

A photograph showing three young adults (two men and one woman) sitting around a table, looking down at some papers or documents they are holding. They appear to be engaged in a collaborative activity, possibly reviewing course materials or assignments. The lighting is warm and focused on their faces and the documents.

**UNRN**

Universidad Nacional  
de Río Negro



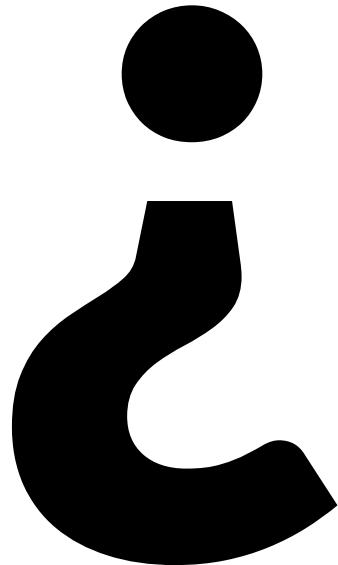
| unrnionegro

# Programación en C #2

**UNRN**

Universidad Nacional  
de Río Negro





**¿Compiló el  
saludo.ç  
del TPO?**



**Es todavía más  
importante que  
todos estemos  
compilando y  
ejecutando**

---

# Abran hilo

---

# ¿Preguntas?



—

# Volvamos con el hola mundo II

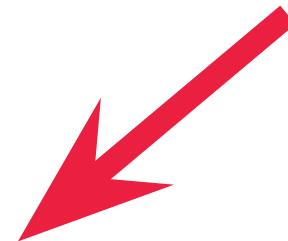
```
#include <stdio.h>
int main()
{
    printf("Hola mundo C. \n");
    return 0;
}
```

-

# La anatomía de un programa

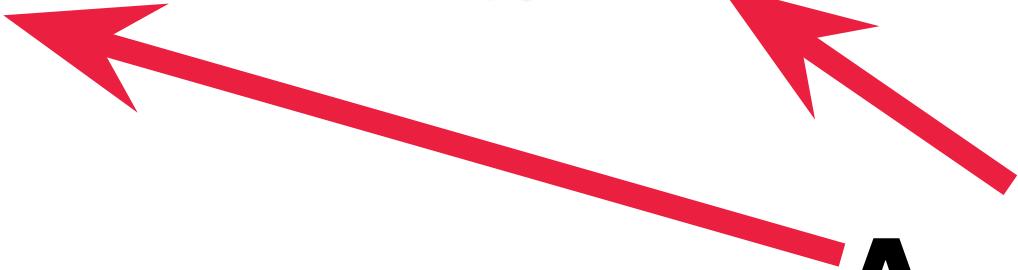
# 1 - El punto de entrada

```
#include <stdio.h>
int main()
{
    printf("Hola mundo C. \n");
    return 0;
}
```



# 2 - Sentencias

```
#include <stdio.h>
int main()
{
    printf("Hola mundo C. \n");
    return 0;
}
```



A B

# 3 - Bloques

```
#include <stdio.h>
int main()
{
    printf("Hola mundo. \n");
    return 0;
}
```

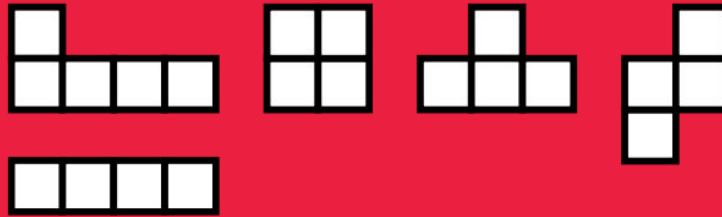
# “parecido” a Python

```
#include <stdio.h>
int main()
{
    printf("Hola mundo C. \n");
    return 0;
}
```

