

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 #1

UNRN

Universidad Nacional
de Río Negro



comisión 1

Ivan Nomdedeu
inomdedeu@unrn.edu.ar

Daniel Teira
deteira@unrn.edu.ar

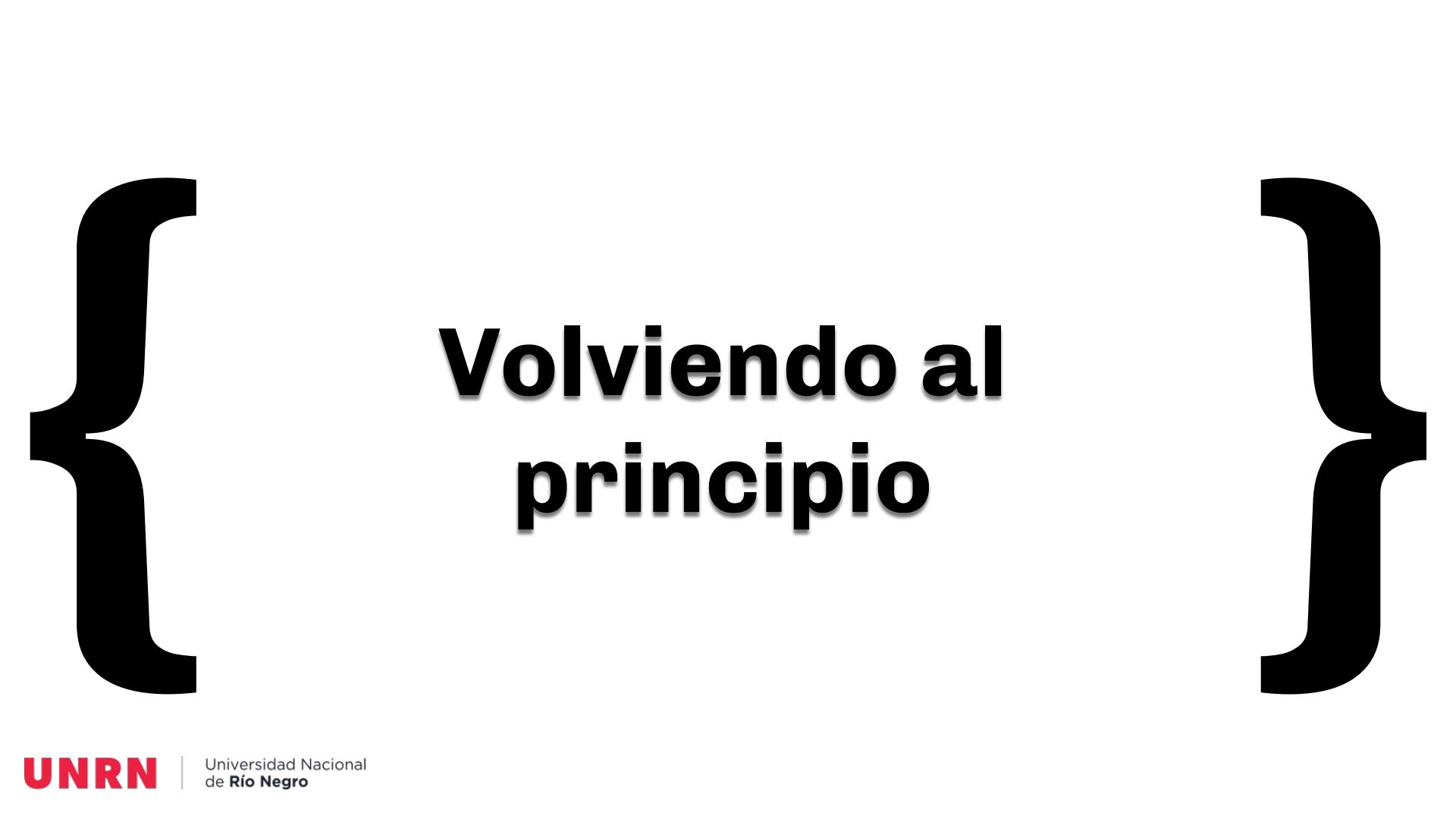
comisión 2

Martín René Vilugrón
mrvilugron@unrn.edu.ar

comisión 3

Miguel Mariguín
mmariguin@unrn.edu.ar

Mauro Fermín
mafermin@unrn.edu.ar



Volviendo al principio

¿Qué es Programar?

