



**UNRN**

Universidad Nacional  
de Río Negro



| unrionegro

# Programación en C #3

**UNRN**

Universidad Nacional  
de Río Negro

**4**  
2023





**Recordatorio #1:**

**La entrega de los TP  
es con el formulario**





**Recordatorio #2:**

**No cierren el Pull  
Request**

---

# Funciones II

---

# funciones

O como hacer de un problema grande,  
varios más chicos

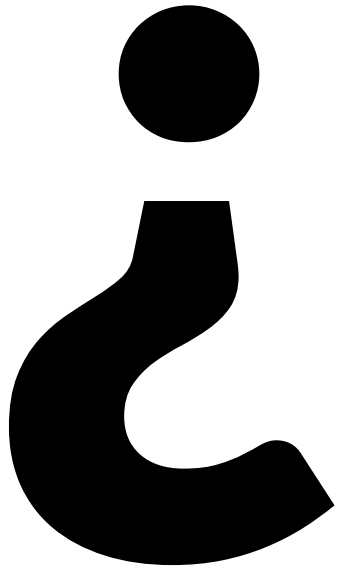
---

---

$$f(x) = x^2$$

Dado un valor de  $x$ , obtenemos  $x^2$

---



**Que tiene que ver  
esto con la  
programación**





**La noción es la  
misma.**

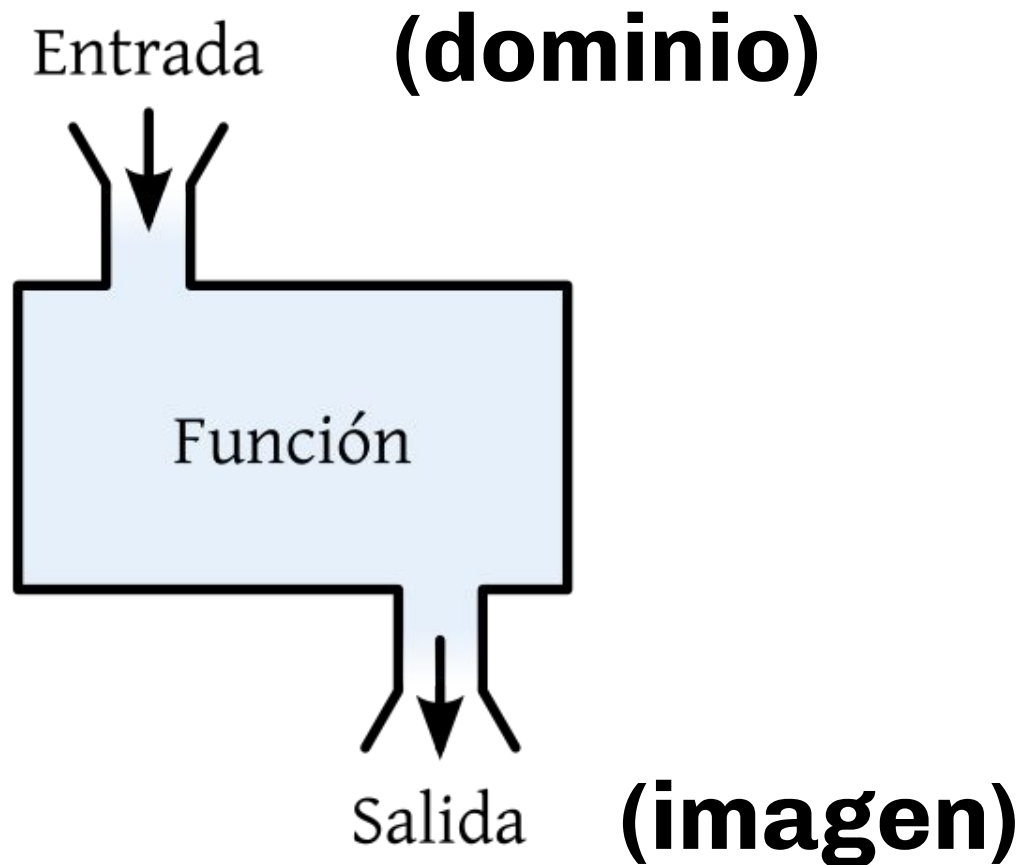
---

# Para simplificar un problema

---

# Funciones

# ¿Como es una función?



**entrada**

---

**tipo    funcion(tipo nombre, ...)**

---

**salida**

**de entrada**  
**tipo nombre**  
**todos los necesarios**

**de salida**  
**tipo**  
**solo uno\***

**\*mas adelante vamos a ver como nos podemos saltar esto**

`tipo_retorno` `identificador`(`lista_argumentos`)



