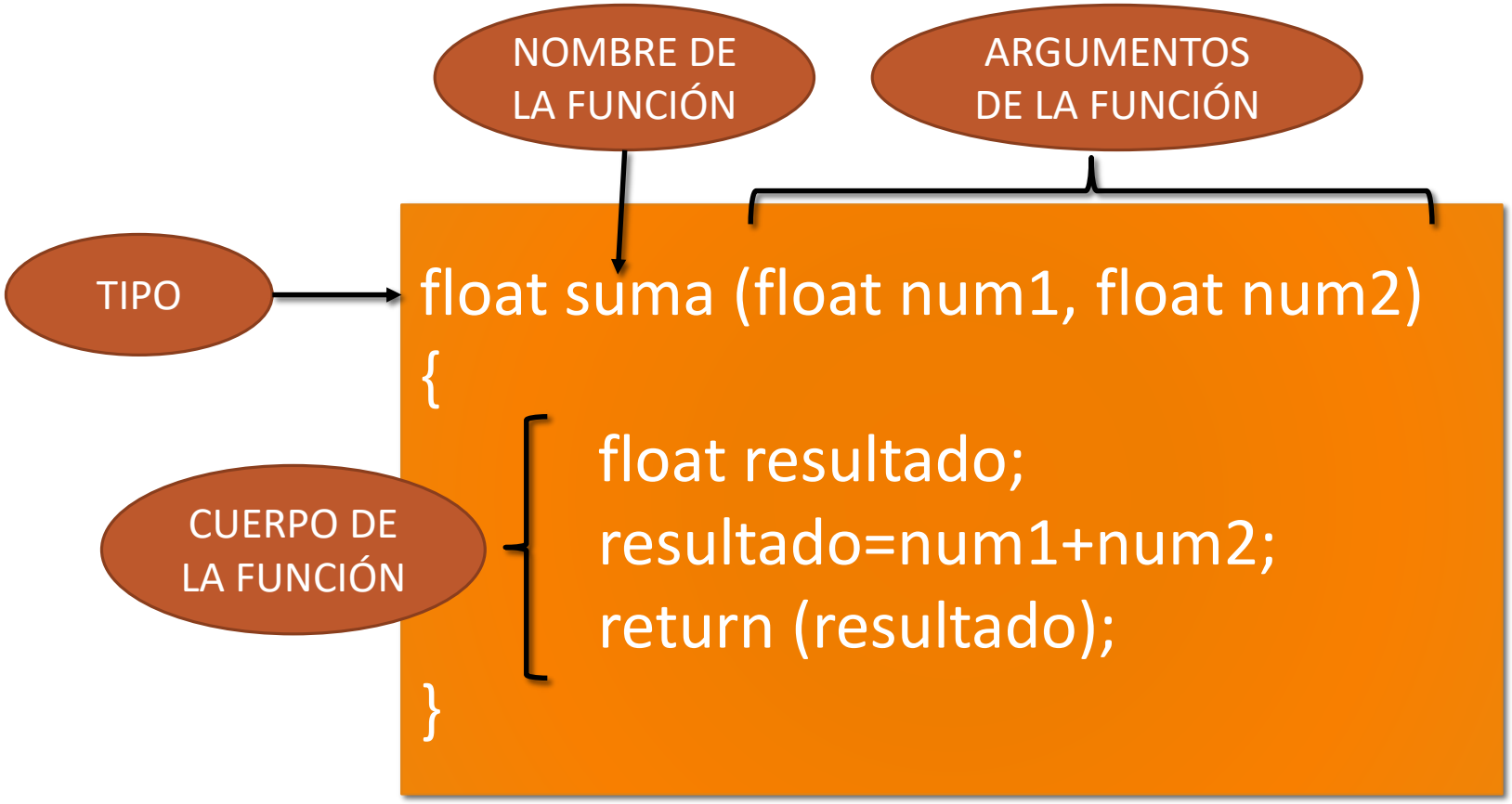
```
tipo  nombre_funcion (lista de parámetros)

{
  cuerpo de la función

}
```