Analizar

Es traducir lo que los futuros usuarios quieren en algo que pueda ser implementado

Diseñar

Es ordenar las solicitudes de los usuarios en una arquitectura implementada o que pueda ser implementada asegurándose de que todo lo que ha sido pedido esté cubierto

Implementar

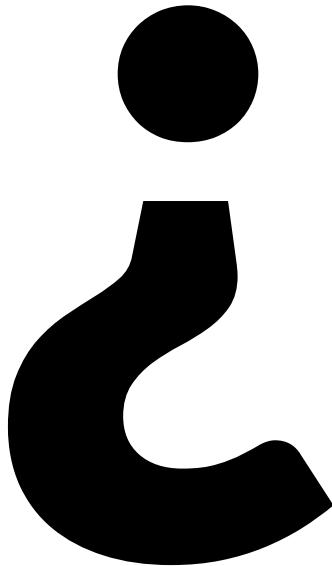
Esto si es la programación, y es traducir lo que el arquitecto de software
diseñó a algo que la computadora puede ejecutar

Probar

Es asegurarse de que lo implementado responde a lo que el arquitecto diseñó y que lo hace sin errores.

¿Preguntas?





**Pero...
¿cómo?**

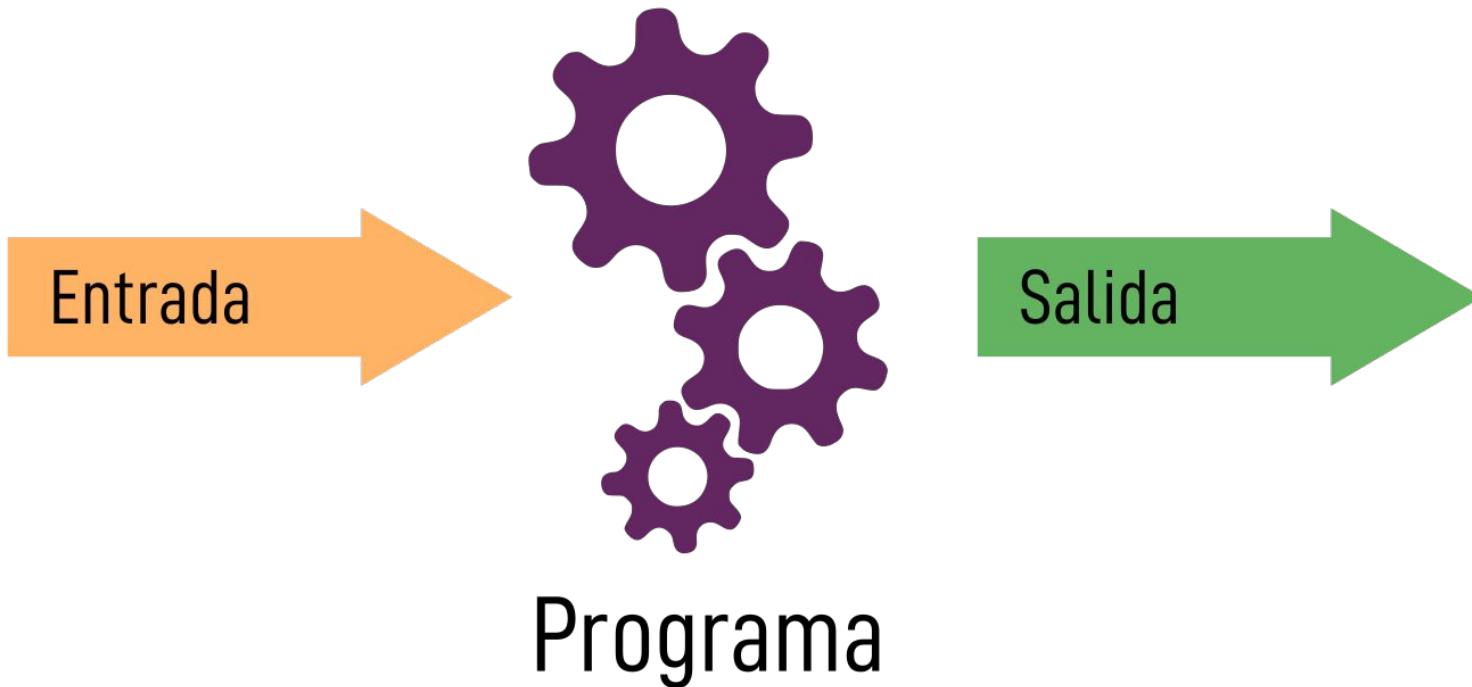


—

Estrategias

*sin un orden en
particular*

**Piensen en lo que
entra y en lo que
sale**



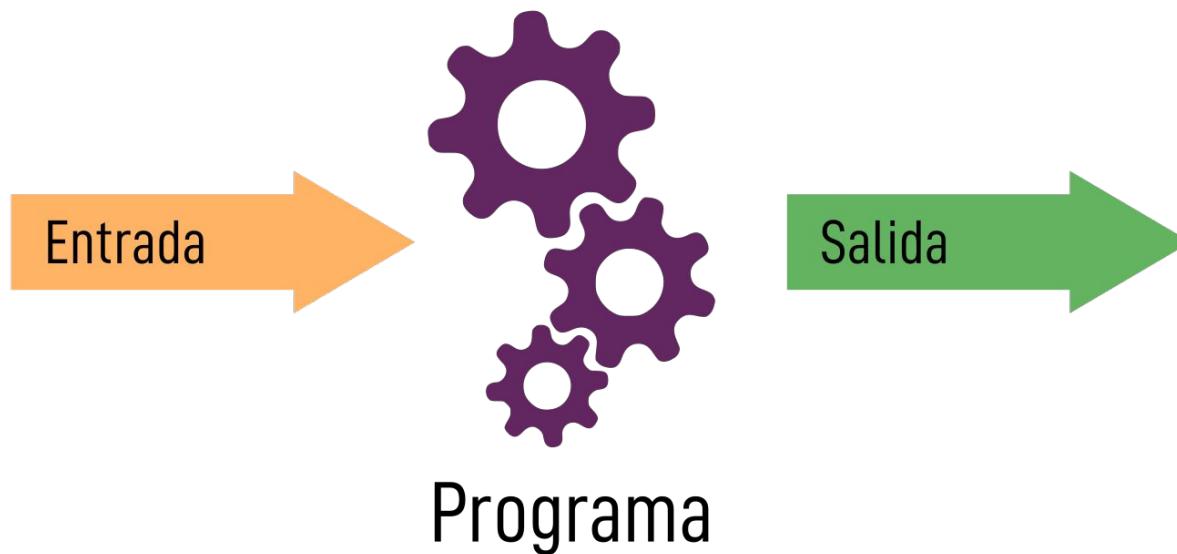
**Esto permite pensar en
cómo probar que
funciona correctamente**

¿Cómo es lo que entra?

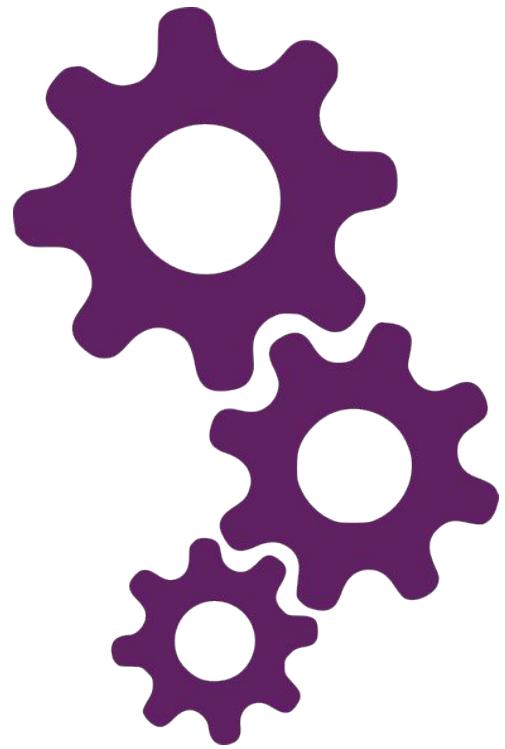
¿Qué es lo que sale?