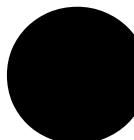


---

# Los bloques básicos

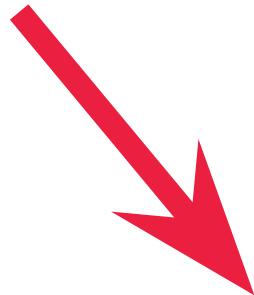


# Variables

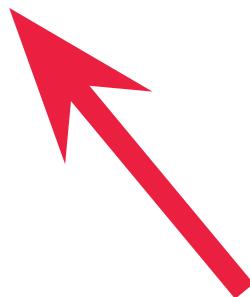


# Qué son las variables

# declaración



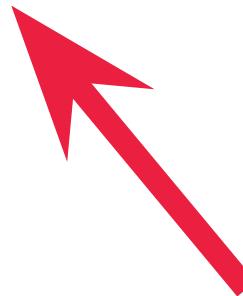
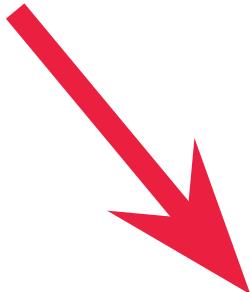
```
int variable;
```



# **Solo al principio de la función**

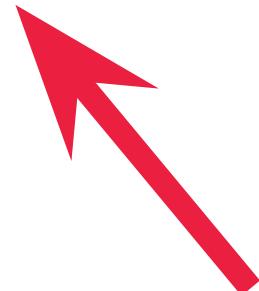
# asignación

**variable = valor;**



# combinado

```
int variable = inicial;
```



# **tipos de datos**



```
int variable;
```

# Números enteros

**short int**

**int**

**long int**

**long long int**

# Números enteros

**short int**

**int**

**long int**

**long long int**

opcionalmente  
unsigned

# Números reales

**float**

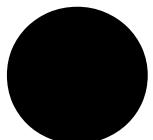
**double**

# Caracteres

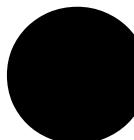
char

—

# Al asignar



**Qué es la  
operación de  
asignación**



# Qué asignamos

# **Hay dos partes**

**l-value = r-value;**

**l-value  
es el “destino”**

**Siempre y exclusivamente una variable**

**I-value  
es el “origen”**

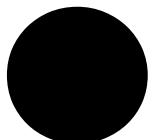
---

**¡Tienen que ser  
del mismo tipo!**

---

# *Por ahora*





**¿Qué puede ser  
un r-value?**

# Variable

# Literales

# Números enteros

1234 - decimales

# Formas alternativas

**0b1010**

- binario [0-1]

**0173**

- octal [0-7]

**0xCAFE**

- hexadecimal [0-16]

binario	0100 1101 0010
octal	2322
decimal	1234
hexadecimal	4D2

# Números reales

**float**                    **1.0f**

**double**                **1.0**

# caracteres

literal: 'a'

Ver tabla ASCII para ver cuales son los validos

**No dejan de ser un  
número!**

Caracteres ASCII de control			Caracteres ASCII imprimibles		
00	NULL	(carácter nulo)	32	espacio	64
01	SOH	(inicio encabezado)	33	!	65
02	STX	(inicio texto)	34	"	66
03	ETX	(fin de texto)	35	#	67
04	EOT	(fin transmisión)	36	\$	68
05	ENQ	(consulta)	37	%	69
06	ACK	(reconocimiento)	38	&	70
07	BEL	(timbre)	39	'	71
08	BS	(retroceso)	40	(	72
09	HT	(tab horizontal)	41	)	73
10	LF	(nueva línea)	42	*	74
11	VT	(tab vertical)	43	+	75
12	FF	(nueva página)	44	,	76
13	CR	(retorno de carro)	45	-	77
14	SO	(desplaza afuera)	46	.	78
15	SI	(desplaza adentro)	47	/	79
16	DLE	(esc.vínculo datos)	48	0	80
17	DC1	(control disp. 1)	49	1	81
18	DC2	(control disp. 2)	50	2	82
19	DC3	(control disp. 3)	51	3	83
20	DC4	(control disp. 4)	52	4	84
21	NAK	(conf. negativa)	53	5	85
22	SYN	(inactividad sínc)	54	6	86
23	ETB	(fin bloque trans)	55	7	87
24	CAN	(cancelar)	56	8	88
25	EM	(fin del medio)	57	9	89
26	SUB	(sustitución)	58	:	90
27	ESC	(escape)	59	;	91
28	FS	(sep. archivos)	60	<	92
29	GS	(sep. grupos)	61	=	93
30	RS	(sep. registros)	62	>	94
31	US	(sep. unidades)	63	?	95
127	DFI	(sobrescribir)			-

