

# PROGRAMACIÓN

**Unidad 2 - Parte 1:** Lenguajes de programación. Programación estructurada. El lenguaje C. Elementos básicos de un programa. Análisis de programas. Estilo de programación.

# Repasemos lo visto



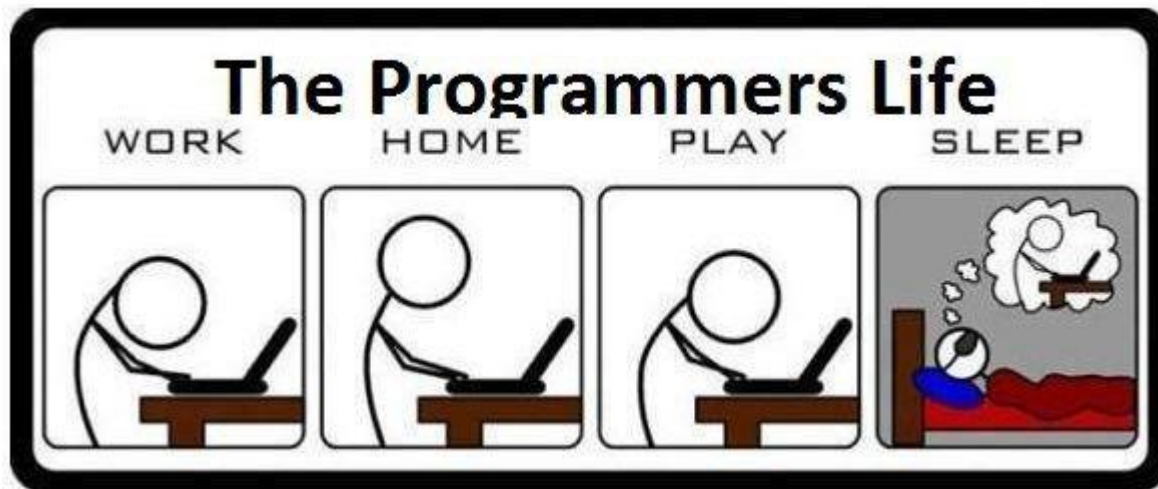
# Etapas de Resolución de Problemas



- Análisis del problema.
- Diseño de la Solución.
- Especificación del Algoritmo.
- Escrituras del Programas
- Verificación.
- Documentación

# Escritura de programas

- ❑ El algoritmo se “traduce” al lenguaje de programación elegido.
- ❑ Escribir las instrucciones.
- ❑ Programar NO ES LO MISMO QUE codificar.

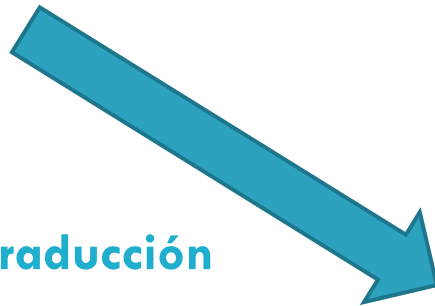


# Verificación: corrección, prueba y optimización

**En la ejecución**



traducción



**Lenguaje de maquina**



# Verificación: corrección, prueba y optimización

## En la compilación y prueba:

### ➤ Los errores de ejecución

Son condiciones que afectan la operación normal del programa y pueden originarse por múltiples circunstancias, entre ellas el uso inadecuado del programa por parte del usuario: datos incorrectos o con un formato diferente al esperado, etc.

### ➤ Los errores de tipo lógico

Son todos aquellos derivados de un mal diseño de los algoritmos, como ser: bucle infinito, resultados incorrectos, etc.

### ➤ Los errores de sintaxis

Son todos aquellos que se generan por no cumplir con las “normas” de escritura de un lenguaje, como ser: falta o mal uso de elementos separadores (comas, puntos y comas), palabras mal escritas, etc.

# Verificación: corrección, prueba y optimización

**Conjuntos de datos  
de prueba**



**Correctitud**

# Documentación



## Documentación interna:

- ☐ Tabulación en el código.
- ☐ Uso de comentarios.

## Documentación externa:

- ☐ Para el usuario.
- ☐ Para el programador.



# Conclusión

Un programador debe asociar inmediatamente el proceso de desarrollo de software con el proceso de refinamiento y abstracción que abarca desde el problema real hasta su solución algorítmica con un lenguaje de programación.



