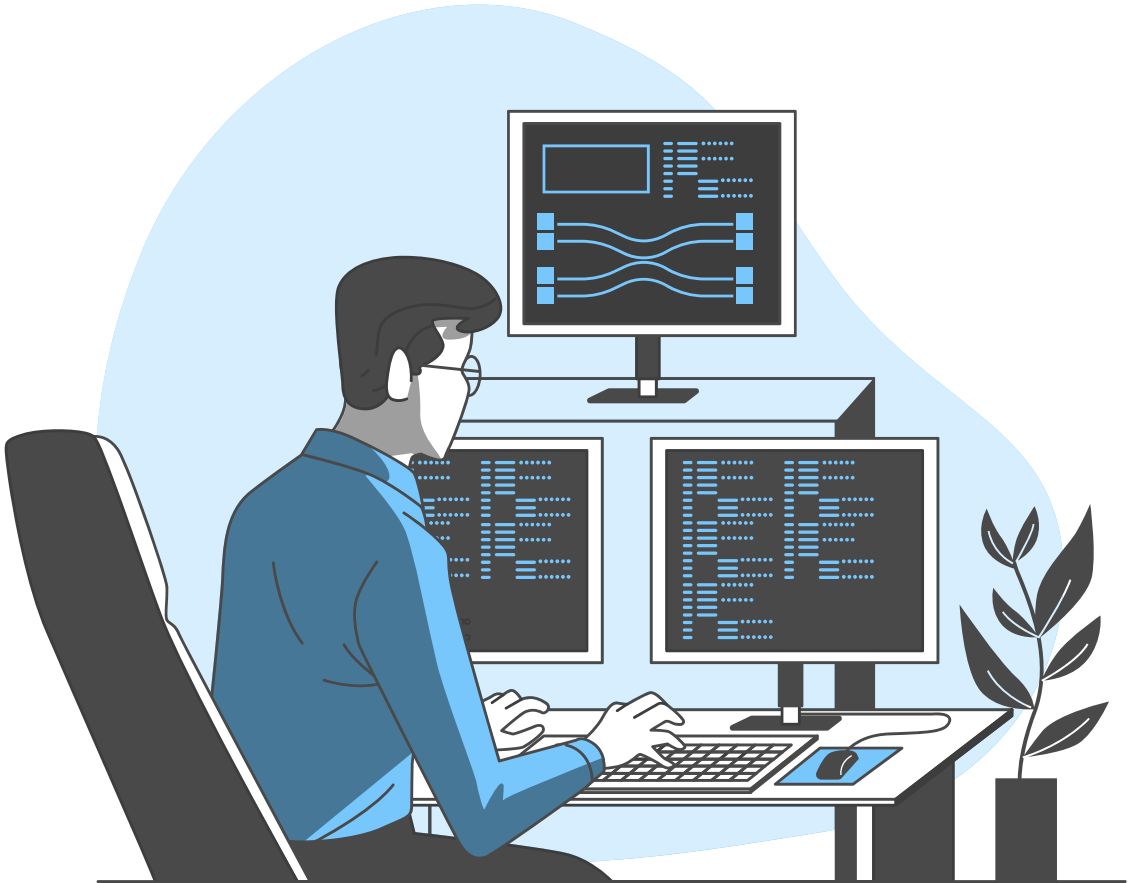


Introducción a la programación

Clase 1

Profesor – Martín Gómez Vega

Da Vinci
PRIMERA ESCUELA DE ARTE MULTIMEDIAL

Agenda

1. Concepto de algoritmo
2. Concepto de programa
3. Concepto de instrucciones
4. Diferencia entre lenguajes de programación compilados e interpretados
5. Proceso de compilación e interpretación de un programa
6. Importancia de los algoritmos en la programación y cómo se relacionan con la solución de problemas
7. Uso de instrucciones en la programación para lograr una tarea específica
8. Importancia de la sintaxis en la programación
9. Función, propósito y uso de los lenguajes de programación

Concepto de algoritmo

Un algoritmo es una secuencia finita de pasos precisos y ordenados que resuelven un problema específico o realizan una tarea.

Los algoritmos son fundamentales en programación, ya que son la base para crear programas.

El programador tiene un rol clave planteando de manera correcta los algoritmos para que la computadora lo interprete.

Concepto de programa

Un programa es un conjunto de instrucciones que le indican a una computadora cómo realizar una tarea o resolver un problema.

Los programas son escritos por programadores en lenguajes de programación.

