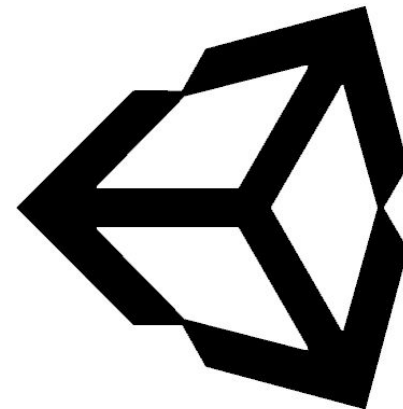




# Programación Estructurada

# Dani Coyotzi Borja

---



---

# **Diseño de programas**

# Fases

- Análisis
- Diseño
- Programación
- Codificación
- Prueba
- Mantenimiento
- Documentación

---

# Tipos de dato

---

# ¿Qué es un tipo de dato?

Es una clasificación que el programador le da a la información almacenada para avisarle al compilador cómo va a ser interpretada.

---

Un tipo de dato es, en esencia, un espacio en memoria con restricciones.



---

# Tipos de datos primitivos



int, float

Númericos



char

Caracteres



bool

Booleanos



**int**

Números enteros  
p/e 4, 52, -999, 1853

**float**

Números decimales  
p/e 3.1416, 0.001

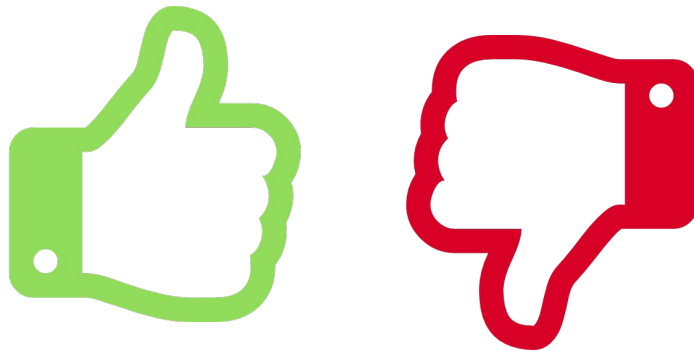
**char**

Caracteres p/e  
'P' 'L' 'A' 'T' 'Z' 'I'

---

# Booleanos

Tipo de dato que representa algo que puede ser verdadero o falso.



---

# Variables

---

# ¿Qué es una variable?

Es un **espacio** reservado en **memoria**, definido por un **tipo de dato** y un **nombre** asignado, en el cual se puede guardar un **valor** y se puede **modificar**.

---

# Declaración de una variable en C

```
main()
```

```
{
```

```
    int healthpoints;
```

```
}
```

↑  
tipo de  
dato

↑  
nombre de  
la variable

# Asignación de datos a una variable

```
main()
```

```
{
```

```
    int healthpoints;
```

```
    healthpoints = 100;
```

valor

```
}
```

nombre de  
la variable

operador  
asignación

---

# Inicialización de una variable

```
main()
```

```
{
```

```
    int healthpoints = 100;
```

```
}
```

---

```
main()
```

```
{
```

```
    int healthpoints = 100;
```

```
    float damage = 13.05;
```

```
    char favoriteLetter = 'D';
```

```
    bool hasFun = false;
```

```
}
```



# Retos!



## Primer Reto:

- Crea dos variables.
- Ingresa y guarda el valor de ambas variables.
- Intercambia el valor de las variables con ayuda de una variable auxiliar.
- Imprime el valor de las variables ya intercambiadas.

# Resultado:

"C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\02-Entrada y s... — □ ×

Da el valor de x: 3

Da el valor de y: 5

El valor intercambiado del entero x es: 5

El valor intercambiado del entero y es: 3

Process returned 0 (0x0)    execution time : 3.679 s

Press any key to continue.

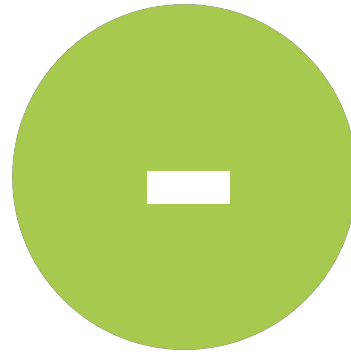
—

---

# Operadores numéricos



Suma



Resta



Multiplicación



División



Módulo

---

# ¿Qué es módulo?

El resultado de  $2\%5 = 1$

$$\begin{array}{r} 2 \\ 2 \overline{) 5} \\ \underline{2} \phantom{0} \\ 1 \end{array} \quad \leftarrow \text{Residuo}$$

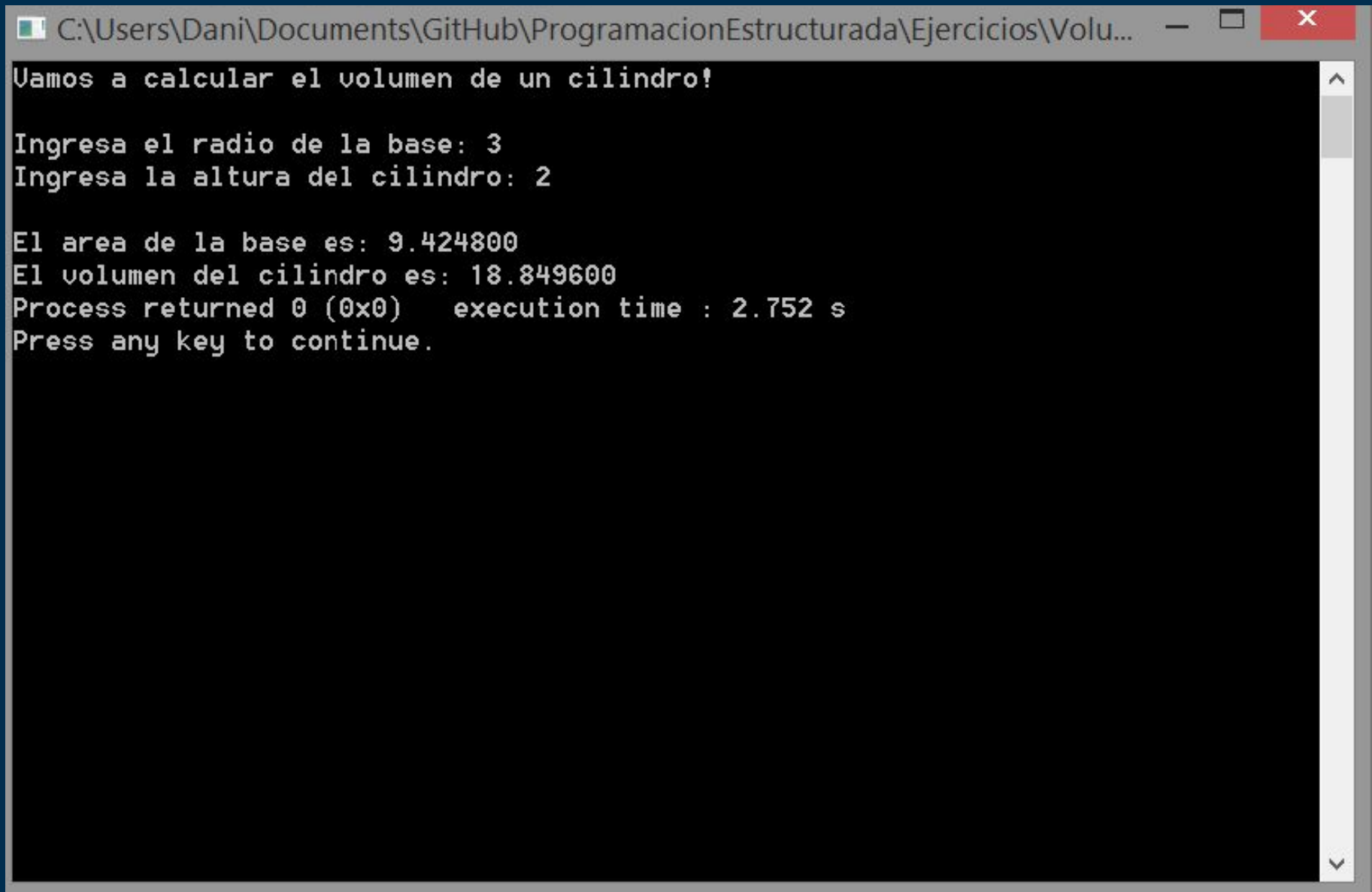
# Retos!



## Primer Reto:

- Ingresa los valores necesarios para calcular el área de un cilindro (radio, altura, área y volumen)
- Realiza la operación.
- Imprime el resultado del área.

# Resultado:



```
C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Volu...  
Vamos a calcular el volumen de un cilindro!  
  
Ingresa el radio de la base: 3  
Ingresa la altura del cilindro: 2  
  
El area de la base es: 9.424800  
El volumen del cilindro es: 18.849600  
Process returned 0 (0x0)   execution time : 2.752 s  
Press any key to continue.
```



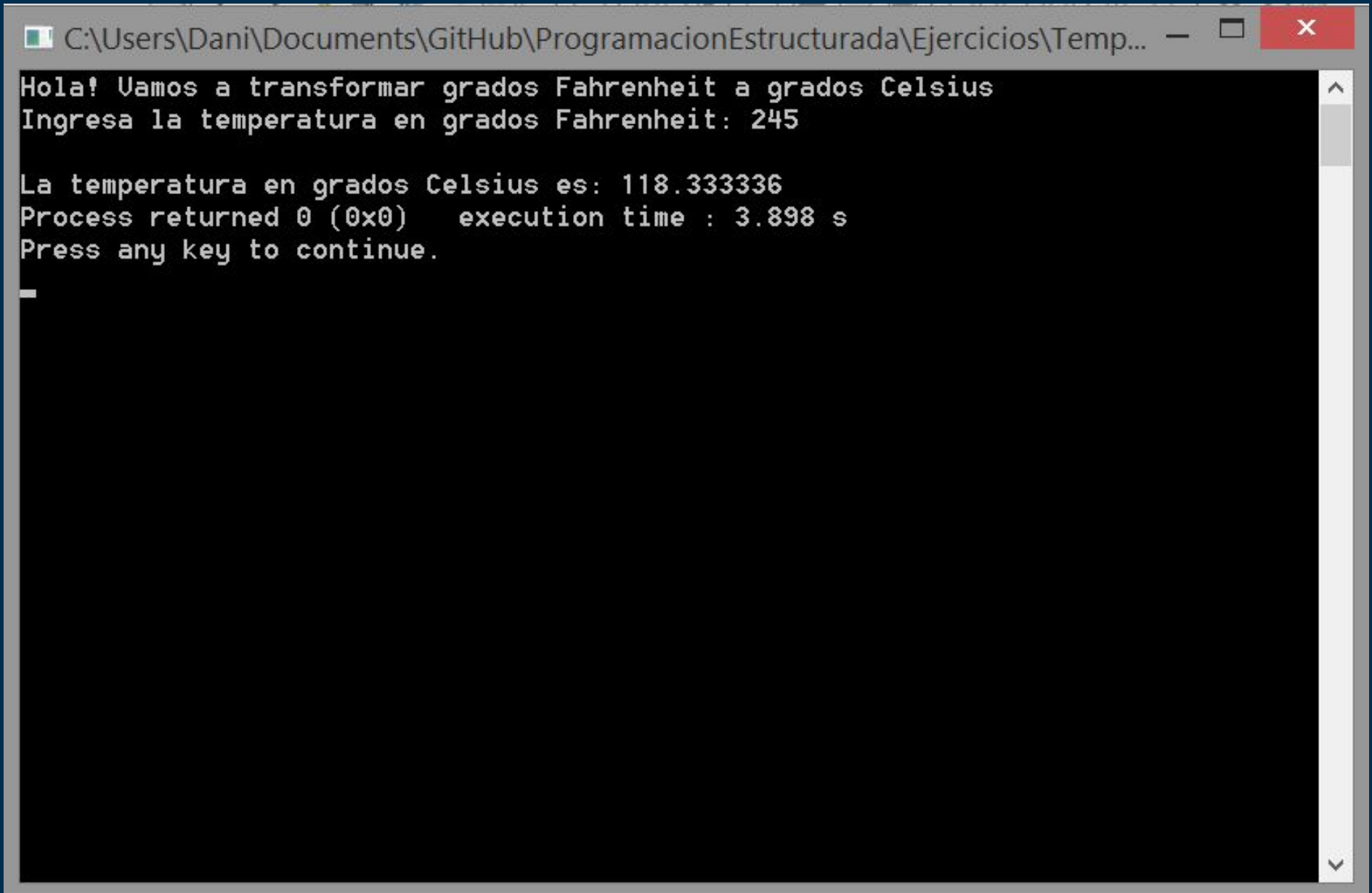


# Retos!

Segundo Reto:

- Ingresa la temperatura en grados Fahrenheit.
- Realiza la operación para convertir de grados Fahrenheit a Celcius.
- Imprime el resultado de la conversión.

