

MATERIA: Fundamento de Computadores Digitales

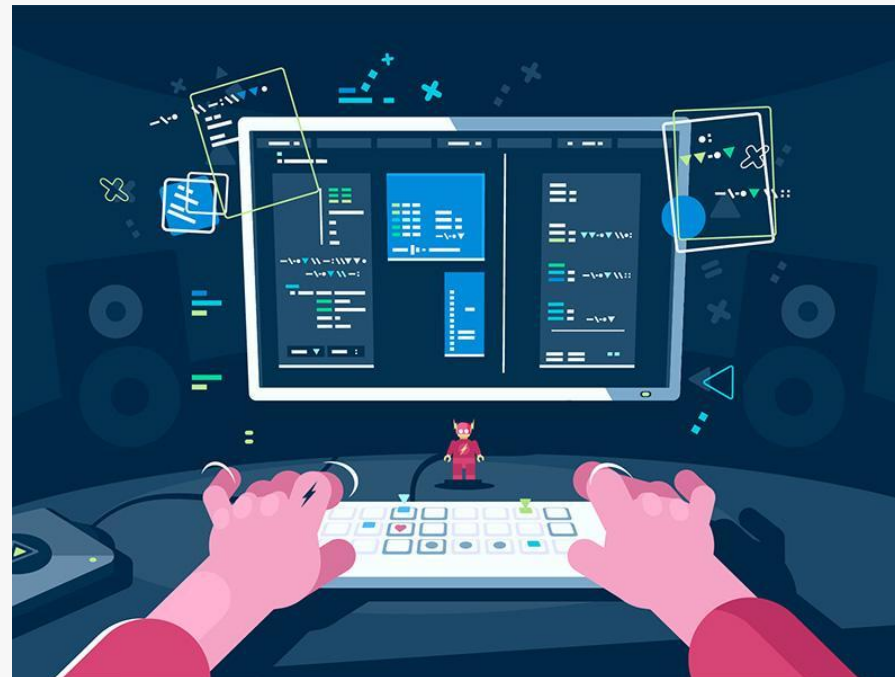


INTRODUCCIÓN AL C

INGRESO DE DATOS Y SALIDA DE INFORMACIÓN

Introducción a la Programación

Hasta el momento hemos realizado algoritmos para resolver problemas, y los hemos representado tanto en Diagramas de Flujo, como en Pseudocódigo. Pero para que un ordenador pueda ejecutar dichos algoritmos es necesario **codificarlos**.



Según su nivel de abstracción

De Máquina

Utilización de lenguaje o código máquina interpretable directamente por el microprocesador. Lenguaje compuesto por conjunto de instrucciones que indican acciones para la máquina.

- 
- Lenguaje Máquina

Bajo nivel

Está relacionado íntimamente con el hardware del ordenador, ya que ejerce un control directo sobre este, y depende directamente de su arquitectura. En este lenguaje las instrucciones se escriben en códigos alfabéticos, conocidos como mnemotécnicos (abreviaturas de palabras).

- 
- Ensamblador

Medio nivel

Los lenguajes de nivel medio son aquellos que permiten un grado de abstracción como los lenguajes de alto nivel, pero no son tan confortables en su programación al no tener automatizaciones en el control errores, control de sintaxis, etc.