# Definición de Lenguaje

El lenguaje es el sistema a través del cual el hombre **comunican sus ideas y sentimientos**, ya sea a través del habla, la escritura u otros signos convencionales

El término lenguaje es de origen latín **lingua**



# Lenguaje de programación



Un **lenguaje de programación** es un lenguaje formal que proporciona una serie de instrucciones que permiten a un programador escribir secuencias de órdenes y algoritmos a modo de controlar el comportamiento físico y lógico de una computadora con el objetivo de que produzca diversas clases de datos. A todo este conjunto de órdenes y datos escritos mediante un lenguaje de programación se le conoce como programa.

# Tipos de lenguajes

La maquina sólo entiende un lenguaje conocido como código binario o código máquina, consistente en ceros y unos. Es decir, sólo utiliza 0 y 1 para codificar cualquier acción.

Existen dos tipos de lenguajes claramente diferenciados:

## Lenguaje de bajo nivel.



Los lenguajes más próximos a la arquitectura hardware se denominan lenguajes de **bajo nivel**

## Lenguaje de alto nivel.



Los lenguajes que se encuentran más cercanos a los programadores y usuarios se denominan lenguajes de **alto nivel**.

# Lenguajes de bajo nivel

Son lenguajes totalmente dependientes de la máquina, es decir que el programa que se realiza con este tipo de **lenguajes no se pueden migrar o utilizar en otras máquinas.**

Al estar prácticamente diseñados a medida del hardware, aprovechan al máximo las características del mismo.

# Dentro de los lenguajes de bajo nivel se encuentra:



**El lenguaje maquina:** este lenguaje ordena a la máquina las operaciones fundamentales para su funcionamiento. Consiste en la combinación de 0's y 1's para formar las ordenes entendibles por el hardware de la maquina.

Este lenguaje es mucho más rápido que los lenguajes de alto nivel. La desventaja es que son bastantes difíciles de manejar.

# Dentro de los lenguajes de bajo nivel se encuentra:

**El lenguaje ensamblador:** es un derivado del lenguaje maquina y esta formado por abreviaturas de letras y números llamadas mnemotécnicos. Con la aparición de este lenguaje se crearon los programas traductores para poder pasar los programas escritos en lenguaje ensamblador a lenguaje máquina. Como ventaja con respecto al código máquina es que los códigos fuentes eran más cortos y los programas creados ocupaban menos memoria, pero seguía siendo difícil de probar y mantener.


# Lenguaje de Alto Nivel

Son aquellos que se encuentran más cercanos al lenguaje natural que al lenguaje máquina. Están dirigidos a solucionar problemas mediante el uso de Estructuras Dinámicas de Datos. **Se tratan de lenguajes independientes de la arquitectura del ordenador.** Por lo que, en principio, un programa escrito en un lenguaje de alto nivel, lo puedes migrar de una máquina a otra sin ningún tipo de problema.

Estos lenguajes permiten al programador olvidarse por completo del funcionamiento interno de la máquina para la que están diseñando el programa. Tan solo necesitan un traductor que entienda el código fuente como las características de la máquina.



# Podemos llamar HTML como Lenguaje de Programación?



En principio diremos que HTML no es un lenguaje de programación, aunque de forma coloquial muchas veces se escuche referencias a HTML como si lo fuera.







HTML es un lenguaje de etiquetas. Estas etiquetas (tag) comunican al navegador cuál es la información a mostrar por pantalla, además del formato de dicha información.

# Ejemplo:

Lenguaje	Código	Salida por pantalla
C	<pre>#include &lt;stdio.h&gt;  int main() {     int i;      for(i=1; i&lt;=10; i++)     {         printf("%d", i);     }     return 0; }</pre>	1 2 3 4 5 6 7 8 9 10
HTML	<pre>&lt;html&gt;   &lt;body&gt;     &lt;p&gt;1&lt;/p&gt;     &lt;p&gt;2&lt;/p&gt;     &lt;p&gt;3&lt;/p&gt;     &lt;p&gt;4&lt;/p&gt;     &lt;p&gt;5&lt;/p&gt;     &lt;p&gt;6&lt;/p&gt;     &lt;p&gt;7&lt;/p&gt;     &lt;p&gt;8&lt;/p&gt;     &lt;p&gt;9&lt;/p&gt;     &lt;p&gt;10&lt;/p&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	1 2 3 4 5 6 7 8 9 10

# Los lenguajes mas usados en la actualidad

El índice de la comunidad de programación TIOBE es un indicador de la popularidad de los lenguajes de programación. Las calificaciones se basan en la cantidad de informáticos calificados en todo el mundo, cursos y proveedores externos.

Aug 2024	Aug 2023	Change	Programming Language	
1	1			Python
2	3	▲		C++
3	2	▼		C
4	4			Java
5	5			C#
6	6			JavaScript

TIOBE Index for August 2024 (<https://www.tiobe.com/tiobe-index/>)



C

Lo utilizan la mayoría de los sistemas operativos lo cual hace que sea un lenguaje muy flexible.