# Resultado:



```
C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Temp...  
Hola! Uamos a transformar grados Fahrenheit a grados Celsius  
Ingresa la temperatura en grados Fahrenheit: 245  
  
La temperatura en grados Celsius es: 118.333336  
Process returned 0 (0x0)   execution time : 3.898 s  
Press any key to continue.  
-
```

---

# Operadores de asignación

**`+=`**



**`a += 5`**

**`a = a + 5`**

**`-=`**



**`a -= 5`**

**`a = a - 5`**

**`*=`**



**`a *= 5`**

**`a = a * 5`**

**`/=`**



**`a /= 5`**

**`a = a / 5`**

**`%=`**



**`a %= 5`**

**`a = a % 5`**

---

# Operadores de incremento y decremento

**++**



**a++**

**a = a + 1**

**--**



**a--**

**a = a - 1**

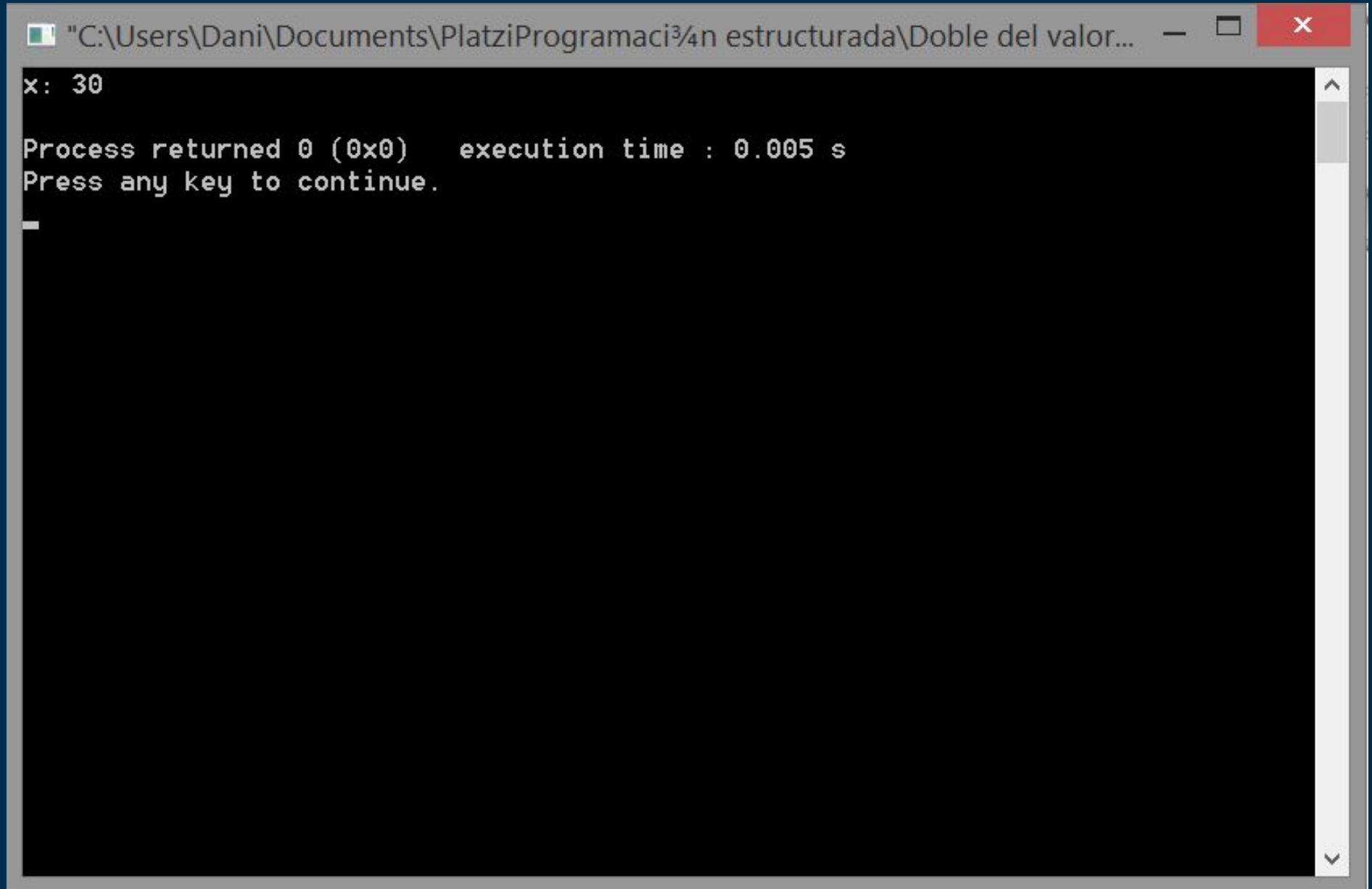
# Retos!



## Primer Reto:

- Crea una variable llamada x con valor 10.
- Utilizando operadores de asignación, que esta variable se sume a sí misma el doble de su valor.
- Imprime el resultado.

# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Users\Dani\Documents\PlatziProgramaci3/4n estructurada\Doble del valor...". The window contains the following text: "x: 30", "Process returned 0 (0x0) execution time : 0.005 s", and "Press any key to continue.". A small horizontal line is visible below the "Press any key to continue." message.

```
x: 30

Process returned 0 (0x0)   execution time : 0.005 s
Press any key to continue.
_
```



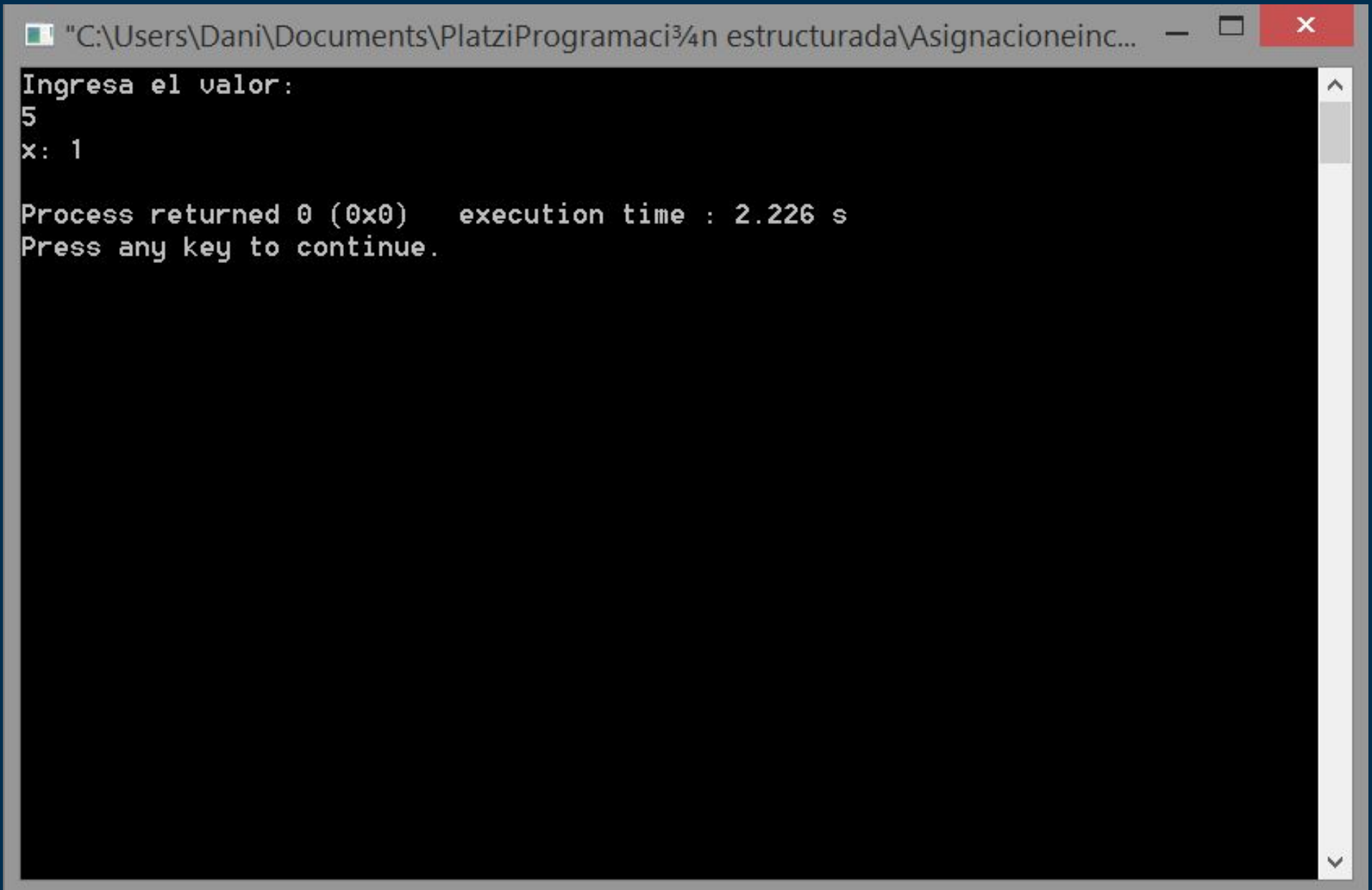
# Retos!



## Segundo Reto:

- Ingresa un numero entero.
- Usando operadores de asignación realiza la operación de módulo del valor ingresado con 5.
- Imprime el resultado más uno usando operadores de incremento.

# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Users\Dani\Documents\PlatziProgramaci3/4n estructurada\Asignacioneinc...". The window has standard Windows window controls (minimize, maximize, close). The command prompt displays the following text:

```
Ingresa el valor:  
5  
x: 1  
  
Process returned 0 (0x0)   execution time : 2.226 s  
Press any key to continue.
```

The text is displayed in a monospaced font on a black background. The window has a vertical scrollbar on the right side.