- 
- Lenguaje C

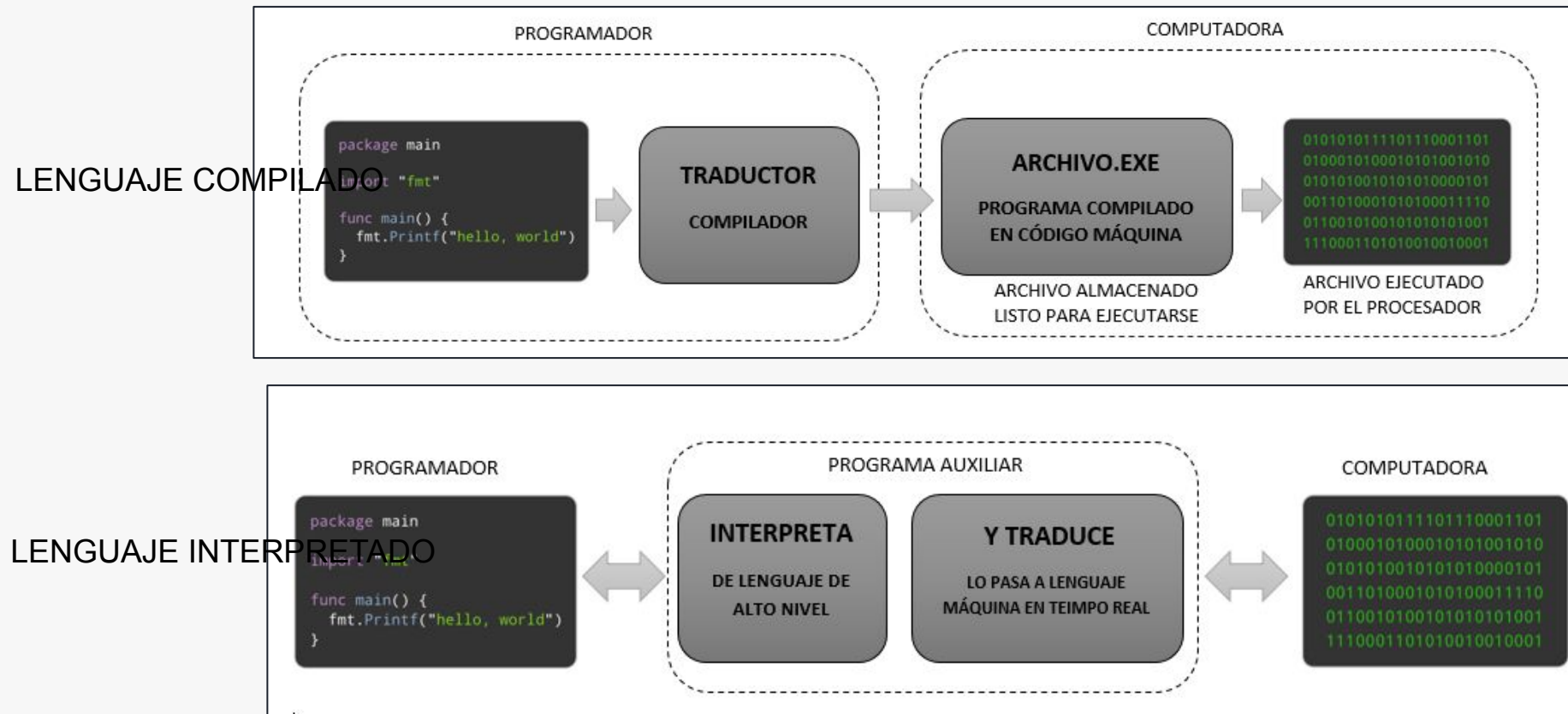
Alto nivel

De fácil comprensión. Esto se debe a que se escribe con palabras similares a las del lenguaje humano, lo que le da un alto nivel de abstracción del lenguaje máquina. Se pueden ejecutar en diferentes tipos de ordenadores con ninguna o muy pocas modificaciones. Son independientes de la máquina,

- 
- Java
 - PHP
 - Python

Según su ejecución

Como estos lenguajes **no pueden ser entendidos directamente por la máquina**, ya que se “alejan” del procesador, deben ser **traducidos**. En líneas generales, pueden clasificarse en: lenguajes interpretados y lenguajes compilados.