# Caracteres especiales

# ¡Secuencias de escape!

\n - salto de línea

\t - tabulador

\\ - la barra invertida misma

\ " - la comilla doble

\ ' - la comilla simple

Como darle un significado alternativo a los mismos caracteres

# ¿Qué podemos decir de variable?

```
variable = 'A';
```

**Le vamos a dar uso  
en un rato, nomá**

**Los literales pueden  
ir donde se espera  
un valor**

# Instrucciones como r-value

**Porque sin asignar  
el resultado es  
como si no pasara  
nada**

# **Operadores aritméticos**

# Aparte de los clásicos

+ - / \* %

**a + 1**

**b \* c**

**2 / d**

**var++;**     **var = var + 1**

**var--;**     **var = var - 1**

# Llamadas a funciones\*

**suma(argumento);**

# **Estos son r-values**

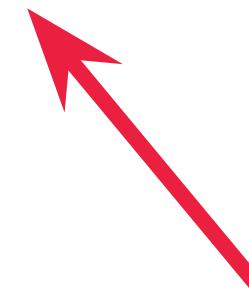
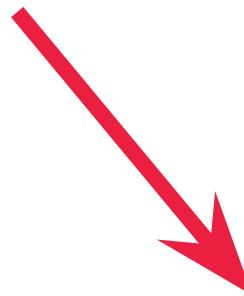
# Porque la idea es que se pueden combinar

```
variable = 1 + suma(3) * variable;
```

Y agrupar instrucciones con paréntesis

# ¡No hacen lo mismo!

```
variable = (1 + suma(3)) * variable;
```



Uno agrupa instrucciones y el otro los argumentos

**Cada operación  
individual, tiene una  
entrada y salida**

**En donde ambos  
tienen un tipo**

# ¿Que resultado obtenemos?

```
int a = 10;  
float b = 2.5;  
int c = a * b;
```

Para empezar, **no mezclar tipos es una buena idea**

# ¿Preguntas?



# Cadenas

(primera pasada)

# Tipo char[ ]

# Literales

## "programacion 1"

# **Operaciones**

## **Ninguna\***

**Por ahora :-D**

```
char[] cadena;
char[] cad = "programacion 1"
```

# Lo mínimo para interactuar con el usuario

# ¿Preguntas?



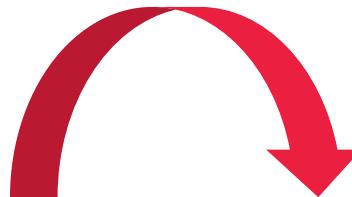
# Interactuar con el usuario

```
#include <stdio.h>
```

# printf

# ¿Como se usa?

```
int variable = 10;  
printf("cadena de formato %d", variable);
```



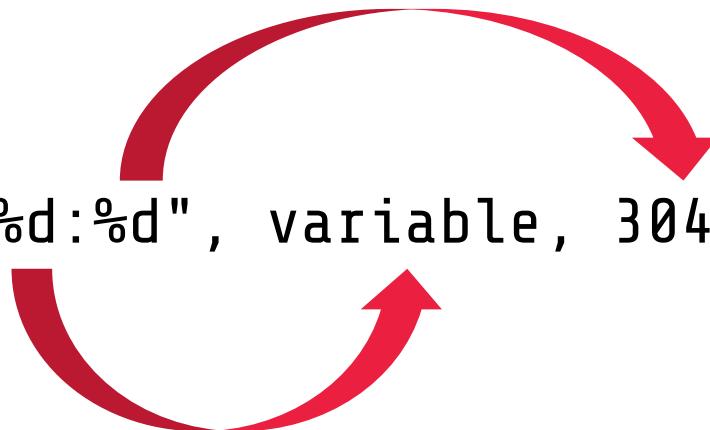
Y la salida queda como:

```
cadena de formato 10
```

# Pueden ir varios valores

```
int variable = 10;
```

```
printf("%d:%d", variable, 304);
```



**El marcador de  
formato tiene  
muchas opciones  
opcionales**

# **Lo importante es**

**Hay un marcador de  
formato por tipo de  
variable**

# Los más importantes

%d %i	int	Números enteros
%u	unsigned int	Números enteros sin signo
%e %f %g	float/double	Números decimales
%c	char	Caracter individual
%s	char[ ]	Cadena de caracteres