---

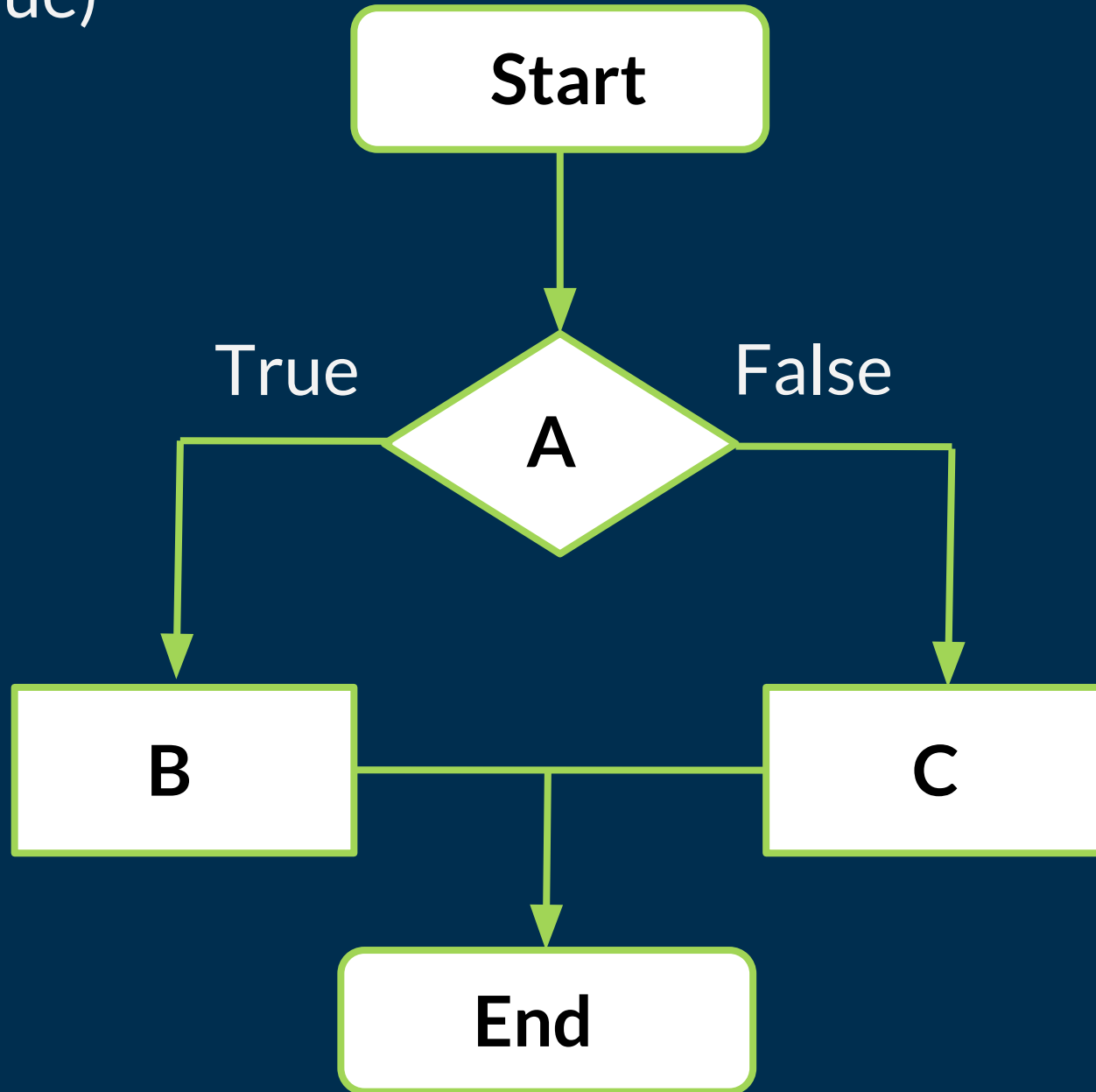
# Condicionales

---

# ¿Condicionales o sentencias condicionales?

Son instrucciones que evalúan resultados booleanos, generalmente usados para alterar el flujo de un programa.

If(a == true)  
Then B  
Else C  
End if



---

# Condicional if

```
main()
{
    if( vidasExtras == 0)
    {
        printf("Game over");
    }
}
```

# Condicional if

```
main()
```

```
{
```

*Instrucción  
condicional*

```
if( vidasExtras == 0)
```

*Operaciones*

*Bloque al  
cumplimiento  
de la condición*

```
{
```

```
printf("Game over");
```

```
}
```

```
}
```

# Retos!

Primer Reto:

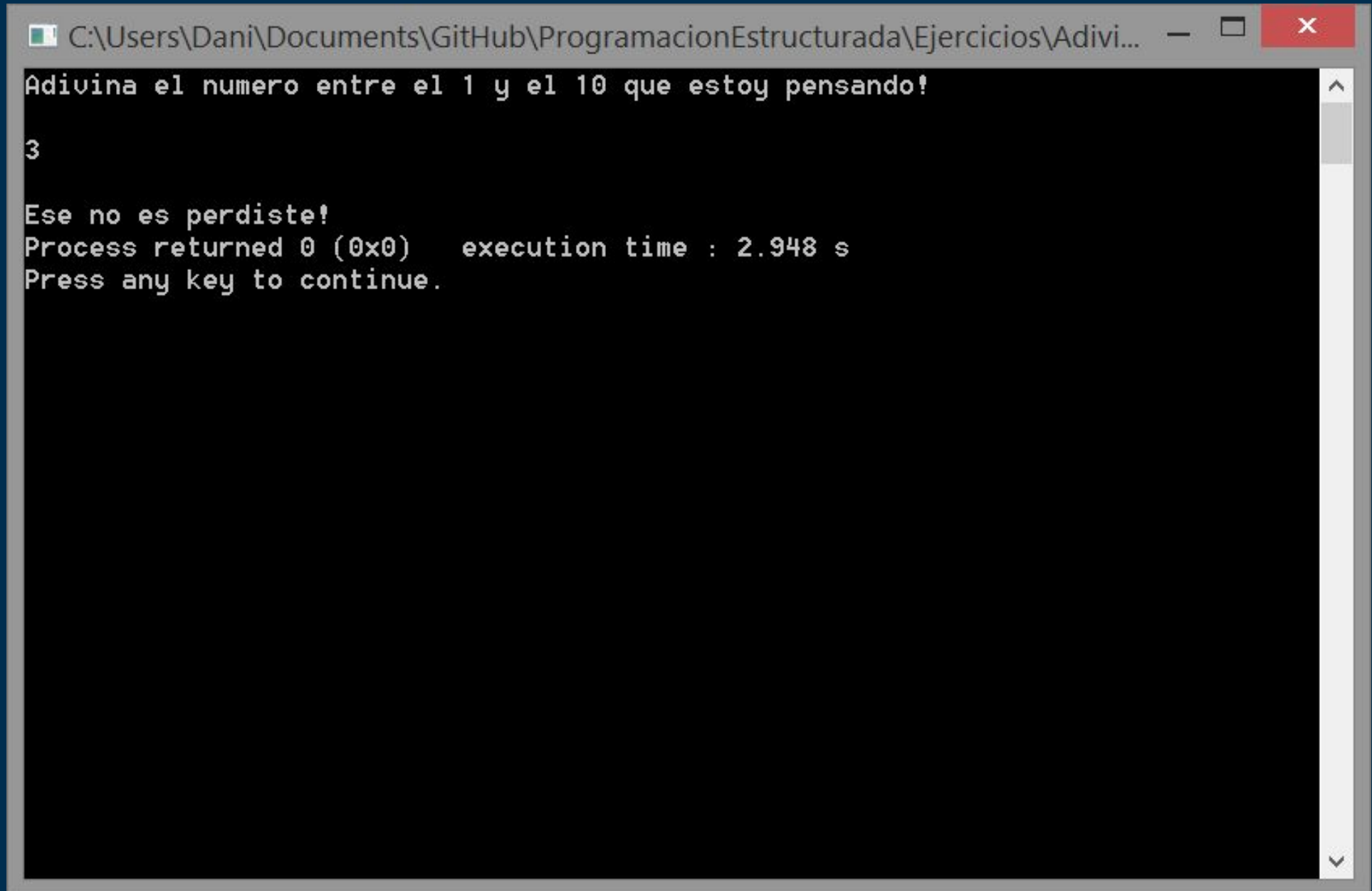
Haremos un programa que adivine número de la computadora.



- Definir una variable cuyo valor es 5.
- En otra variable ingresar un número con la instrucción, entre el 1 y el 10.
- Si el número ingresado es igual a la variable definida (5). Imprimir “Adivinaste”. Si no imprimir “Ese no es, perdiste!”



# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Adivi... The window contains the following text:

```
Adivina el numero entre el 1 y el 10 que estoy pensando!  
3  
Ese no es perdiste!  
Process returned 0 (0x0)    execution time : 2.948 s  
Press any key to continue.
```

---

# Operadores relacionales



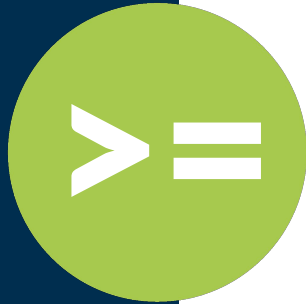
Menor que



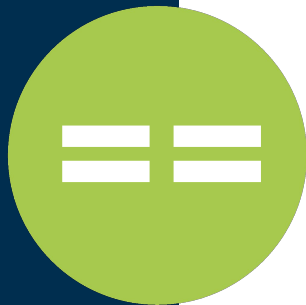
Menor o igual que



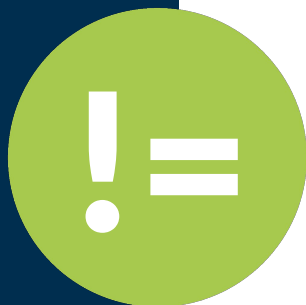
Mayor que



Mayor o igual que



Igual que (Comparación)



No igual que (Diferente)

---

# Operadores lógicos



AND (Y) El resultado será verdadero si ambas condiciones son verdaderas.

p/e `if(key1 == true && key 2 == true)`



OR (O) El resultado será verdadero si alguna de las condiciones es verdadera.

p/e `if(key1 == true || key 2 == true)`



NOT (No) El resultado será inverso al operando.

p/e `hasKey = !hasKey;`

# Retos!

Primer Reto:



- Imprime la instrucción ingresa el primer número.
- Ingresa el número en una variable.
- Repite para una segunda variable.
- Si el primer número es menor que el segundo, imprime el primer número.
- Si no, imprime el segundo número.

# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Men...

Si me das dos numeros enteros, te dare el menor entre ellos!

Ingresas el primer numero: 3

Ingresas el segundo numero: 6

El numero menor es: 3

Process returned 0 (0x0) execution time : 4.075 s

Press any key to continue.



