



Introducción a la programación



Índice



- ¿Qué es programar?
- Lenguaje de alto/ bajo nivel
- Paradigmas de programación
- Concepto de variable
- Algoritmos
- Traza
- Realizar un algoritmo
- Del código fuente al ejecutable
- ¿Por qué usar c?
- Partes de un fichero c



¿Qué es programar?

¿Qué es programar?

- ▀ Traducir de una idea a un lenguaje de programación para resolver un problema.

Problema ->



Método de resolución ->

Lenguaje de programación ->



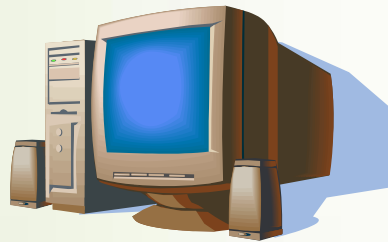
Solución



Lenguaje de alto/bajo nivel

Lenguaje de alto/bajo nivel

- Alto nivel: Lenguaje humano.
- Bajo nivel: Lenguaje maquina.



Lenguaje
máquina



C/ C++



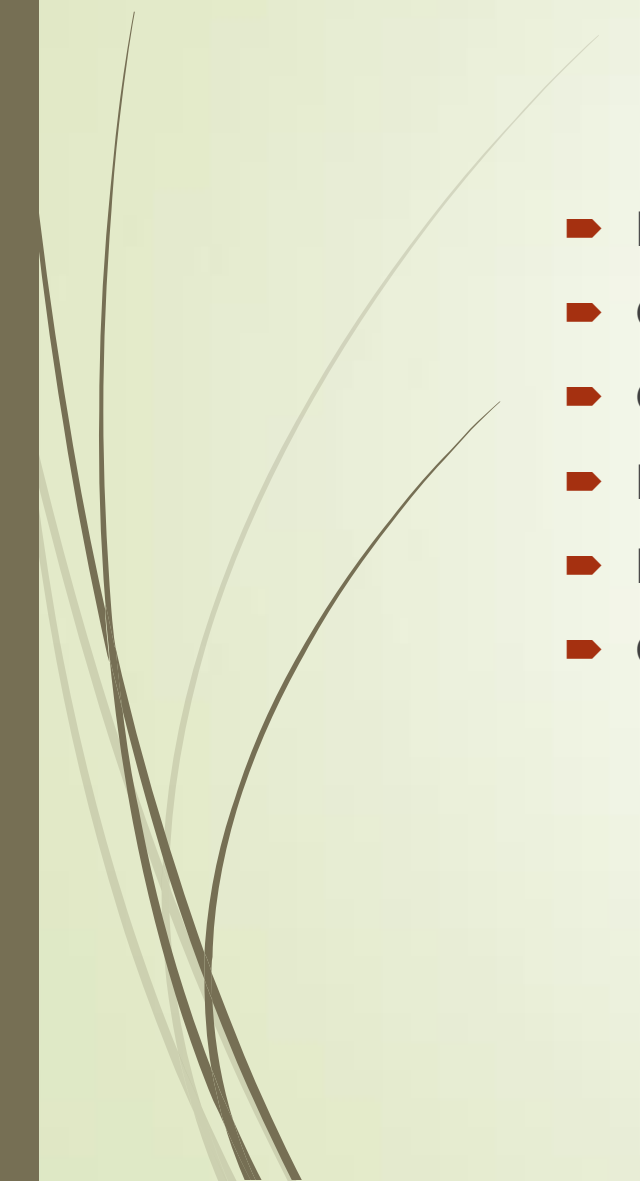
Java, PHP,
Python



Paradigmas de programación



Paradigmas de programación

- Estructurada.
 - Orientada a objetos.
 - Orientada a aspectos.
 - Paralela.
 - Procesos e hilos
 - Cíclica.
- 



Concepto de variable



Concepto de variable

- 1. Que varía o puede variar.
- 2. Que está sujeto a cambios frecuentes o probables.
- En **programación**, las **variables** son espacios reservados en la memoria que, como su nombre indica, pueden cambiar de contenido a lo largo de la ejecución de un programa. Una **variable** corresponde a un área reservada en la memoria principal del ordenador.
- Tipos de variables.
- Operaciones con variables.
- Ámbito de las variables.



