INGENIERÍA – DESARROLLO DE SOFTWARE

DESARROLLO DE SOFTWARE BACKEND I

ROMERO CARRASCO SAMANTHA BELEN

HERNANDEZ GARCIA AARON

UNIDAD 1: ACTIVIDAD TEMA 1.1 LENGUAJES IMPERATIVOS

Resumen de los Lenguajes Imperativos

Definición

Los lenguajes de programación imperativos son aquellos en los que el programador escribe una secuencia de instrucciones para que la computadora las ejecute de manera secuencial. Estos lenguajes se basan en el concepto de cambiar el estado del programa mediante la asignación de valores a variables y el uso de estructuras de control como bucles y condicionales.

Características Principales

1. **Secuencialidad**: Las instrucciones se ejecutan en un orden específico, una tras otra.
2. **Variables**: Uso de variables para almacenar datos que pueden ser modificados durante la ejecución del programa.
3. **Estructuras de Control**: Instrucciones como *if*, *for*, *while* permiten controlar el flujo de ejecución.
4. **Asignaciones**: Las operaciones de asignación son fundamentales, cambiando el valor de las variables a lo largo del tiempo.
5. **Estado del Programa**: El estado del programa es una combinación de los valores actuales de todas las variables.

Ejemplos de Lenguajes Imperativos

- **C**: Un lenguaje de propósito general, conocido por su eficiencia y control de bajo nivel.
- **Pascal**: Diseñado para la enseñanza de la programación estructurada.
- **Java**: Aunque es también un lenguaje orientado a objetos, sigue un enfoque imperativo en su núcleo.
- **Python**: Un lenguaje de alto nivel que soporta varios paradigmas, incluyendo el imperativo.
- **BASIC**: Conocido por su simplicidad y uso en la enseñanza inicial de la programación.

Ventajas

- **Claridad en la Secuencia de Operaciones**: Es fácil entender cómo los datos son manipulados paso a paso.
- **Control Detallado**: Permite un control preciso sobre el hardware y la memoria.
- **Amplio Conjunto de Herramientas y Bibliotecas**: Los lenguajes imperativos tienen una larga historia y una comunidad amplia, proporcionando muchas herramientas y recursos.

Desventajas

- Complejidad en Programas Grandes: La gestión del estado puede volverse complicada y propensa a errores en sistemas grandes.
- Menos Naturales para Problemas Concurrentes: No son tan adecuados para programas que requieren ejecutar múltiples procesos en paralelo.
- Posibilidad de Errores de Bajo Nivel: Errores como la manipulación incorrecta de punteros o la gestión de memoria pueden ser difíciles de detectar y solucionar.

Diferencias entre Lenguaje Imperativos y Declarativos

Enfoque:

- **Imperativo**: Enfocado en cómo se debe hacer algo.
- **Declarativo**: Enfocado en qué se debe lograr.

Control de Flujo:

- **Imperativo**: Uso intensivo de estructuras de control para gestionar el flujo de ejecución.
- **Declarativo**: No se especifica el flujo de ejecución, se confía en que el sistema subyacente lo gestione.

Estado del Programa:

- **Imperativo**: Cambios continuos en el estado del programa mediante asignaciones a variables.
- **Declarativo**: Menos énfasis en el estado, algunos lenguajes no tienen estado mutable.

Legibilidad y Mantenimiento:

- **Imperativo**: Puede volverse complejo y difícil de mantener, especialmente en proyectos grandes.
- **Declarativo**: Suele ser más fácil de leer y mantener debido a su enfoque en la descripción de resultados.

Ejemplos de Uso:

- **Imperativo**: Aplicaciones donde el control detallado del hardware y la eficiencia son críticos, como sistemas operativos y software de tiempo real.
- **Declarativo**: Sistemas de gestión de bases de datos, diseño de interfaces web, y aplicaciones donde la lógica de negocio es más importante que el control del flujo.

Conclusión

Los lenguajes de programación **imperativos** han sido fundamentales en el desarrollo de la informática y siguen siendo ampliamente utilizados debido a su simplicidad y control detallado del hardware. Sin embargo, a medida que los sistemas y los requisitos de software se han vuelto más complejos, se han desarrollado otros paradigmas, como los lenguajes funcionales y orientados a objetos, que abordan mejor ciertos tipos de problemas, como la concurrencia y la gestión de grandes sistemas de software. A pesar de estas nuevas tendencias, el paradigma **imperativo** continúa siendo relevante, especialmente en áreas donde el control fino y la eficiencia son críticos. La elección del lenguaje y el paradigma de programación depende en gran medida del problema específico que se está resolviendo y de las preferencias del desarrollador.

Referencias

Apple, L. F. (2020, junio 13). *Lenguaje de Programación Imperativo*. La Factoría Apple. https://www.lafactoriaapple.com/ciencias-de-la-computacion/lenguaje-imperativo.php

Palacios, D. (s/f). *Programación declarativa vs imperativa*. Styde.net. Recuperado el 16 de mayo de 2024, de https://styde.net/programacion-declarativa-vs-imperativa/