# Retos!

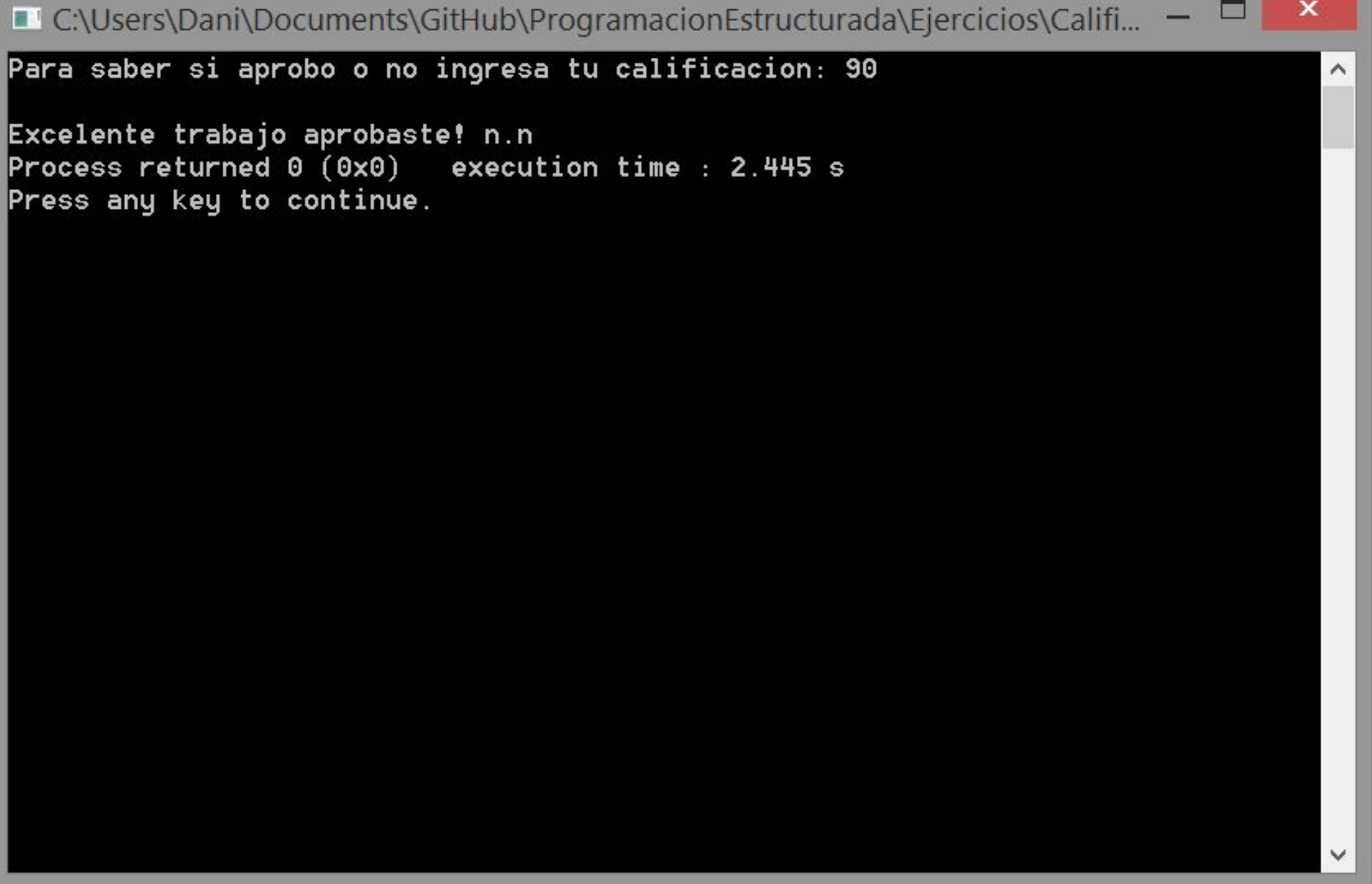
Segundo Reto:

- Hacer un programa que reciba la calificación de un alumno.
- Si el alumno sacó menos de 60, imprimir que el alumno está reprobado.
- Si el alumno sacó más de 60, imprimir que el alumno está aprobado.

Extra: Si el alumno sacó más de 90, imprimir que está aprobado y una carita feliz.



# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Califi... with standard minimize, maximize, and close buttons. The command prompt has a black background with white text. The text displayed is: 'Para saber si aprobo o no ingresa tu calificacion: 90', 'Excelente trabajo aprobaste! n.n', 'Process returned 0 (0x0) execution time : 2.445 s', and 'Press any key to continue.'.

```
C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Califi...  
Para saber si aprobo o no ingresa tu calificacion: 90  
Excelente trabajo aprobaste! n.n  
Process returned 0 (0x0) execution time : 2.445 s  
Press any key to continue.
```

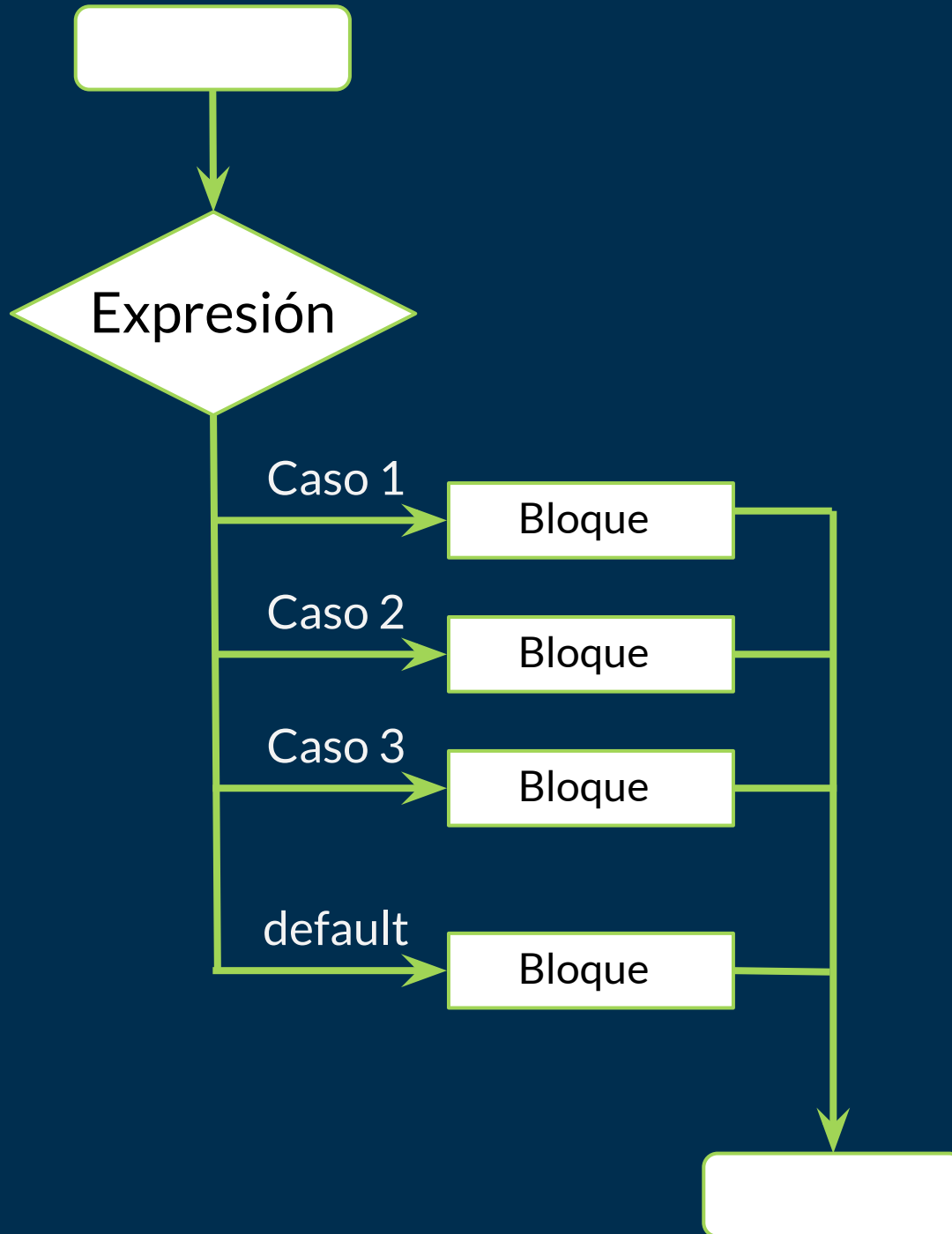
---

# Switch

---

# Switch!

Es una estructura de control para agilizar el flujo del programa en opciones múltiples.



```
switch( balas )  
{  
  case 0:  
    printf("No hay balas!");  
    break;  
  
  case 1:  
    printf("Te queda una bala");  
    break;  
  
  case 5:  
    printf("Tu arma está llena");  
    break;  
  
  default:  
    printf(":");  
    break;  
}
```

```
switch( balas )
{
    case 0:
        printf("No hay balas!");
        break;

    case 1:
        printf("Te queda una bala");
        break;

    case 5:
        printf("Tu arma está llena");
        break;

    default:
        printf(":");
        break;
}
```

```
if(balas == 0)
{
    printf("No hay balas!");
}
else if( balas == 1 )
{
    printf("Te queda una bala");
}
else if( balas == 5 )
{
    printf("Tu arma está llena");
}
else
{
    printf(":");
}
```



# Retos!

## Primer Reto:

- Vamos a hacer un pequeño juego de texto.
- Imprimir una pequeña introducción, con tres opciones a elegir, numeradas del 1 al 3.
- Cada una de ellas te debe de imprimir un resultado distinto en la historia.

# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Jueg...

```
Te encuentras en un sueño y tienes tres caminos.  
Escribe 1 si quieres ir por el camino de dulces  
Escribe 2 si quieres ir por el camino de madera  
Escribe 3 si quieres ir por el camino de gatitos
```

```
3
```

```
Los gatitos lindos te llevan una puerta y al cruzarla te despiertas!  
Felicidades!
```

```
Process returned 0 (0x0)    execution time : 3.349 s
```

```
Press any key to continue.
```



---

# Loops (Bucles)

Estructuras de control repetitivas

---

# ¿Qué es un loop?

Una estructura iterativa que permite repetir un bloque de instrucciones.

Esta repetición es controlada por una condición booleana.

---

# While

---

# ¿Qué es un while?

Es una estructura de control en la que la repetición se realizará tantas veces como se indique mientras se cumpla una condición.

