

Programación I

Docentes

❑ Ariel Sebastian Tapia – Profesor
atapia@docentes.frgp.utn.edu.ar

❑ Walter Pintos - Profesor
wpintos@docentes.frgp.utn.edu.ar

❑ Juan Manuel Fernandez – Ayudante de TPs
jmfernandez@red.frgp.utn.edu.ar

Contenidos

- Tipos de datos
- Variables y constantes
- Operadores
- Estructura de secuencia
- Estructura de decisión: Simple, doble y múltiple
- Estructura de repetición: exacta e inexacta. Ciclos combinados
- Funciones. Tipos de parámetros
- Vectores
- Proyectos de software

Evaluaciones

- Dos exámenes de diagramación
- Un trabajo práctico de desarrollo grupal (hasta cuatro personas) con defensa con preguntas y/o modificaciones individuales y/o realización de ejercicio individual en código.
- En el **Contrato Didáctico** pueden encontrarse más detalles.
- En el **Cronograma** están las fechas de las evaluaciones y los recuperatorios.

Software

Code::Blocks Versión 20.03 (Windows/Linux)

Alternativas:

Visual Studio Code Con extensión C ++ (Multiplataforma)

Dev C++ (Windows)

Xcode (Mac)

Visual Studio Community (Windows)

Software

PSeInt para diagramación (Windows/Linux/Mac)

Otros:

GitHub para control de versiones y la colaboración en proyectos de software)

Aula virtual

- Foros: Avisos, Dudas generales, Cafetería
- Contrato didáctico
- Cronograma
- Guía de actividades
- Repositorio de archivos
- Videotutoriales

Aula virtual



<https://frgp.cvg.utn.edu.ar/>

Usuario: DNI.frgp

Contraseña: DNI

¿Qué es programar?

¿Es lo mismo programar que codificar?

Proceso de desarrollo de un programa

1 Análisis del problema

Se analizan las características del problema. Se determinan los datos clave de entrada y salida. Se analizan requisitos y restricciones.

2 Diseño de la solución

Se determina en detalle las estructuras de datos y de programación que se utilizarán. Se hace un esquema general que resuelva claramente el problema.

3 Codificación de la solución en un lenguaje de programación

Se eligen las tecnologías más apropiadas para desarrollar la solución y se procede a codificarla.

4 Compilación

Dependiendo la tecnología, es necesario realizar un proceso de compilación para transformar el código en un programa ejecutable.

¿No compila?
Volver a (3)

5 Testing

Se ejecuta el programa creado y se comprueba su correcto funcionamiento.

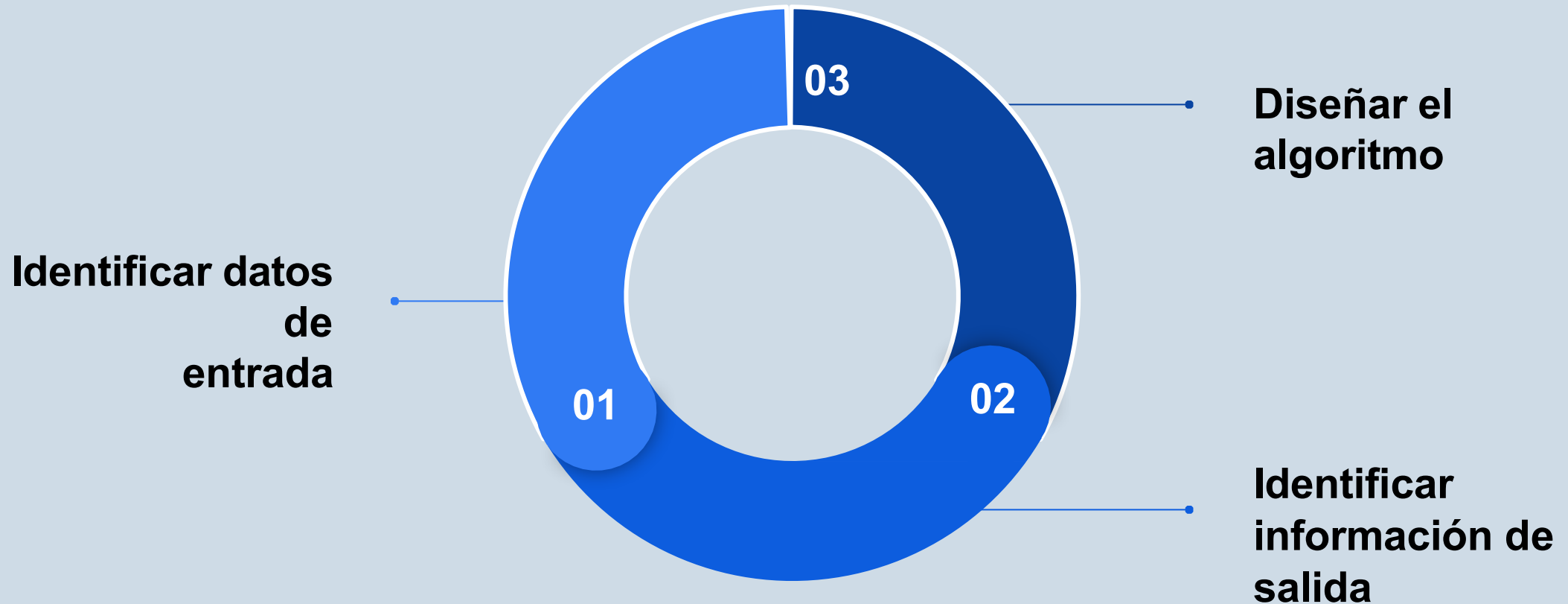
¿No funciona correctamente?
Volver a (2) o (3)