Introducción a Lenguaje C



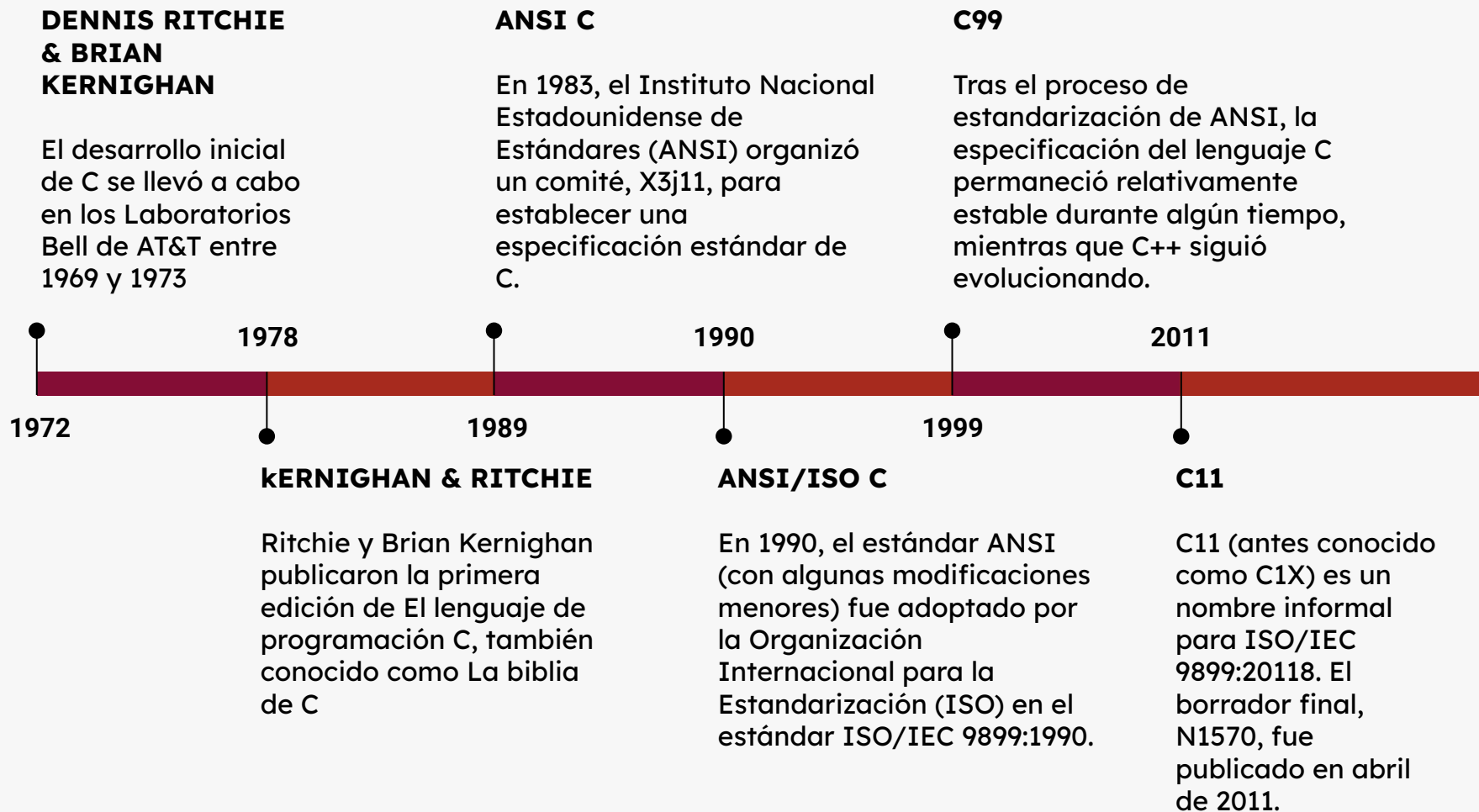
- Historia y Estándares
- Estructura de programa en C
- Tipos de Datos
- Variables - Constantes - literales
- Operadores
- Ingreso y Egreso de Datos
- Secuencias de escape

