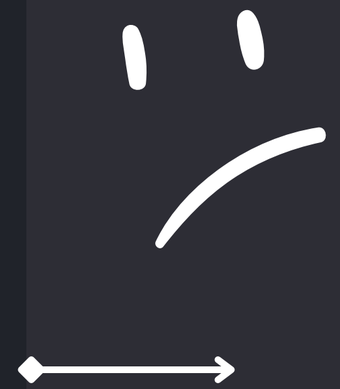
```
1  import { useState } from "react";
2
3  // Programación I - Unidad 01
4
5
6  class Algoritmos() {
7
8
9      const [prof, setProf] = useState({
10
11         setProf: "Miguel Silva."
12     });
13
14
15
16     return prof;
17
18
19 };
20
21
22 export default Algoritmos;
23
```

UTN INSPT

// Laboratorio de Informática - Unidad 02

Programación???
Algoritmos???
Códigos???

"Cosme Fulanito"



Realidad simulada.

```
// Laboratorio de Informática - Unidad 02


```

Concepto de Programacion() {

Proceso de crear instrucciones que una computadora puede seguir para realizar una tarea específica a través de lenguajes de programación. Están diseñados para ser entendidos tanto por humanos como por máquinas.

};

```
export default Programacion;
```

```
// Laboratorio de Informática - Unidad 02


```

Concepto de Algoritmo() {

Es un conjunto ordenado de instrucciones (o pasos) que permite solucionar un problema o realizar una tarea en un número finito de pasos.

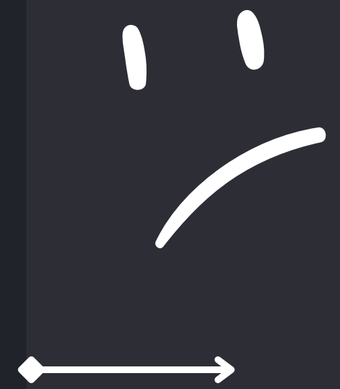
};

```
export default Algoritmo;
```

// Laboratorio de Informática - Unidad 02

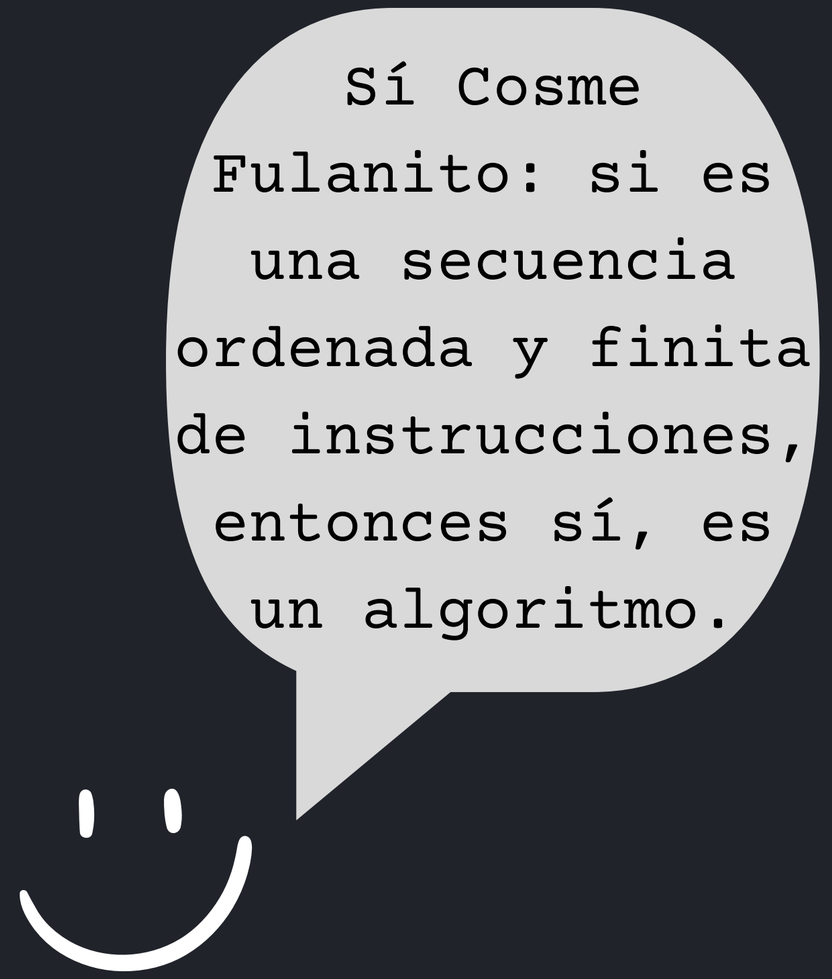
¿Entonces la receta de un alto guiso es un algoritmo?