Concepto de variable

Tipos de variables

- Tipo de dato a almacenar por una variable.
- Diferentes lenguajes usan diferentes tipos de variables.
- Tipos genéricos: Numérico, Texto, Lógico.
- Tipos más usados:
 - char -> almacena caracteres,
 - Int -> almacena números enteros.
 - float, double -> almacenan números reales.
- Prefijos: unsigned, short, long

Concepto de variable

Tipos de variables

Tipo	Tamaño	Rango
unsigned char	1B	[0,255]
char	1B	[-127,127]
short int	2B	[-32768,32768]
unsigned int	4B	[-2'.147.483.648,2'.147.483.648]
unsigned long	4B	[0, 4'.294.967.295]
enum	2B	[2'.147.483.648, 2'.147.483.648]
float	4B	[3.4x10-38,3.4x10+38]
double	8B	[1.7x10-308,1.7x10+308]
long double	9B	[3.4x10-4932,3.4x10+4932]
bool	1B	true, false



Concepto de variable

Tipos de variables

- “Otros tipos” de variables más complejos son según su paradigma de programación:
 - Vectores (arrays o listas).
 - Matrices (vectores multidimensionales).
 - Objetos. (POO)
 - Estructuras.
 - Pilas.
 - Árboles.
 - Aspectos (POA)

Concepto de variable


Operaciones con variables

- Operaciones aritméticas (+ - * / % += -= ++ --)
- Operaciones de comparación (> >= < <= !=)
- Operaciones lógicas o booleanas (AND, OR, NOT).
- Operaciones de entrada/salida (leer(x), escribir(x))
- Asignación o comparación (a=1 a==1)
- Operaciones con memoria del sistema. (punteros * &, constructores, destructores, liberar espacio de memoria reservado)
- Operaciones de cadenas de texto (concatenar, contar letras, buscar subcadenas ...)



Concepto de variable

Ámbito de las variables

- Tipado fuerte o débil, dinámico o estático
 - Global.
 - Local.
 - Constante.
- 



Algoritmo



Algoritmo



- **Programa:** Un programa es una serie de instrucciones ordenadas, codificadas en lenguaje de programación que expresa un **algoritmo** y que puede ser ejecutado.
- **Algoritmo:** Conjunto de instrucciones que realizadas en orden conducen a obtener la solución de un problema.
 - Características de un Algoritmo.
 - Partes de un Algoritmo.
 - Formas de representar un algoritmo.



Algoritmo: Características de un Algoritmo

➤ **Todo algoritmo debe tener las siguientes características:**

1. Debe ser preciso, porque cada uno de sus pasos debe indicar de manera precisa e inequívoca que se debe hacer.
2. Debe ser finito, porque un algoritmo debe tener un número limitado de pasos.
3. Debe ser definido, porque debe producir los mismos resultados para las mismas condiciones de entrada.
4. Puede tener cero o más elementos de entrada.
5. Debe producir un resultado. Los datos de salida serán los resultados de efectuar las instrucciones.



Algoritmo: Partes de un Algoritmo

- **Todo algoritmo debe tener las siguientes partes:**
 - **Entrada de datos**, son los datos que el algoritmo necesita para ser ejecutado.
 - **Proceso**, es la secuencia de pasos para ejecutar el algoritmo.
 - **Salida de resultados**, son los datos obtenidos después de la ejecución del algoritmo.



Algoritmo: Formas de representar un Algoritmo

- Para la representación de un algoritmo, antes de ser convertido a lenguaje de programación, se utilizan algunos métodos de representación escrita, gráfica o matemática. Los métodos más conocidos son:
 - Diagramas de flujo.
 - Pseudocódigo

Algoritmo: Formas de representar un Algoritmo

Datos de entrada: dos números enteros A y B

Datos de salida: el MCD

1. Si B es mayor que A
intercambiar los valores.

2. Calcular el resto de dividir A
por B y poner ese valor en R

3. Si R es igual a 0, el MCD es B, y
FIN

4. Ponemos en A el valor
contenido en B, y
ponemos en B el valor contenido
en R

5. Volver a 2.

Algoritmo Positivo Negativo

Variables

x: Entero

Inicio

Leer (x)