Técnicamente
Todo puede ser una
cadena o un ‘print’

Desde este punto de vista



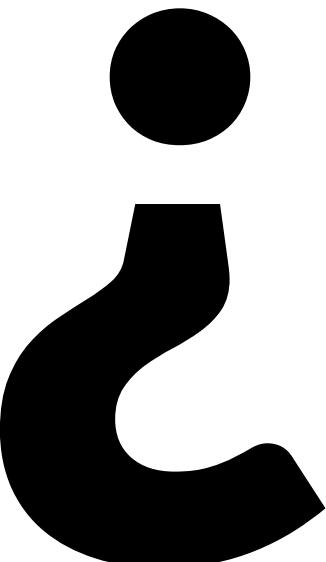
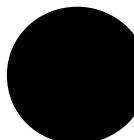
**Pero no siempre para
los ‘engranajes’
individuales**



**Las respuestas
deben quedar
registradas**

¿Cómo es la salida del programa?

**¿Es una frase, un
número, o alguna
otra cosa?**



**qué otras
preguntas se les
ocurren hacer**

Aunque de manera incompleta...
esta es la consigna

¿Preguntas?

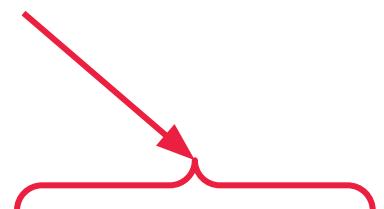


suma lenta

**La suma con pasos
adicionales**

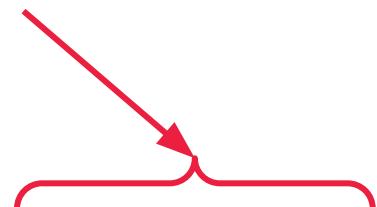
Suma paso a paso (positivo)

$$2 + 3 = 5$$


$$2 + 1 + 1 + 1 = 5$$

Suma paso a paso (negativo)

$$2 - 3 = -1$$



$$2 - 1 - 1 - 1 = -1$$

Suma lenta
→ que sumar
 el resultado

¿Preguntas?



Qué podemos hacer con esta información

Verificar el programa

Conociendo lo que entra

Sabemos lo que sale

**Por ejemplo:
números enteros**

$$\mathbb{N} = \{0, 1, 2, 3, 4, \dots\}$$

$$\mathbb{Z} = \{-2, -1, 0, 1, 2, \dots\}$$

A simple vista

**Algunos números
importantes,
 $\{\pm 0, \pm 1, \pm 2, \pm 3, \pm 10\}$**

¿Por qué esos números?

en particular

Cuanto es

 $n \times 0 = ?$ $n \times 1 = ?$ $n + 0 = ?$

en particular

Cuanto es

$$n \times 0 = 0$$
$$n \times 1 = n$$
$$n + 0 = n$$

**No son ejemplos
exhaustivos**

Dependiendo del resultado y las operaciones involucradas

Es razonable pensar en la combinación de valores importantes

—

Por lo que es importante identificarlos

Y con esto, armar combinaciones

Para

int suma_lenta(int a, int b);

