
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL AVELLANEDA

# CLASE 3

## Funciones:

Porque dividir el programa en Módulos?

1-Cada módulo realiza una tarea específica.

2-La depuración es puntual del módulo.

3-Las modificaciones se centran en una porción del programa y no todo.

4-Se puede utilizar muchas veces.

5-Se gana independencia en cada módulo.

# CLASE 3

## Funciones:

Cada Módulo se llama FUNCION.

Una función es un programa que realiza una tarea determinada y bien acotada a la cual le pasamos datos y nos devuelve datos. Este programa se ejecuta cuando se lo llama o invoca.

Ej.: scanf(), printf()…etc..

# CLASE 3

## Funciones:

Getche()

No recibe datos y devuelve el caracter que leyó del teclado.

```
rta=getche();
```

El código que se ejecuta cuando se llama a la función *getche* está dentro de las bibliotecas que vienen con el compilador, el código no se conoce (independencia del código) pero se sabe qué es lo que hace y que lo hace bien.

# CLASE 3

## Funciones:

Cómo crear y utilizar funciones?

Se deben seguir 3 pasos:

1-Declaración

2-Llamada

3-Desarrollo

# CLASE 3

## Funciones: -Declaración

Se declaran todas las funciones propias que se van a utilizar durante el programa.

Tipo_devuelto Nombre_de_funcion ( tipo variable_1, tipo variable_2 , ..... , tipo variable_N)

- Tipo_devuelto: Es el tipo de dato que devuelve la función luego de terminada su ejecución
- Nombre_de_función: Es el nombre de la función
- tipo variable_1: Es el tipo y nombre de la variable que se le pasa a la función.

# CLASE 3

## Funciones: -Declaración

A las declaraciones que se realizan en este punto se las llama **prototipo** de la función.

Los **prototipos** son necesarios para que el compilador verifique que sean coherentes los tipos de datos declarados en el prototipo contra los que realmente se usan en la llamada a la función.

# CLASE 3

## Funciones: -Llamada

-Se llama a la función para usarla, es decir para que realice su trabajo.

-Desde cualquier parte de *main* o desde otra función.

-En el momento en que se produce la llamada a la función, se produce un salto en el flujo del programa. En ese momento se pasa a ejecutar el código de la función que va a terminar de ejecutarse al encontrar la sentencia *return* o llegar a la llave que cierra la función.

-Cuando la función finaliza, se sigue ejecutando el código de la función que produjo la llamada.

# CLASE 3

## Funciones: -Desarrollo

Esta es la parte en la cuál se escribe el código adentro de de la función donde se dan las órdenes para realizar una tarea determinada y devuelve o no un valor.

# CLASE 3

## Funciones:

**Ejercicio:**

**-Crear una función que permita ingresar un numero al usuario, lo retorne y lo muestre.**

# CLASE 3

Funciones:

**Ejercicio:**

**-Crear una función que reciba el radio de un círculo y retorne su área.**

# CLASE 3

## Funciones:- Ejemplo

Ingresar 2 números enteros y por medio de una función calcular la suma de los mismos

**PARAMETROS FORMALES** **PARAMETROS ACTUALES**

```
#include <stdio.h>
int suma(int a, int b); //(DECLARACIÓN,
void main(void)          //PROTOTIPO)
{
   int x,y,z;
   printf("ingrese numero a sumar: ");
   scanf("%d",&x);
   printf("ingrese numero a sumar: ");
   scanf("%d",&y);
   z=suma(x,y);   //(LLAMADA)
   printf("La suma es %d",z);
}
int suma(int a, int b)  //(DESARROLLO)
{
   int total;
   total=a+b;
   return total;
}
```