6 Instalación y mantenimiento

De ser necesario, se realiza la instalación del programa, su puesta a punto y posterior mantenimiento de solución de bugs y mejoras.

¿No resuelve el problema?
Volver a (1)

Proceso de análisis de un problema



Proceso de resolución del problema

Datos de entrada

Determinar cuántos y cuáles son los datos de entrada de nuestro programa.

Ponerles un nombre y determinar su tipo de datos.

Proceso

El algoritmo ideado debe poder transformar los datos de entrada en la información de salida.

Se apreciará que el algoritmo sea eficiente en su resolución. Resolviendo el problema de la mejor manera y con el menor costo de recursos posibles.

Información de salida

La información de salida debe ser clara, prolija e informar estrictamente lo necesario.

Diseño de un algoritmo

Estructuras de datos

Variables simples o complejas (vectores, matrices, etc.)



Estructuras de programación

Decisiones, estructuras de repetición, funciones, etc.



Esquema general del algoritmo

Secuencia de instrucciones no ambiguas y finitas que resuelven un problema en particular. El conjunto de instrucciones en un lenguaje determinado se llama código fuente.

Del código al programa

El proceso de transformar un **código fuente** en un **programa** que contiene las instrucciones que sean comprensibles por la computadora se llama **compilación**.



Codificación

Cada elemento que utilicemos en un **lenguaje de programación** debe estar sujeto a una estricta sintaxis. Los elementos que el lenguaje admite son:

- Variables y constantes
- Operadores
- Expresiones
- Palabras reservadas del lenguaje

Como muchos lenguajes, C y C++ son case-sensitive. Esto significa que hace diferencias entre mayúsculas y minúsculas.

Palabras reservadas del lenguaje

Palabras que el lenguaje utiliza para identificar tipos de datos, estructuras de programación, etc. Tienen un significado especial para el lenguaje y no pueden ser utilizados como **identificadores de nombre**.

Propósito	Palabras reservadas
Tipos de datos	bool, int, float, char, short, void, long, double
Elementos de programación	if, else, switch, default, break, for, while, do, return, auto, struct, class, static, virtual
Operadores	new, delete, sizeof

Variables

Representación simbólica de espacio de memoria. Es donde se almacenan los datos en procesamiento.

Una variable se identifica con un **tipo de dato** y un **identificador de nombre** (los elige el programador), y permite escribir un dato en la memoria o leer un dato de la memoria. Se puede modificar su valor las veces que sea necesario.