# Sintaxis

while (a < 5) → *Condición*

```
{  
    a++;  
}
```

→ *Bloque al cumplimiento de la condición*



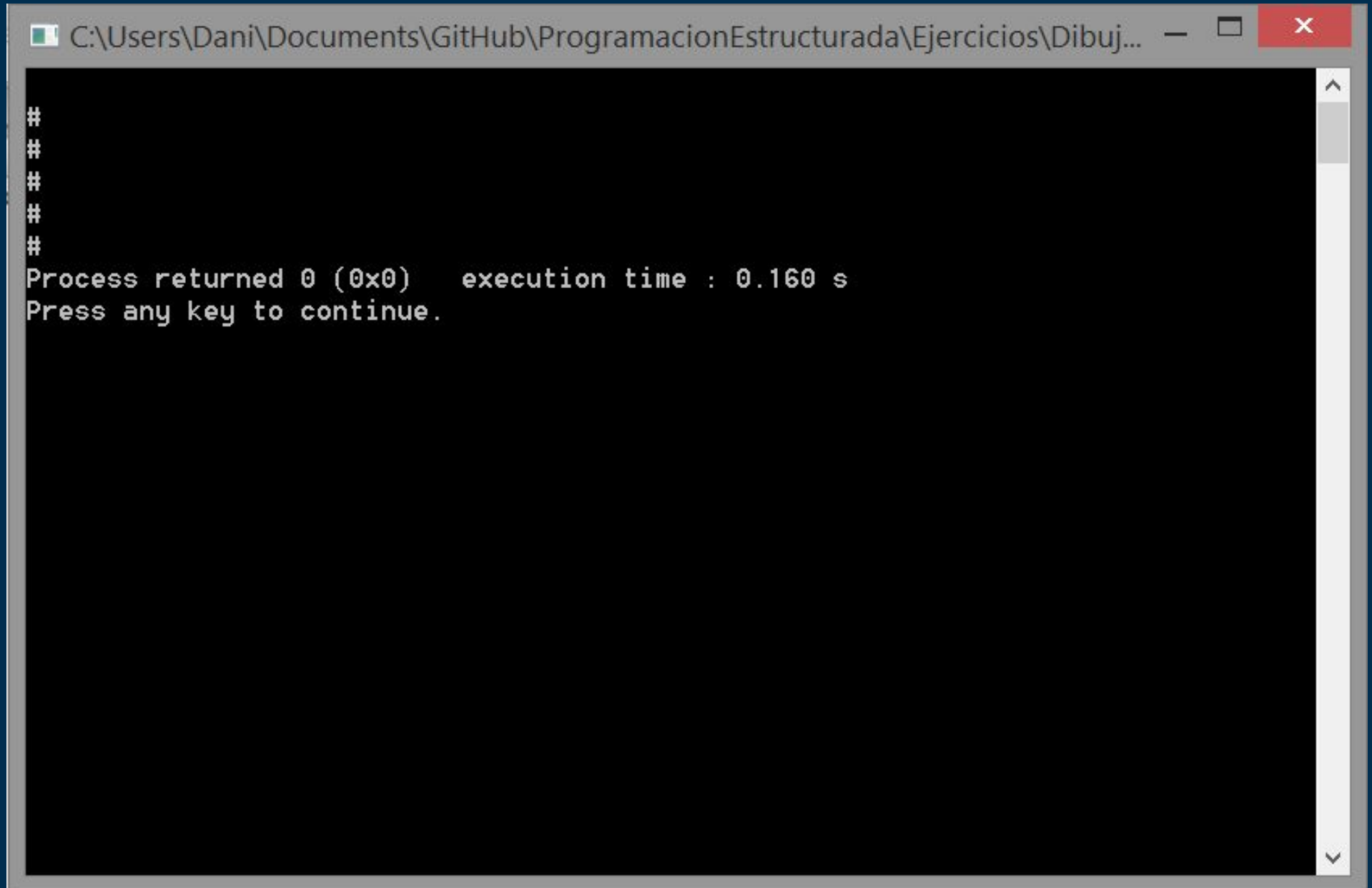
# Retos!

Primer Reto:

- Hacer un programa que imprima el símbolo de # en 5 filas

Tip: Puedes usar operadores de incremento y decremento.

# Resultado:



A screenshot of a Windows command prompt window. The title bar at the top shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Dibuj... with standard minimize, maximize, and close buttons. The command prompt area has a black background with white text. It displays five hash symbols (#) on separate lines. Below these, it shows the message "Process returned 0 (0x0) execution time : 0.160 s" followed by "Press any key to continue." on the next line. A vertical scrollbar is visible on the right side of the window.

```
#  
#  
#  
#  
#  
Process returned 0 (0x0)   execution time : 0.160 s  
Press any key to continue.
```

---

**For**



---

# ¿Qué es un for?

Es una estructura de control que nos permite repetir un bloque de comandos un número de veces específico.

# Sintaxis

*Inicialización*

*Condición*

*Incremento*

for (int i = 0; i < 10; i++)

```
{  
  
  
  
  
  
  
  
  
  
}
```

*Bloque al  
cumplimiento de  
la condición*

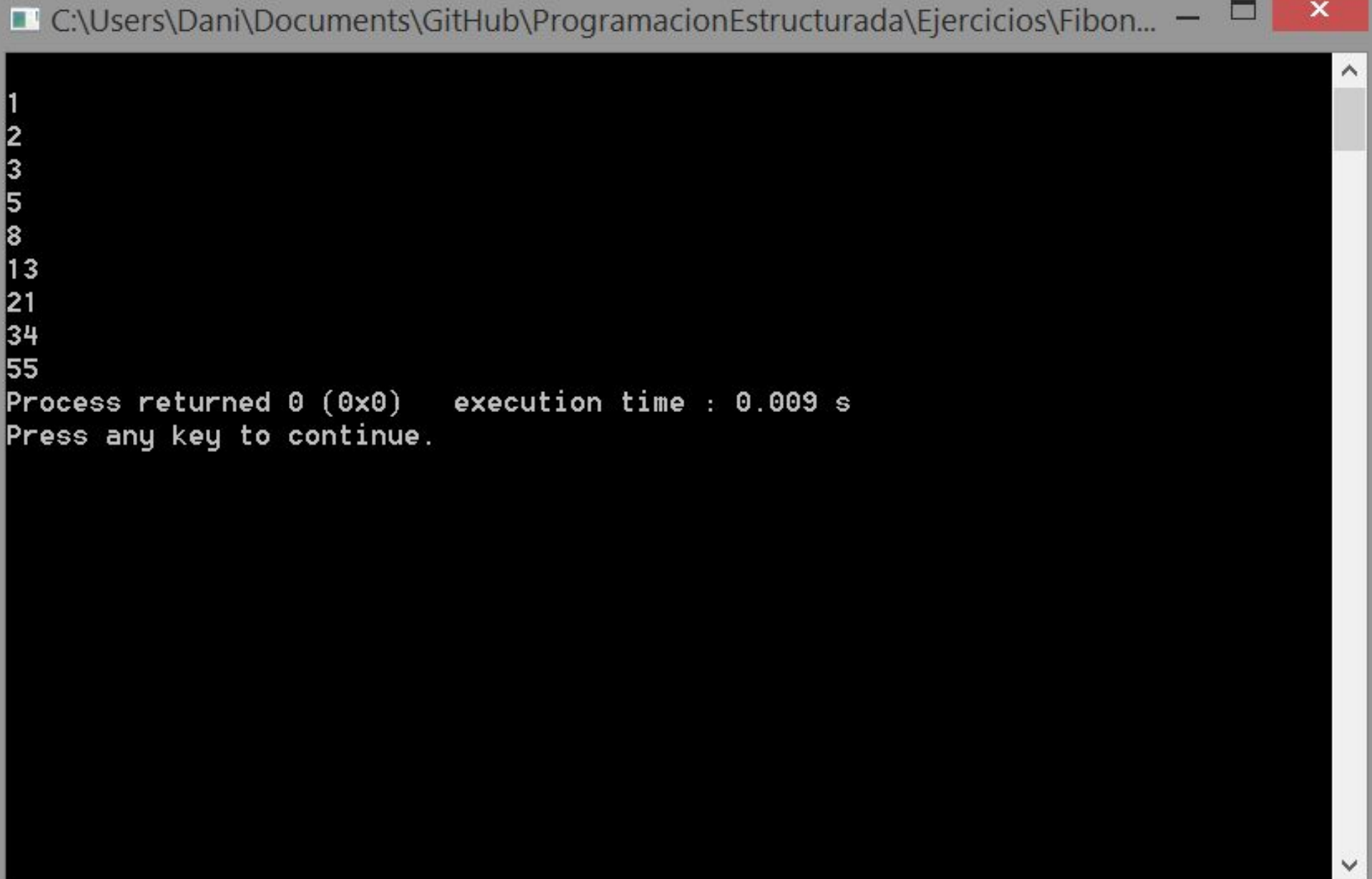


# Retos!

Primer Reto:

- Usando for, imprime la secuencia Fibonacci hasta la novena vuelta

# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Fibon... The window contains the following text:

```
1
2
3
5
8
13
21
34
55
Process returned 0 (0x0)    execution time : 0.009 s
Press any key to continue.
```

---

# Do While

---

# ¿Qué es un Do While?

- Es una estructura de control donde la condición de continuación del ciclo se prueba al final del mismo.
- Funciona de manera similar a la estructura while, la diferencia es que esta evalúa **al final**.

# Sintaxis

do

```
{  
    a++;  
} while (a <= 5);
```

*Bloque antes del  
cumplimiento de  
la condición*

*Condición*



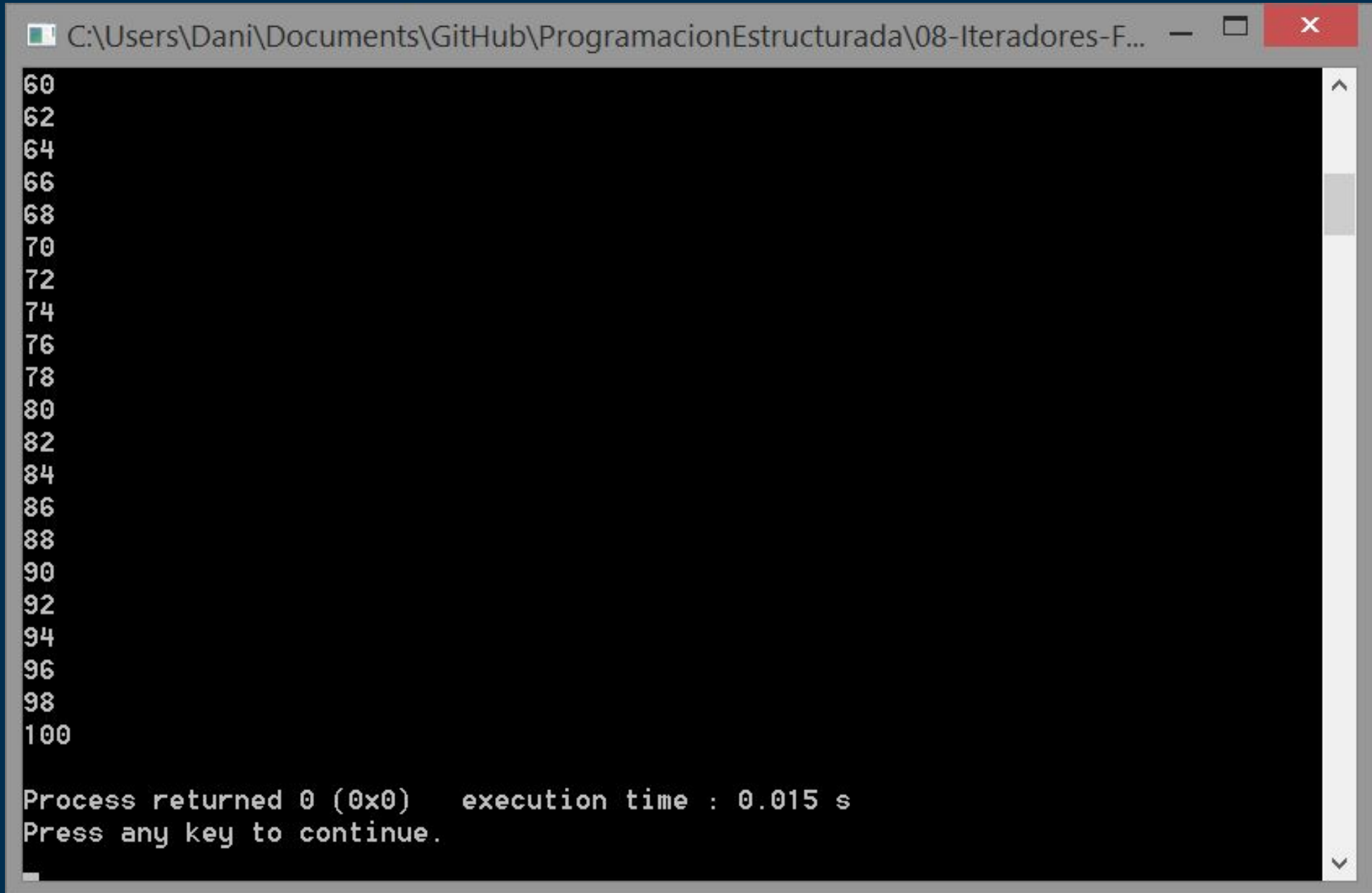
# Retos!