**en donde cada ítem separado por coma de  
lista\_argumentos es**

**tipo\_argumento identificador**

# ¿Y la salida?

**return variable;**

**funciona igual que  
Python**

**Y la salida de la función**

```
int suma(int op1, int op2)
{
    int suma = op1 + op2;
    return suma;
}
```

# ¿Puede no tener salida?

---

**si**

---

**Cuando no devuelve nada, se  
indica con la palabra  
reservada**

**void**

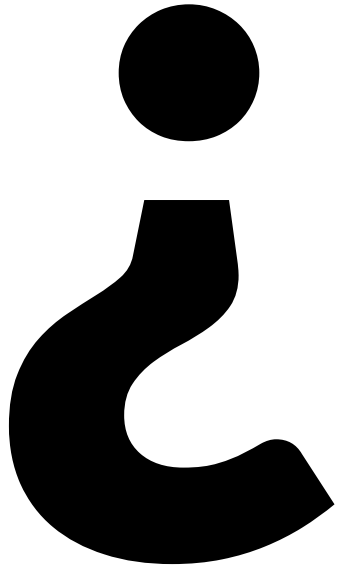
```
void mi_funcion()  
{  
    bloque;  
}
```

**Técnicamente, esto se llama 'procedimiento'**



# Un ejemplo de procedimiento

```
void muestra_mensaje()  
{  
    printf("Hola Mundo!\n");  
}
```



**Los argumentos,  
son la única  
forma de pasar  
información a  
una función**



---

**nope\***

---

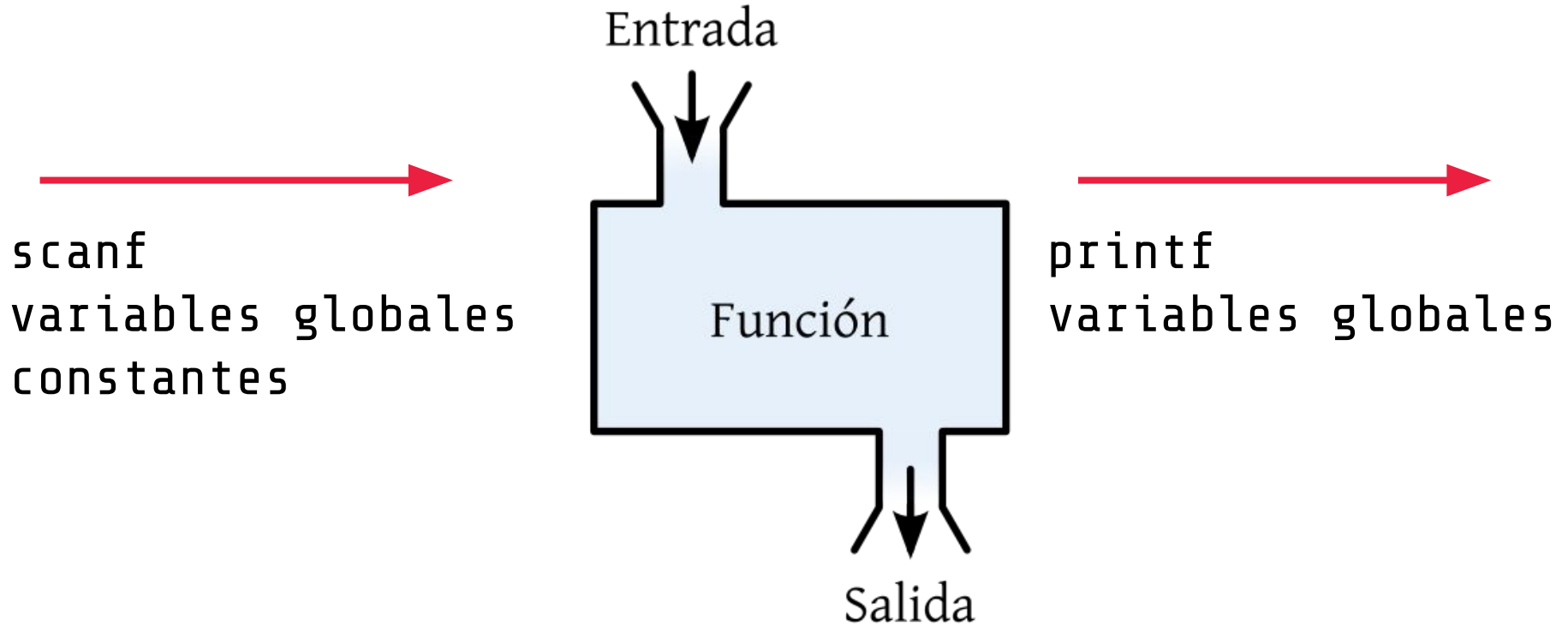
---

**\*Por ahora**

---

# Solo argumentos

# Otras formas de pasar información



## Salvo para funciones como por ejemplo:

```
int leer_entero()
{
    int valor = 0;
    scanf("pasame un valor entero %d", &valor);
    return valor;
}
```



**¿Preguntas?**



**¿El orden de las  
funciones es  
importante?**

```
void function_1()  
{  
    function_2();  
}
```

```
void function_2()  
{  
    function_1();  
}
```

# Prototipos de función

**Es la función definida  
pero sin cuerpo.**

```
void mi_funcion();