Si $(x < 0)$ entonces

Escribir ('Numero negativo')

sino

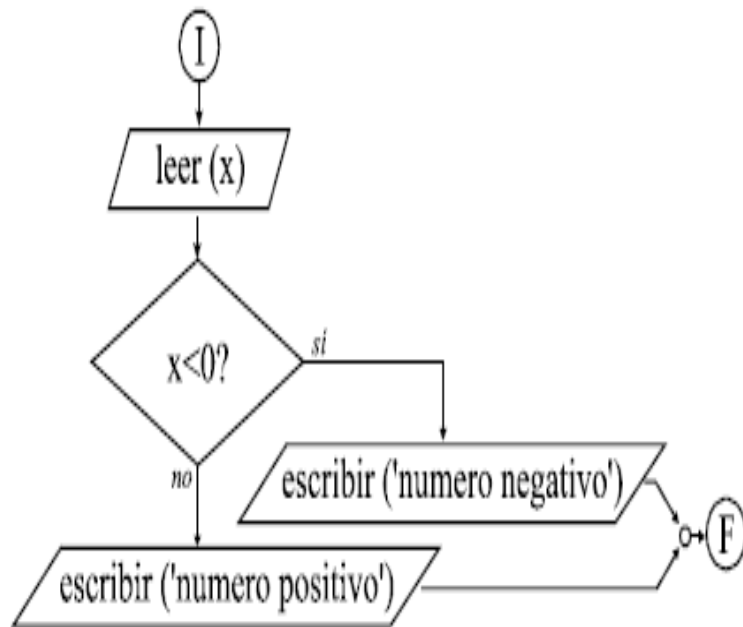
Escribir ('Numero positivo')

Fin_si

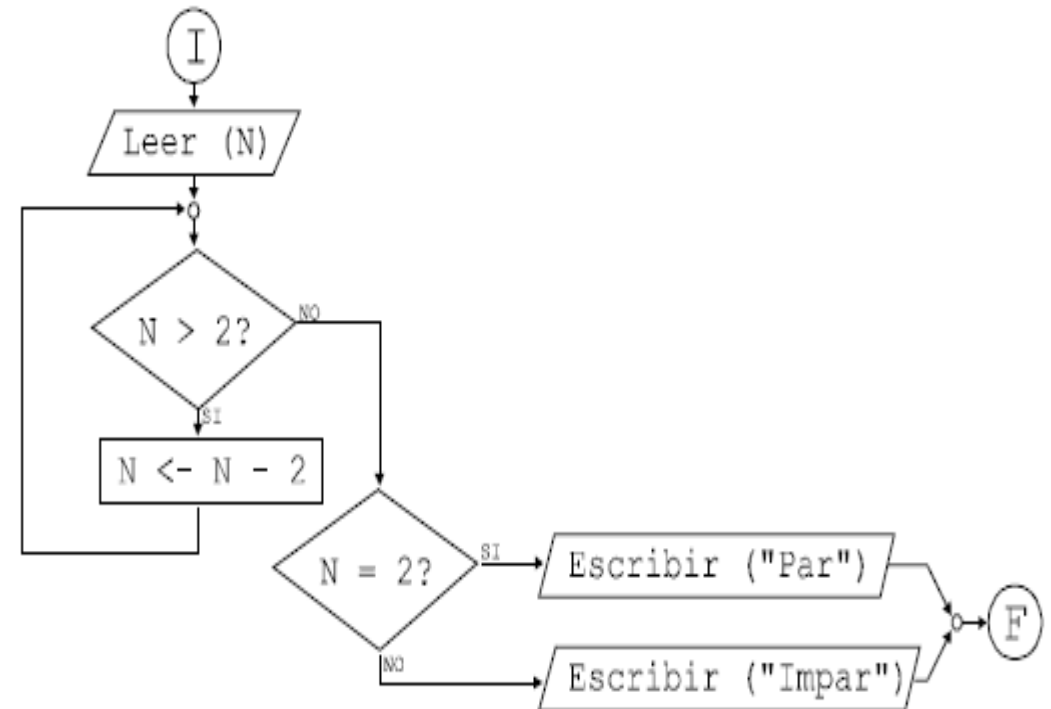
Fin

Algoritmo: Formas de representar un Algoritmo

Ejemplo: Dado un número, decir si es positivo o negativo.



Ejemplo: Algoritmo para decir si un número es par (organigrama)





Traza

Traza

- La **traza** de un algoritmo (o programa) indica la secuencia de acciones (instrucciones) de su ejecución, así como, el valor de las variables del algoritmo (o programa) después de cada acción (instrucción).

Algoritmo suma

Inicio

Variable: Entera a,b,c

Escribir "Indique el primer sumando"

Leer a

Escribir "Indique el segundo sumando"

Leer b

$c = a + b$

Escribir "El resultado es"; c

INICIO con valores 5 y 4

A	=	5	5	5	5
B	=		4	4	4
C	=			5+4	9



Traza

Pseudocódigo: Método 2: *Criba de Eratóstenes*

- .1. Poner todos los números entre 2 y MAX uno detrás de otro.
- .2. Si hay números sin tachar, el primero de ellos es primo
- .3. Tachar de la lista todos los múltiplos del primer número
- .4. Borrar el primer número
- .5. Borrar los tachados y pasar a .2.