Primer Reto:

- Usando do while, imprime los primeros 100 números naturales.



# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\08-Iteradores-F... The window contains a list of even numbers from 60 to 100, followed by the text "Process returned 0 (0x0) execution time : 0.015 s" and "Press any key to continue." The cursor is at the bottom left.

```
60
62
64
66
68
70
72
74
76
78
80
82
84
86
88
90
92
94
96
98
100

Process returned 0 (0x0)   execution time : 0.015 s
Press any key to continue.
```

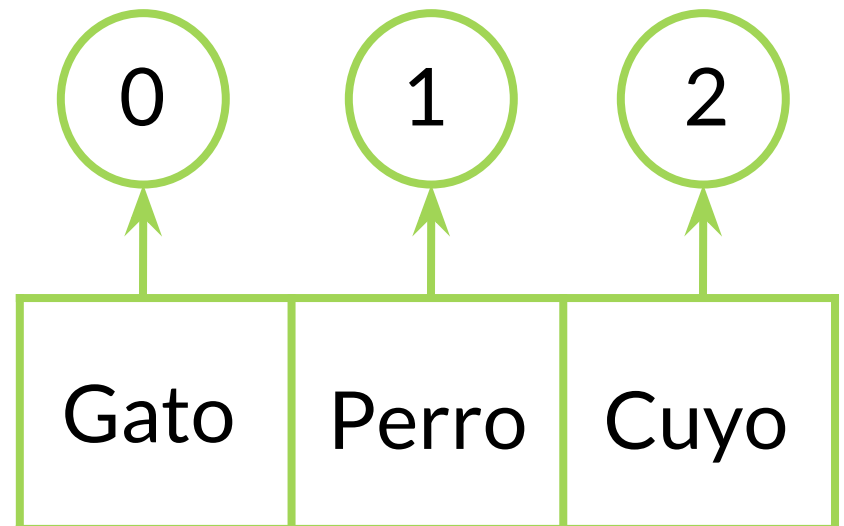
---

# Arreglos

---

# ¿Qué es un arreglo?

Son una serie de elementos, del mismo tipo de dato y son almacenados de manera consecutiva.



# Declarar un arreglo

```
Data_type name[size];
```



Tipo de  
dato      nombre      tamaño

```
int maxElements[5];
```

# Inicializar un arreglo

Data\_type name [size] = {contenido};

Tipo de  
dato

tamaño

Contenido  
del arreglo

int maxElements[5] = {0, 12, 5, 9, 55};

nombre



```
int maxElements[5] = {0, 12, 5, 9, 55};
```

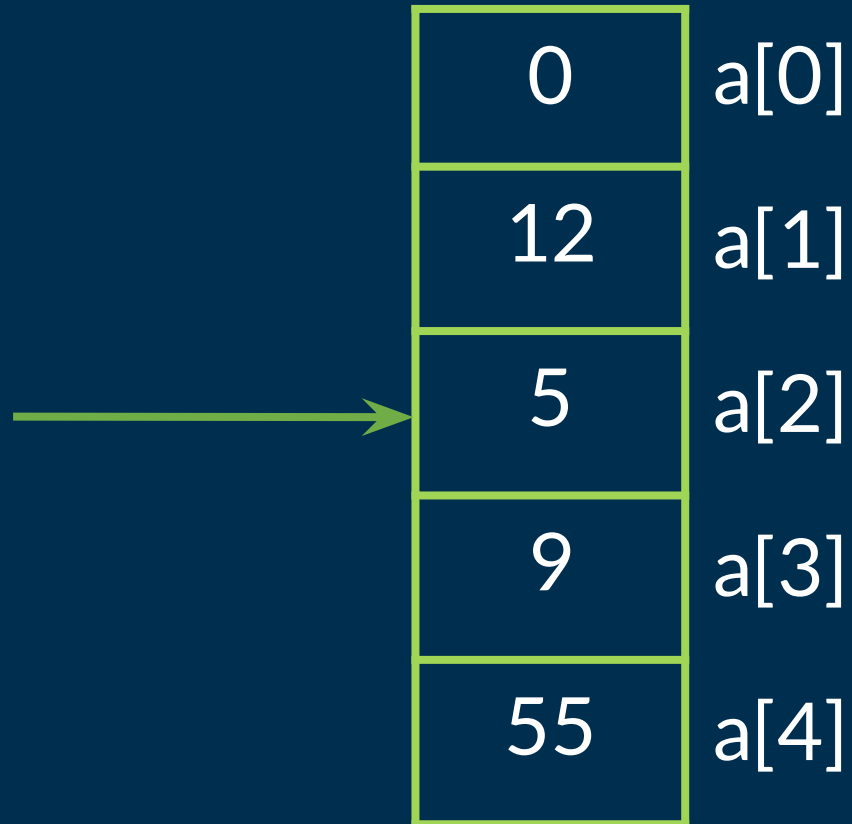
`a[0] = 0;`

`a[1] = 12;`

`a[2] = 5;`

`a[3] = 9;`

`a[4] = 55;`





# Retos!

Primer Reto:

- Ingresar valores a un arreglo con un tamaño de 5.
- Multiplicar todos sus valores.
- Imprimir el resultado.



# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Multi...

Multiplicar todos los elementos de un arreglo:

Ingresa los valores:

Valor[1]: 3

Valor[2]: 2

Valor[3]: 1

Valor[4]: 4

Valor[5]: 5

El resultado es: 120

Process returned 0 (0x0) execution time : 5.832 s

Press any key to continue.

---

# Arreglos bidimensionales

Matrices

---

# ¿Qué es un arreglo bidimensional?

Los arreglos bidimensionales son también llamados **tablas** o **matrices**.

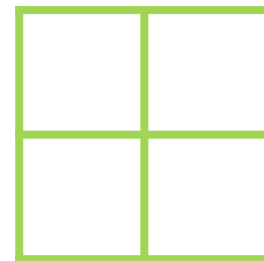
Tiene dos índices: el primero indica el número de **fila** y el segundo el número de **columna** en que se encuentra el elemento.

# Declarar un arreglo bidimensional

```
Data_type name [size][size];
```

Tipo de dato   nombre   filas   columnas

```
int matrix[2][2];
```



# Inicializar un arreglo bidimensional

```
Data_type name [size][size];
```



Tipo de dato   nombre   filas   columnas

```
int matrix[3][4] = {1{1,2,3,4},  
                    2{5,6,7,8},  
                    3{9,1,2,3} };  
                   1 2 3 4
```

```
int maxElements[5][2] = {{0, -1}  
                          {12, 3}
```

a[0][0]

0

-1

a[0][1]

{5, 4}

a[1][0]

12

3

a[1][1]

{9, 10}

{55, 7}};

a[2][0]

5

4

a[2][1]

a[3][0]

9

10

a[3][1]

a[4][0]

55

7

a[4][1]

# Retos!

Primer Reto:

Crea un arreglo de 3 filas por 4 columnas en donde:

- Los elementos de la primer fila sumen un total de 4.
- Los elementos de la segunda fila sumen un total de 10.
- Los elementos de la tercer fila sumen un total de 26.
- Imprime las sumatorias de cada fila.



# Resultado:

"C:\Users\Dani\Documents\PlatziProgramaci3/4n estructurada\SumaFilasBidim...

Array Bidimensional - Suma de filas

Primer fila: 4  
segunda fila: 10  
Tercer fila: 26

Process returned 0 (0x0) execution time : 0.006 s  
Press any key to continue.



---

# **Arreglos unidimensionales e iteradores**

---

# ¿Porqué?

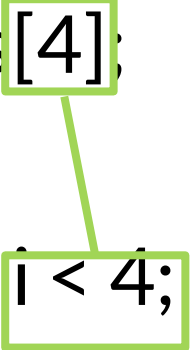
- Para manipular todos los elementos de un arreglo podemos utilizar una estructura repetitiva. La más usual es el ciclo for
- Cuando desea imprimir el contenido del arreglo.
- Cuando se suman todos los elementos.
- También cuando se va a inicializar.

---

# ¿Cómo utilizarlos?

```
int numbers[4];
```

```
for(int i = 0; i < 4; i++)  
{  
    numbers[i] = i;  
}
```

A green box highlights the number 4 in the array declaration 'int numbers[4];'. A green line connects this box to another green box that highlights the expression 'i < 4' in the for loop condition 'for(int i = 0; i < 4; i++)'. This visualizes that the loop iterates from 0 to 3, covering all elements of the array.



# **Retos!**

Primer Reto:

- Escribir un programa que nos diga el número más grande de un arreglo. Utilizando arreglos e iteradores.

# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Num...

Encontrar el numero mas grande de un arreglo.

Tamaño del arreglo: 5

Ingresar los valores:

Valor[0]: 1

Valor[1]: 4

Valor[2]: 7

Valor[3]: 9

Valor[4]: 13

El numero mayor es: 13

Process returned 0 (0x0) execution time : 8.570 s

Press any key to continue.

---

# Arreglos bidimensionales e iteradores

# ¿Cómo recorrer un arreglo bidimensional?

For anidado:

```
for(i = 0; i < 2; i ++ )  
{  
    for(j = 0; j < 1; j ++ )  
    {  
        printf("%i", matriz[i][j]);  
    }  
}
```

Primer  
for

Segundo  
for

# Retos!

Primer Reto:

Crea un arreglo de 5 filas por 6 columnas en donde:



- Los primeros 5 elementos cada fila tengan calificaciones aprobatorias entre 6 y 10.
- El sexto elemento de cada fila debe ser 0.
- Calcula el promedio de los primeros 5 elementos de cada fila y asignalo al sexto elemento.
- Imprime el promedio de cada fila de calificaciones.