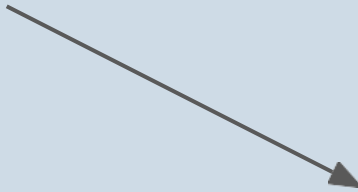
Ejemplos

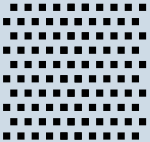
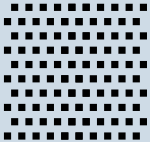
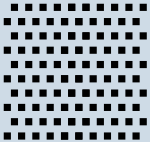
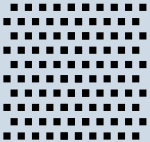
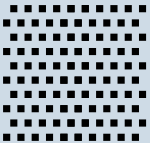
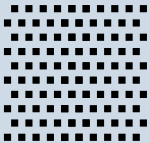
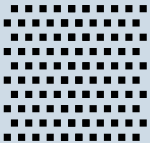
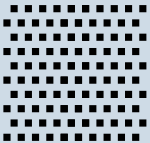
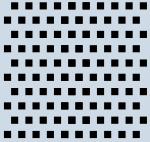
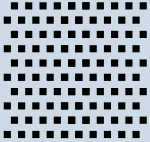
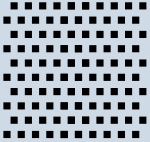
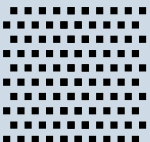
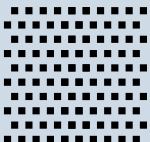
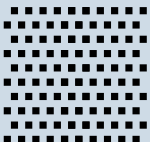
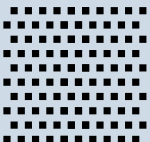
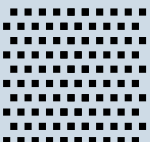
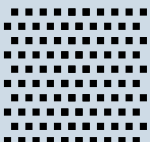
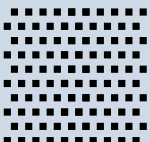
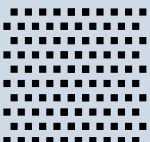
```
int edad;  
char caracter;  
float precio;  
bool aprobado;  
string nombre;
```

Variables

Ejemplo

```
int edad;  
edad = 20;
```



			
			
20 edad			
			
			

Representación simbólica de la memoria.

Al espacio de memoria identificado por **edad** se le asignó el valor 20.

Constantes

Representación simbólica de espacio de memoria. Es donde se almacenan los datos en procesamiento.

Una constante se identifica con la palabra reservada **const**, un tipo, un nombre (lo elige el programador), y un valor que no puede ser modificado durante el transcurso del programa en ejecución.

Ejemplos

```
const int EDAD_MIN = 18;  
const float IMPUESTO = 10.5;  
const char PAIS = 'A';
```

Expresiones

Conjunto de variables, constantes, números y operadores ordenados de acuerdo a las reglas sintácticas establecidas en el lenguaje de programación.

Tienen como objetivo la construcción de instrucciones para la resolución del problema (o de parte del problema) planteado.

Ejemplos

10

50 + 100

aux – 20

'B'

Operadores

Conjunto de símbolos y palabras reservadas que nos permiten hacer operaciones con expresiones.

Existen diferentes categorías de operadores:

- Asignación
- Matemáticos
- Relacionales
- Lógicos
- De dirección e indirección
- Para memoria dinámica
- Etcétera

Operadores matemáticos

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la separación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División real Cociente de la división entera
%	Resto de la división entera

Ejemplos

$((2+3)*5) + 10$

$2 + 3 * 5 + 10$

$5 / 2$

$5.0 / 2$

$5 \% 2$

Operadores matemáticos

Necesarios para realizar cálculos matemáticos. Los paréntesis tienen el mismo efecto que en la matemática en las expresiones algebraicas. Sin embargo, en programación no se utilizan corchetes ni llaves para la separación de términos.

Operador	Operación
+	Suma
-	Resta
*	Producto
/	División real Cociente de la división entera
%	Resto de la división entera

Ejemplos

$$((2+3)*5) + 10 \rightarrow 35$$

$$2 + 3 * 5 + 10 \rightarrow 27$$

$$5 / 2 \rightarrow 2$$

$$5.0 / 2 \rightarrow 2.5$$

$$5 \% 2 \rightarrow 1$$

¿División real vs División entera?

Una división puede resolverse de dos maneras. Obteniendo un resultado con expresión decimal o resultados enteros. Veamos estos ejemplos:



Tenemos seis kilos de helado para repartir entre ocho personas. ¿Cuántos kilos les corresponde a cada uno?



Tenemos diecisiete chupetines para repartir entre cinco personas. ¿Cuántos chupetines les corresponde a cada uno?

¿División real vs División entera?



Tenemos seis kilos de helado para repartir entre ocho personas. ¿Cuántos kilos les corresponde a cada uno?

6	8
60	0.7
40	0.75
0	

Corresponde 0.75 kilos por persona



Tenemos diecisiete chupetines para repartir entre cinco personas. ¿Cuántos chupetines les corresponde a cada uno?

