Proyecto Final del Curso CC112

MORALES ROSSELL, CARLOS DANIEL

Diciembre, 2024

Abstract

Los lenguajes de programación juegan un papel fundamental en el desarrollo de software moderno, siendo Python y C++ dos de los más utilizados globalmente. Este informe busca hacer un análisis comparativo de las características fundamentales de ambos lenguajes, con el objetivo de identificar sus diferencias clave y contextos de aplicación óptimos. La metodología empleada consistió en realizar el análisis de una serie ejercicios realizados en el lenguaje python identificando sus ventajas con respecto a C++ así como aprendiendo una nueva sintaxis. Los resultados revelan que C++ destaca por su rendimiento superior y control preciso de memoria, siendo ideal para sistemas que requieren alta eficiencia, mientras que Python sobresale por su simplicidad sintáctica y extensa biblioteca de paquetes, haciéndolo más adecuado para desarrollo rápido de programas. Se concluye que la elección entre ambos lenguajes debe justificarse considerando factores como la velocidad de ejecución necesaria, la complejidad del desarrollo y el tiempo disponible para la implementación.

1 Introducción

Actualmente en el desarrollo de software, la elección del lenguaje de programación adecuado se ha convertido en una decisión crucial que puede determinar el éxito de un proyecto. Python y C++ se destacan como dos de los lenguajes más influyentes y ampliamente utilizados en la industria tecnológica.

C++, creado en 1983 por Bjarne Stroustrup, se ha consolidado como un lenguaje robusto y eficiente, especialmente en aplicaciones que requieren alto rendimiento y control preciso sobre los recursos del sistema. Por otro lado, Python, desarrollado por Guido van Rossum y lanzado en 1991, ha revolucionado el mundo de la programación con su filosofía de simplicidad y legibilidad, ganando popularidad especialmente en campos como la ciencia de datos y el aprendizaje automático.

La relevancia de comprender las diferencias entre estos dos lenguajes radica en su impacto directo sobre la productividad del desarrollador, el rendimiento de las aplicaciones y la facilidad de mantenimiento del código. Mientras C++ ofrece un control detallado sobre el hardware y una ejecución más rápida, Python privilegia la rapidez de desarrollo y la facilita el aprendizaje. Esta dualidad presenta un interesante campo de estudio para desarrolladores y estudiantes que buscan tomar decisiones informadas sobre qué lenguaje utilizar en sus proyectos.

2 Objetivos

2.1 Objetivo General

Analizar las diferencias fundamentales entre los lenguajes de programación Python y C++ para determinar sus fortalezas, debilidades y casos de uso óptimos en el desarrollo de software moderno.

2.2 Objetivos Específicos

1. Examinar las características sintácticas y estructurales de ambos lenguajes.
2. Comparar los sistemas de gestión de memoria y rendimiento de Python y C++.
3. Determinar los contextos y tipos de proyectos más apropiados para cada lenguaje.

3 Desarrollo

3.1 PARTE 1

3.1.1 Funciones en Python

Se implementó una función recursiva para calcular el enésimo término de una sucesión de Fibonacci. Para desarrollar este ejercicio se tomó en cuenta la idea conseguida al estudiar C++ de una función llamandose a sí misma, teniendo en consideración las diferencias en sintaxis entre ambos lenguajes de programación.

3.1.2 Cadenas en Python

Se trabajó una función que tome una cadena como parámetro y devuelva la cadena invertida.En este problema utilizamos el bucle for que en el lenguaje de programación Python se utiliza para recorrer los elementos de un objeto iterable (lista, tupla, conjunto, diccionario) y ejecutar un bloque de código.

3.1.3 Referencias y asignación dinámica en Python

Se creó una función que aceptara una lista de enteros, calculara su suma y devolviese el resultado, donde el tamaño de la lista se le fue solicitado al usuario, para su desarrollo recurrimos al uso de la palabra asignada While que similar a C++ representa la repetición de un bloque de código a nuestra conveniencia. Además utilizamos las sentencias Try y Except las cuales nos permitieron subsanar errores que pudiesen cometer los usuarios al rellenar los datos solicitados a través de condiciones dadas por nosotros.

