

CFGM SISTEMAS MICROINFORMÁTICOS Y REDES

Sistemas Operativos Monopuesto

TEMA 04: Fundamentos de Programación II

Modularización: Funciones en C

Contenido

- Concepto de función
- Estructura de una función
- Prototipo de una función
- Paso de parámetros

Concepto de función (I)

- Las funciones son un mecanismo de abstracción del lenguaje de programación que nos permite gestionar la complejidad de un programa.
- Divide y vencerás.
 - ¿Cómo te comes un elefante? Pártelo en trocitos que seas capaz de comerte y cómete un trocito cada día.
 - ¿Cómo resuelves un programa muy complicado? Divídelo en trocitos más pequeños que sepas programar y haz un miniprograma para cada uno de los trocitos. Esos *miniprogramas* es lo que en C se conocen como funciones
- Un programa que se descompone en funciones es:
 - Más fácil de entender y por tanto de programar y de mantener.
 - Más conciso. Las funciones posibilitan reutilizar código.
- Ya sabemos llamar a algunas funciones que han hecho otros: printf, scanf,... Ahora debemos aprender a crear nuestras propias funciones.

Concepto de función (II)

- Una función es un miniprograma que recibe unos parámetros con los que trabaja y retorna un **ÚNICO** resultado.



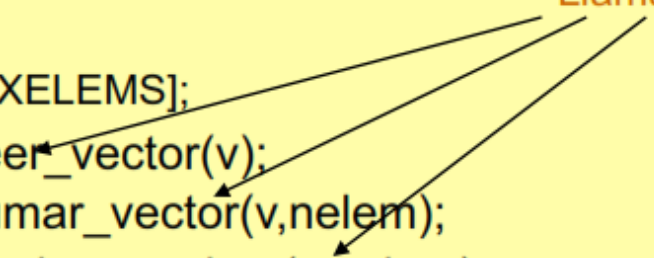
Concepto de función (III)

- Programa que lee un vector de enteros y calcula la suma de los elementos del vector y su máximo.
- Podemos dividir el programa en 3 subproblemas más pequeños:
 - Leer un vector de enteros hasta introducir el elemento 0
 - Calcular la suma de los elementos de un vector de enteros
 - Calcular el máximo de un vector de enteros
- Asignamos cada subproblema a una función.

```
#define NUM_MAXELEMS 100
```

```
void main() {  
    int v[NUM_MAXELEMS];  
    int nelem = leer_vector(v);  
    int suma = sumar_vector(v,nelem);  
    int max = calcular_maximo(v,nelem);  
    printf("La suma del vector es %d y el maximo es %d\n",suma,max);  
}
```

Llamada o invocación a función



Estructura de función (definición)

- Lo siguiente que debemos hacer es definir las funciones, por ejemplo, la que suma los elementos de un vector:

Tipo de resultado

Lista de parámetros

Cabecera de la función

```
int sumar_vector(int v[ ],int nelem){
```

```
    int suma = 0;
```

```
    int i;
```

```
    for (i = 0 ; i < nelem ; i++) {
```

```
        suma = suma + v[i];
```

```
    }
```

```
    return suma;
```

```
}
```

Declaración de variables

Devolución del resultado

Ejecución de un programa con funciones

```
#include <stdio.h>
void func1(); //Función 1 que imprime un mensaje
//Función 2 que imprime un mensaje y el valor del parámetro de entrada
void func2(int param);
void main(){
    printf("Vamos a ver como se ejecutan las funciones\n");
    func1();
    func2(123);
    printf("He terminado\n");
}
void func1() {
    printf("Soy la funcion 1\n");
}
void func2(int param) {
    printf("Soy la funcion 2, mi parametro vale %d\n",param);
}
```

Prototipo de una función

- El prototipo de una función define únicamente:
 - El nombre de la función
 - Los parámetros que recibe la función
 - El tipo del valor que retorna la función.
- El prototipo debe ir acompañado de un comentario donde se explica qué hace la función. Juntos, prototipo y comentario deben proporcionar toda la información que un usuario necesita conocer de la función para poder utilizarla. Para usar una función necesitamos saber **qué hace** no **cómo lo hace**

Prototipo + Comentario = Declaración



Definición



Declaración de funciones

```
/* Función que dado un entero, retorna el entero al cuadrado */
```

```
int cuadrado(int n);
```

```
/* Función que recibe dos números en punto flotante y retorna el  
máximo entre ellos */
```

```
float maximo(float x,float y);
```

```
/* Función que recibe un vector de enteros y su número de elementos  
y retorna la suma de los cuadrados de sus elementos */
```

```
int suma_cuadrados(int v[ ],int num_elems);
```

```
/* Función que recibe un vector de números en punto flotante y su  
número de elementos y calcula la media de los elementos del  
vector */
```

```
float media(float v[ ],int num_elems);
```

Paso de parámetros

En C cada vez que se envía un parámetro a una función se realiza una copia del mismo, excepto en el caso de los arrays. Este mecanismo se denomina paso de parámetros **por valor**.