"Cosme Fulanito"

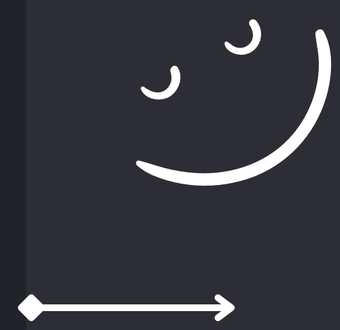


Realidad simulada.

// Laboratorio de Informática - Unidad 02



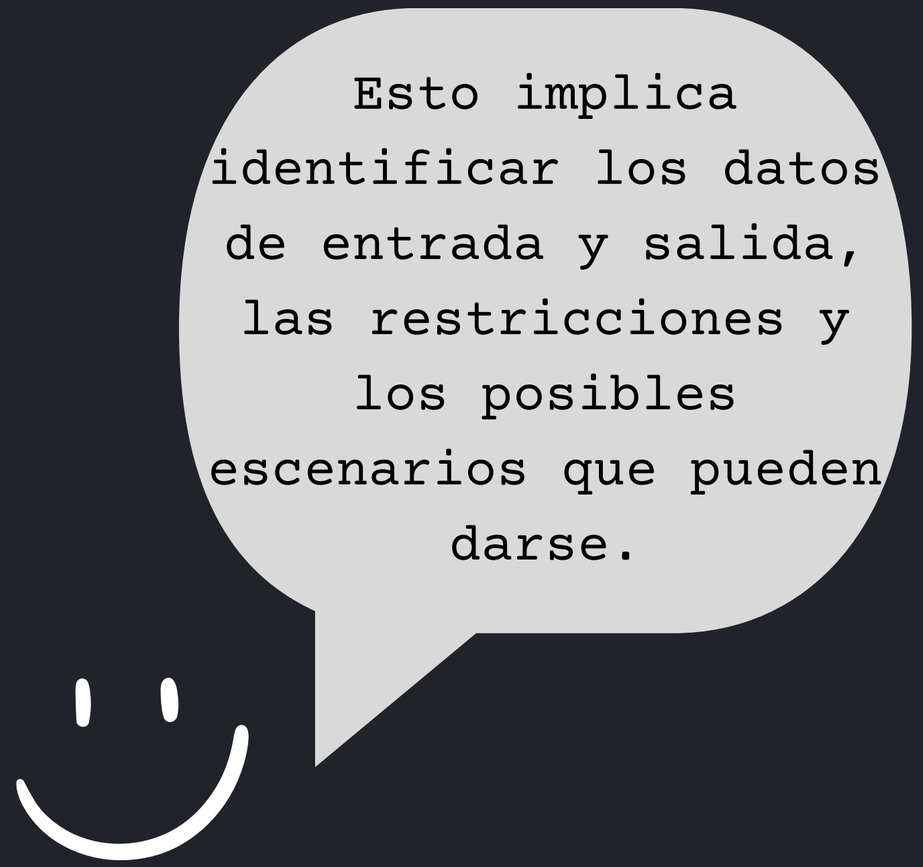
"The ticher"



Realidad simulada.

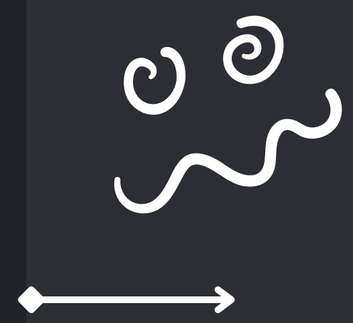
// Comprensión de enunciados:

Para resolver un problema mediante un algoritmo, es fundamental comprender el enunciado del mismo.