Realizar un algoritmo

Realizar un algoritmo

Entradas y salidas

- Escribir
- Leer



```
graph TD; A[ ] --> B[Escribir ("Hola")]; B --> C[ ]
```

Escribir ("Hola")

The diagram illustrates a single step in an algorithm. It features a central red parallelogram containing the text "Escribir ("Hola")". A red arrow points down into the top of this parallelogram, and another red arrow points down from its bottom, indicating the flow of the process.

Realizar un algoritmo

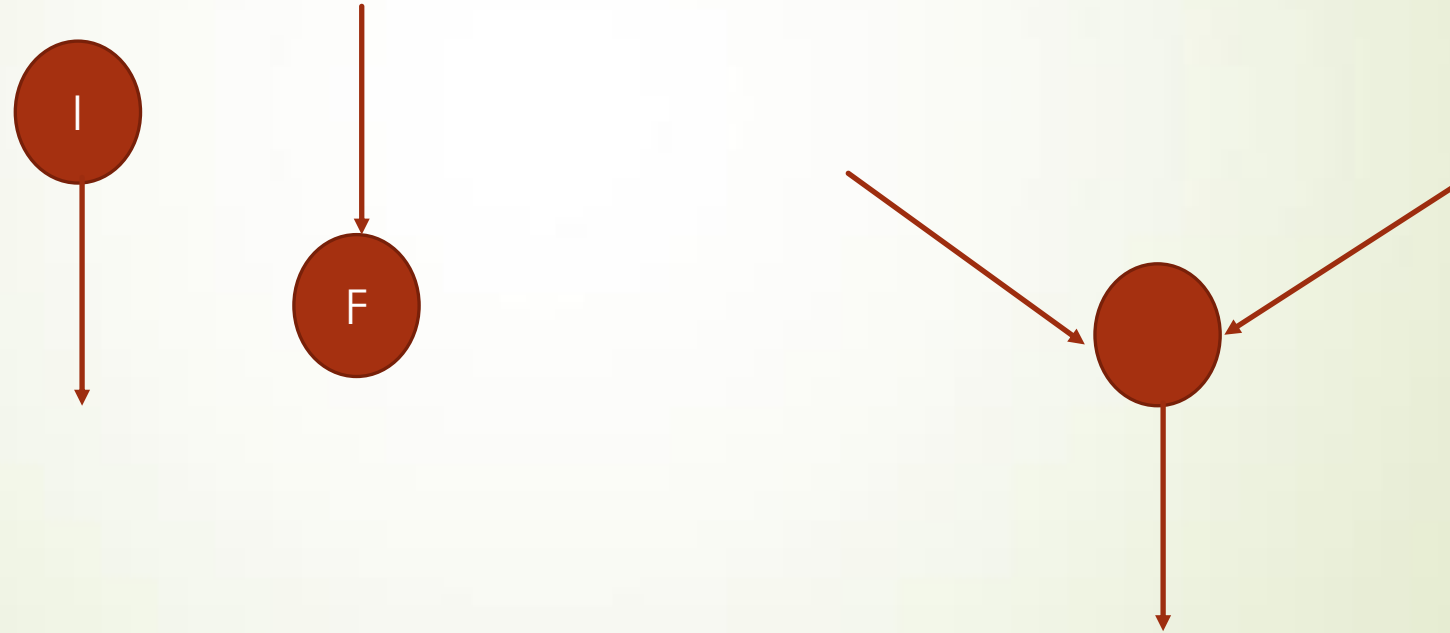
Instrucciones

- Operaciones aritméticas
- Acciones de búsqueda u ordenación.
- Operaciones Max, min, media...