```

```
void funcion_1();  
void funcion_2();
```

```
void funcion_1()  
{  
    funcion_2();  
}  
void funcion_2()  
{  
    funcion_1();  
}
```



# **Un ejemplo más concreto**

# Un ejemplo más completo

```
void saludo();
```

 Es necesario

```
int main()
```

```
{
```

```
    saludo();
```

```
    return 0;
```

```
}
```

 Se usa antes que se implementa

```
void saludo()
```

```
{
```


```
    printf("Hola Mundo");
```

```
}
```

# El 'prototipo' no es necesario en este caso

```
void saludo()  
{  
    printf("Hola Mundo");  
}
```

Pero acá se  
implementa antes  
de usarse



```
int main()  
{  
    saludo();  
    return 0;  
}
```

Se usa pero ya esta  
implementada





---

# Conectando funciones



**una función  
puede llamar a  
otra**



---

**isi!**

---

**Y con esto pueden  
simplificar sus  
programas**

---

# Cuestiones generales

# Los parámetros con un nombre descriptivo

# Descripción de la función

```
/*  
* Toda funcion debe tener un comentario sobre que describa que  
es lo que hace.  
Indicando que hace cada argumento.  
Esta funcion se encarga de sumar dos numeros enteros  
@param termino1 es el primer termino de la suma  
@param termino2 es el segundo termino a sumar  
@returns la suma de ambos terminos  
*/  
int suma(int termino1, int termino2);
```