3.1.4 Estructuras en Python

Se implementó un programa que permita representar la información de un estudiante: su nombre, edad y promedio. Así como una función para mostrar los detalles del estudiante. Se utilizó la sintaxis Class, ya que en el lenguaje de programación Python no existe la palabra designada Struct para la formación de estructuras.Adicionalemente conocimos las funciones fstring que nos ayudaron a la lectura y el ingreso de correcto de datos así como nos dieron un código mucho menos denso y más practico.

3.1.5 Archivos en Python

Para este tema se trabajó un programa que escribiera datos en un archivo de texto creado por nosotros. Se utilizaron las sentencias open y close para definir los límites de nuestro archivo de texto creado en nuestro código así como la sentencia write para introducirle datos al mismo, de manera muy similar a lo visto en C++.

3.1.6 Clases en Python

Se definió una clase Persona con atributos nombre y edad, así como un método para mostrar estos datos. De manera muy similar a C++ se implementó constructores y métodos de instancia a nuestra clase creada, teniendo en cuenta las variaciones en la sintaxis del lenguaje de programación trabajado.

3.2 Resultados

Logramos identificar diferencias en la sintaxis utilizada entre ambos lenguajes de programación, destacando Python por ser mucho más practico e intuitivo que C++, encontrándose entre las características más importantes, el uso de la indentación por parte de Python para definir los bloques de código que funcionan como unidad así como diferencias el inicio de funciones implementadas y el fin de estas.

Otro importante punto develado durante la elaboración de los códigos fue la diferencia en las que se programaban su variables mientras que para C++ era necesaria una declaración explícita de estas en Python no era necesario realizar dicha aclaración destacando que en este estaba instaurado

Aspecto	C++	Python
Sintaxis	Usa ; y {} para bloques	Usa indentación y es más simple
Variables	Tipado estático (declaración explícita)	Tipado dinámico (tipos inferidos)
Memoria	Gestión manual (new/delete)	Recolección de basura automática
Ejecución	Compilado (más rápido)	Interpretado (más lento)
POO	Acceso explícito (public/private)	Convenciones de nombres (_)
Bibliotecas	Instalación compleja	Fácil gestión con pip
Aprendizaje	Curva pronunciada	Más amigable para principiantes

Figure 1: Diferencias más importantes entre Python y C++

el manejado dinámico de sus variables en la memoria, pudiendo evitar el uso de punteros, punteros dobles y la escritura de símbolos de referencia pues ya estaban implícitos en sus variables.

Finalmente se notó que al momento de ejecutar nuestros programas entre más complejo era, la velocidad de ejecución era más lenta, mientras que en C++ nos podíamos permitir ser más aclarativos en el código y extendernos.

3.3 Conclusiones

Al momento de trabajar con Python se notó una mayor facilidad para entender el código y para intuir nuestro siguiente paso al desarrollar nuestro programa esto influyó a que la implementación del código se sienta más ligera por lo que este sería un excelente lenguaje para iniciarse en el mundo de la programación.

Sin embargo C++ permite la manipulación directa de hardware y memoria, lo que le añade poderosas características orientadas a objetos. Que para un nivel más intermedio puede venir muy bien.

Si bien Python pareciese ser más atractivo a la hora de elegir un lenguaje de programación, este cumple ciertas metas muy distintas a C++, por lo que no podemos devaluar o comparar al cien por ciento la utilidad mayor de una sobre otra. Y la mejor decisión sería conocer que objetivo tenemos para desarrollar nuestro programa pues esto nos permitirá elegir correctamente el método que se adapte mejor a nuestras necesidades.

References

[1] . JETIR January 2019, Volume 6, Issue

[2] . Conference: 2014 International Conference on Teaching and Learning in Computing and Engineering (LaTiCE)