- tipo
  - Especifica el tipo del valor que la función devolverá cuando ésta sea invocada. Si no se coloca explícitamente la función retorna un entero como resultado.
- nombre_funcion
  - Es único para cada función dentro del mismo programa. Con este nombre se pueden ejecutar las sentencias contenidas en la función, en cualquier parte del programa (llamada de la función).
- lista de parámetros
  - Es un conjunto de nombres de variables que reciben los valores de los argumentos cuando se llama a la función. Una función puede no tener parámetros, pero aunque no existan se requieren los paréntesis.
- cuerpo de la función
  - Es el conjunto de sentencias programadas en C que realizan una tarea específica.

Forma general de una función



Prototipo, llamada y definición de una función

EJEMPLO FUNCIÓN CUBO

```
#include <stdio.h>
#include <stdlib.h>
long cubo (long x);
void main() {
      printf("Ingrese un valor entero: ");
      scanf("%d", &entrada);
      respuesta = cubo(entrada);
      printf("\n El cubo de %ld es %ld", entrada, respuesta);
      system("PAUSE");
}
long cubo(long x) {
      long x_cubo;
      x_cubo=x*x*x;
      return x_cubo;
}
```

PROTOTIPO, LLAMADA Y DEFINICIÓN DE LA FUNCIÓN

- prototipo de la función cubo():
  - tipo nombre_funcion (parámetros);
- llamada a la función cubo():
  - asignándole el valor que retorna a la variable respuesta, que es del mismo tipo (long)
  - nombre_funcion(argumentos)
  - los argumentos que le enviamos a la función entre los paréntesis deben coincidir en cantidad, orden y tipo con los parámetros declarados en el prototipo
- definición de la función cubo():
  - tipo nombre_funcion (argumentos) { ...cuerpo de la función ... }

Prototipo, llamada y definición de una función

EJEMPLO FUNCIÓN CUBO

```
#include <stdio.h>
#include <stdlib.h>
long cubo(long x) {
    long x_cubo;
    x_cubo=x*x*x;
    return x_cubo;
}
void main() {
    printf("Ingrese un valor entero: ");
    scanf("%d", &entrada);
    respuesta = cubo(entrada);
    printf("\n El cubo de %ld es %ld", entrada,
    respuesta);
    system("PAUSE");
}
```

PROTOTIPO, LLAMADA Y DEFINICIÓN DE LA FUNCIÓN

- Otra manera es agregar la definición antes del mail, donde se coloca el prototipo.

Valores de retorno de una función

- Todas las funciones, con excepción de las del tipo void, retornan un valor.
- Este valor puede estar explicitado mediante la sentencia return o, si no se especifica, su valor es cero.
- Ejemplos de llamadas a un función que devuelven un valor:
  - x=pot(y);
  - if (max(x,y) >100)
- Ejemplo de llamadas a una función que no devuelven ningún valor:
  - mostrar_hola_mundo();

Llamada a funciones

- Una función puede ser llamada de las 2 siguientes maneras:
  1. nombre_función (lista de parámetros);
  2. variable = nombre_función (lista de parámetros);
- Una llamada a la función implica los siguientes pasos:
  1. A cada parámetro se le asigna el valor real de su correspondiente argumento.
  2. Se ejecuta el cuerpo de la función.
  3. Se devuelve el valor de la función y se retorna al punto de llamada.