# Resultado:

```
"C:\Users\Dani\Documents\PlatziProgramaci3n estructurada\SumaFilasContl...
Array Bidimensional - Promedio de calificaciones.

La sumatoria de arrayB[0][5], es: 6.000000
La sumatoria de arrayB[0][5], es: 13.000000
La sumatoria de arrayB[0][5], es: 19.000000
La sumatoria de arrayB[0][5], es: 26.000000
La sumatoria de arrayB[0][5], es: 34.000000

    El promedio de la fila 0, es: 6.800000

La sumatoria de arrayB[1][5], es: 8.000000
La sumatoria de arrayB[1][5], es: 16.000000
La sumatoria de arrayB[1][5], es: 23.000000
La sumatoria de arrayB[1][5], es: 32.000000
La sumatoria de arrayB[1][5], es: 39.000000

    El promedio de la fila 1, es: 7.800000

La sumatoria de arrayB[2][5], es: 10.000000
La sumatoria de arrayB[2][5], es: 20.000000
La sumatoria de arrayB[2][5], es: 29.000000
La sumatoria de arrayB[2][5], es: 39.000000
La sumatoria de arrayB[2][5], es: 47.000000

    El promedio de la fila 2, es: 9.400000

La sumatoria de arrayB[3][5], es: 10.000000
La sumatoria de arrayB[3][5], es: 19.000000
La sumatoria de arrayB[3][5], es: 28.000000
La sumatoria de arrayB[3][5], es: 37.000000
La sumatoria de arrayB[3][5], es: 45.000000

    El promedio de la fila 3, es: 9.000000

La sumatoria de arrayB[4][5], es: 8.000000
La sumatoria de arrayB[4][5], es: 15.000000
La sumatoria de arrayB[4][5], es: 21.000000
La sumatoria de arrayB[4][5], es: 28.000000
La sumatoria de arrayB[4][5], es: 36.000000

    El promedio de la fila 4, es: 7.200000
```

---

# Cadena de caracteres

---

# ¿Que es una cadena de caracteres?

Generalmente se dice que es un arreglo de caracteres cuando lo que se almacenó son caracteres y no existe el carácter nulo al final.

Cuando el arreglo de caracteres termina con el carácter nulo se llama cadena de caracteres

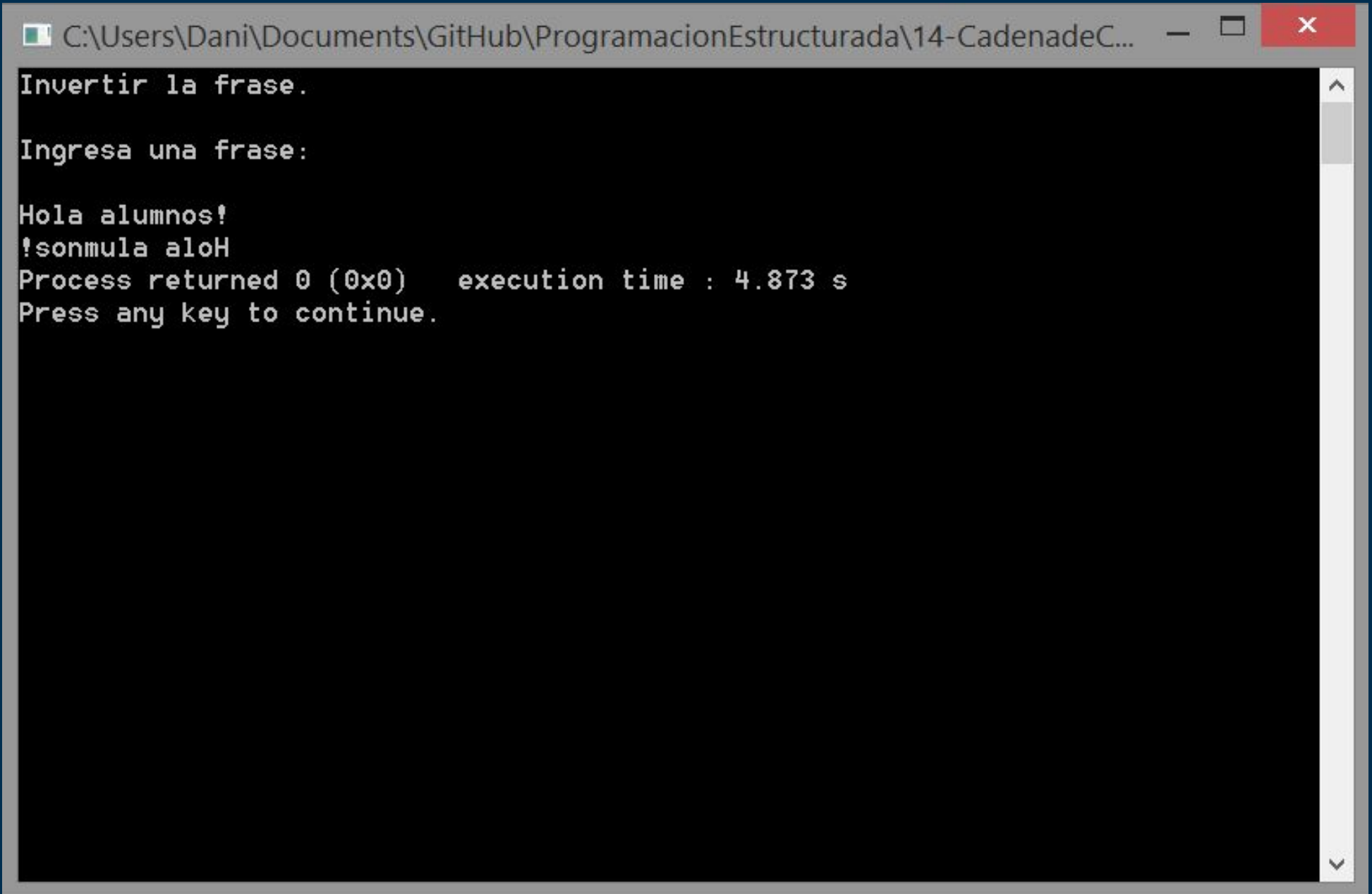


# Retos!

Primer Reto:

- Haz un programa que reciba una cadena de caracteres e imprima de regreso la misma cadena de forma invertida.

# Resultado:



```
C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\14-CadenadeC...  
Invertir la frase.  
Ingresa una frase:  
Hola alumnos!  
!sonmula aloH  
Process returned 0 (0x0)   execution time : 4.873 s  
Press any key to continue.
```

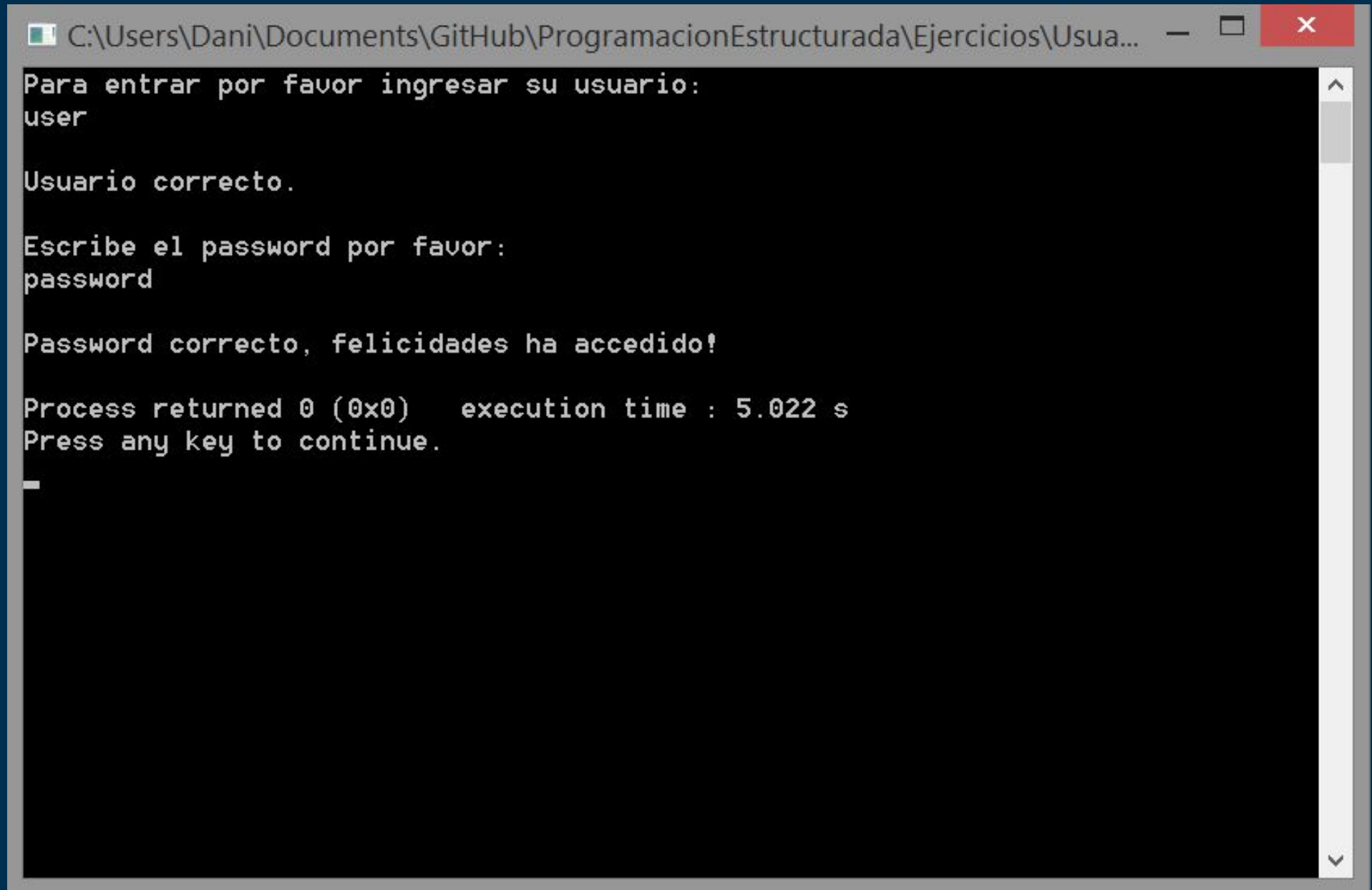
# Retos!

## Segundo Reto:



- Crear un usuario y contraseña pre-escritos.
- Pedirle al usuario que ingrese su usuario y contraseña.
- Comparar la información de entrada con la información pre-escrita.
- Si el usuario es correcto imprimir: “Usuario correcto, escriba el password por favor”
- Si el password es correcto imprimir: “Password correcto, felicidades ha accedido!”

# Resultado:



A screenshot of a Windows command prompt window. The title bar shows the file path: C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Usua... with standard minimize, maximize, and close buttons. The command prompt has a black background with white text. The text displayed is as follows:

```
Para entrar por favor ingresar su usuario:  
user  
  
Usuario correcto.  
  
Escribe el password por favor:  
password  
  
Password correcto, felicidades ha accedido!  
  
Process returned 0 (0x0)   execution time : 5.022 s  
Press any key to continue.  
_
```

---

# Funciones



---

# ¿Qué es una función?

Las funciones son bloques de código que realizan alguna operación.

Pueden aceptar datos de **entrada** (Parámetros) y devolver un dato de **salida**.