Introducción a Lenguaje C



- De **nivel medio**
- Sirve para crear aplicaciones y software de sistemas.
- Es **portable**, es decir, posibilita la adaptación de programas escritos para un tipo de computadora en otra.
- Es de fácil aprendizaje.
- Posee un completo conjunto de **instrucciones de control**.
- Al ser un **lenguaje estructurado** se divide el programa en módulos, lo que permite que puedan compilarse de modo independiente.
- Trabaja con **librerías** de funciones en las que básicamente solo se necesita cambiar los valores dentro de una aplicación dada.
- Es uno de los lenguajes más populares. Muy utilizado, especialmente en el campo de la **ingeniería** y el campo científico.
- Dispone de excelentes **compiladores de C gratuitos**, para casi cualquier plataforma sobre la que se quiera trabajar y con entornos de programación claros y funcionales.

Historia y estándares



Estructura básica de un programa en C

La estructura general básica de un programa escrito en C es la siguiente:

Inclusión de librerías
Declaración de constantes
Comienzo del programa
Declaración de variables
Cuerpo del programa

```
#include <stdio.h>
/* Esto es un comentario
   en varias líneas*/

int main()
{
    printf("Hola Mundo!\n");
    // otro comentario: printf es una función (de
    stdio)
    return 0;
}
```


Inclusión de librerías

Consisten en archivos de código, que poseen funciones y rutinas estandarizadas escritas en un lenguaje de programación, que podemos incluir en nuestros programas.

Para poder utilizarlas, el programador debe invocar todas aquellas librerías que necesite al principio del programa.

Ejemplos de librerías estándar:

- `#include <stdio.h>`
- `#include <stdlib.h>`
- `#include <string.h>`
- `#include <math.h>`
- `#include <time.h>`

`#include <nombre_libreria.h>`

Declaración de constantes

Una constante es un valor que **permanece inalterable** durante todo el programa, es decir, que no puede ni debe cambiar de valor.

Su declaración se indica antes del comienzo del programa, y su valor debe ser conocido previamente.

La manera de declarar constantes en C es utilizando la palabra reservada `#define`.

```
#define PI 3.1416
```

Atención: la declaración de constantes no necesita ; (punto y coma) para finalizar la línea de comando.

Comienzo del programa

La filosofía en la que se basa el diseño del Lenguaje C es el empleo de funciones. Por esta razón, un programa en C contiene al menos una función, la función main. Esta función es particular dado que la ejecución del programa se inicia con las instrucciones contenidas en su interior, y controla las llamadas a otras funciones.

La sintaxis general del main es la siguiente:

```
int main ()  
{  
    //Comienzo del programa...  
  
    .  
  
    .  
  
    .  
  
    return 0; //Fin del programa  
}
```

Instrucciones y Sentencias

Una **instrucción** es lo que le ordenamos a la maquina para que ejecute, por eso se dice que un programa es un conjunto de **instrucciones**; ya que cuando ejecutamos un programa, se ejecutan así mismo en su interior muchas **instrucciones** que hacen que la maquina realice algo.

Las sentencias son las que realmente realizan las operaciones. Una sentencia puede tener varias instrucciones. Para que una instrucción se convierta en sentencia en **Lenguaje C** se utiliza el carácter de ; (punto y coma) al final.

```
int a;  
  
int numB, numC;  
  
float numD = 5.21;  
  
numD = numD * 5;
```

Declaración de variables

Declarar es identificar con un **nombre** y un **tipo** a una variable. La sintaxis general es:

tipo_dato_variable nombre_variable;

tipo_dato_variable nombre_variable = valor;

Ejemplos:

- **int** A;
- **int** A,B;
- **float** suma;
- **char** C1;
- **float** S1,S2;
- **int** D = 5;
- **float** num = 5.21;

Atención: en este lenguaje no puede quedar ninguna variable sin declarar. Además es importante saber que se diferencian las minúsculas de las mayúsculas.

Tipos de datos

Un tipo de dato es la propiedad de un **valor** que determina su **dominio** (qué valores puede tomar), qué operaciones se le pueden aplicar y cómo es representado internamente por la computadora.