Todas las combinaciones de

{±0, ±1, ±2, ±3, ±10}

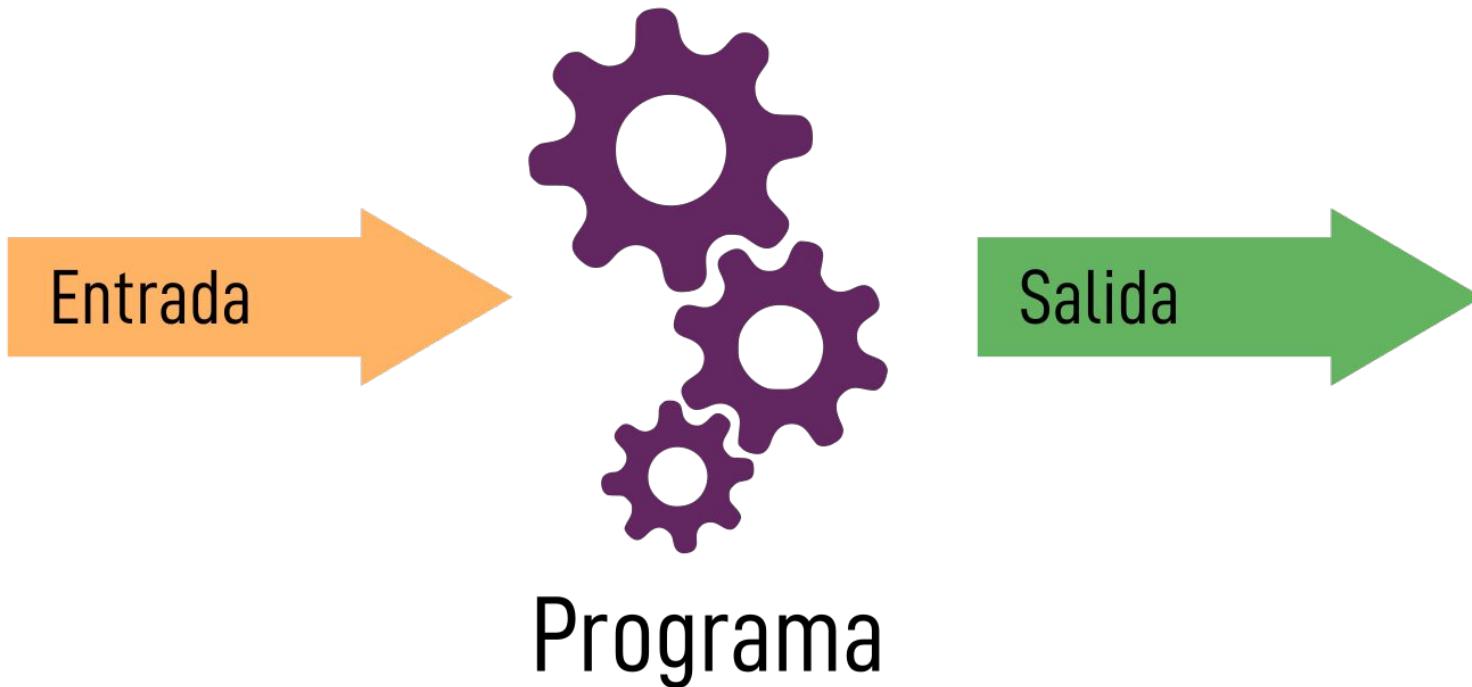
y en ese ejemplo en particular
**Siempre que puedan, usen
algo ya hecho como apoyo**

¿Preguntas?