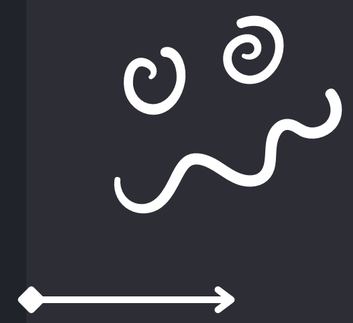


"The ticher"

```
// Estrategias de resolución de problema
// Estado inicial y estado final:

estrategias para Problemas() {

    Existen diferentes estrategias para la
    resolución de problemas, como la
    división del problema en subproblemas,
    la aplicación de la técnica de fuerza
    bruta o el uso de heurísticas.

};

export default Problemas;
```



```
// Estrategias de resolución de problema
// Estado inicial y estado final:

estado InicialyFinal() {
    Es importante definir el estado
    inicial del problema (es decir, la
    situación en la que se encuentra el
    problema en el momento en que se
    plantea) y el estado final (la
    situación deseada).
};

export default InicialyFinal;
```



```
// Lenguajes algorítmicos:


distintos lenguajes Algoritmicos() {

    Como el código y el diagrama de flujo.
    Los lenguajes permiten representar de
    manera textual el conjunto de
    instrucciones que conforman el
    algoritmo, mientras que los diagramas
    de forma gráfica..
};

export default Algoritmicos;
```

```
// Lenguajes algorítmicos:


lenguajes de_programacion() {
    Notación o conjunto de símbolos y caracteres que se combinan entre sí, siguiendo las reglas de una sintaxis predefinida, con el fin de posibilitar la transmisión de instrucciones a un ordenador.
};

export default de_programacion;
```

// Lenguajes algorítmicos:

Este es un ejemplo de algoritmo con C...

"The ticher"

C_projects > C Untitled1.c > ...

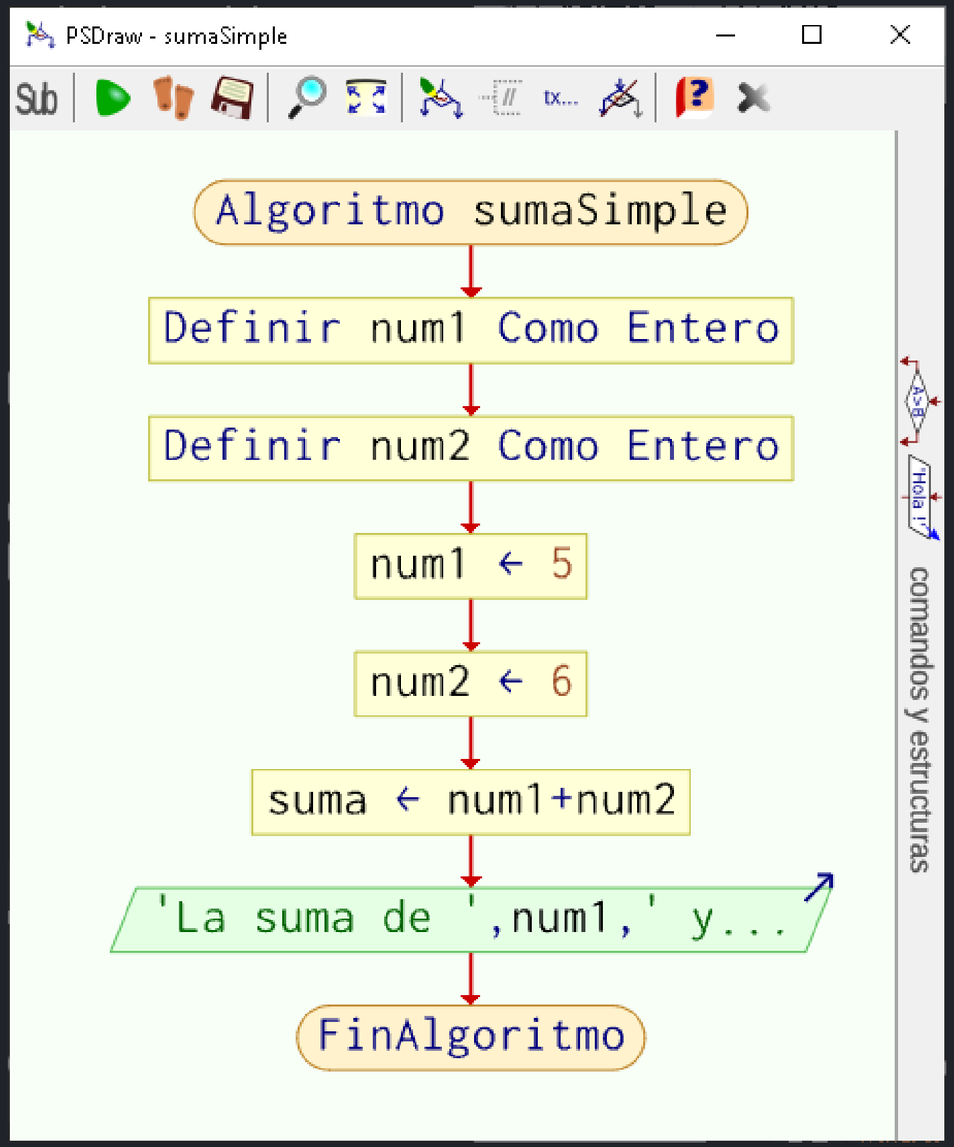
```c
#include <stdio.h>

int main()
{
    int num1, num2, suma;

    num1 = 5;

    num2 = 6;

    suma=num1+num2;

    printf("La suma es: %d", suma);

    return 0;
}
```

