

# LENGUAJES DE PROGRAMACIÓN

Elementos de Informática

# Lenguajes de Programación

Se denominan lenguajes de programación a una serie de palabras predefinidas que se combinan según reglas establecidas (sintaxis) para dar origen a un programa que puede ser ejecutado por una computadora

Existen tres grandes tipos:

- Lenguaje de máquina,
- Lenguaje ensamblador
- Lenguajes de Alto nivel.

# Lenguajes de Bajo Nivel

Los lenguajes de bajo nivel están formados por cadenas de 0 y 1.

Es el único lenguaje comprendido por una computadora.

Programa en lenguaje de máquina  
que multiplica dos números e  
imprime el resultado

1.	00000000	00000100	0000000000000000
2.	01011110	00001100	11000010 0000000000000010
3.		11101111	00010110 00000000000000101
4.		11101111	10011110 00000000000001011
5.	11111000	10101101	11011111 0000000000010010
6.		01100010	11011111 0000000000010101
7.	11101111	00000010	11111011 0000000000010111
8.	11110100	10101101	11011111 0000000000011110
9.	00000011	10100010	11011111 000000000100001
10.	11101111	00000010	11111011 000000000100100
11.	01111110	11110100	10101101
12.	11111000	10101110	11000101 000000000101011
13.	00000110	11111011	11000101 000000000110001
14.	11101111	00000010	11111011 000000000110100
15.			00000100 000000000111101
16.			00000100 000000000111101

# Lenguajes ensambladores

Los lenguajes ensambladores también se consideran lenguajes de “bajo nivel”.

Grace Hooper, miembro de la Armada de los Estados Unidos desarrolló un lenguaje de símbolos basados en una ayuda nemotécnica que representan instrucciones del lenguaje de máquina.

El ensamblador es el programa que se encarga de traducir el código simbólico en lenguaje de máquina

---

```
1. entry main, ^m<r2>
2. sub12 #12, sp
3. jsb    C$MAIN_ARGS
4. movab  $CHAR_STRING_CON
5.
6. pushal -8(fp)
7. pushal (r2)
8. calls  #2, read
9. pushal -12(fp)
10. pushal 3(r2)
11. calls  #2, read
12. mull3  -8(fp), -12(fp), -
13. pusha  6(r2)
14. calls  #2, print
15. clrl                      r0
16. ret
```

---

# Lenguajes de alto nivel

Los lenguajes de alto nivel facilitan la tarea del programador permitiendo que la atención se focalice en el problema y no en los detalles de la computadora. Estos lenguajes se pueden portar a muchas plataformas.

El compilador es el programa que se encarga de traducir el código en lenguaje de máquina

```
1. /* Este programa lee dos números enteros desde el
2. teclado e imprime su producto.
3. Escrito por:
4. Fecha:
5. */
6. #include <iostream.h>
7.
8. int main (void)
9. {
10. // Declaraciones locales
11.     int number1;
12.     int number2;
13.     int result;
14.
15. // Instrucciones
16.     cinn >> number1;
17.     cinn >> number2;
18.     result = number1 * number2;
19.     cout << result;
20.     return 0;
21. } // principal
```

# Algoritmo – Programa - Proceso

Al abordar el concepto de programación se deben conocer los siguientes conceptos:

**Algoritmo:** sucesión finita y ordenada de pasos, libre de ambigüedades, que permite obtener la resolución de un problema dado.

```
algoritmo Sumar
```

```
variables
```

```
    entero a, b, c
```

```
inicio
```

```
    escribir( "Introduzca el primer número (entero): ")
```

```
    leer( a )
```

```
    escribir( "Introduzca el segundo número (entero): ")
```

```
    leer( b )
```