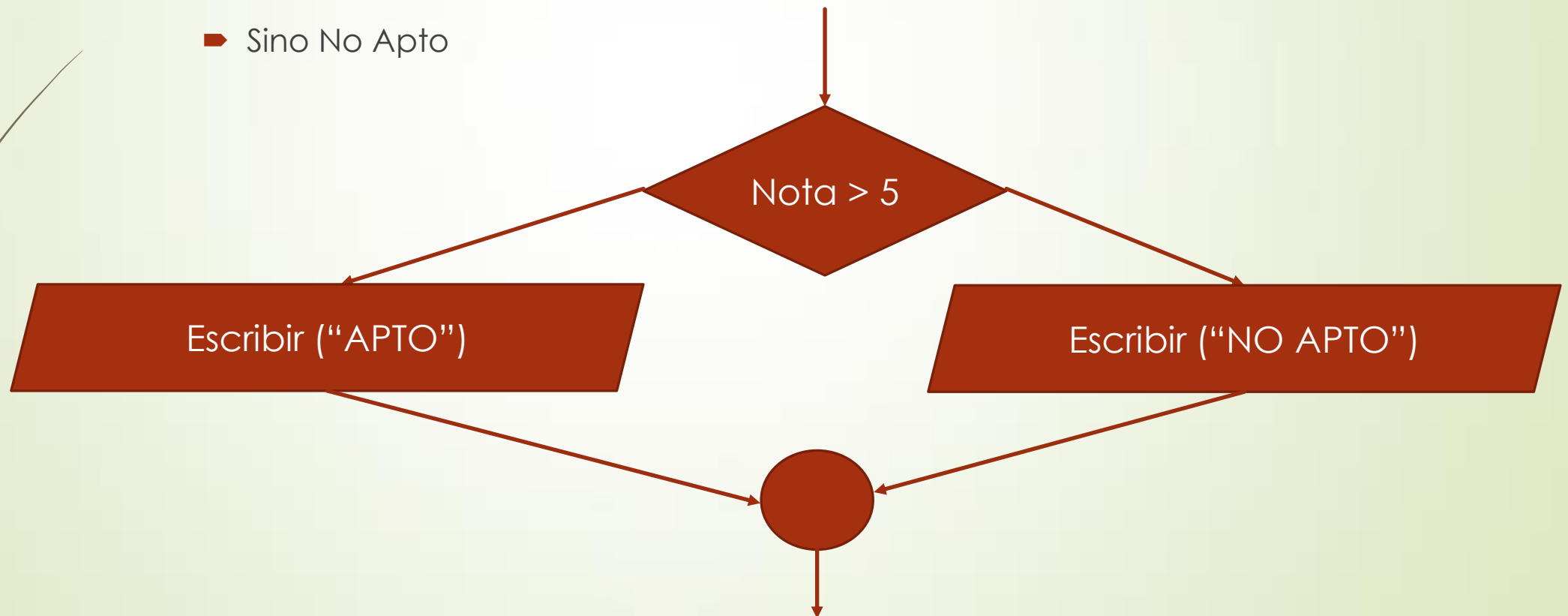
Realizar un algoritmo Inicio y Final

- El inicio, el final, los puntos de reunión de flujo.



Realizar un algoritmo Condiciones

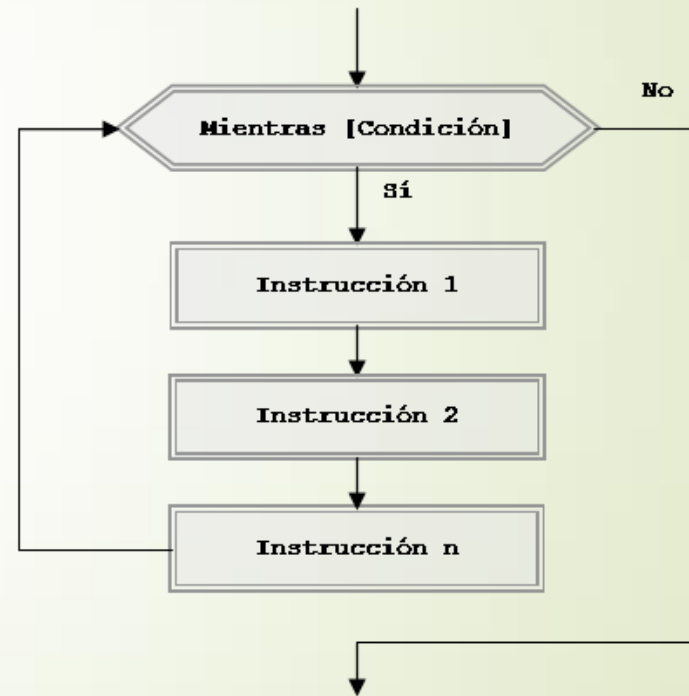
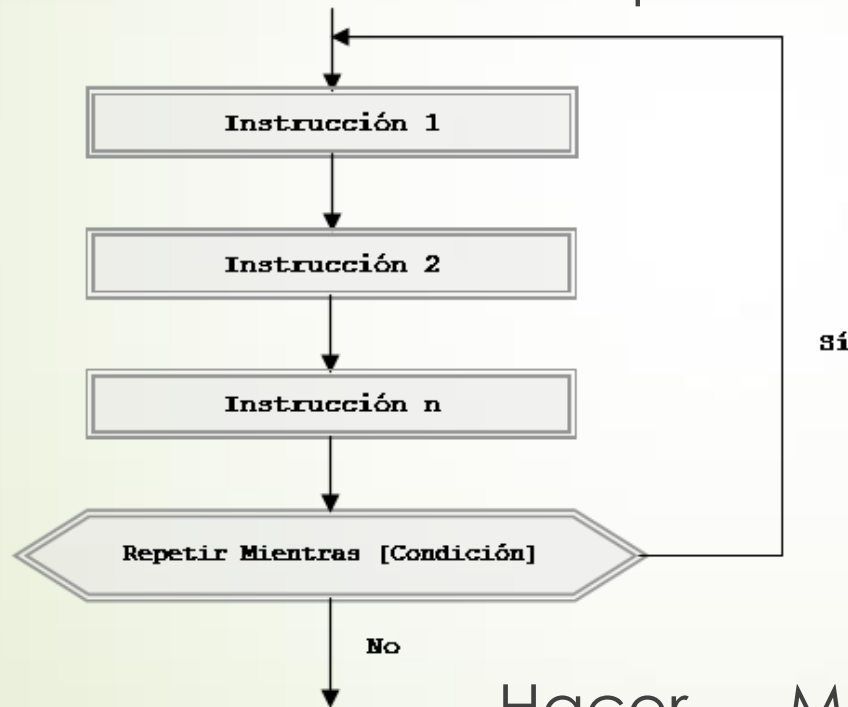
- Al evaluar una expresión condicional resultado afirmativo o negativo.
 - Si $\text{nota} > 5$ Apto
 - Sino No Apto




