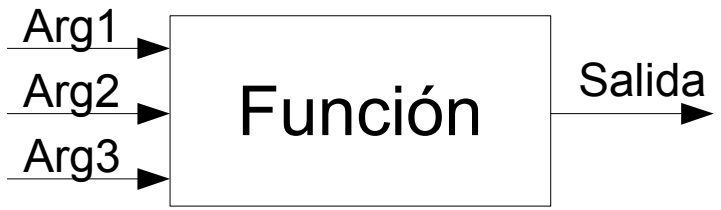
Tema 6: Funciones

Departamento de Automática
Universidad de Alcalá

Índice

Conceptos básicos
Concepto de función

- Una función es una caja negra que contiene código
  - Admite argumentos
  - Puede devolver un valor



Definición de función

```
tipo_devuelto nombre_funcion(lista_argumentos)
```

- ¡Las funciones deben ser definidas antes que utilizadas!
- Se puede usar el tipo void cuando
  - No se admiten argumentos
  - No se revuelve un valor
- La instrucción return indica qué valor devolver
  - Se termina la ejecución de la función
  - Puede usarse sin argumento cuando la función no devuelve

Conceptos básicos
Ejemplos de uso de funciones

Ejemplo 1

```
int sumar(int a, int b) {
  return (a+b);
}
```

Ejemplo 2

```
int sumar(int a, int b) {
  int variable;
  variable = a + b;
  return(variable);
}
```

Ejemplo 3

```
void sumar(int a, int b) {
  printf("a+b=%d", a+b);
}
```

Ejemplo 4

```
void factorial(int a) {
  int temp = 1;
  if (a <= 0) return;
  while (a>1) {
    temp *= a;
    a--;
  }
  printf("a! = ", temp);
}
```

Ejemplo 5

```
void error() {
  printf("Error");
}
```

Ámbito de las variables

Variables locales

- Las variables existen dentro del bloque en que se definieron
  - Una variable definida en una función sólo existe en dicha función
  - Dos variables con el mismo nombre en dos funciones son variables distintas
- Variables locales: Variables definidas dentro de una función
  - Son las variables vistas hasta el momento

Variables locales

```
int sumar(int a, int b) {
  int c = a+b;

  return c;
}

int main() {
  int a=2, b=4, c=0;

  print("%d", sumar(a,b));
  print("%d", c);

return 0;
}
```

Ámbito de las variables
Variables globales

- Existen variables globales
  - Se definen fuera de todo bloque
  - Están definidas para todas las funciones
- Cuidado con mezclar variables locales y globales con el mismo nombre

Variables globales

```
int a, b, c;

void sumar() {
  int c = a+b;
}

int main() {
  a = 2;
  b = 4;
  sumar(a,b);
  print("%d", c);
  return 0;
}
```

Paso de parámetros

Paso de argumentos por valor (I)

- Los argumentos que se pasan a una función son copias locales
  - Se crea una variable local
  - Se copia el valor del argumento en esa variable
  - Por eso se dice que son argumentos por valor

Paso por valor

```
int sumar(int a, int b) {
  return (a+b);
}

int main() {
  int valor1=2, valor2=3;

  printf("%d", sumar(
      valor1, valor2))
  return 0;
}
```

Paso de parámetros

Paso de argumentos por valor (II)

- Una función no puede cambiar el argumento (es una copia)
  - Además, sólo puede devolver un valor

Paso por valor

```
void sumar(int a, int b, int c) {
  c = a+b;
}

int main() {
  int valor1=2, valor2=3;
  int c=0;
  sumar(valor1, valor2, c);
  printf("%d", c)
  return 0;
}
```

Paso de argumentos

Paso de argumentos por referencia

- Envía como argumento la dirección en la que se guarda la variable
  - Esto es, se envía un puntero
  - Hay que poner un antes de la variable
  - Se antepone un & antes de la variable al invocar la función
- Cualquier cambio a la variable pasada por referencia se propaga

Paso por referencia

```
void sumar(int a, int b,
    int *c) {
  *c = a+b;
}

int main() {
  int valor1=2, valor2=3;
  int c=0;
  sumar(valor1, valor2, &c)
      ;
  printf("%d", c)
  return 0;
}
```