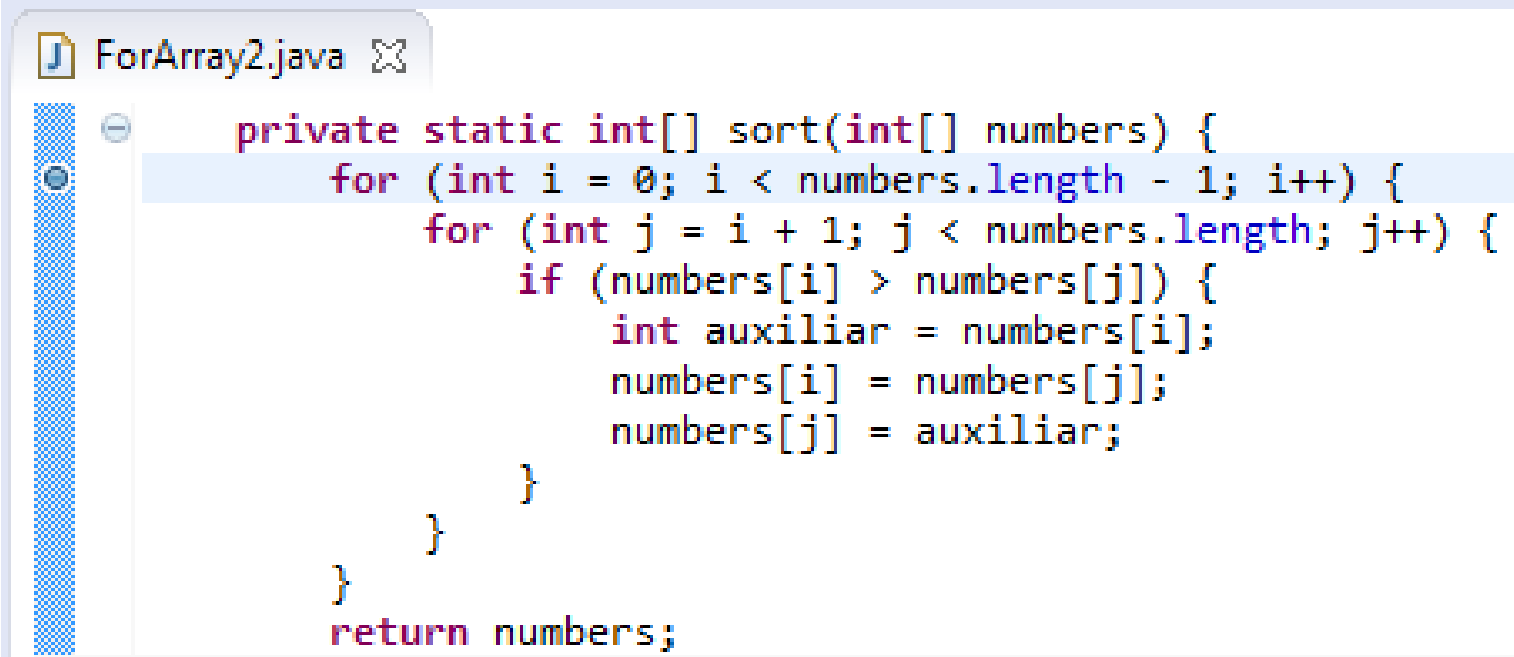
```
     $c \leftarrow a + b$ 
```

```
    escribir( "La suma es: ", c )
```

```
fin
```

# Algoritmo – Programa - Proceso

**Programa:** Es un algoritmo escrito en lenguaje de programación.



The image shows a screenshot of a Java code editor with a tab labeled "ForArray2.java". The code implements a bubble sort algorithm. The first line of the code is highlighted in blue. The code is as follows:

```
private static int[] sort(int[] numbers) {  
    for (int i = 0; i < numbers.length - 1; i++) {  
        for (int j = i + 1; j < numbers.length; j++) {  
            if (numbers[i] > numbers[j]) {  
                int auxiliar = numbers[i];  
                numbers[i] = numbers[j];  
                numbers[j] = auxiliar;  
            }  
        }  
    }  
    return numbers;  
}
```

**Proceso:** Es la ejecución de un programa.

# Programación

A partir de un problema real, la programación consta de los siguientes pasos:

1. Entender el problema
2. Encontrar un algoritmo de resolución
3. Representar el algoritmo en un Pseudo lenguaje y realizar las pruebas
4. Codificación: traducir el algoritmo de un pseudo lenguaje a un lenguaje de programación
5. Compilar: depurar errores de sintaxis.
6. Ejecución
7. Ajuste
8. Mantenimiento



# Criterios de evaluación de un Lenguaje de Programación

La elección de un lenguaje de programación para un proyecto específico, debe tener en cuenta las características del problema a resolver. Entre los criterios que se utilizan durante la evaluación de los lenguajes disponibles están:

# Criterios de evaluación de un Lenguaje de Programación

- 1) Área de aplicación general.
- 2) Complejidad algorítmica y computacional.
- 3) Entorno de ejecución del soft.
- 4) Consideraciones de rendimiento.
- 5) Complejidad de las estructuras de datos.
- 6) Disponibilidad de un buen compilador.