Ejemplo de función que devuelve un valor

```
#include<stdio.h>

#include<stdlib.h>

float suma (float x, float y);

main()

{
  float a=10.4, b=12.1, c;
  c=suma(a,b);

}

float suma(float x, float y)

{
  return (x+y);

}
```

Ejemplo de función que no devuelve ningún valor

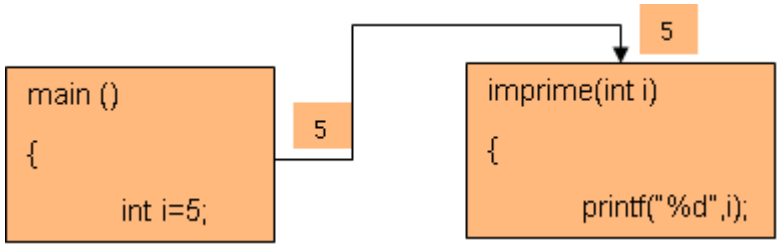
```
#include<stdio.h>
#include <stdlib.h>
void media3(float n1,float n2,float n3);
int main()
{
 float n1, n2, n3;
 printf("Introduzca un numero real:\n");
 scanf("%f",&n1);
 printf("Introduzca otro numero real:\n");
 scanf("%f",&n2);
 printf("Introduzca otro numero real:\n");
 scanf("%f",&n3);
 media3 (n1,n2,n3);
 system("pause");
}
void media3(float n1, float n2, float n3)
{
 float resultado;
 resultado=(n1+n2+n3)/3;
 printf("La media es:%.2f\n",resultado);
}
```

Transferencia de parámetros entre funciones

- La transferencia resulta necesaria porque las variables locales son mejores que las globales pero no pueden ser accedidas desde fuera del bloque donde se encuentran definidas.
- Las variables locales quedan así protegidas contra posibles sobrescrituras de sus valores, pero algunas veces deben ser compartidas con otras rutinas.
- Hay 2 maneras de transferir variables entre funciones en C:
  - Transferencia por valor (o por copia)
  - Transferencia por referencia (o por dirección)

Transferencia por valor

- Cuando se transfiere un argumento por valor, se envía a la función receptora una copia del valor de la variable, la que es asignada al respectivo parámetro.
- El valor de “i” es transferido. A partir de entonces hay dos variables i. Como ambas son locales (una de main y la otra de imprime) no se genera conflicto aunque se llamen igual.
- Como se transfiere una copia, si la función imprime modificara el valor de su variable “i” esto no afectaría a la variable “i” de main.
- Este es el método predeterminado para transferir variables entre funciones
- Sólo hay que incluirlas en la lista de argumentos de la función llamante y en la de parámetros de la función receptora.



Ejemplo de función que usa transferencia por valor

```
/* Programa que calcula el peso en la luna del usuario */
#include <stdio.h>
#include <stdlib.h>
void luna(int);                    //prototipo de la función “luna”
void main()
{
    int peso;                                   //variable local de main
    printf(“¿Cuántos kilos pesa usted?);
    scanf (“%d”, &peso);
    luna(peso);                    //llama a la función “luna” y le transfiere el valor de “peso”
}
void luna(int pesoEnTierra)        //declara el parámetro a serle transferido
{
    int pesoEnLuna;      //variable local de esta función
     pesoEnLuna=pesoEnTierra;
    printf(“¡En la Luna usted pesa sólo %d kilos”,pesoEnLuna);
}
```

Transferencia por referencia (por dirección)

- En este método es la dirección de la variable lo que se envía a la función receptora y ésta lo asigna a su parámetro.
- Es decir que si la función receptora modifica la variable, ésta es modificada también en la función invocante.
- Este es el método que se utiliza siempre que la variable transferida sea un arreglo.

Ejemplo de función que usa transferencia por referencia

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
void potrat2(float *x, float *y);
int main()
{
    float a, b;
    a = 5.0; b = 10.0;
    printf("valores antes de llamar a la funcion: ");
    printf("\n a = %.2f b = %.2f\n",a,b);
    potrat2 (&a, &b);
    printf("valores despues de llamar a la funcion: ");
    printf("\n a = %.2f b = %.2f\n",a,b);
    system ("pause");
}
void potrat2(float *x, float *y)
{
    *x = (*x)*(*x);
    *y = sqrt (*y) ;
}
```