También se utiliza frecuentemente para el desarrollo de aplicaciones de escritorio como por ejemplo GIMP.



# JAVA

JAVA sigue desde hace muchos años siendo el lenguaje programación más usado, quizás sea por su gran legibilidad y simplicidad.

Actualmente cuenta con más de 9 millones desarrolladores que lo usan y está presente en más de 7 mil millones de dispositivos en todo el mundo.



# C++

C++ es un lenguaje de programación orientado a objetos y una evolución del lenguaje C.

Es un lenguaje muy utilizado para desarrollar programas y paquetes como por ejemplo el paquete de programas de Adobe.



# Python

Python es un lenguaje de programación  
multiplataforma y multiparadigma.

Es muy fácil de utilizar lo cual lo hace un lenguaje de  
programación ideal para principiantes.



# C#

C# es un lenguaje de programación orientado a objetos fue desarrollado en el año 2000 por Microsoft para ser empleado en una amplia gama de aplicaciones empresariales.

C# es una evolución de los lenguajes de programación C y C++, y destaca por su sencillez.





# Javascript

Javascript es un lenguaje de programación que puede ser utilizado para crear programas que son integrados a una página web o dentro de aplicaciones más grandes.

Además lo podemos utilizar para crear efectos y realizar acciones interactivas.

Algunos ejemplos de este lenguaje son el chat, calculadoras, buscadores de información y un sin fin de utilidades más.

# La programación estructurada

La **programación estructurada** es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora



# La programación estructurada



La programación estructurada propone segregar los procesos en estructuras elementales:

- Secuencia
- Selección
- Iteración



# Lenguaje C

# Un poco de historia...

Los laboratorios Bell lo desarrollaron a principios de la década del 70.



Los autores del lenguaje son: Brian Kernighan (Canadá) , y Dennis Ritchie(Estados Unidos).



El objetivo de su creación fue para que los programadores de Bell pudiesen redactar su sistema operativo UNIX para una nueva computadora producida por DEC (Digital Equipment Corporation).

Debido a que los otros lenguajes de alto nivel existentes en aquel tiempo (COBOL, FORTRAN, etc), eran demasiados lentos para ser utilizados en la codificación de un sistema operativo. Los programadores de laboratorios Bell decidieron desarrollar su propio lenguaje, basado en Algol y BCPL, dos eficientes lenguajes de alto nivel.

# Características de C



- Es un lenguaje para la programación estructurada.
- Es tipificado.
- Contiene muy pocas palabras reservadas.
- No contiene órdenes para trabajar con objetos compuestos (cadenas, registros, etc).
- Distingue entre mayúsculas y minúsculas.

# Ventajas

- Es el lenguaje más portado en existencia, habiendo compiladores para casi todos los sistemas conocidos.
- Proporciona facilidades para realizar programas modulares y/o utilizar código o bibliotecas existentes.
- Es un lenguaje muy flexible, muy veloz y potente, lo que permite un software efectivo.
- Posibilita una programación estructurada o modular.
- Acceso a memoria de bajo nivel mediante el uso de punteros.

# Desventajas

- No tiene instrucciones propias para la asignación dinámica de memoria ni instrucciones de entrada/salida. Todas estas operaciones de alto nivel pueden ser realizadas por funciones explícitamente.
- Se requiere más tiempo en conseguir el ejecutable, porque cada vez compila todo el fichero.
- No dispone de sistemas de control automáticos y la seguridad depende casi exclusivamente de la experiencia del programador.



**ENTONCES SATANAS DIJO,  
USEMOS EL LENGUAJE C**



Imagen creada en [GeneradorMemes.com](https://www.GeneradorMemes.com)