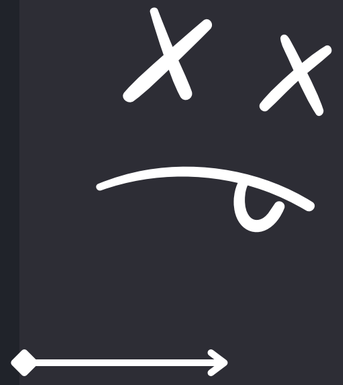
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\Mike> cd 'd:\C_projects\output'
PS D:\C_projects\output> & .\'Untitled1.exe'
La suma es: 11
PS D:\C_projects\output>
```

// Lenguajes algorítmicos:

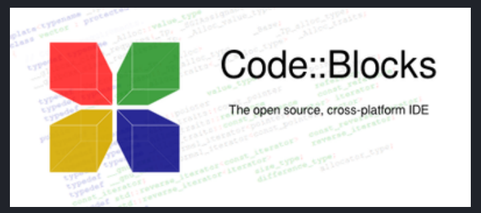
...Y este es un ejemplo de diagrama de flujo.

"The tichers"

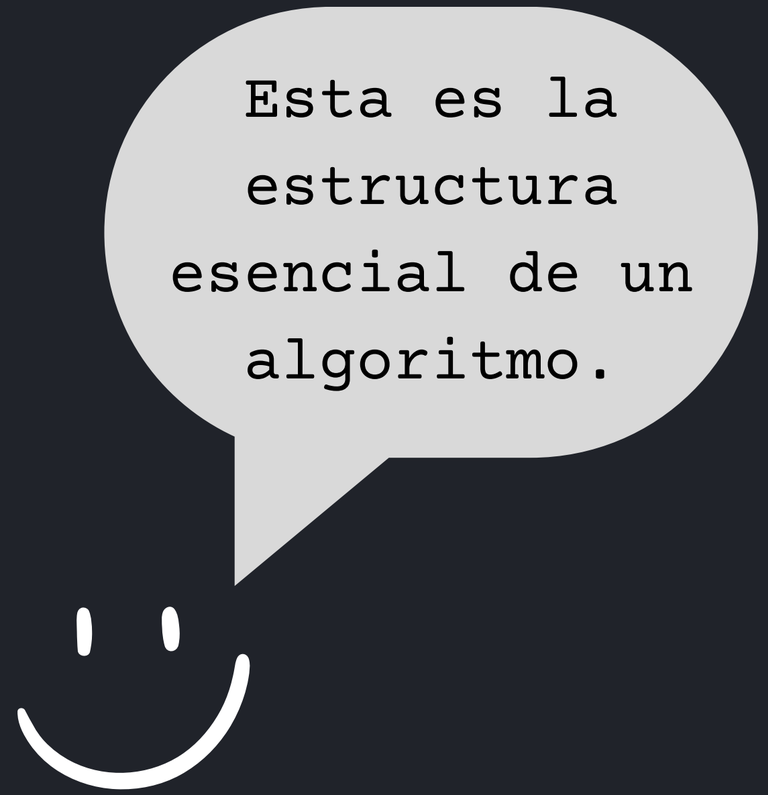




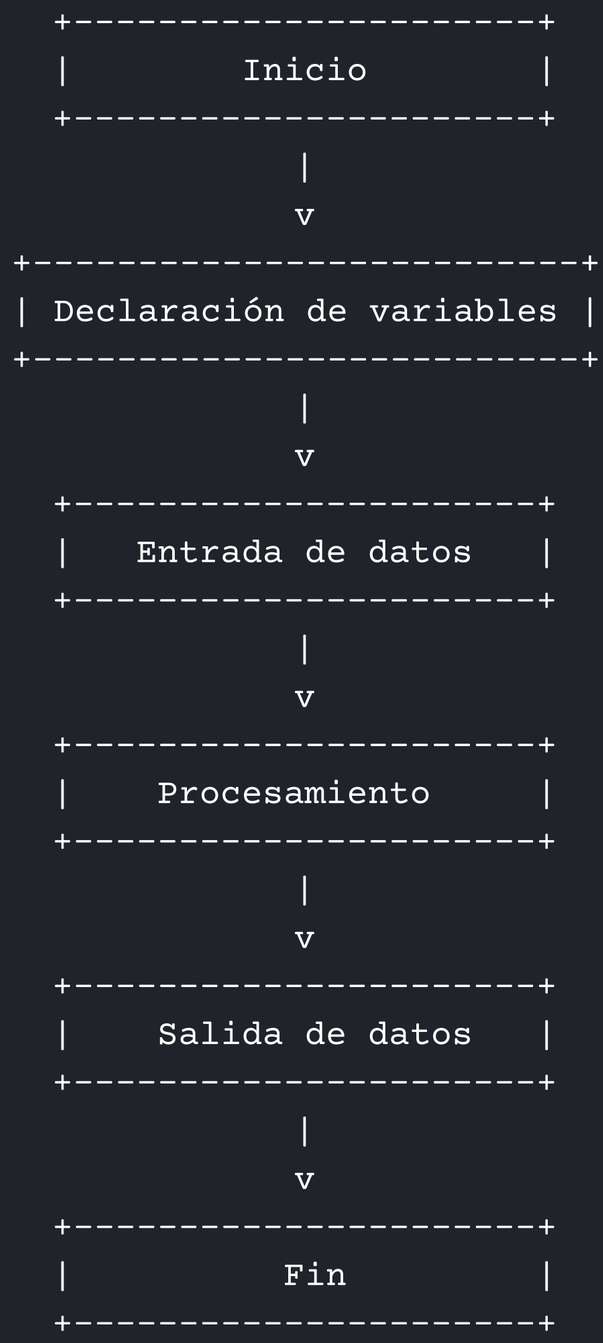
// Diferentes IDEs para diferentes lenguajes:



// Representación básica:

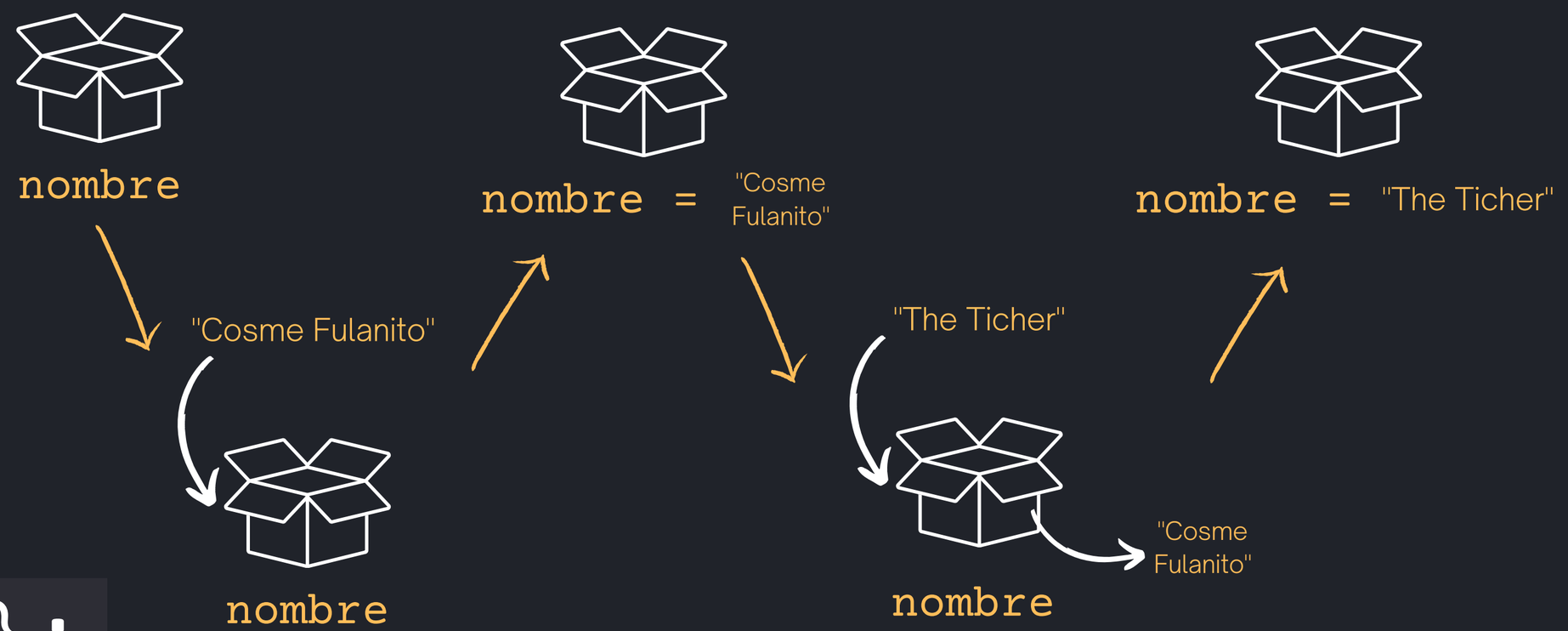


"The tichers"



