


Laboratorio de Computación I

Docentes:

Angel Simón, Brian Lara, Soledad Arena

Contenidos

- Estructura de secuencia
 - Estructura de decisión
 - Simple
 - Múltiple
 - Estructura de repetición
 - Ciclo exacto
 - Ciclo inexacto
 - Ciclos combinados
 - Vectores y matrices
 - Cadenas de caracteres
 - Punteros
 - Funciones
 - Proyectos de software
- 

Evaluaciones



Primer parcial

- Resolución de una actividad online individual (teórico/práctica).



Segundo parcial

- Trabajo práctico de desarrollo grupal (hasta tres personas)
- Defensa grupal con preguntas y/o modificaciones individuales

Software



Codeblocks 20.03



Alternativas:

Dev C++

Visual Studio

Visual Studio Code + plugin C++

Xcode

GitHub



Elementos del lenguaje

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.

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 y nombre (lo 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.

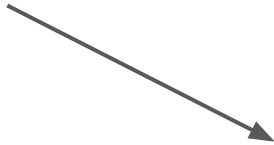
Ejemplos

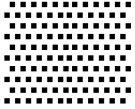
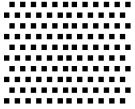
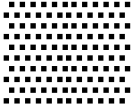
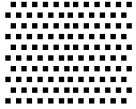
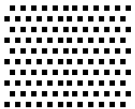
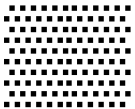
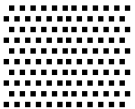
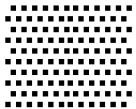
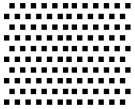
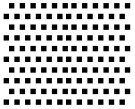
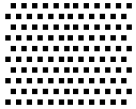

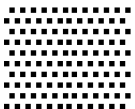
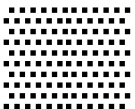


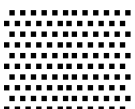
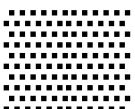

```
int edad;  
char caracter;  
float precio;  
bool aprobado;
```

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 | ■ | De dirección e indirección |
| ■ | Matemáticos | ■ | Para memoria dinámica |
| ■ | Relacionales | ■ | Etcétera |
| ■ | Lógicos | | |

Operadores matemáticos

Necesarios para realizar cálculos matemáticos y expresiones algebraicas. Los paréntesis tienen el mismo efecto que en la matemática para la separación de términos. Sin embargo, 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

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

$$2+3*5+10$$

$$5 / 2$$

$$5.0 / 2$$

Operadores matemáticos

Necesarios para realizar cálculos matemáticos y expresiones algebraicas. Los paréntesis tienen el mismo efecto que en la matemática para la separación de términos. Sin embargo, 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

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

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

$$5 / 2 \rightarrow 2$$

$$5.0 / 2 \rightarrow 2.5$$

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.

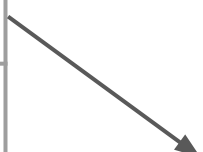
Operadores lógicos

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.



```
var = 50;  
a = b;  
character = 'X';  
precio = cant * pu;  
cont = cont + 1;
```



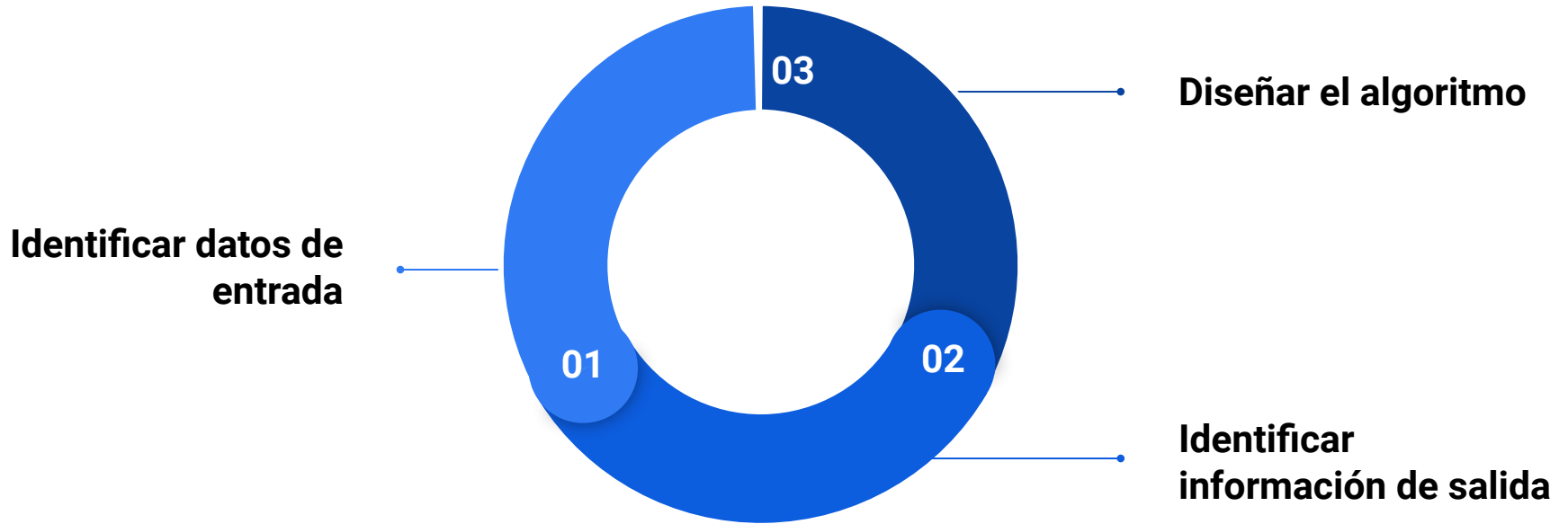
```
50 = var;  
50 = 50;
```

Error de sintaxis

```
var = var;
```

Sintácticamente correcto pero sin sentido.