Ejemplo:
Un programa en Java que muestra "Hola, mundo!" en la pantalla:

```
public class HolaMundo {
   public static void main(String[ ] args) {
      System.out.println("Hola, mundo!");
   }
}
```

Concepto de programa

Ejemplo:
Un programa en Java que muestra "Hola, mundo!" en la pantalla:

```
public class HolaMundo {
  public static void main(String[ ] args) {
    System.out.println("Hola, mundo!");
  }
}
```

1. **public class HolaMundo {**
   Aquí estamos definiendo una "clase" en Java llamada "HolaMundo". En programación, una clase es como un plano o una plantilla que define cómo se comportará un objeto.

2. **public static void main(String[ ] args) {**
   Dentro de la clase "HolaMundo", estamos definiendo un "método" llamado "main". El método "main" es especial en Java, ya que es el punto de entrada para la ejecución de un programa.

3. **System.out.println("Hola, mundo!");**
   Dentro del método "main", estamos usando una "sentencia" que muestra un mensaje en la pantalla. La sentencia es System.out.println("Hola, mundo!");, y hace que el programa imprima la cadena de texto "Hola, mundo!" en la consola (la ventana de texto en la que se ejecuta el programa).

Conceptos claves

Variable

Es una ubicación de memoria que se reserva para almacenar datos.
Las variables en Java deben declararse con un tipo de dato específico antes de ser utilizadas.
Pueden tener un valor asignado inicialmente y ese valor puede cambiar durante la ejecución del programa.

```
public class EjemploVariables {
    public static void main(String[ ] args) {
        int edad; // Declaración de la variable "edad" de tipo entero
        edad = 25; // Asignación del valor 25 a la variable "edad"
        System.out.println("La edad es: " + edad);
    }
}
```

Parámetro

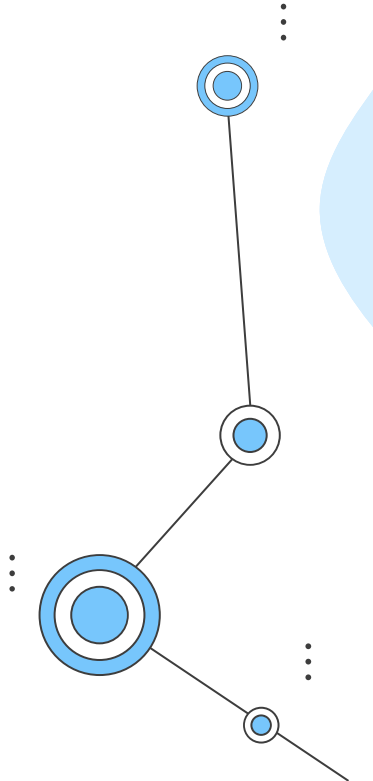
Es una variable local declarada en la lista de parámetros de un método o función.
Los parámetros permiten que un método reciba valores específicos cuando es llamado, lo que le permite operar con esos valores de manera dinámica.

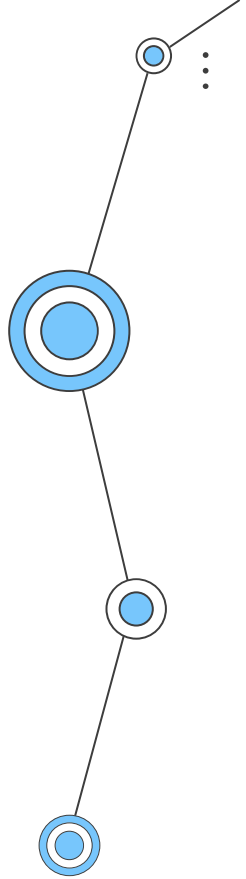
```
public void saludar(String nombre) {
    System.out.println("Hola, " + nombre);
}
```