**scanf**

# De a una variable por vez

```
int variable;
```

```
scanf( "%d", &variable);
```



**Ya vamos a ver que  
es el  
&variable**

**Usa los mismos  
marcadores de tipo  
que printf**

**Es poco probable usar  
otros calificadores  
aparte de  
%d y %f**

# ¿Preguntas?



# Operaciones de comparación

$n > m$

$n < m$

$n \geq m$

$n \leq m$

$n == m$

**Dan como resultado  
un número entero  
int**

# ¿Pero que significa?

**numero > 4**

**verdadero**



**4**

**numero >= 4**

**verdadero**



**numero < 7**

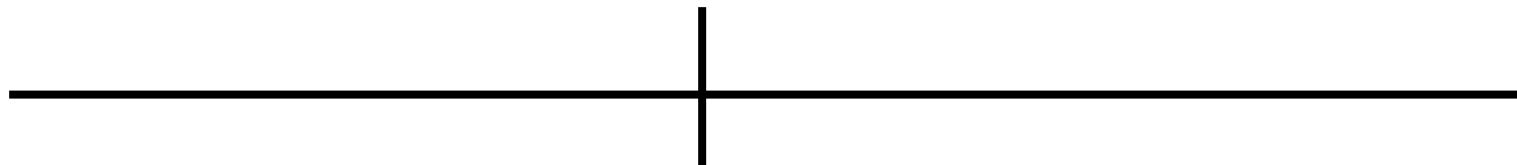


**numero <= 7**



**numero == 5**

**verdadero**



**5**

**numero != 5**

**verdadero**



**verdadero**



**5**

**0 es falso**

**1 es verdadero\***

**\*Cualquier otra cosa que no sea cero es verdadero**

**También esta  
#include <stdbool.h>**

# Sin olvidarnos qué if acepta int

```
#include <stdio.h>
#include <stdbool.h>

int main ()
{
    bool b = true;

    if ( b )
    {
        printf ( "Si!\n" );
    }
    else
    {
        printf ( "No!\n" );
    }
    return 0;
}
```