C ofrece cuatro tipos de datos básicos:

- Números enteros
- Números reales
- Letras o caracteres
- Booleanos

Números enteros

Se definen con “**int**” y admiten de forma opcional dos prefijos modificadores:

- “**short**” y “**long**”: Modifica el tamaño en bits del entero. Existen por tanto tres tipos de enteros: “int”, “short int” (que se puede abreviar como “short”), y “long int” (que se puede abreviar como “long”).
- “**unsigned**”: define un número natural (mayor o igual a cero).

Nombre	Tamaño	Alcance
short	16 bits	desde -32.768 hasta 32.767
unsigned short	16 bits	desde 0 hasta 65535
int	32 bits	desde -2.147.483.648 hasta 2.147.483.647
unsigned int	32 bits	desde 0 hasta 4.294.967.295
long	32 bits	desde -2.147.483.648 hasta 2.147.483.647
unsigned long	32 bits	desde 0 hasta 4.294.967.295
long long	64 bits	desde -9.223.372.036.854.775.808 hasta 9.223.372.036.854.775.807
unsigned long long	64 bits	desde 0 hasta 18.446.744.073.709.551.615

Números reales

Los números reales se definen con “**float**” o “**double**”. La diferencia entre ambas es la precisión que ofrece su representación interna. Hay un número infinito de reales, pero se representan con un número finito de bits.

Nombre	Tamaño	Alcance
float	32 bits	desde <u>1.17549e-38</u> hasta <u>3.40282e+38</u>
double	64 bits	desde <u>2.22507e-308</u> hasta <u>1.79769e+308</u>
long double	96 bits (80 bits reales)	desde <u>1.68105e-4932</u> hasta <u>1.18973e+4932</u>

Letras o caracteres

Involucra un conjunto ordenado y finito de símbolos que el procesador puede reconocer.

Si bien no existe un conjunto estándar, podemos decir que dicho conjunto está básicamente integrado por:

- Letras mayúsculas (desde la A hasta la Z), sin incluir la CH y la LL (eventualmente puede no ser incluida la Ñ).
- Letras minúsculas (desde la a hasta la z), con las mismas restricciones que para las mayúsculas.
- Dígitos (del 0 al 9).
- Caracteres especiales

Atención: en C los caracteres van delimitados por comillas simples.
Por ejemplo: categoría = 'A'

Booleanos

En programación, el tipo de dato lógico o booleano es aquel que puede representar valores lógicos de forma binaria, por lo general representan “**verdadero**” o “**falso**”.

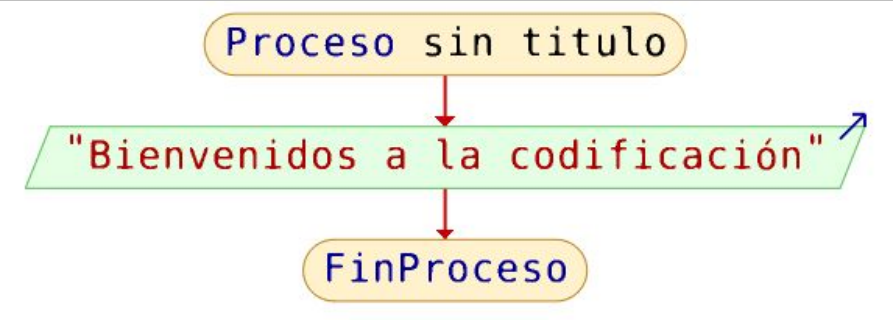
En C puede tomar los valores de “**true**” o “**false**” respectivamente, y su sintaxis general es:

```
bool nombre_variable;
```