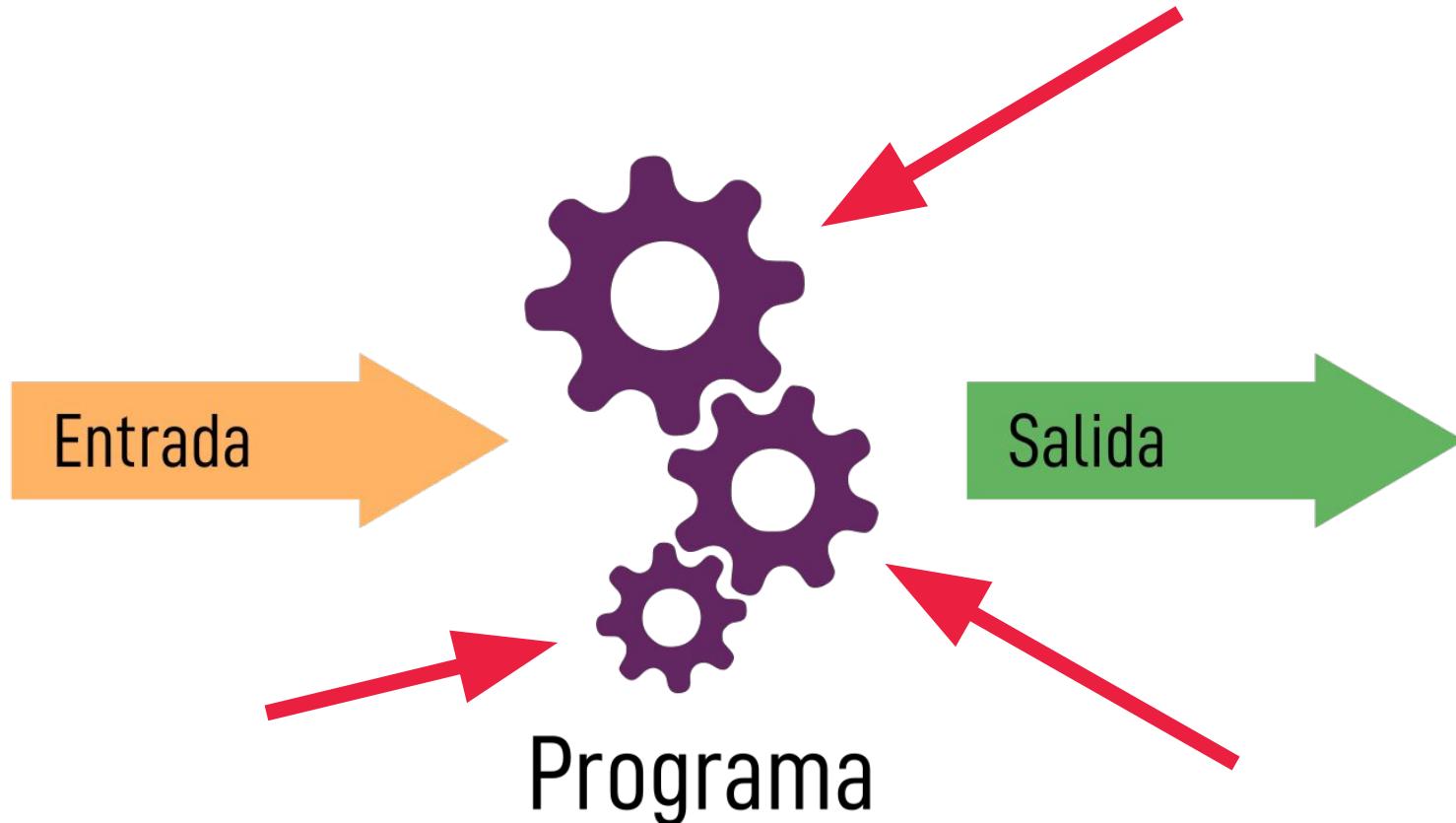
**Pero eso no es
suficiente**

**Es el punto de vista
externo**



**Es necesario ir más
profundo**





**Que veremos
la proxima**



¿Por qué C?

Tenemos que ver

datos

Información en la memoria

literales

Como representamos un dato en el código

variables

Donde guardamos un dato

constantes

datos fijos

expresiones

Que producen y modifican datos a partir de literales

Asignación y operaciones

estructuras de control

Repeticiones y decisiones.

estructuras de organización

Separación funcional +
Funciones y procedimientos*

-

Empecemos por el hola mundo

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hola mundo C. \n");
    return 0;
}
```

Compilando y ejecutando

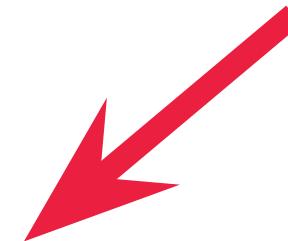
```
$> gcc hola.c  
$> ./a.out    (linux)  
$> ./a.exe    (windows)
```

—

Paso a paso

1 - El punto de entrada

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    printf("Hola mundo C. \n");
    return 0;
}
```

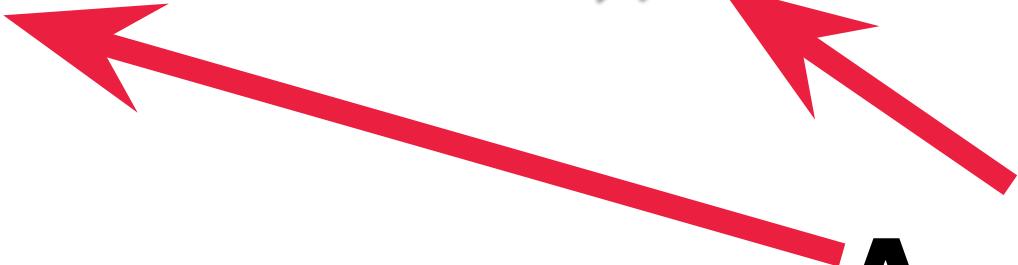


2 - ¡esto compila!

```
#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(int argc, char *argv[])
{
    printf("Hola mundo C. \n");
    return 0;
}
```



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

¿Preguntas?



Declaración de variables

A diferencia de Python

```
int variable;
```

```
int variable = 0;
```

Las variables van
con tipo.

¿Qué tipo de datos hay?

