

Fundamentos de Informática

Docentes


- Daniel Kloster - Profesor
dkloster@docentes.frgp.utn.edu.ar
- Angel Simón - Jefe de trabajos prácticos
asimon@docentes.frgp.utn.edu.ar
- Brian Lara - Ayudante de primera
blara@docentes.frgp.utn.edu.ar

Contenidos

Programación

- Estructura de secuencia
- Estructura de decisión
 - Simple
 - Múltiple
- Estructura de repetición
 - Ciclo exacto
 - Ciclo inexacto
 - Ciclos combinados
- Vectores
- Funciones

Planilla de Cálculo

- Fórmulas y funciones estadísticas simples.
 - Funciones de decisión
 - Funciones de búsqueda
- 

Evaluaciones

Primer parcial

- Resolución de problemas de programación. (Individual)

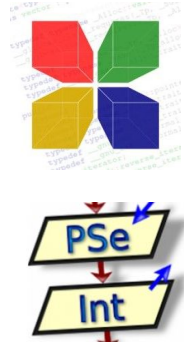
Segundo parcial

- Resolución de problemas de programación. (Individual)

Trabajo práctico

- Resolución de un caso práctico con planilla de cálculo. (Grupal)
- Defensa de una situación problemática con la planilla del caso práctico. (Individual)

Software



Programación

Codeblocks
PSeInt

Planilla de cálculo

Microsoft Excel
LibreOffice Calc
Google Sheets

Puede usarse cualquier software y lenguaje de programación.

Definiciones básicas

- El objetivo de la materia es la **introducción a los conocimientos básicos de la programación**.
- Para ello es necesario hacer programas utilizando un **lenguaje de programación**.
- Un lenguaje de programación proporciona un conjunto de instrucciones, mecanismos y reglas mediante las cuales construir el código capaz de resolver un problema.
- El conjunto de instrucciones que componen a un programa se denomina **código fuente**. Programar escribiendo código se denomina **codificación** y realizando diagramas se llama **diagramación**.

Definiciones básicas

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. Algunos de los elementos que el lenguaje admite son:

- **Variables y constantes**

- **Operadores**

- **Expresiones**

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

Diagramación

Equivalente a las instrucciones que podemos realizar en la codificación.
En la diagramación tenemos componentes que las representan.