**Los bool no forman parte del C original**

# Operadores lógicos

para condiciones más complejas

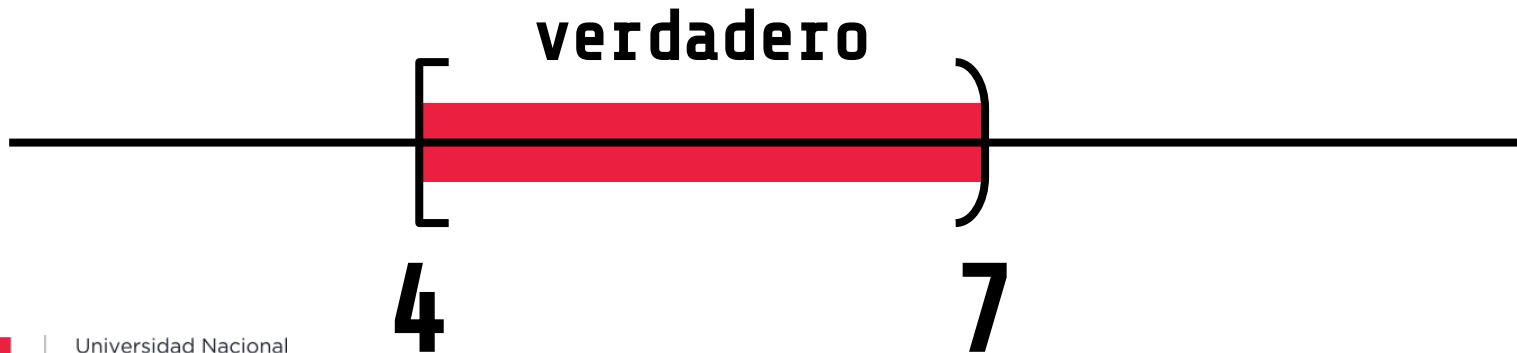
! & ||

# negación y lógico o lógico

# **Un ejemplo rápido**

*al que volveremos en instantes nada más*

**(nota >= 4) && (nota < 7)**



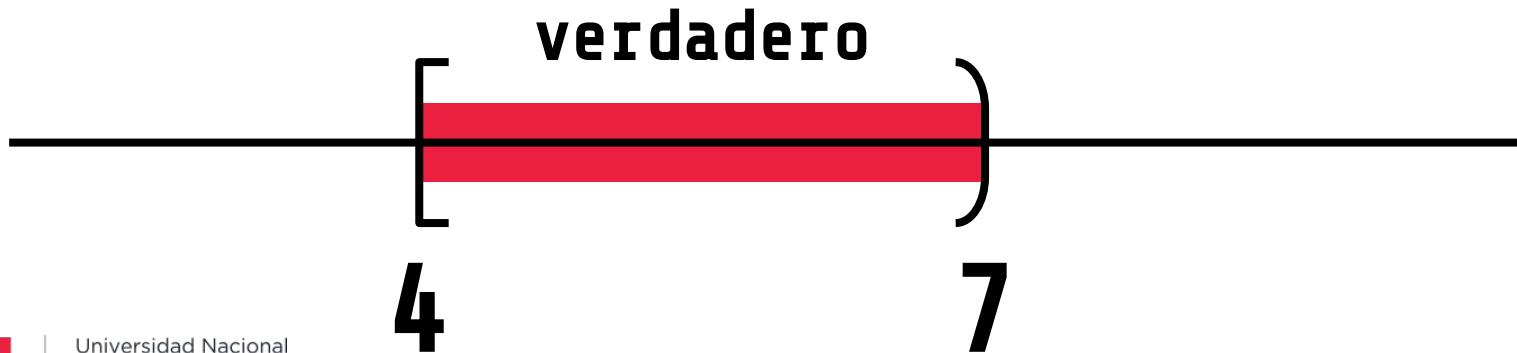
# ¿Preguntas?



# Tomando decisiones

```
if (condición)
{
    bloque verdadero;
}
```

**(nota >= 4) && (nota < 7)**



```
if ((nota >= 4) && (nota < 7))  
{  
    bloque;  
}
```

```
if (condición)
{
    bloque verdadero;
}
else
{
    bloque falso;
}
```

```
if (condición)
{
    bloque condición verdadero;
}
else if(condición_2)
{
    bloque condicion_2 verdadero;
}
```

-

# Cuestiones a tener en cuenta:

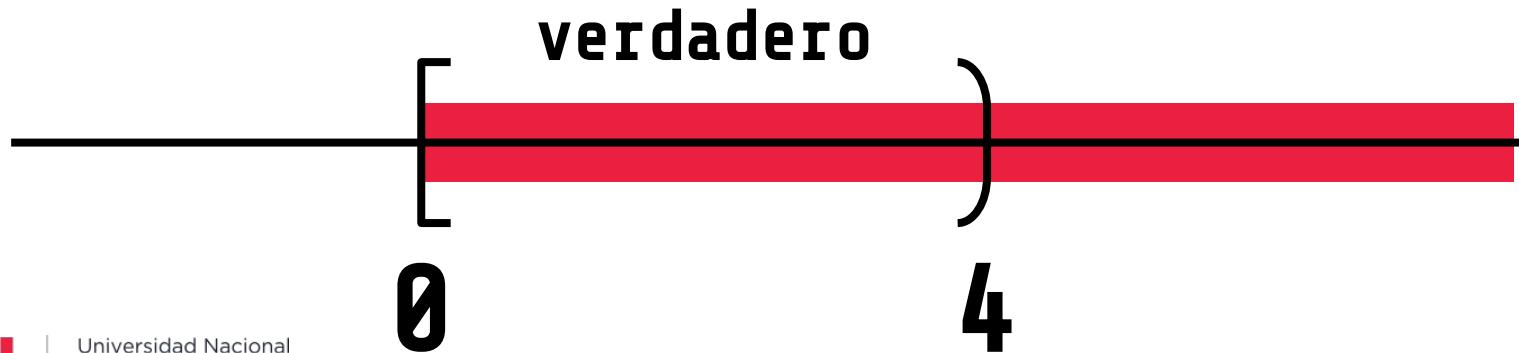
# ¡Cuidado con las superposiciones!

```
int numero = scanf("%d");

if (numero >= 0)
{
    printf("Es cero");
}
else if (numero < 4)
{
    printf("menor a 4");
}
else
{
    printf("que va aca?");
}
```

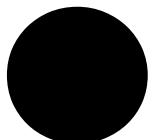
Pueden cambiar el significado del condicional