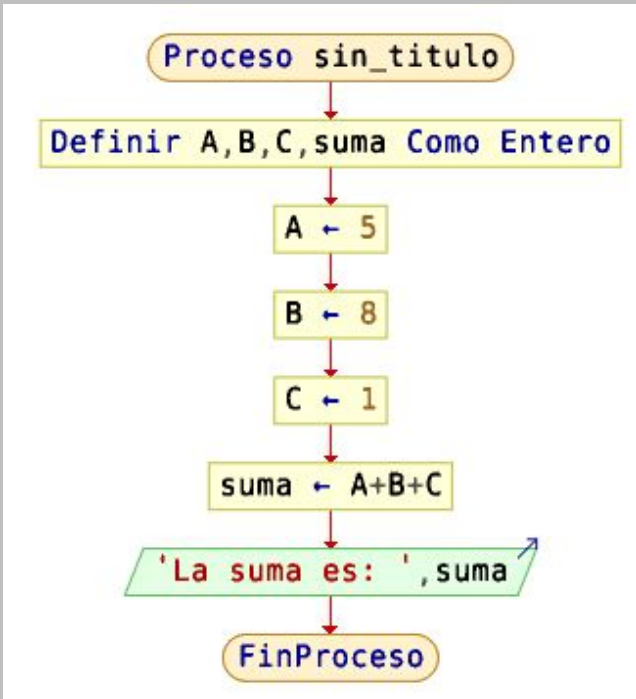
```
bool nombre_variable2 = false;
```

Cuerpo del programa

Aquí es donde debemos introducir nuestro algoritmo. En nuestro caso, traduciremos nuestro diagrama de flujo y/o pseudocódigo en **lenguaje c**.

Diagrama de flujo	Código
 <pre>graph TD; A([Proceso sin titulo]) --> B[/"Bienvenidos a la codificación"/]; B --> C([FinProceso]);</pre>	<pre>#include <stdio.h> int main() { printf("Bienvenidos a la Codificación"); return 0; }</pre>

Cuerpo del programa

Diagrama de flujo	Código
 <pre>graph TD; Inicio([Proceso sin_titulo]) --> Def[Definir A,B,C,suma Como Entero]; Def --> A[A ← 5]; A --> B[B ← 8]; B --> C[C ← 1]; C --> Suma[suma ← A+B+C]; Suma --> Print[/'La suma es: ', suma/]; Print --> Fin([FinProceso]);</pre>	<pre>#include <stdio.h> int main() { int A, B, C, suma; A = 5; B = 8; C = 1; suma = A + B + C; printf("La suma es %i ",suma); return 0; }</pre>

Printf()

El printf puede mostrar el contenido de una o más variables y, por lo general, va acompañado de carteles o literales.

- Pertenecen a la biblioteca estándar “stdio” (standard input output).
- Se utilizan máscaras, secuencias de escape, para indicar donde van los datos. Estas secuencias comienzan con %.

<i>Máscara</i>	<i>Imprime</i>
<i>%i o %d</i>	<i>Un entero</i>
<i>%f</i>	<i>Un float</i>
<i>%lf</i>	<i>Un double</i>
<i>%c</i>	<i>Un único carácter</i>
<i>%s</i>	<i>Una cadena de caracteres</i>
<i>%(número)s</i>	<i>Una cadena de caracteres limitada por un número, por ejemplo:</i>
<i>%5s</i>	<i>(en estos casos imprimirá los cinco primeros caracteres)</i>
<i>%%</i>	<i>Esto es por si queremos imprimir el símbolo de porcentaje</i>
<i>%(número1).(número2)f</i>	<i>Un número con decimales. El tamaño es número1 y la cantidad de decimales es número2. Por ejemplo: %6.2f (el tamaño del número será 6 y tendrá 2 decimales)</i>