Realizar un algoritmo estructuras iterativas

(repetición, o de bucle)

Mientras se cumple
expresión ir a paso N



Hacer ... Mientras,
Mientras, Repetir



Realizar un algoritmo estructuras iterativas

(repetición, o de bucle)

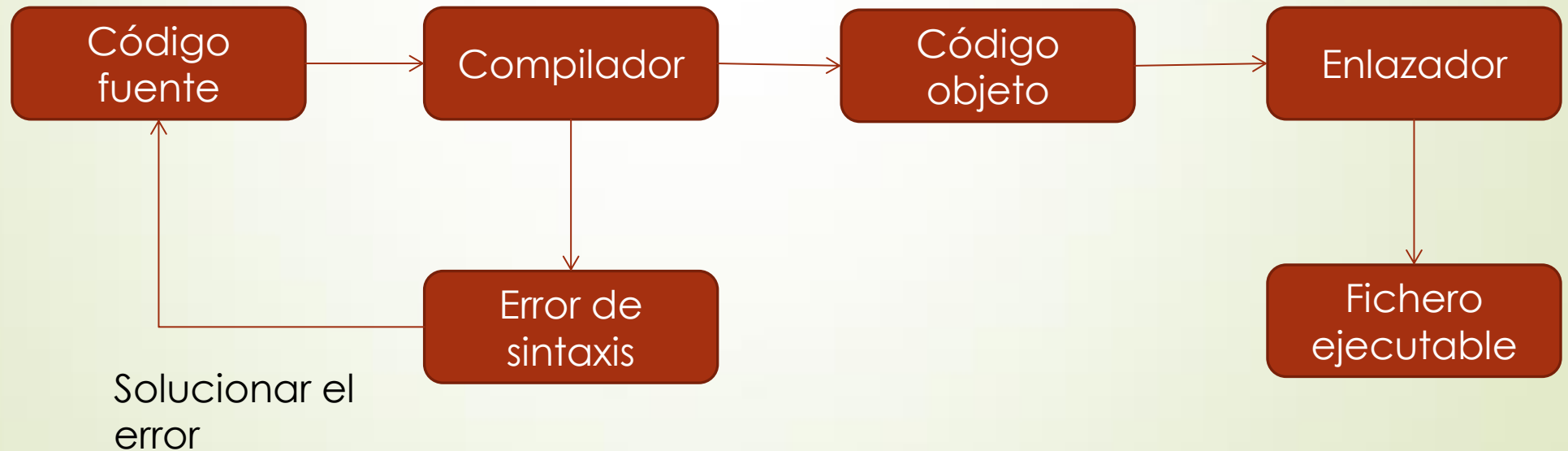
- Dos estrategias clásicas de estructuras iterativas:
 - Repeticiones en función de una condición:
 - Mientras se cumpla una condición se repiten las instrucciones.
 - Repeticiones cierto numero de veces:
 - Las instrucciones se realizarán un cierto numero de veces.
 - La condición de salida de la repetición es que una variable (llamada contador), llegue al valor deseado, mediante operaciones de suma o resta.