# CLASE 3

## Funciones:

**PARAMETROS FORMALES** **PARAMETROS ACTUALES**

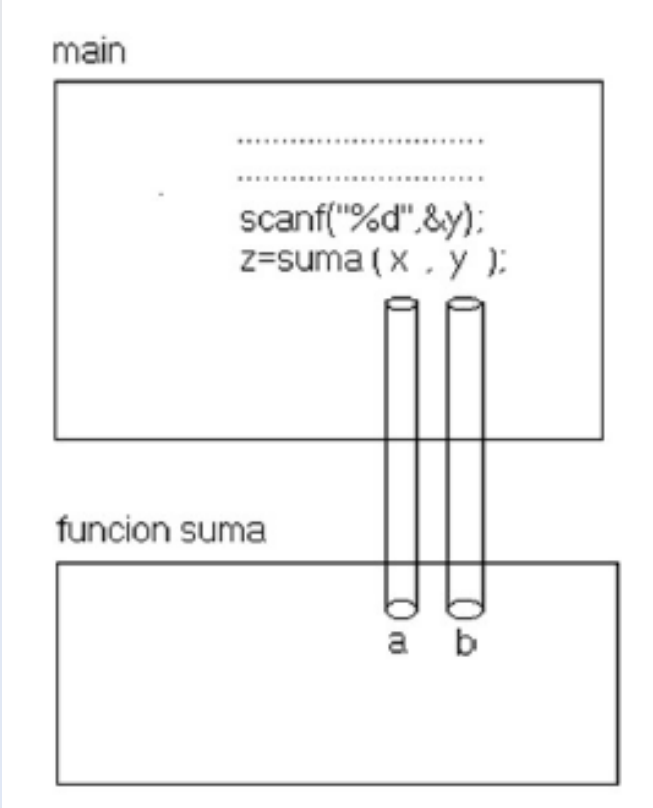
```
#include <stdio.h>
int suma(int a, int b); //(DECLARACIÓN,
void main(void)          //PROTOTIPO)
{
   int x,y,z;
   printf("ingrese numero a sumar: ");
   scanf("%d",&x);
   printf("ingrese numero a sumar: ");
   scanf("%d",&y);
   z=suma(x,y);   //(LLAMADA)
   printf("La suma es %d",z);
}
int suma(int a, int b)  //(DESARROLLO)
{
   int total;
   total=a+b;
   return total;
}
```

La instrucción *return* lo que hace es terminar la función y devolver el valor que se encuentra a continuación, en este caso devuelve el valor que tiene la variable “total”.

# CLASE 3

## Funciones:

Gráficamente podemos imaginarnos a la función trabajando con el *main* de acuerdo al siguiente esquema



# CLASE 3

## Funciones:

**PARAMETROS FORMALES** **PARAMETROS ACTUALES**

```
#include <stdio.h>
int suma(int a, int b); //(DECLARACIÓN,
void main(void)          //PROTOTIPO)
{
   int x,y,z;
   printf("ingrese numero a sumar: ");
   scanf("%d",&x);
   printf("ingrese numero a sumar: ");
   scanf("%d",&y);
   z=suma(x,y);   //(LLAMADA)
   printf("La suma es %d",z);
}
int suma(int a, int b)  //(DESARROLLO)
{
   int total;
   total=a+b;
   return total;
}
```

Las variables definidas como parámetros formales de la función son variables locales de la misma. Es decir solo se pueden utilizar dentro de la función.

# CLASE 3

## Funciones:-Ámbito de las variables

Dependiendo en el lugar donde se define una variable, serán las características de visibilidad y ámbito de validez que ellas tienen.

Todas las variables que se encuentren definidas dentro de las llaves de una función (*main* también es una función) tienen validez dentro de dicha función y se llaman variables locales, al ámbito donde dichas variables son visibles, se lo conoce como *scope*.

# CLASE 3

## Funciones:-Ámbito de las variables

Si se desea utilizar una variable desde cualquier función y durante el transcurso de todo el programa, esa es una variable **global**.

Las variables globales de definen fuera de cualquier función , normalmente debajo de los *include* que se colocan al comienzo del programa.

# CLASE 3

## Funciones:-Ámbito de las variables

```
#include <stdio.h>
int var;
void carga(void);
void main(void)
{
   int x,y,z;
   var=5;
   carga();
   printf("%d",var);
}
```

```
void carga(void)
{
   var=3;
}
```

La variable *var* es una variable global. Esta puede ser usada desde el *main* o desde la función “carga”.

# CLASE 3

## Funciones:-Ámbito de las variables

```
#include <stdio.h>
int var;
void carga(void);
void main(void)
{
   int x,y,z;
   var=5;
   carga();
   printf("%d",var);
}
```

```
void carga(void)
{
   var=3;
}
```

Cuando el programa comienza se le asigna a la variable *var* el valor 5, luego se llama a la función y se le asigna el valor 3. Finalmente cuando se muestra el valor de la variable *var* aparecerá un 3 que es el último valor asignado.