```
// Elementos de las partes de un algoritmo:


entidades Primitivas() {
    Son elementos básicos que se utilizan
    en la construcción de algoritmos.
    Entre las entidades primitivas más
    comunes se encuentran las variables,
    las constantes, los operadores
    aritméticos y las funciones.
};

export default Primitivas;
```

```
// Elementos de las partes de un algoritmo:

function Variables() {
```




```
};

export default Variables;
```

```
// Elementos de las partes de un algoritmo:

tipos Variables() {

    123    "Cosme Fulanito"

                                true

    3,14e        'F'

                                false

};

export default Variables;
```

```
// Elementos de las partes de un algoritmo:

tipos variablesEnC() {
```

Tipo de dato	Descripción.
char	Carácter o entero pequeño (byte)
int	Entero
float	Punto flotante
double	Punto flotante (mayor rango que *float*)
void	Sin tipo (uso especial)

```
};

export default variablesEnC;
```

```
// Elementos de las partes de un algoritmo:

tipos modificadorVariable() {
```

Modificador	Tipos de actuación		Descripción
signed	char	int	Con signo (por defecto)
unsigned	char	int	Sin signo
long	int	double	Largo
short	int		Corto

```
};

export default modificadorVariable;
```

```
// Elementos de las partes de un algoritmo:

tipos modificadorVariable() {
```

	Rango de valores posibles en (notación matemática)	
Tipo de variable declarada	**16 bits**	**32 bits**
char / signed char	[-128 , 127]	[-128 , 127]
unsigned char	[0 , 255]	[0 , 255]
int / signed int	[-32768 , 32767]	[-2147483647 , 2147483648]
unsigned int	[0 , 65535]	[0 , 4294967295]
short int / signed short int	[-32768 , 32767]	[-32768 , 32767]
unsigned short int	[0 , 65535]	[0 , 65535]
long int / signed long int	[-2147483647 , 2147483648]	[-2147483647 , 2147483648]
unsigned long int	[0 , 4294967295]	[0 , 4294967295]
float	[-3.4E+38 , -3.4E-38], 0 , [3.4E-38 , 3.4E+38]	[-3.4E+38 , -3.4E-38], 0 , [3.4E-38 , 3.4E+38]
double	[-1.7E+308 , -1.7E-308], 0 , [1.7E-308 , 1.7E+308]	[-1.7E+308 , -1.7E-308], 0 , [1.7E-308 , 1.7E+308]
long double	[-3.4E+4932 , -1.1E-4932], 0 , [3.4E-4932 , 1.1E+4932]	[-3.4E-4932 , -1.1E+4932], 0 , [3.4E-4932 , 1.1E+4932]

```
};

export default modificadorVariable;
```

```
// Elementos de las partes de un algoritmo:
```

Combinaciones modificadores y de variables

Tipo	Cantidad de bits	Rango numérico
char	8	-128 a 127
unsigned char	8	0 a 255
signed char	8	-128 a 127
short (int)	16	-32768 a 32767
int OJO!!!!	16	-32768 a 32767
unsigned int	16	0 a 65535
signed int	16	-32768 a 32767
short int	16	-32768 a 32767
unsigned short int	16	0 a 65535

```
// Elementos de las partes de un algoritmo:

tipos modificadorVariable() {
```

Modificador	Efecto
const	Variable de valor constante
volatile	Variable cuyo valor es modificado externamente

```
};

export default modificadorVariable;
```

```
// Elementos de las partes de un algoritmo:

tipos operadoresAritmeticos() {
```

Unarios	**Signo negativo**	−
	Incremento	++
	Decremento	−−
Binarios	**Suma**	+
	Resta	−
	Multiplicación	*
	División	/
	Módulo	%

```
};

export default operadoresAritmeticos;
```

```
// Elementos de las partes de un algoritmo:

tipos operadoresLogicos() {
```

Menor que	<
Mayor que	>
Menor o igual que	<=
Mayor o igual que	>=
Igual que	==
Distinto que	!=

Conjunción ó Y lógico	&&
Disyunción u O lógico	||
Negación ó NO lógico	!

```
};

export default operadoresLogicos;
```

```
// Elementos de las partes de un algoritmo:

operaciones Logicas() {
```

Negación (NO,-) Conjunción (Y, AND, &&) Disyunción (O, OR, ||)

A	B	! A	A && B	A || B
Cierto	Cierto	Falso	Cierto	Cierto
Cierto	Falso	Falso	Falso	Cierto
Falso	Cierto	Cierto	Falso	Cierto
Falso	Falso	Cierto	Falso	Falso

```
};

export default Logicas;
```

```
// Elementos de las partes de un algoritmo:

variables otrasBases() {() {
```

Prefijo	Base
0x	Hexa
0B	Binario
0	Octal
	Decimal (aunque puede variar)

```
};

export default otrasBases;
```

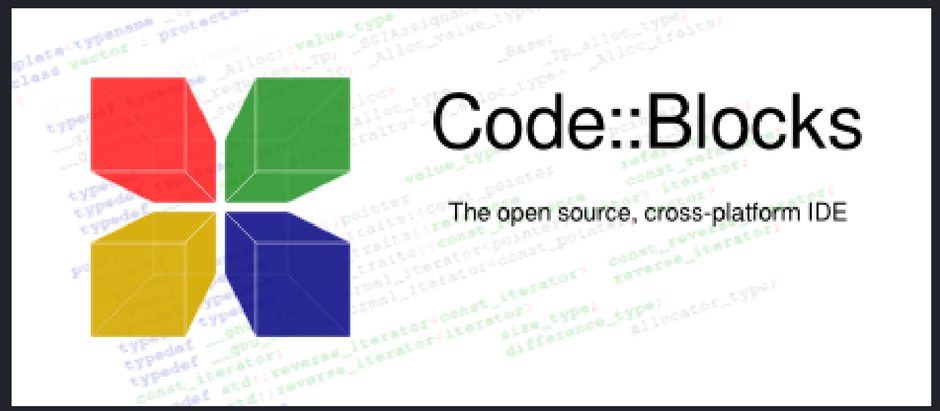
```
// Introducción a la programación en C:


programacion en_C() {
```

- Lenguaje de prog. de medio nivel
- Utilizado en el desarrollo de SO, compiladores y aplicaciones de bajo nivel.
- Eficiente y flexible.
- Requiere un mayor nivel de atención a los detalles.