Del código fuente al ejecutable

Del código fuente al ejecutable

- Los programas son escritos en algún lenguaje de programación, creando un **código fuente** y luego son traducidos mediante un proceso de **compilación** o **interpretación** hacia el sistema, que desembocará en un fichero ejecutable o proceso de ejecución.



Del código fuente al ejecutable

```
main.c  main.cpp [X]
1  #include <iostream>
2  #define dolar 12.8000
3  #define yen 0.1659
4  #define euro 18.2108
5
6  using namespace std;
7
8  void cabecera();
9  void conversion();
10
11 int main()
12 {
13     cabecera();
14     conversion();
15
16     return 0;
17 }
18
19 void cabecera () {
20     cout << "Cambio de divisas" << endl;
21     cout << "-----" << endl;
22     cout << "Pesos\tDolares\tYenes\tEuros" << endl;
23 }
24
25 void conversion() {
26     int i;
27     double Dolar, Yen, Euro;
28     for (i = 1; i <= 20; i++) {
29         Dolar = i * dolar;
30         Yen = i * yen;
31         Euro = i * euro;
32         cout << i << "\t" << Dolar << "\t" << Yen << "\t" << Euro << endl;
33     }
34 }
35
```



Del código fuente al ejecutable

- Durante el proceso de compilación el IDE (Entorno de Desarrollo Integrado) particular del lenguaje elegido, alcanzará el código fuente en busca de errores: léxicos, sintácticos.
- Error léxico: Palabras propias del lenguaje mal escritas.
- Error sintáctico: Instrucciones mal escritas.
- Error semántico o lógico: Imposible de detectar por el compilador, el programa se ejecuta pero no realiza lo que el programador pretendía.



Del código fuente al ejecutable

- Una vez compilado el código fuente se genera un archivo llamado **archivo objeto** o **programa objeto** que es luego enlazado mediante el **enlazador**, para generar el archivo ejecutable.



Del código fuente al ejecutable

► El **depurador** es una herramienta que nos proporciona una serie de utilidades para identificar los errores de nuestros programas de una manera más eficaz, mediante una serie de pasos:

1. Colocar un punto de parada.
2. Ejecutar el programa en modo depuración.
3. Ejecutar el programa paso a paso.
4. Inspeccionar variables.



¿Por qué usar c?



¿Por qué usar c?

- Es un lenguaje potente
 - Se pueden realizar multitud de programas con este lenguaje.
- Es un lenguaje eficiente
 - Es un lenguaje que crea aplicaciones con carga rápida
- Es un lenguaje multiplataforma.
 - Podemos programar en c y ejecutar los programas tanto en Windows, Linux o Mac entre otros
- Es la base de otros lenguajes

¿Por qué usar c?