# CLASE 3

## Funciones:-Ámbito de las variables

```
#include <stdio.h>
int var;
void carga(void);
void main(void)
{
   int x,y,z;
   var=5;
   carga();
   printf("%d",var);
}
```

```
void carga(void)
{
   var=3;
}
```

Se debe tener en cuenta que al tener una variable definida en forma global se puede acceder a ella desde cualquier parte del programa ya sea para asignarle un nuevo valor o para leerla.

# CLASE 3

## Funciones:-Ámbito de las variables

Modificación al programa anterior.

¿Cuál se supone que será el resultado mostrado en pantalla?

```
#include <stdio.h>
int var;
void carga(void);
void main(void)
{
   int x,y,z;
   var=5;
   carga();
   printf("%d",var);
}
```

```
void carga(void)
{
   int var;
   var=3;
}
```

# CLASE 3

Funciones:-Ámbito de las variables

Modificación al programa anterior.

¿Cuál se supone que será el resultado mostrado en pantalla?

```
5
```

**porque var se declara dentro de la función carga y no es la misma var declarada en main**

# CLASE 3

## Funciones:-Ámbito de las variables - Reglas

1-La variable local tiene validez solo dentro de la función en la que fue definida, fuera de ella no existe.

2-Si existe una variable global y otra local con el mismo nombre tiene validez la variable local dentro de esa función.

3-Si en la función “a” tenemos la variable “nombre” y en la función “b” existe la variable “nombre” no existe problema de ningún tipo ya que ambas son variables locales de funciones distintas (aunque tengan el mismo nombre).

4-Es conveniente usar variables con distinto nombre para evitar confusiones.

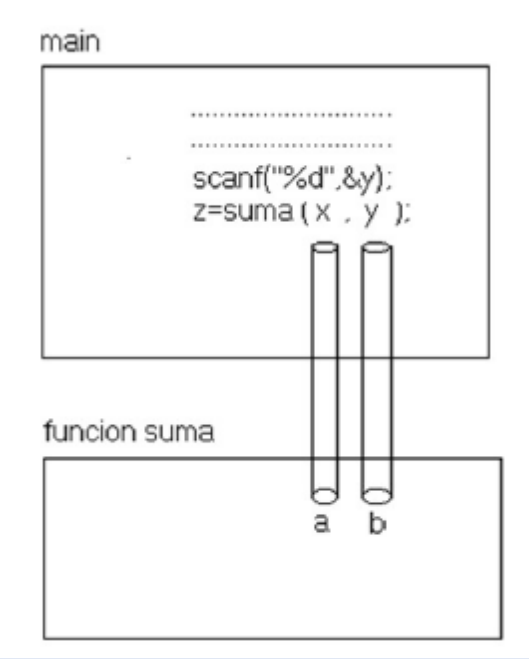
# CLASE 3

## Funciones:-Parámetros por Valor y Por Referencia

Se llama pasaje **por valor** cuando a la función se le pasa como parámetro actual el valor de la variable.

**Ej: el valor de la variable x, pasa a la variable a**

**Y el valor de la variable y, pasa a la variable b**



# CLASE 3

## Funciones:-Parámetros por Valor

```
#include <stdio.h>
void muestra(int x,int y);
void main(void)
{
  int x,y;
  printf("Ingrese un numero entero");
  scanf("%d",&x);
  printf("Ingrese un numero entero");
  scanf("%d",&y);
  muestra(x,y);
  printf("\n-----valores dentro de main----");
  printf("\nx vale %d \ny vale %d",x,y);
}
```

```
void muestra(int x,int y)
{
x=y;
printf("\n-----valores dentro de la funcion----");
printf("\nx vale %d \ny vale %d",x,y);
}
```

# CLASE 3

## Funciones:-Parámetros por Valor

```
Ingrese un numero entero2
Ingrese un numero entero3

-----valores dentro de la funcion----
x vale 3
y vale 3
-----valores dentro de main----
x vale 2
y vale 3
Process returned 19 (0x13)   execution time : 6.615 s
Press any key to continue.
```