**Aquellas funciones  
que terminen  
describiendo con un  
“y”**



**probablemente  
sean dos funciones**

# Una función que probablemente sean dos

Esta función se encarga de sumar y multiplicar

---

# Consideraciones generales

# **Una sola instrucción**

## **return**

## **por función**

**Todas las funciones  
van con comentario  
de documentación**



**¿Preguntas?**

---

# **Descomposición funcional**

---

# Cohesión



# Reusabilidad

# Legibilidad



**¿Preguntas?**

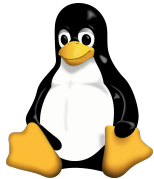
---

# La compilación (y ejecución del programa)

# Para ejecutar



**`./a.exe`**



**`./a.out`**

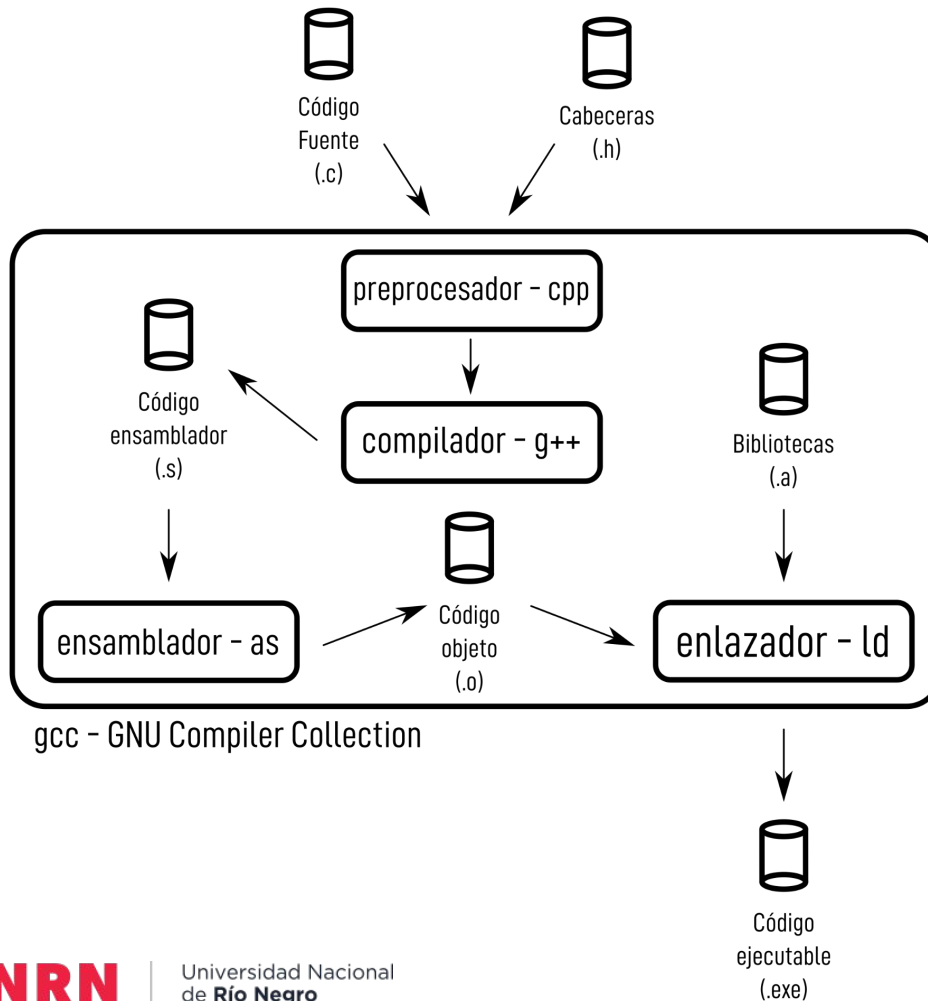
**¿Pero qué pasa  
cuando compilamos?**

# ¿expecto binarium?!



**¡Solo un conjunto de  
programas!**





**Es posible  
llamar a cada  
programa  
individualmente**

# **Pero si hacemos**

```
$> gcc --verbose programa.c
```

# **vemos por qué no es viable**

—

**Usualmente uno  
quiere indicar el  
nombre del binario  
generado.**

```
$> gcc -o binarium programa.c
```

# Opciones de compilador recomendadas

---

# **-Wall**

Activa más mensajes de advertencia (todos)

---

---

# **-Werror**

Las advertencias ahora son errores que frenan la compilación

---



**¿Preguntas?**



---

# El preprocesador

**#include <archivo>**  
**#include "archivo"**

**#define PALABRA valor**

(por  
ejemplo)

```
#define TRUE 1  
#define FALSE 0
```

---

# ¡A trabajar!

# TP2

**unrn.edu.ar**

**UNRN**

Universidad Nacional  
de **Río Negro**



| **unrionegro**