A diagram illustrating the division $17 : 3 = 5 \text{ R}2$. It features a large light blue square divided into four quadrants by a blue L-shaped line. The top-left quadrant contains the number 17. The top-right quadrant contains the number 5. The bottom-left quadrant contains the number 2, which is circled in blue. The bottom-right quadrant contains the number 3, which is also circled in blue. Below the circled 2 is the word "Resto" (Remainder), and below the circled 3 is the word "Cociente" (Quotient).

17	5
2	3
Resto	Cociente

**Corresponden 3 chupetines por persona y
sobran 2 chupetines.**

Operadores relacionales

Son necesarios para decisiones y ciclos. Nos permiten establecer proposiciones lógicas. El resultado de una proposición lógica puede ser **verdadero** o **falso**.

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

Ejemplos

Si me pregunto `5 > b`, el resultado va a depender en función del valor de la variable `b`.

Por ejemplo:


Si `b` es igual a 6, el resultado es falso.

Si `b` es igual a 1, el resultado es verdadero.

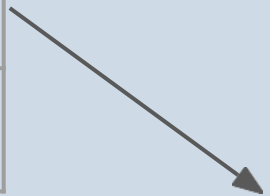
Operadores lógicos

Nos permiten combinar dos o más proposiciones lógicas

Operador		Significado
&&	and	Y lógico
	or	O lógico
!	not	Negación



A	B	A && B
verdadero	verdadero	verdadero
verdadero	falso	falso
falso	verdadero	falso
falso	falso	falso



A	B	A B
verdadero	verdadero	verdadero
verdadero	falso	verdadero
falso	verdadero	verdadero
falso	falso	falso

Operador de asignación

Nos permite asignar la expresión que se encuentra a la derecha del operador en la variable que se encuentra a la izquierda.



```
edad = 50;  
a = b;  
nombre = "Angel";  
caracter = 'X';  
precio = cant * pu;  
cont = cont + 1;
```



```
50 = edad;  
50 = 50;
```

Errores de sintaxis

edad = edad; Sintácticamente correcto pero sin sentido.

Ejercicio

Hacer un programa que permita ingresar dos números enteros por teclado. Luego calcular e informar la suma de ellos.

1

Datos de entrada: Dos números enteros.

2

Proceso: Operación matemática de suma de ambos números.

3

Información de salida: Resultado de la suma

Resolución: Instrucciones

01

Declarar las variables
necesarias

```
int num1, num2, resultado;
```

02

Ingresar el primer número
por teclado

```
cin >> num1;
```

03

Ingresar el segundo
número por teclado

```
cin >> num2;
```

04

Realizar la suma

```
resultado = num1 + num2;
```

05

Mostrar por pantalla el
resultado

```
cout << resultado;
```



Resolución: Código fuente

```
#include <iostream>
using namespace std;

int main(){
    int num1, num2, resultado;
    cin >> num1;
    cin >> num2;
    resultado = num1 + num2;
    cout << resultado;
    return 0;
}
```

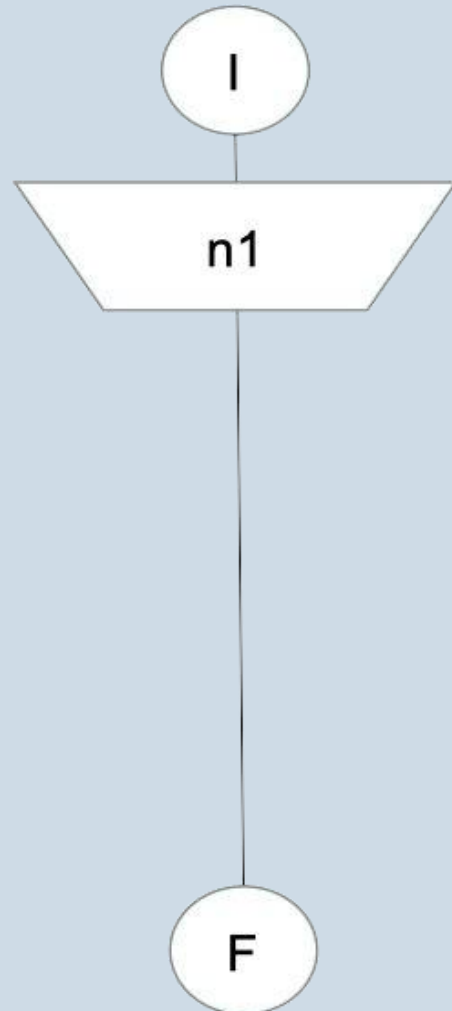


¿¡Qué es todo eso!?