Números enteros

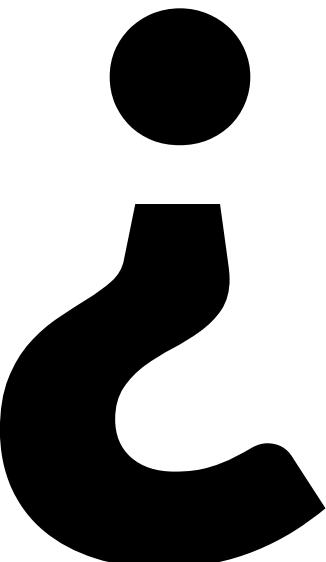
short int

int

long int

long long int

**opcionalmente
unsigned
sin signo**



Como le damos un valor



Con los *literales*

0b1010

- binario (0b)

0173

- octal (0)

1234

- decimal

0xCAFE

- hexadecimal (0x)

Números reales

float (4-bytes) **1.0f**

double (8-bytes) **1.0**

¿Cuándo usar uno u otro?

Caracteres

tipo: char

literal: 'a'

No hay strings^{*}
No hay booleanos

**Son algo *diferente* y
vamos a ver mas
adelante**

¿Preguntas?



Expresiones

Asignación

No se olviden



variable = expresion;

Operadores aritmeticos

Aparte de los de Python:

+ - / * %

var++; //var = var + 1

var--; //var = var - 1

Operadores condicionales

> < = =

-

Operadores lógicos

! & ||

negación y lógico o lógico

Estructuras de control

y el estilo a usar

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

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

```
if (condición)
{
    bloque;
}
else if(condición_2)
{
    bloque_2;
}
```

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

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

A cartoon illustration of Wile E. Coyote running from Road Runner. Wile E. Coyote is in the foreground, looking back over his shoulder with a worried expression. He has his arms outstretched and is running on all fours. Road Runner is visible behind him, looking forward. The background is a simple yellow gradient. There are two speech bubbles: one on the left containing Wile E. Coyote's code, and one on the right containing Road Runner's code.

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

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

```
for (inicio; condición; cambio_var)  
{  
    bloque;  
}
```

```
int i;  
for (i=0; i<10; i++)  
{  
    bloque;  
}
```

```
for (;;)  
{  
    bloque;  
}
```

**¡Es un lazo
infinito!**

break;
para cortar el lazo

continue;
para pasar a la siguiente iteración

```
switch (expresión)
{
    case valor:
        sentencia
    break;
    default:
        sentencia;
}
```

**El famoso
switch**

```
switch (expresión)
{
    case valor:
        sentencia
    break;
    default:
        sentencia;
}
```

**El famoso
switch**

¡Es opcional!



Refuerzo

Premio si llegamos hasta este punto

—

Empecemos a hablar de la documentación



escririendo el código

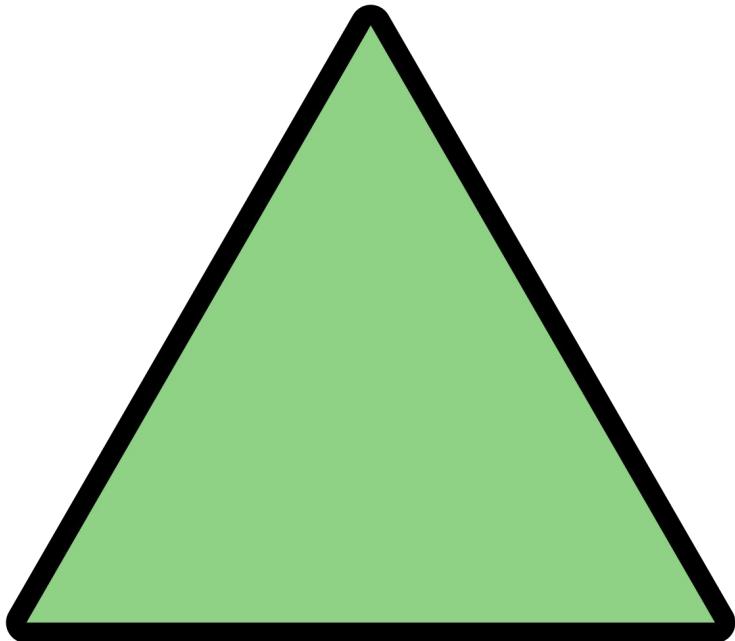


documentando el código

El triángulo de la documentación

o por qué el código no es documentación

qué



cómo

por qué

El objetivo de la documentación

**qué
el código en sí**

WHAT

por qué

Los comentarios

SUPER
ZHONG
中國紅

TACOS
SIEMPRE
PRECIOS LOCOS!!!



cómo El contexto



UNRN

unrn.edu.ar

UNRN

Universidad Nacional
de Río Negro



| unrnionegro