---

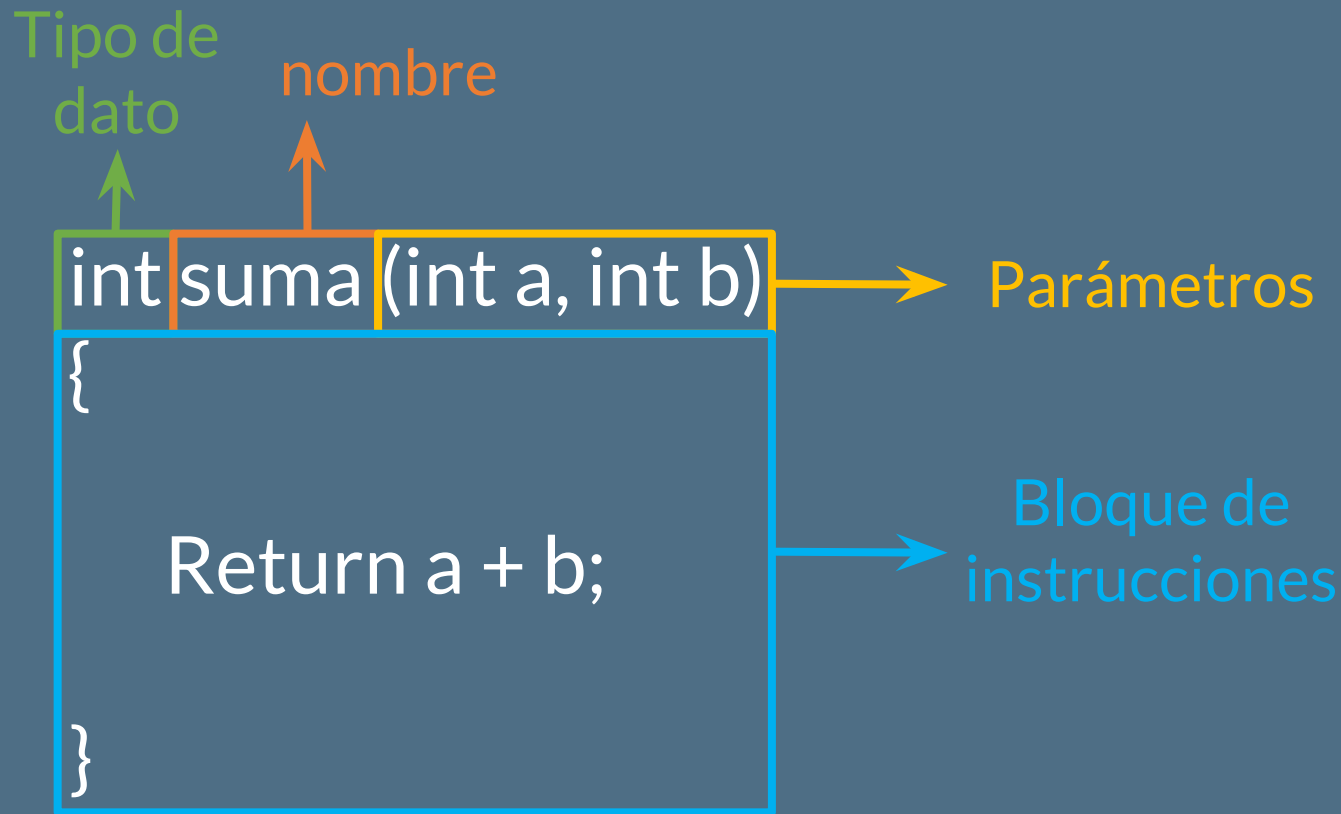
# ¿Para qué?

- Encapsulamiento
- Reusabilidad
- Separar Tareas
- Cambios a futuro

# Declaración de una función

```
tipo_de_dato nombre (parametros)
{
    Bloque de instrucciones;
}
```

# Declaración de una función



# Retos!

Primer reto:



- Vamos a calcular la potencia de un número.
- Ingresar un valor base.
- Ingresar un valor de exponente.
- Dentro de una función calcular el exponente del número base.
- Imprime el resultado.

# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Pote...

Potencia de un numero.

Base: 2

Exponente: 3

El resultado es: 8.000000

Process returned 0 (0x0) execution time : 2.136 s

Press any key to continue.

# Retos!



Primer reto:

- Hacer un programa de cambio de dólares a tu moneda y de tu moneda a dólares.
- Usa funciones.

# Resultado:

```
C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Conv...  
Vamos a convertir cantidades de dinero  
Si quieres convertir de dolares a pesos presiona P  
Si quieres convertir de pesos a dolares presiona DD  
  
Escribe la cantidad que quieres cambiar45  
El resultado es: 2.205000  
Process returned 0 (0x0)    execution time : 9.255 s  
Press any key to continue.  
-
```



---

# Variables globales

---

# ¿Qué es una variable global?

Las variables globales se declaran fuera de cualquier función y, según donde se declaren, varias funciones pueden tener acceso a ellas.

# Retos!



Primer reto:

- Utilizando variables globales, ingresa el nombre de un alumno y su calificación.
- En una función evalúa si el alumno ha sido aprobado o no.
- La calificación mínima aprobatoria es 7.
- Imprimir desde la función si el alumno aprobó.

# Resultado:

C:\Users\Dani\Documents\GitHub\ProgramacionEstructurada\Ejercicios\Califi...

Calificacion.

Ingresa el nombre: Platzi

Ingresa la calificacion: 100

El alumno Platzi ha sido aprobado.

Process returned 0 (0x0) execution time : 5.816 s

Press any key to continue.

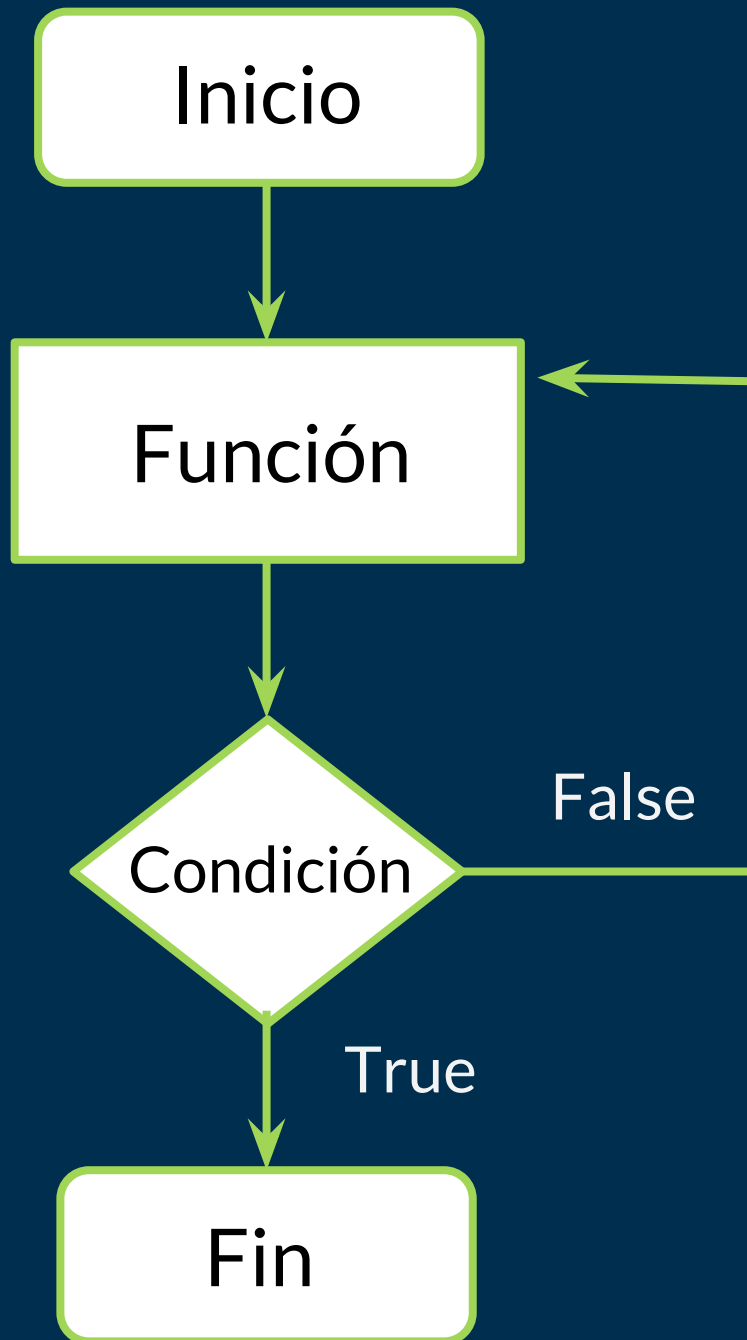
---

# Recursividad

---

# ¿Qué es recursividad?

- En C, las funciones pueden llamarse a sí mismas.
- Si una expresión en el cuerpo de una función llama a la propia función, se dice que ésta es recursiva.
- La recursividad es el proceso de definir algo en términos de sí mismo y a veces se llama **definición circular**.



---

# Apuntadores



---

# ¿Qué es un apuntador?

Un apuntador es una variable que guarda la dirección de memoria de otra variable.

---

# Operación de dirección

El Operador de Dirección (&) regresa la dirección de una variable.

---

# Operación de indirección

El Operador de Indirección ( \* ), toma la dirección de una variable y regresa el dato que contiene esa dirección.

---

# Sintaxis

float val;

Declaración del apuntador

float\* apVal;

Regresa la dirección de una variable

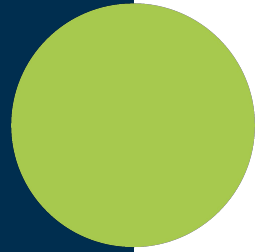
apVal = &val;

Asignar el dato en la dirección de memoria guardada

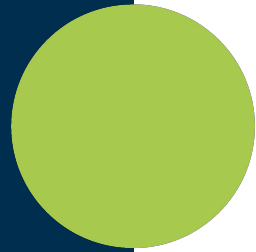
\*apVal = 3.1416;

---

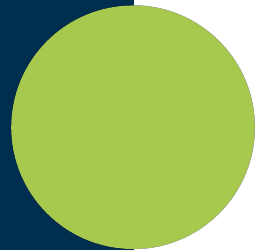
# **Manejo de archivos**



Creación, apertura y  
cierre



Escritura



Lectura

---

# Creación y apertura

Parámetros para la función `fopen()` son:

- `"rb"` : Abre un archivo en modo binario para lectura, el fichero debe existir.
- `"w"` : abrir un archivo en modo binario para escritura, se crea si no existe o se sobrescribe si existe.

# Creación y apertura

```
fopen("DatosPersonales.dat", "r");
```

Nombre y  
extensión del  
archivo

Modo de  
accesibilidad  
del archivo

## Cierre

```
fclose(archivo);
```

Archivo por cerrar



---

# Escritura

Cuenta con cuatro parámetros, estos son:

```
fwrite(void *apuntador, size_t tamaño, size_t cantidad, FILE archivo);
```

↓  
Apuntador con la  
dirección de la  
información que estamos  
guardando

↓  
Tamaño de cada  
elemento que será  
escrito

↓  
Número de  
elementos

↓  
Apuntador con la  
dirección al archivo  
donde estamos  
guardando/escribiendo

---

# Lectura

Cuenta con cuatro parámetros, estos son:

```
fread(void *apuntador, size_t tamaño, size_t cantidad, FILE archivo);
```

↓  
Apuntador con la  
dirección de la  
información que estamos  
leyendo

↓  
Tamaño de cada  
elemento que será  
leído

↓  
Número de  
elementos

↓  
Apuntador con la  
dirección al archivo  
donde estamos leyendo

---

# Librerías

---

# ¿Qué es una librería?

Una librería es código de programación ya escrito; un conjunto de funciones independientes para solucionar problemas concretos.

Nombre del código donde se llamará la librería

↑  
miCodigo.c

```
#include "libreria.h"
```

Hay que llamar la librería previamente hecha con #include

```
main()
```

```
{
```

```
    miFuncion();
```

```
}
```

Función creada en libreria.h

Nombre del código

↑  
libreria.h → Extensión

```
void miFuncion()
{
    //Instrucciones
}
```

↓  
Función a llamar en cualquier otro código