No se preocupen. En breve lo codificamos en un IDE y lo mostramos paso a paso.

Resolución: Instrucciones

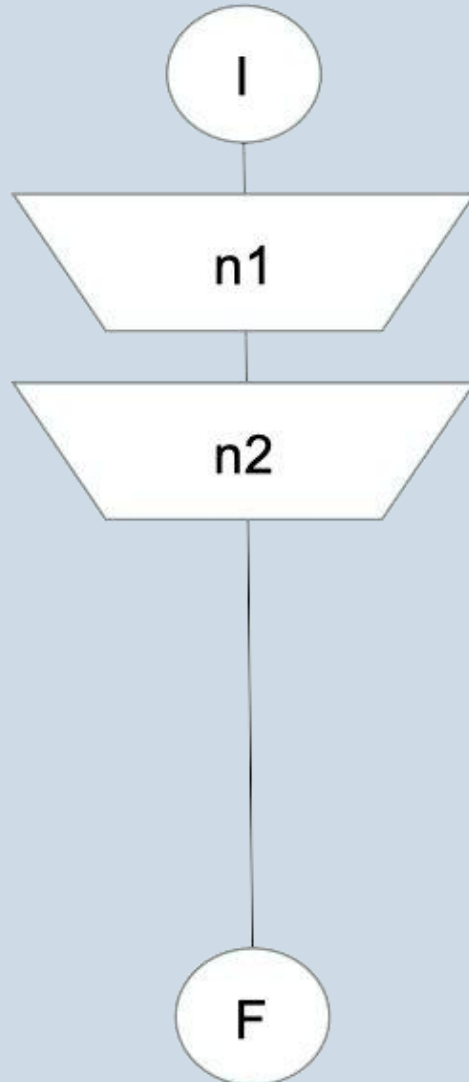
01



Ingresa el primer número

Resolución: Instrucciones

02

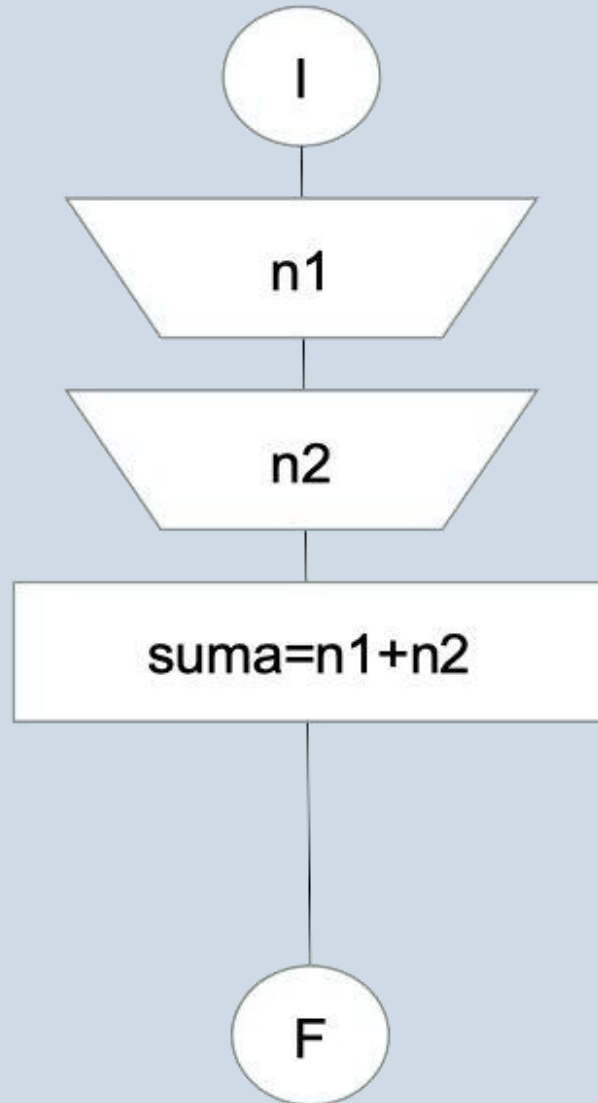


Ingresar el primer número

Ingresar el segundo número

Resolución: Instrucciones

03



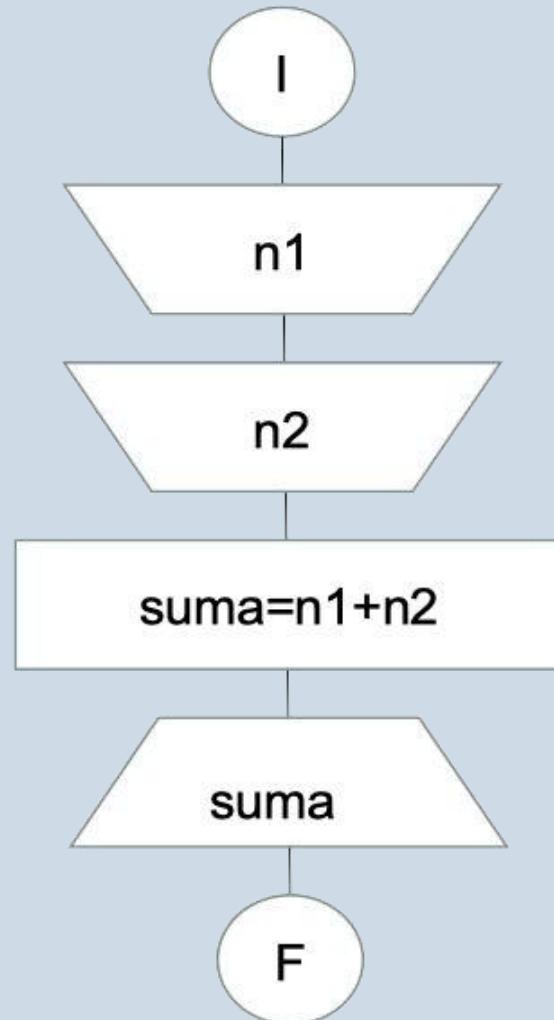
Ingresar el primer número

Ingresar el segundo número

Calcular la suma

Resolución: Instrucciones

04



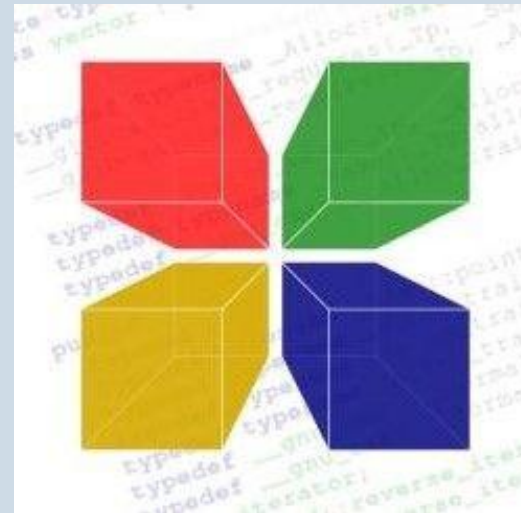
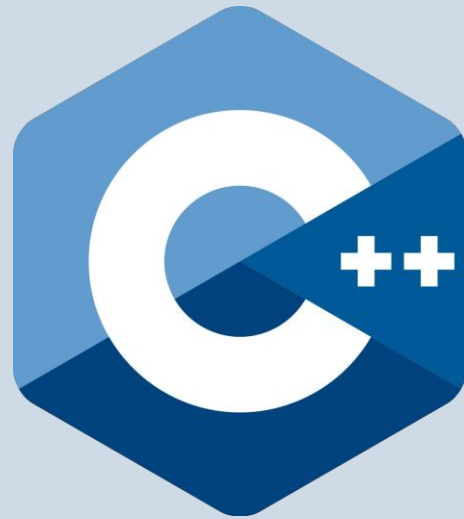
Ingresar el primer número

Ingresar el segundo número

Calcular la suma

Mostrar el resultado por pantalla

Resolución en C/C++



Más ejercicios: en la guías de ejercicios de la materia

Créditos

- Angel Ruben Simón, docente UTN FRGP
- Brian Lara Campos, docente UTN FRGP
- Logo de CodeBlocks: captura de pantalla de la aplicación
- Logo de C++: autor Jeremy Kratz
- Ícono de persona dudando: por turkkub de www.flaticon.com
- Ícono de helado: por Icongeek26 de www.flaticon.com
- Ícono de chupetín: por Freepik de www.flaticon.com