Punto 4

Comparación del rendimientode un lenguaje de programación compilado e interpretado

Introducción

Los lenguajes de programación se dividen en dos categorías principales: compilados e interpretados. Cada enfoque tiene sus ventajas y desventajas, y es importante comprender cómo afectan el rendimiento de los programas. En este informe, nos centraremos en comparar C++ (un lenguaje compilado) y Python (un lenguaje interpretado).

Metodología

Para evaluar el rendimiento de estos lenguajes, consideraremos los siguientes aspectos:

- Velocidad de ejecución: Compararemos el tiempo que tarda cada lenguaje en ejecutar tareas específicas.
- Uso de recursos: Analizaremos la cantidad de memoria y CPU utilizada por los programas escritos en C++ y Python.
- Facilidad de desarrollo: Discutiremos la productividad y la facilidad de escribir código en ambos lenguajes.

Marco Teórico

Lenguajes Compilados: Los lenguajes compilados se traducen directamente a código máquina que el procesador puede ejecutar.

- **Ventajas:**
  - Rendimiento: Suelen ser más rápidos y eficientes en tiempo de ejecución.
  - Control: Los desarrolladores tienen más control sobre la gestión de memoria y el uso del CPU.
- **Ejemplos:** C, C++, Erlang, Haskell, Rust y Go.

Lenguajes Interpretados: Los lenguajes interpretados ejecutan línea por línea el programa y a la vez ejecutan cada comando.

- **Ventajas:**
  - **Flexibilidad:** Permite cambios en tiempo real sin necesidad de recompilación.
  - **Facilidad de desarrollo:** Más cómodo para escribir y depurar código.
  - **Ejemplos:** Python, Ruby, PHP y JavaScript.

Desarrollo

Para la realización de comparación de rendimiento entre un lenguaje de programación compilado y uno interpretado, elegí C++ como lenguaje compilado y Python como lenguaje

interpretado. Se realizó un programa que calcula la suma de los primeros N números naturales donde en este caso N = 1000000

Programa en C++

```cpp
#include <iostream>
#include <chrono>

int main() {
    const int N = 1000000;
    int sum = 0;

    auto start_time = std::chrono::high_resolution_clock::now();

    for (int i = 1; i <= N; ++i) {
        sum += i;
    }

    auto end_time = std::chrono::high_resolution_clock::now();
    auto duration = std::chrono::duration_cast<std::chrono::microseconds>(end_time - start_time);

    std::cout << "Suma: " << sum << std::endl;
    std::cout << "Tiempo tomado por c++ " << duration.count() << " microsegundos" << std::endl;

    return 0;
}
```

Resultado

```
/tmp/aFVcaAiSsc.o
Suma: 1784293664
Tiempo tomado por c++ 918 microsegundos
```

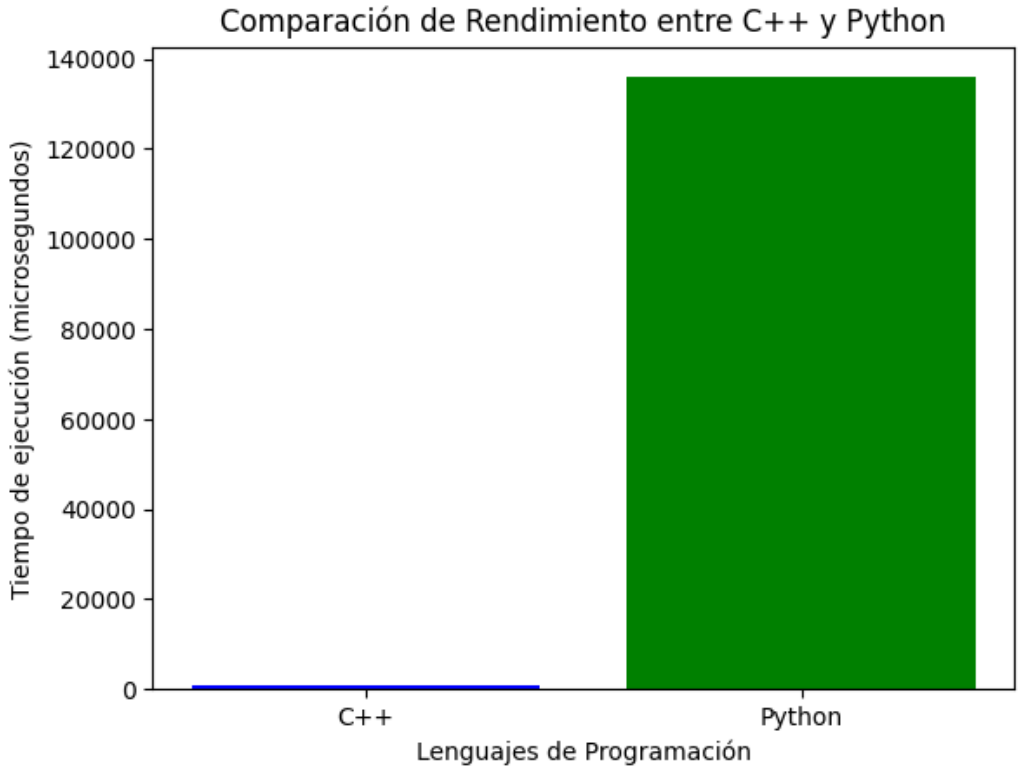
Programa en Python

```python
import time

N = 1000000
sum_value = 0

start_time = time.time()

for i in range(1, N + 1):
    sum_value += i

end_time = time.time()
duration = (end_time - start_time) * 1e6

print(f"Suma: {sum_value}")
print(f"Tiempo tomado por python : {duration} microsegundos")

```

Resultado

```
Suma: 500000500000
Tiempo tomado por python : 171632.7667236328 microsegundos
```

Comparación final



Resultados

Comparación

- **Velocidad:** C++ es más rápido debido a su naturaleza compilada, mientras que Python es más lento debido a la interpretación.
- **Recursos:** C++ utiliza menos recursos en tiempo de ejecución, pero Python es más fácil de desarrollar y depurar.

En resumen, C++ es excelente para aplicaciones de alto rendimiento, mientras que Python es ideal para prototipos rápidos y desarrollo ágil. Considera estos aspectos al elegir el lenguaje adecuado para tu proyecto.

Conclusiones

C++ muestra un rendimiento significativamente superior en términos de tiempo de ejecución en este caso específico, mientras que el tiempo de ejecución de Python es notablemente más alto en comparación con C++, lo que sugiere que la ejecución de operaciones simples puede llevar más tiempo en lenguajes interpretados.