**numero >= 0**



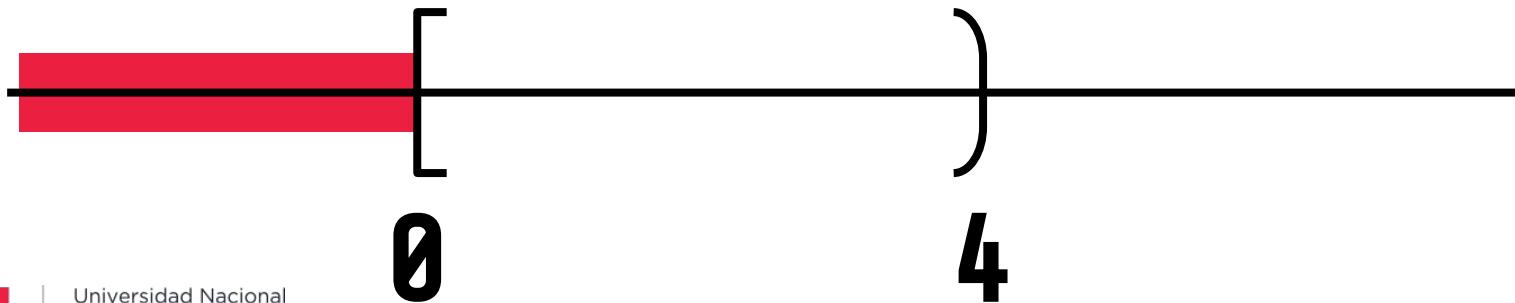
# ¿Preguntas?





# Como solucionarlo

**! (numero >= 0)**



**numero < 4**



# ¿Preguntas?



**¿Cuántos caminos nos da  
una condición individual?**

## Con la condición compuesta y más compleja

```
if ((numero >= 4) && (numero < 7)){
```

**Solo obtenemos un resultado**

# ¿Preguntas?



# Lazos

# Repetiendonos

```
while (condición)
{
    bloque;
}
```

```
int numero = 0;
while(numero < 10)
```



```
do
{
    bloque;
}
while (condición);
```

```
int numero = 10;
do
{
    numero = numero -1;
}
while (numero > 0);
```

A classic Looney Tunes scene. Wile E. Coyote is shown in mid-stride, running towards the right. He has a determined expression, his mouth is open as if he's shouting, and his arms are pumping. He wears his signature brown suit and bow tie. In the upper left, Road Runner stands on a rocky ledge, looking back over his shoulder at Wile E. with a smug, knowing smile. He has his signature blue feathers and long, thin body. The background is a simple yellow gradient. Two speech bubbles are present: one above Wile E. containing code for a while loop, and one above Road Runner containing code for a do-while loop.

```
while (not edge) {  
    run();  
}
```

```
do {  
    run();  
} while (not edge);
```

—

# Lazos mas comunes

# De menor a mayor

```
int numero = 0;
while (numero < 10)
{
    printf("numero:%d", numero);
    numero = numero + 1;
}
```

**Este es uno de los mas comunes.**

# De menor a mayor

```
int numero = 0;  
do  
{  
    printf("numero:%d", numero);  
    numero = numero + 1;  
}  
while (numero < 10);
```

**Este es uno de los más comunes.**

# De mayor a menor

```
int numero = 10;
while (numero > 0)
{
    printf("numero:%d", numero);
    numero = numero - 1;
}
```

**Una variación simple**

# Cuando tenemos **menos** idea de cuantas veces ejecutar

```
int numero = 10;
bool bandera = true;
while (bandera)
{
    printf("numero:%d", numero);
    numero = numero - 1;
    if (numero > 0)
    {
        bandera = false;
    }
}
```

**¡El if interno puede ser mucho más complejo!**

# ¡La condición puede ser compuesta!

```
int numero = 10;
while (((numero > 0) && (numero < 20)))
{
    printf("numero:%d", numero);
    numero = numero + 1;
    if (numero % 2 == 0)
    {
        numero = -numero * 2;
    }
}
```

Aunque este ejemplo sea *un poco rebuscado* :-)

# Suma de una cantidad abierta de números

```
int numero;
int cantidad = 0;
int suma = 0;
bool activo = true;
while (activo)
{
    printf("numero %d", cantidad);
    scanf("%d", &numero);
    cantidad = cantidad + 1;
    if ((cantidad > 100) && (numero < 0))
    {
        activo = false;
    }
}
```