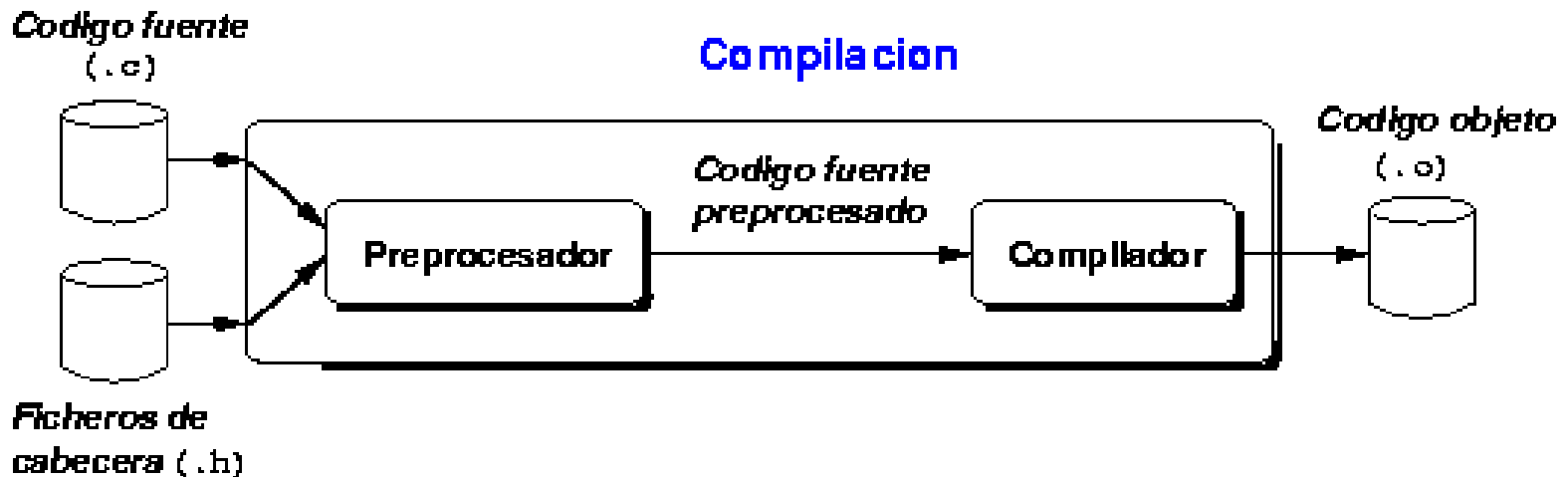
# En un programa intervienen



- ❑ Ordenes para el preprocesador
- ❑ Variables
- ❑ Constantes
- ❑ Aritmética
- ❑ Funciones
- ❑ Funciones de entrada y salida
- ❑ Comentarios

# Ordenes para el preprocesador

- ❑ Conceptualmente es un paso previo a la compilación.
- ❑ Las mas usadas son: `# include`  
`# define`
- ❑ Para inclusión de archivos.



# Funciones

“ Un programa escrito en código C es una reunión de funciones “.

**main** {  
función principal debe estar , presente en todos los programas escritos en C  
Puede invocar a otra funciones .

- Un método para comunicar datos entre las funciones, es que la función que llama proporciona una lista de valores, llamados argumentos.
- Los paréntesis después del nombre de la función, están para encerrar una lista de valores que serán argumentos.
- Puede ocurrir que una función este definida para ser una función que no espera argumentos, tal es el caso del main, lo cual se indica con una lista de argumentos vacía.

**main()**

# Funciones de Entrada y Salida

La biblioteca estándar `stdio.h` provee al programador de una extensa gama de funciones para lectura y escritura.

Es necesario escribir una orden para el preprocesador para usar dichas funciones.

```
#include <stdio.h> { printf("Hola Mundo");  
                    scanf("%d", &num);
```

# De las variables y constantes

- ❑ Objetos sobre los que actúan las instrucciones que componen el programa.
- ❑ Deben declararse como tales.
- ❑ Deben tener un identificador asociado.
- ❑ El 1º carácter debe ser una letra. Letras mayúsculas y minúsculas son diferentes.
- ❑ Debe indicarse que tipo de datos pueden contener.
- ❑ Las variables pueden inicializarse en forma grupal.
- ❑ Las constantes se definen en una línea `#define` , que es una directiva del preprocesador.

*`#define` nombretexto de reemplazo*

Por ejemplo: `#define MAX 100`

- ❑ Las constantes pueden ser enteras, reales y de carácter.

# Aritmética

- ❑ Interacción entre los operadores aritméticos y las variables y/o constantes declaradas (si el tipo es numérico).
- ❑ El tipo de operación permitido está **ligado tipo de dato** con que fue declarada la variable y/o constante.
- ❑ Tipos de datos básicos: int, float, char, short, long, double.

# Comentarios



Dos modos de comentar las acciones del código escrito:

// comentarios de una sola línea

/\*comentarios de varias líneas

..... \*/



# Algunas recomendaciones

- ❑ La metodología Top-down.
- ❑ Divide and conquer
- ❑ Recursos del lenguaje
- ❑ Elección del tipo de dato apropiado.
- ❑ Comentarios.
- ❑ Uso de sangría apropiada.



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    printf("Hasta la próxima clase!!\n");
```

```
    return 0;
```

```
}
```