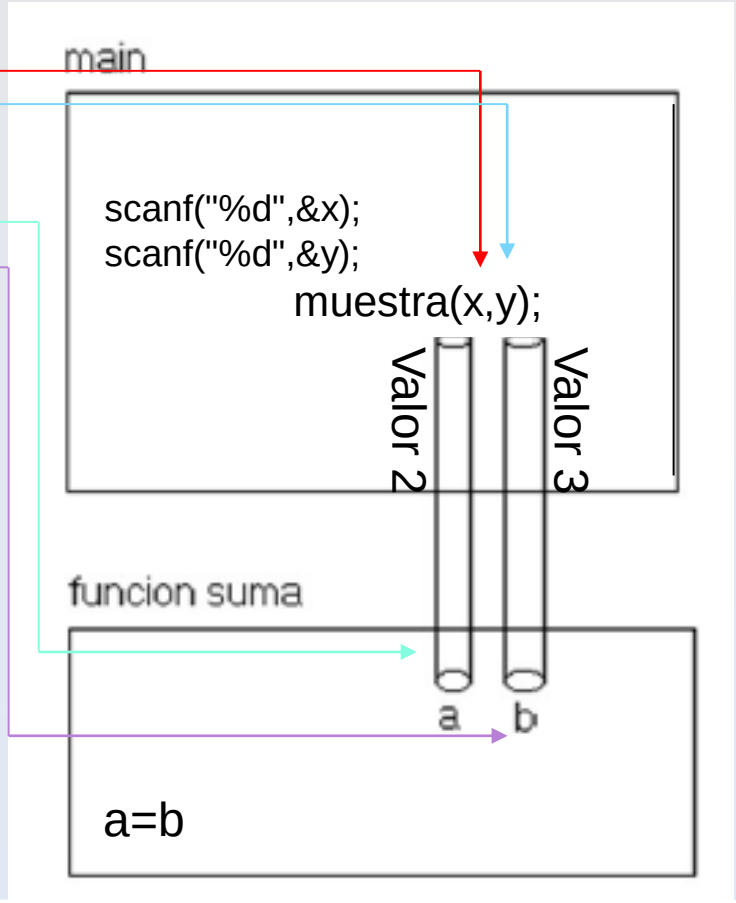
Esto ocurre por que aunque las variables tengan el mismo nombre, son distintas ya que son locales de cada función y como sabemos tienen validez solamente dentro de la función en la cual están definidas.

# CLASE 3

## Funciones:-Parámetros por Valor

```
Ingrese un numero entero2
Ingrese un numero entero3

-----valores dentro de la funcion----
x vale 3
y vale 3
-----valores dentro de main----
x vale 2
y vale 3
Process returned 19 (0x13)   execution
Press any key to continue.
```