Ingreso de datos por teclado



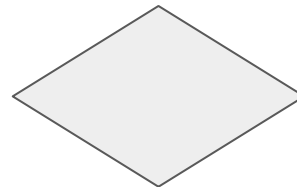
Salida de datos por pantalla



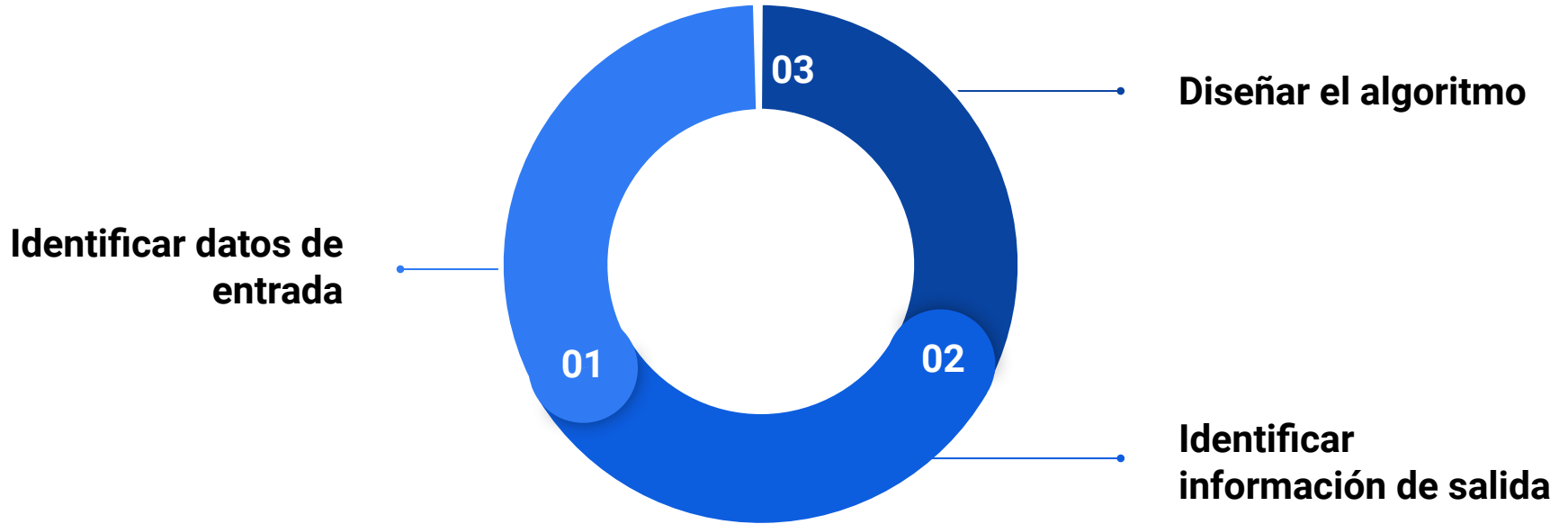
Operación



Decisión simple



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.

Variables, constantes, expresiones y operadores

Variables

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

Una variable se identifica con un **tipo de dato (sólo en codificación)** 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.

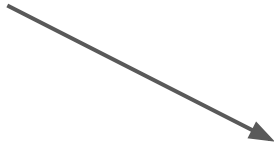
Ejemplos

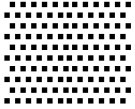
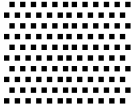
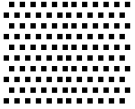
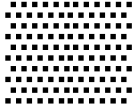
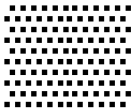
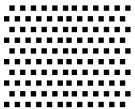
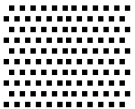
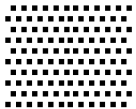
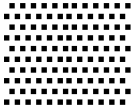
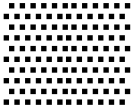
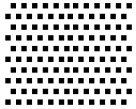

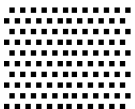
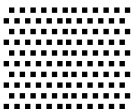


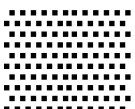
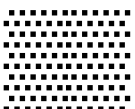

```
int edad;  
char caracter;  
float precio;  
bool aprobado;  
string nombre;
```

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 (sólo en codificación), 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**

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

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

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

$$5 / 2 \rightarrow 2$$

$$5.0 / 2 \rightarrow 2.5$$

$$5 \% 2 \rightarrow 1$$

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.

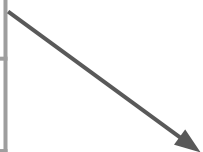
Operadores lógicos

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;
```

Error de sintaxis

```
edad = edad;
```

Sintácticamente correcto pero sin sentido.

Programa de ejemplo

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;
}
```



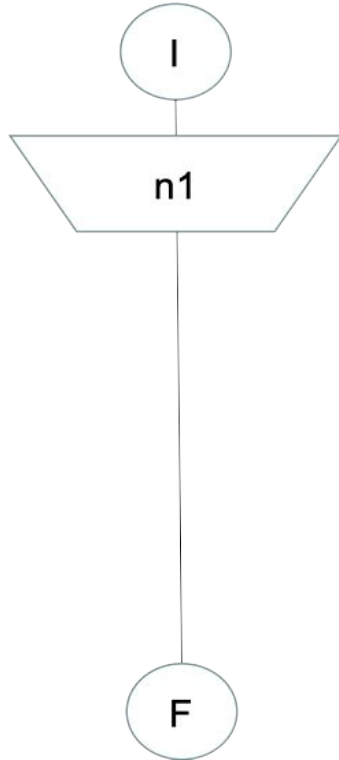
¿Qué es todo eso!?

No se preocupen. En breve lo codificamos y diagramamos paso a paso.