Concepto de instrucciones

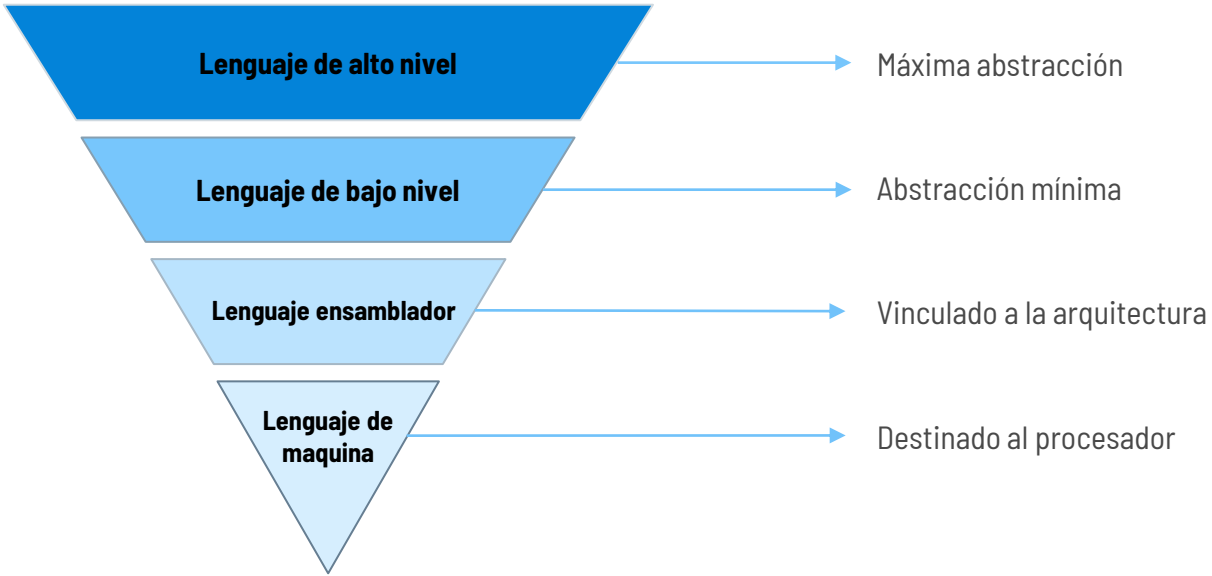
Las instrucciones son comandos específicos que se le dan a la computadora para realizar una acción.

Las instrucciones forman parte de un programa y se ejecutan en secuencia.





Lenguajes de programación



Diferencia entre lenguajes de programación compilados e interpretados

La diferencia se basa en cómo se traduce y ejecuta el código fuente de un programa.

Lenguajes compilados
El código fuente del programa se traduce completamente a un lenguaje de bajo nivel o código máquina antes de que se ejecute.
El compilador toma todo el código y lo convierte en un archivo ejecutable que se puede ejecutar sin necesidad de recompilación en cada ejecución.

Ventajas	Desventajas
Rendimiento: Los programas compilados tienden a ser más rápidos, ya que están completamente traducidos antes de la ejecución.	**Proceso de compilación**: Requiere un paso de compilación antes de ejecutar el programa.
Errores de sintaxis: El compilador verifica la sintaxis del código antes de la ejecución, lo que ayuda a detectar errores antes de tiempo.	**Portabilidad**: Los programas compilados suelen ser específicos de la plataforma, lo que significa que pueden requerir ajustes para ejecutarse en diferentes sistemas operativos.

Ejemplos:
C, C++, **Java**, C#, Swift, Kotlin, Golang, entre otros.

Diferencia entre lenguajes de programación compilados e interpretados

Lenguajes Interpretados

El código fuente se traduce y ejecuta línea por línea en tiempo real por un intérprete en lugar de ser traducido completamente antes de la ejecución.

Ventajas	Desventajas
Portabilidad: Los programas interpretados son más portátiles ya que solo necesitas el intérprete en la plataforma destino para ejecutar el código.	**Rendimiento:** Los programas interpretados suelen ser más lentos porque la traducción y ejecución ocurren en tiempo real.
Facilidad de depuración: Los errores pueden ser detectados y corregidos inmediatamente después de escribir el código.	**Errores de tiempo de ejecución:** Los errores de sintaxis pueden no ser detectados hasta que la línea de código se interprete durante la ejecución.

Ejemplos:

Python, JavaScript, Ruby, PHP, entre otros.

Proceso de compilación e interpretación de un programa

El proceso de compilación e interpretación son dos enfoques distintos para traducir y ejecutar programas escritos en un lenguaje de alto nivel, como Java, C++, Python, etc., a instrucciones entendibles por la máquina.

Proceso de compilación
La compilación es como la construcción de un objeto a partir de piezas. Imagina que estás construyendo un automóvil: primero necesitas las piezas, como el motor, las ruedas y las puertas, y luego un proceso de montaje para ensamblar todo.

Paso 1: Escritura del Código:
Imagina que estás escribiendo una carta a un amigo. El código fuente es como esa carta, pero en lenguaje de programación. Escribimos estas instrucciones para decirle a la computadora lo que queremos que haga.

Paso 2: Compilación:
Una vez que hemos escrito el código, necesitamos traducirlo a un lenguaje que la computadora pueda entender. Esto es similar a traducir una carta a un idioma que tu amigo pueda comprender.

Ejemplo:
Supongamos que tenemos un programa simple en el lenguaje C que imprime "Hola, mundo!" en la pantalla. El proceso de compilación tomará este código fuente y lo traducirá en un programa ejecutable, como un archivo que puede ser ejecutado por la computadora para mostrar el mensaje.

