PARTE 1

ANALISIS REFLEXSIVO:

Al comparar cómo se resuelve el cálculo del factorial en C++ (POO), Prolog (programación lógica), y PSeInt (programación estructurada), se pueden observar las distintas fortalezas y debilidades de cada paradigma.

**Programación Orientada a Objetos (POO) en C++:**

**Ventajas:**

* **Encapsulamiento y Modularidad:** La POO en C++ permite encapsular datos y funciones en clases, lo que facilita la organización del código, su mantenimiento y su reutilización. Esto es esencial para proyectos grandes y complejos.
* **Reutilización:** La creación de una clase específica para el cálculo del factorial hace que sea fácil reutilizar este código en otros programas o proyectos futuros.

**Desventajas:**

* **Complejidad:** Para un problema sencillo como el cálculo del factorial, la POO introduce complejidad innecesaria. Crear clases y métodos para una tarea tan simple puede hacer que el código sea más difícil de entender.
* **Rendimiento:** La sobrecarga de la POO, como la creación de objetos, puede reducir el rendimiento, aunque esto es menos significativo en problemas pequeños.

**Programación Lógica en Prolog:**

**Ventajas:**

* **Declaratividad y Simplicidad:** En Prolog, el código se centra en describir lo que se quiere lograr (el factorial de un número), sin necesidad de especificar cómo debe realizarse paso a paso, lo que hace que el código sea muy directo y fácil de seguir.
* **Manejo de Recursión:** Prolog maneja de forma natural la recursión, lo que es ideal para calcular factoriales, ofreciendo un código conciso y elegante.

**Desventajas:**

* **Rendimiento:** Aunque la recursión es elegante, puede ser menos eficiente en Prolog, especialmente si se compara con un enfoque iterativo en otros lenguajes.
* **Familiaridad:** Prolog no es tan común como otros lenguajes, lo que limita su uso en entornos industriales o educativos más amplios.
* **Interactividad Limitada:** Las operaciones de entrada y salida en Prolog pueden ser menos intuitivas, lo que podría complicar la creación de programas interactivos.

**Programación Estructurada en PSeInt:**

**Ventajas:**

* **Simplicidad y Claridad:** PSeInt, diseñado para la enseñanza de la programación, permite escribir código de manera clara y secuencial. Cada paso del proceso (como la lectura de un número, el cálculo del factorial, y la impresión del resultado) se expresa directamente, lo que facilita la comprensión.
* **Eficiencia:** La programación estructurada en PSeInt permite un control directo del flujo del programa, lo que puede hacer que el código sea más eficiente, especialmente cuando se usa un enfoque iterativo para calcular el factorial.
* **Accesibilidad:** PSeInt es una herramienta accesible y fácil de usar, especialmente para principiantes, lo que lo convierte en un buen punto de partida para aprender los conceptos básicos de programación.

**Desventajas:**

* **Falta de Abstracción:** Aunque la simplicidad es una ventaja, también puede ser una limitación. La programación estructurada no ofrece las mismas herramientas para modularidad y reutilización que la POO, lo que puede llevar a la repetición de código.
* **Escalabilidad:** En proyectos grandes, la programación estructurada puede volverse difícil de manejar, ya que no proporciona los mecanismos de organización avanzada que ofrece la POO.

**Conclusión:**

Cada uno de estos paradigmas tiene su propio conjunto de ventajas y desventajas, que reflejan diferentes enfoques para resolver problemas computacionales. **La POO en C++** es excelente para proyectos grandes y modulares, aunque puede ser innecesariamente compleja para problemas simples. **Prolog** ofrece un enfoque lógico y elegante, ideal para problemas que se benefician de una estructura recursiva, aunque su uso está más limitado. **PSeInt** y la programación estructurada son ideales para comenzar, ofreciendo un enfoque directo y sencillo, aunque carecen de la capacidad de abstracción y reutilización que los otros paradigmas pueden ofrecer.

Entender estos paradigmas y sus aplicaciones no solo nos permite elegir la herramienta adecuada para cada tarea, sino que también nos enriquece como programadores, dándonos diferentes perspectivas para abordar los desafíos de la programación.