Resolución: Instrucciones

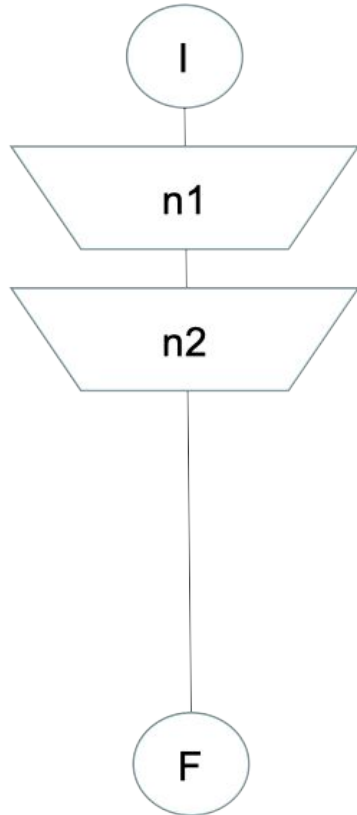
01



Ingresar el primer número

Resolución: Instrucciones

02

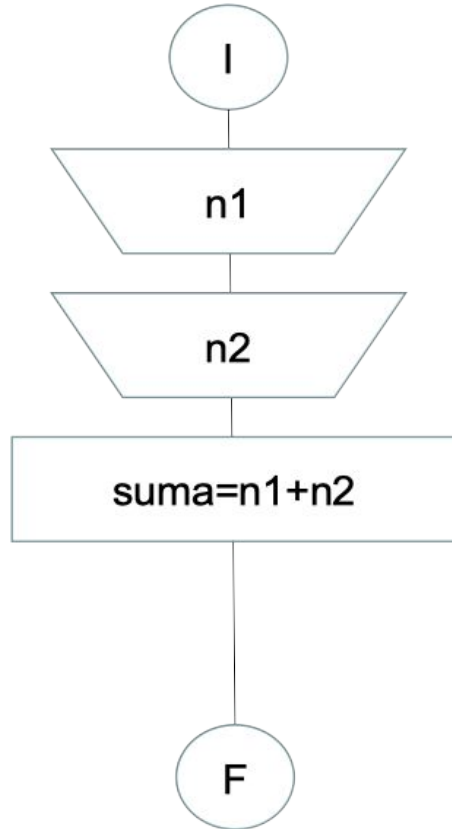


Ingresar el primer número

Ingresar el segundo número

Resolución: Instrucciones

03



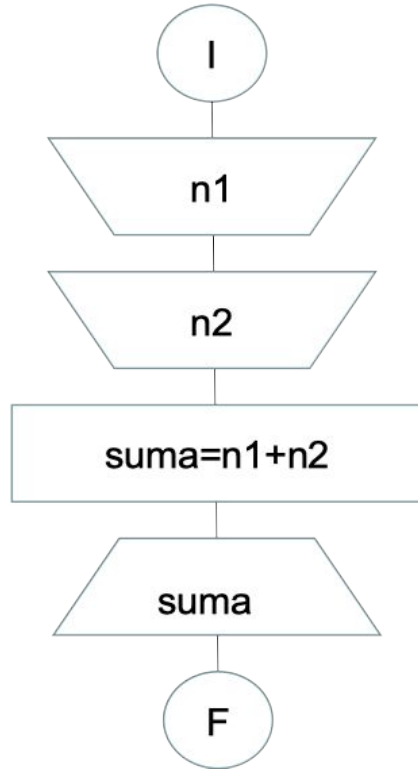
Ingresar el primer número

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

Créditos

- Logo de CodeBlocks: captura de pantalla de la aplicación
- Logo de PSeInt: captura de pantalla de la aplicación
- Logo de Excel: Derechos reservados Microsoft
- Logo de LibreOffice: Derechos reservados Libreoffice Foundation
- Logo de Google Calc: Derechos reservados Google
- Logo de C++: autor Jeremy Kratz
- Ícono de Código fuente y Compilación: por Freepik de www.flaticon.com
- Ícono de Programa binario: por juicy_fish de www.flaticon.com