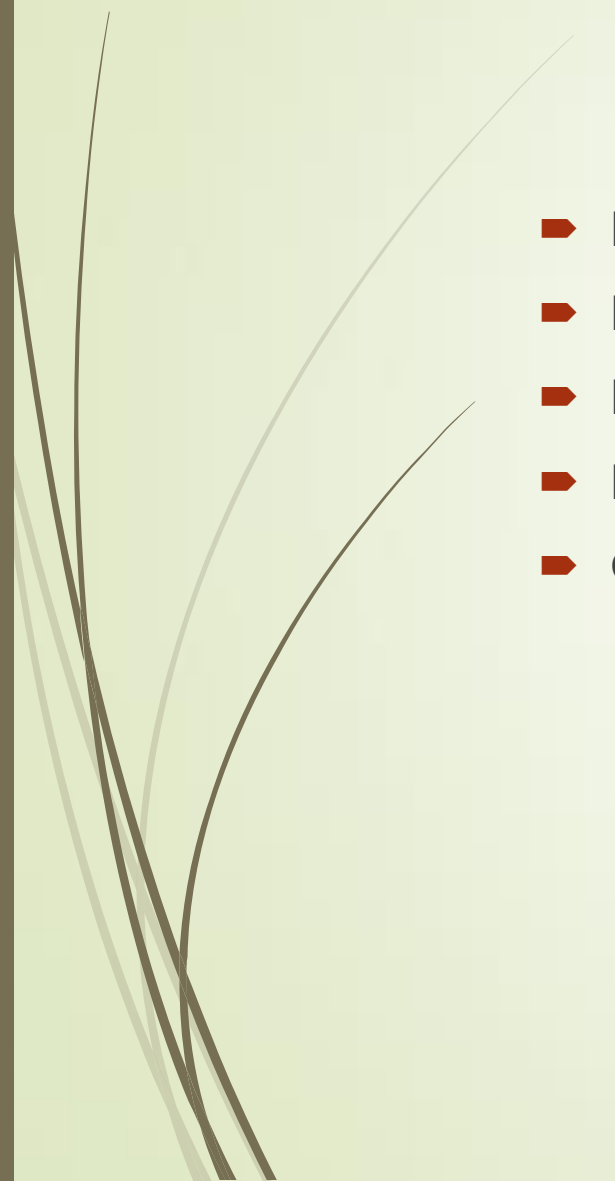




Partes de un fichero c/c++




Partes de un fichero c/c++

- Librerías.
 - Declaración, definición y llamada.
 - Funciones.
 - Declaración y definición de la función principal
 - Comentarios.
- 



Partes de un fichero c/c++ Librerías

- Es donde residen las funciones u operaciones propias del lenguaje de programación.
- Para utilizarlas necesitamos la instrucción **#include** y las vamos a situar en la parte superior del código fuente de nuestro programa
- En función de (la o las) librerías que llamamos podremos usar o no las funciones.



Partes de un fichero c/c++ Declaración, definición y llamada

- El concepto de declaración, definición y llamada, se utiliza tanto para variables , objetos y funciones.
- **Declaración:** Es necesario realizar una declaración antes de usar una función, variable u objeto.
- **Definición:** Una vez realizada una declaración se puede asignar (en el caso de variables u objetos) u valor según el tipo declarado o definir las instrucciones que conforman la función.
- **Llamada:** Se realiza una llamada a una función para hacer uso de ella.

Partes de un fichero c/c++ Funciones

- Es la parte independiente del programa principal.
- Su estructura es la siguiente:

```
Tipo_de_valor_retorno nombre_de_la_funcion(datos_de_entrada)
{
    Declaración de variables;
    instrucciones;
    instrucción de retorno;
}
```

Partes de un fichero c/c++ Declaración y definición de la función principal

- Todo programa en C++, consta de un programa principal, que es con el que empieza la ejecución del programa. Este programa que realmente es una función, se llama **main()** (función principal) y tiene la forma siguiente:

```
int main(int argc, char *argv[])  
{  
    sentencias o operaciones del programa  
}
```

Otro uso mas habitual es:

```
void main(void)  
{  
    sentencias o operaciones del programa  
}
```

Se define una vez declaradas las librerías necesarias, las variables globales y constantes y las funciones que vamos a usar.

Partes de un fichero c/c++ Comentarios

- Los comentarios es la parte de código fuente que el compilador no examina.

// comentario solo para una única línea de código

/*

todo el código que se sitúe entre estos dos símbolos se considera un comentario, indistintamente de las líneas que lo conformen

*/



Ejercicios (1/2)

- Escriba un programa que pida al usuario que escriba dos enteros, que obtenga los números del usuario e imprima el número más grande, seguido de las palabras "es más grande". Si los números son iguales, imprima el mensaje "Estos números son iguales."
- Escriba un programa que reciba tres enteros del teclado e imprima la suma, promedio, producto, menor y mayor de esos números.
- Escriba un programa que lea cinco enteros y que determine e imprima los enteros mayor y menor en el grupo.
- Escriba un programa que lea un entero y que determine e imprima si es impar o par. [*Sugerencia*: use el operador modulo, un número par es un múltiplo de dos. Cualquier múltiplo de dos deja un residuo de cero cuando se divide entre dos.]
- Escriba un programa que lea dos enteros, determine si el primero es un múltiplo del segundo e imprima el resultado. [*Sugerencia*: use el operador de modulo.]



Ejercicios (2/2)

- Sabiendo que una milla son 1 609 Kilómetros, realice un programa que lea por teclado una cantidad en millas y la muestre en kilómetros.
- Realice un programa que pida al usuario una nota, y exprese un resultado en suspenso (nota <5), aprobado (nota entre 5 y 7), notable (nota entre 7 y 9) o sobresaliente (nota entre 9 y 10).
- Escriba un programa que pida al usuario 2 números, y escriba los números pares existentes entre ambos números.
- A una fiesta ingresan personas de diferentes edades, no se permite el ingreso de menores de edad. Se pide la edad menor, la mayor, y el promedio de edades. El ingreso debe terminar cuando la edad ingresada sea cero.
- Determine cuántos dígitos tiene un número entero ingresado por teclado.