# Tipos de Aplicaciones:

## Aplicaciones Científicas

Manipulan predominantemente números y arrays de números, usando principios matemáticos y estadísticos como base de los algoritmos: test estadísticos, programación lineal, análisis de regresión, ecuaciones diferenciales e integrales.

Se caracterizan por: pocos datos, estructuras de datos simples; más trabajo del procesador central que de los dispositivos de E/S.

Lenguajes adecuados a estas aplicaciones son:  
R, Python, SQL, Java, Scala, Julia, MATLAB

## Tipos de Aplicaciones:

### Aplicaciones de Procesamiento de Datos

Se trata de problemas cuyo interés predominante es la creación, mantenimiento, extracción y recopilación de datos en registros y archivos.

Incluyen funciones relativas a las nóminas, contabilidad, facturación, inventario, producción y ventas.

Estas aplicaciones suponen un gran volumen de datos y cantidad de cálculos aritméticos bajo.

Lenguajes adecuados a estas aplicaciones son el PYTHON, R, RPG, SQL...

# Tipos de Aplicaciones:

## Aplicaciones de Inteligencia Artificial

Son programas que han sido diseñados principalmente para emular el comportamiento inteligente.

Incluyen algoritmos de juegos tales como ajedrez, comprensión del lenguaje natural, robótica y “sistemas expertos”;

Lenguajes adecuados para desarrollar estas aplicaciones son: Python, Prolog, Java, C++

## Tipos de Aplicaciones:

### Aplicaciones de Programación de Sistemas

Incluyen el desarrollo de programas que hacen de interfase entre el hardware y el programador. Incluyen: compiladores, intérpretes, rutinas de E/S.

Lenguajes adecuados a estas aplicaciones son el C, Ada, Modula-2...

# Tipos de Aplicaciones:

## Aplicaciones de Sistemas de Tiempo Real

En este tipo de sistemas, la secuencia temporal de entradas es determinada total o parcialmente por el mundo real.

Lenguajes adecuados a estas aplicaciones son el Ada, Modula-2, Java.

# Tipos de Aplicaciones:

## Aplicaciones de Procesamiento de Texto

Su principal actividad consiste en la manipulación de texto del lenguaje natural en vez de números.

Lenguajes adecuados a estas aplicaciones son el SNOBOL, C, Perl.



# Criterios Para Evaluación Y Comparación De Lenguajes

1. Expresividad: es la habilidad que el lenguaje posee para expresar el significado deseado.
2. Nivel de definición: se refiere a que la sintaxis y semántica del lenguaje no presenten ambigüedad.
3. Tipos y Estructuras de Datos: se evalúa la habilidad del lenguaje para soportar distintos.

# Criterios Para Evaluación Y Comparación De Lenguajes

4. Modularidad: se considera el soporte de subprogramas y extensibilidad del lenguaje.
5. Facilidades de E/S: soporte de acceso a archivos y bases de datos.
6. Transportabilidad: diseño del lenguaje independiente de la máquina.

# Criterios Para Evaluación Y Comparación De Lenguajes

7. Eficiencia: se refiere a la rapidez con que el lenguaje realiza la compilación y ejecución.
8. Pedagogía: se considera si el lenguaje es intrínsecamente fácil de enseñar y aprender.
9. Generalidad: se verifica su utilidad en un amplio rango de aplicaciones.

# Paradigma de Programación Imperativa

Los lenguajes imperativos o procedurales ejecutan una serie de instrucciones una después de la otra con la posibilidad de elegir diferentes caminos como así también ejecutar instrucciones más de una vez.

Cada instrucción es un comando para que el sistema realice una tarea específica.

Ejemplo: C, Ada, Pascal

# Paradigma de Programación Lógica

Utiliza el razonamiento lógico, basado en la deducción, para responder una consulta.

Se plantean instrucciones que se suponen verdaderas y a partir de la lógica y las reglas de razonamiento se deducen nuevas reglas.

Se utiliza para programar inteligencia artificial

Ejemplos: Prolog, Mercury, Oz.

# Paradigma de Programación Funcional

Los programas se consideran una función matemática. Una función es una caja negra que relaciona una lista de entradas y devuelve una lista salida.

El lenguaje funcional fomenta la programación modular y permite crear nuevas funciones.

Ejemplos: LISP, Scheme, Haskell, Scala

# Paradigma de Programación Orientada a objetos

La programación orientada a objetos se basa en el concepto de clases y de objetos los cuales tienen las operaciones ligadas.

El programador define el objeto y las operaciones que se le pueden aplicar. Los objetos se consideran activos.

Ejemplo: C++, Java