Printf()

```
printf("Valor de variable: %d \n", numero_ingresado);
```

Salida: Valor de variable: 5

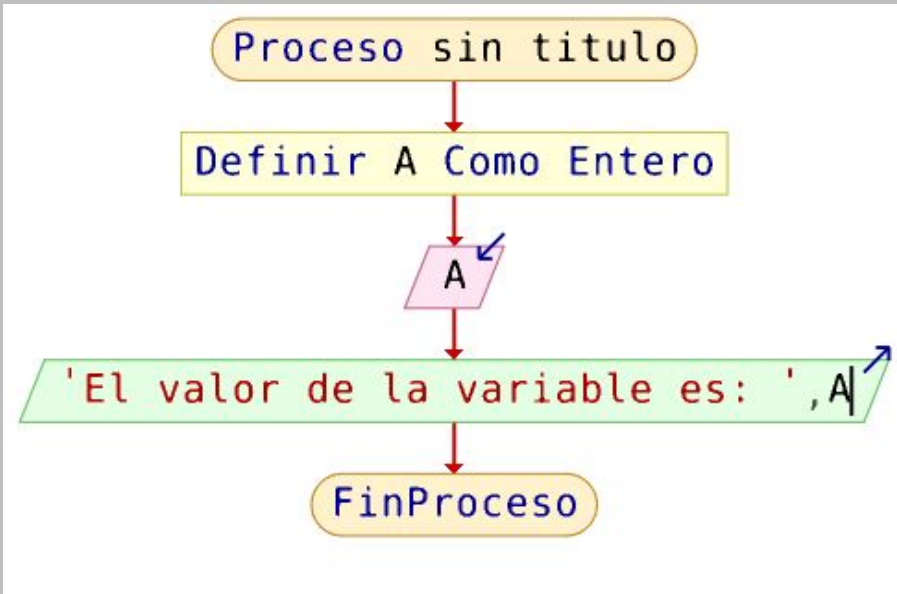
```
printf("Valor de variable multiplicado: %d \n", numero_ingresado * 2);
```

Salida: Valor de variable multiplicado: 10

```
printf("Valor de variable multiplicado: %d \t numero literal: %d \n",  
numero_ingresado * 5, 7);
```

Salida: Valor de variable multiplicado: 10 numero literal: 7

Otro ejemplo:

Diagrama de flujo	Código
 <pre>graph TD; A([Proceso sin titulo]) --> B[Definir A Como Entero]; B --> C[/A/]; C --> D[/El valor de la variable es: ', A/]; D --> E([FinProceso]);</pre>	<pre>#include <stdio.h> int main() { int A; scanf("%i",&A); printf("El valor de la variable es: %i",A); return 0; }</pre>

Scanf()

El **scanf** es la función que nos ayuda a obtener los datos por teclado.

Tiene asociados dos parámetros: la **máscara** y la **variable de ingreso**. Delante de la variable hay un “&” que nos da como resultado la dirección de la variable, donde se va a almacenar el dato.

Para ingresar un valor en la variable dato:

```
int dato ;  
scanf ( "%d", &dato) ;  
printf ( "Valor ingresado en variable dato: %d \n", dato );
```


Comentarios en la codificación

```
#include <stdio.h>

/* Esto es un comentario

En varias líneas */

int main() { //Esto es un comentario en una sola línea

    printf("Bienvenidos a la Codificación");

    return 0;

}
```

La misión de estos comentarios es servir de explicación o aclaración sobre cómo está desarrollado el programa, de forma que pueda ser entendido por cualquier otra persona o por el propio programador un tiempo después.

Caracteres de escape printf

Código	Significado	Valor ASCII (Decimal)	Valor ASCII (Hexadecimal)
'\n'	Nueva línea (dependiente SO)	10	0x0A
'\r'	Retorno de carro	13	0x0D
'\t'	Tabulador (horizontal)	09	0x09
'\f'	Nueva página	12	0x0C
'\a'	Alerta (campana)	07	0x07
'\b'	Retroceder un caracter	08	0x08
'\v'	Tabulador (vertical)	11	0xB
'\\'	Barra invertida	92	0x5C
'\"'	Comilla simple	39	0x27
'\"'	Comilla doble	34	0x22
'\ddd'	El caracter ASCII cuyo código sea ddd en octal		
'\xhh'	El caracter ASCII cuyo código sea nn en hexadecimal		
Nota: Este listado no es completo			