Proceso de compilación e interpretación de un programa

Proceso de Interpretación
La interpretación es como un intérprete que sigue tus instrucciones en tiempo real. En lugar de construir un objeto a partir de piezas, alguien sigue tus indicaciones paso a paso mientras las das.

Paso 1: Escritura del Código:
Al igual que en el proceso de compilación, primero escribimos el código fuente. Imagina que estás dando instrucciones a una persona que sigue esas indicaciones.

Paso 2: Interpretación:
A medida que entregamos cada instrucción, el intérprete (como un asistente personal) lo ejecuta inmediatamente. Cada vez que das una orden, el intérprete la sigue y obtienes una respuesta.

Ejemplo:
Supongamos que estamos usando un lenguaje de script como Java. Es como si estuviéramos dando instrucciones a alguien que sigue esas instrucciones en el acto. Si le decimos "imprime 'Hola, mundo!'", el intérprete mostrará el mensaje "Hola, mundo!" en la pantalla de inmediato.

Importancia de los algoritmos en la programación y cómo se relacionan con la solución de problemas

Los algoritmos son como mapas para resolver problemas. En la programación, usamos algoritmos para decirle a la computadora cómo realizar tareas específicas.
Cada vez que creamos un programa, estamos construyendo un conjunto de instrucciones paso a paso que la computadora debe seguir para lograr un objetivo.

¿Cómo solucionan los problemas?
Piensa en un juego de rompecabezas. Para resolverlo, necesitas un plan, una estrategia. Esa estrategia es como un algoritmo.
En programación, enfrentamos problemas que necesitan ser "resueltos" por la computadora. Ya sea ordenar una lista, buscar información o hacer cálculos complejos, necesitamos algoritmos para decirle a la computadora cómo hacerlo.

Ejemplo
Vamos a imaginar que necesitamos encontrar el número más grande en una lista de números. Eso es un problema, ¿verdad? Aquí hay un algoritmo simple para resolverlo:

1. Empieza por el primer número de la lista.
2. Luego, compara ese número con el siguiente.
3. Si el siguiente número es más grande, guárdalo como el número más grande.
4. Sigue comparando con los números restantes, siempre guardando el más grande.
5. Cuando hayas comparado todos los números, el número guardado será el más grande de la lista.

Este algoritmo es como una receta para encontrar el número más grande. Puedes aplicarlo a cualquier lista de números, grande o pequeña, y siempre obtendrás el resultado correcto.

Uso de instrucciones en la programación para lograr una tarea específica

Las instrucciones en un programa son pasos concretos que permiten realizar una tarea específica, como cálculos, entrada/salida de datos, control de flujo, etc.

Ejemplo

Vamos a ver un ejemplo simple. Supongamos que queremos hacer que la computadora muestre el mensaje "Hola, amigos" en la pantalla. Aquí hay una receta sencilla para hacerlo utilizando instrucciones de programación:

1. *Iniciar*: Enciende la computadora.
2. *Abrir el Programa*: Abre el programa de programación (como un cuaderno especial) donde escribiremos las instrucciones.
3. *Escribir el Mensaje*: Escribe una instrucción que diga: "Mostrar 'Hola, amigos' en la pantalla".
4. *Ejecutar*: Presiona un botón para que la computadora siga la instrucción y muestre el mensaje.

Las instrucciones son como una lista de tareas que la computadora sigue para hacer lo que queremos. Si deseas cambiar el mensaje a "¡Bienvenidos a la programación", solo necesitas modificar la instrucción en el paso 3.

En programación, las instrucciones pueden hacer cosas más complejas, como cálculos matemáticos, tomar decisiones basadas en condiciones y mucho más. Pero, en su esencia, son pasos que le dicen a la computadora cómo hacer una tarea específica.

Importancia de la sintaxis en la programación

La sintaxis se refiere a las reglas gramaticales y estructurales del lenguaje de programación.

Una sintaxis incorrecta puede provocar errores en el programa.

Función, propósito y uso de los lenguajes de programación

Los lenguajes de programación son herramientas para que los programadores puedan comunicarse con las computadoras y expresar soluciones de manera comprensible.

Ejemplos de lenguajes de programación incluyen Java, Python, C#, JavaScript, Golang, etc. Cada lenguaje tiene sus propias características y es adecuado para diferentes tipos de aplicaciones.

Para la clase, puedes utilizar el ejemplo del programa "Hola, mundo!" en Java para demostrar cómo se escriben y ejecutan programas. También puedes presentar ejemplos más sencillos de algoritmos para que los estudiantes comprendan cómo se descomponen los problemas en pasos más pequeños.

¿Qué es lo más importante?
La lógica, la precisión y la práctica en la programación.

¿Dudas?