**¡El if interno puede ser mucho más complejo!**

# Un toque de funciones

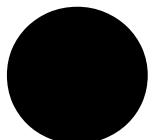
**Volvamos con el  
hola mundo III**

```
#include <stdio.h>
int main()
{
    printf("Hola mundo C. \n");
    return 0;
}
```

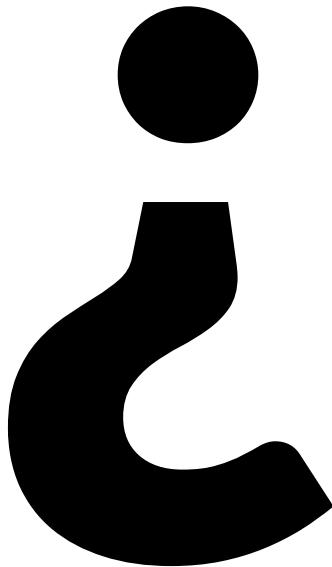
# Anatomía de una función

```
tipo_retorno nombre(lista_argumentos);  
  
tipo_retorno nombre(tipo_argumento nombre_argumento)  
{  
    return valor_generado;  
}
```

\*Ya vimos como llamar a una



**cuál es el  
propósito de una  
función**



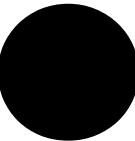
?

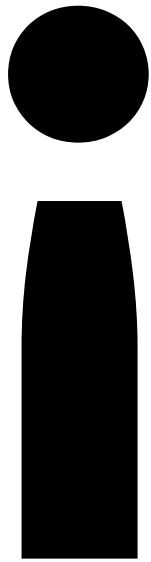
**hacer**

**scanf**

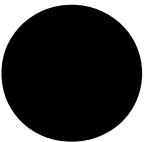
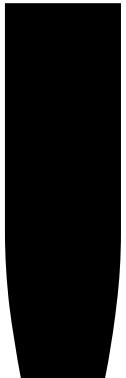
**printf**

?





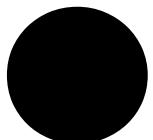
nope\*



**\*a no ser que ese sea  
explícitamente su  
proposito**

# ¿Cuál es el propósito de una función así?

```
int pedir_entero();
```

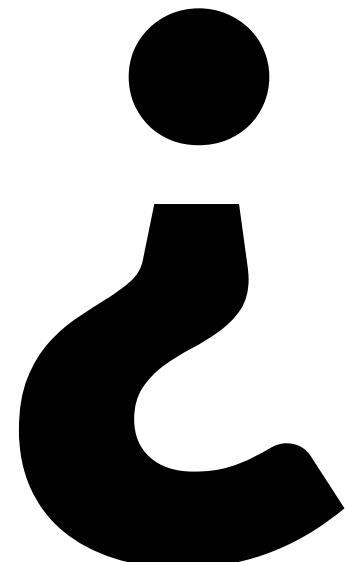
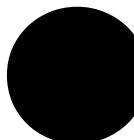


**Puede no tener  
argumentos**

---

**si**

---

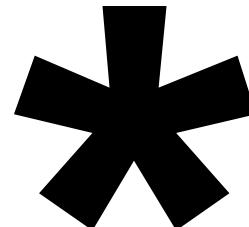
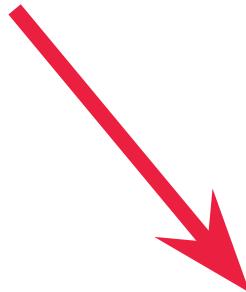


**Puede no tener  
retorno**

---

**si**

---



**void sin\_retorno();**

**\*Más adelante vamos a ver para qué existe algo así**

# Consideraciones generales

**Una sola instrucción  
return  
por función**

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

# ¿Como es un comentario documentación completa?

```
/**  
 * La suma lenta es una forma rebuscada  
 * de reinventar la rueda en pos del  
 * aprendizaje.  
 * @param n es el primer sumando  
 * @param m es el segundo sumando  
 * @return la suma de los sumandos  
 */  
int suma_lenta(int n, int m);
```

**El miércoles vamos a ver un par más de detalle de esto**

# ¿Preguntas?



---

¿TP2?

**unrn.edu.ar**

**UNRN**

Universidad Nacional  
de Río Negro



| unrnionegro