Proceso de resolución de problemas



Proceso de resolución de problemas

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.

Ejercicio

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

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

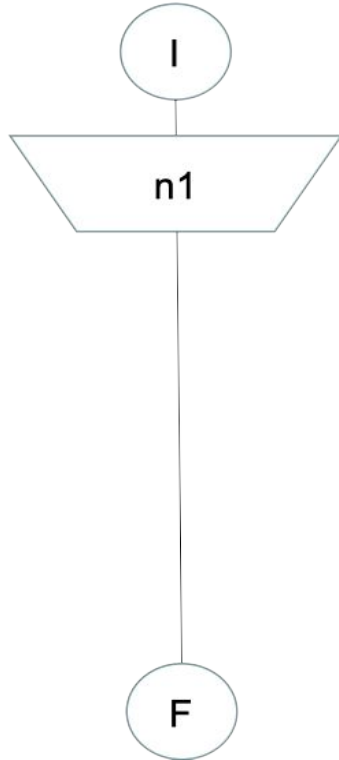
Mostrar por pantalla el resultado

```
cout << resultado;
```



Resolución: Instrucciones

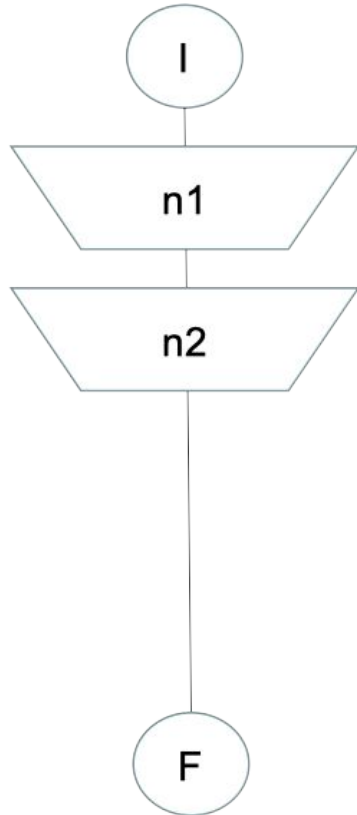
01



Ingresar el primer número

Resolución: Instrucciones

02

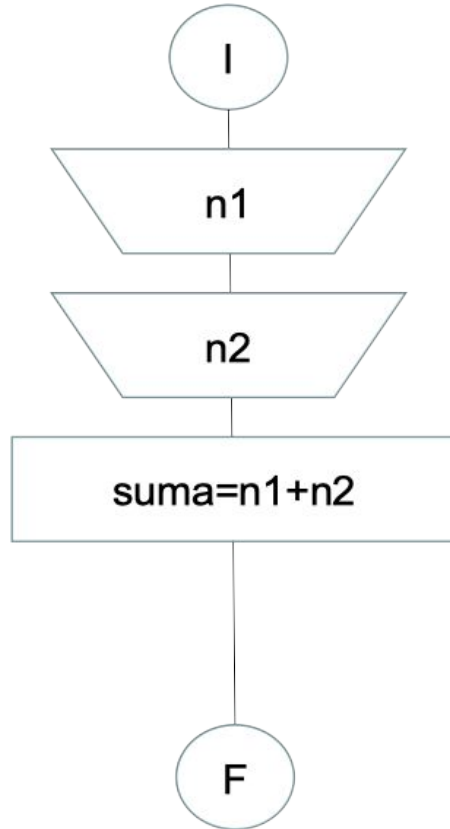


Ingresar el primer número

Ingresar el segundo número

Resolución: Instrucciones

03



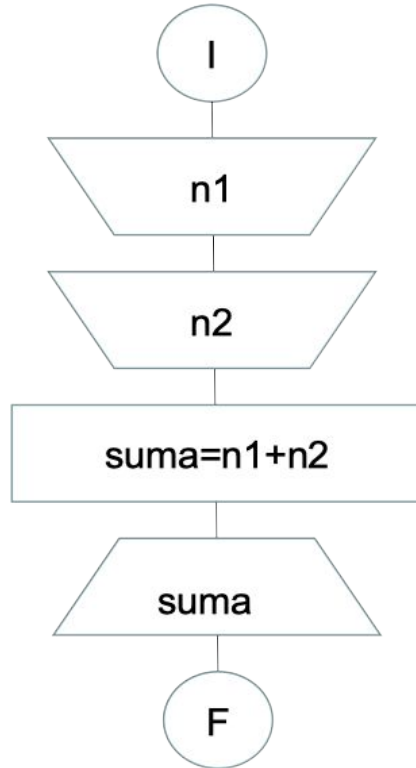
Ingresar el primer número

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

Resolución en C/C++



Más ejercicios: bit.ly/LAB1-25