**Si se llaman a,b o x,y son variables locales a cada función.**

# CLASE 3

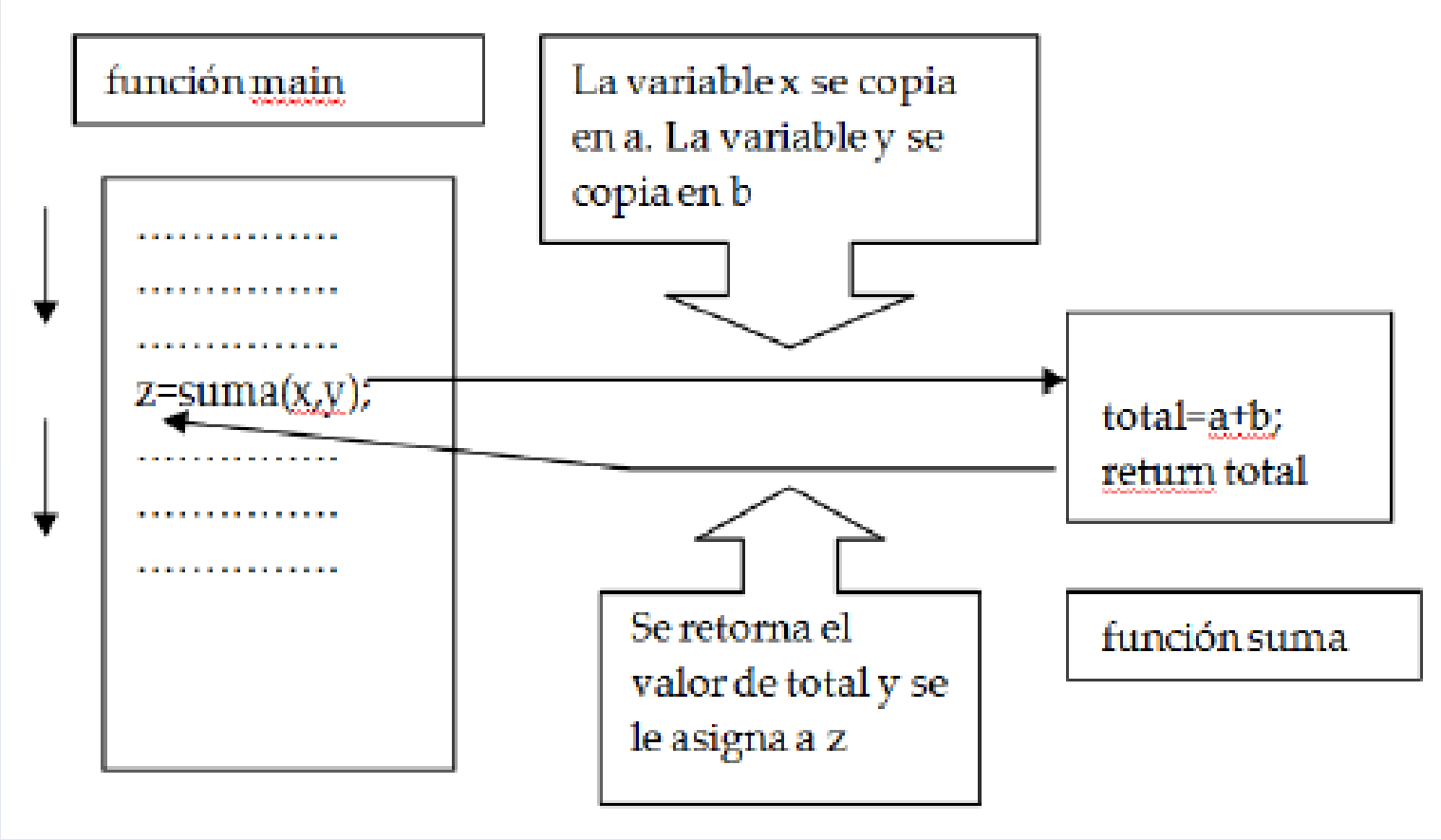
## Funciones:-Diagrama de Flujo

Vamos a analizar cómo es la ejecución de un programa que utiliza funciones como por ejemplo el de la función suma.

```
#include <stdio.h>
int suma(int a, int b);
void main(void)
{
   int x,y,z;
   printf("ingrese numero a sumar: ");
   scanf("%d",&x);
   printf("ingrese numero a sumar: ");
   scanf("%d",&y);
   z=suma(x,y);
   printf("La suma es %d",z);
}
int suma(int a, int b)
{
   int total;
   total=a+b;
   return total;
}
```

# CLASE 3

## Funciones:-Diagrama de Flujo



# CLASE 3

## Funciones:-Funciones y Parametros

## 1)Recibe parámetros y no devuelve nada.

```
#include <stdio.h>
#include <stdlib.h>
void suma(int a, int b);

int main()
{
   suma(2,3);
   return 0;
}
```

```
void suma(int a, int b)
{
    int rta;
    rta = a+b;
    printf("La suma es %d ", rta);
}
```

# CLASE 3

## Funciones:-Funciones y Parametros

1)No recibe parámetros y no devuelve nada.

```
#include <stdio.h>
#include <stdlib.h>
void suma(void);
int main()
{
   suma();
   return 0;
}
```

```
void suma()
{int a,b,rta;
    a=2;
    b=3;
    rta = a+b;
    printf("La suma es %d ", rta);
}
```

# CLASE 3

## Funciones:-Funciones y Parametros

### 1)No recibe parámetros y devuelve algún valor.

```
#include <stdio.h>
#include <stdlib.h>
int numero(void);
int main()
{ int valorDevuelto;
   valorDevuelto=numero();
   printf("Valor= %d ", valorDevuelto);
   return 0;
}
```

```
int numero()
{int valor;
   valor = rand() % 11;
   return valor;

}
```


UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL AVELLANEDA