```
};

export default en_C;
```

```
// Editores de texto para programar en C:
```

programación en_C() {



www.codeblocks.org

code.visualstudio.com

};

```
export default en_C;
```

```
// Partes de un “Hola mundo” en C:


programacion en_C() {
```

```c
#include <stdio.h>

int main() {
    printf("¡Hola, mundo!\n");
    return 0;
}
```

```
};


export default en_C;
```

```
// Palabras reservadas en C:


palabras Reservadas() {
```

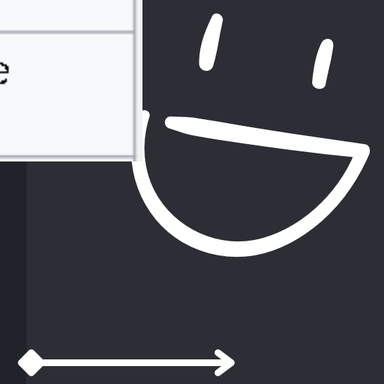
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

```
};


export default Reservadas;
```

```
// Formateadores de variables en C:

formateador printf() {() {
```

Formateador	Salida
%d ó %i	entero en base 10 con signo (int)printf ("el numero enteronen base 10 es: %d" , -10);
%u	entero en base 10 sin signo (int)
%o	entero en base 8 sin signo (int)
%x	entero en base 16, letras en minúscula (int)
%X	entero en base 16, letras en mayúscula (int)
%f	Coma flotante decimal de precisión simple (float)
%lf	Coma flotante decimal de precisión doble (double)

```
};

export default printf;
```



// Formateadores de variables en C:

formateador printf() {() {

Formateador	Salida
%ld	Entero de 32 bits (long)
%lu	Entero sin signo de 32 bits (unsigned long)
%e	La notación científica (mantisa / exponente), minúsculas (decimal precisión simple ó doble)
%E	La notación científica (mantisa / exponente), mayúsculas (decimal precisión simple ó doble)
%c	carácter (char)
%s	cadena de caracteres (string)

};

```
export default printf;
```

```
// Formateadores de variables en C:

formateador printf() {() {

        int main (void){

            int var=3,var2=5;
            printf("el valor de la variable var es: %d\n",var);
            printf("el valor de la variable var2 es: %d\n",var2);
            return 0;
        }

int main (void){

    int var=3,var2=5;
    printf("el valor de la variable var es: %d \nel valor de la variable var2 es: %d\n",var,var2);
    return 0;
}

    };

    export default printf;
```

// Formateadores de variables en C:

formateador scanf()

```c
#include <stdio.h>

int main() {
    int numero;
    printf("Por favor, ingrese un número: ");
    /* En scanf, el "&" se utiliza para obtener la dirección de memoria
       de la variable 'numero'.
       Esto permite que scanf pueda almacenar el valor ingresado por el
       usuario en la ubicación de memoria de 'numero'. */
    scanf "%d"
    printf("El número ingresado es: %d\n", numero);
    return 0;
}
```

```
export default scanf;
```

EJERCITACIÓN:

1. Con uno de sus lados, calcula el área y perímetro de un cuadrado.
2. Con la base y la altura de un rectángulo, calcular su área y su perímetro.
3. Con el diámetro de un círculo, calcula su área.
4. Convierte grados Celsius a Fahrenheit.
5. Determinar si un número es par o impar.
6. Calcula la suma de los números enteros del 1 al 100.
7. Verifica si un año ingresado por teclado es bisiesto o no.
8. Con cantidad de segundos, mostrar por pantalla a cuántas horas, minutos y segundos corresponden.
9. Crear una calculadora que te devuelva las 4 operaciones básicas con dos números.
10. Lo mismo que la pregunta anterior, pero con tres números.

// Programación I - Unidad 01

REALIZADO POR *MIGUEL SILVA C.*

miguel.silva@inspt.utn.edu.ar