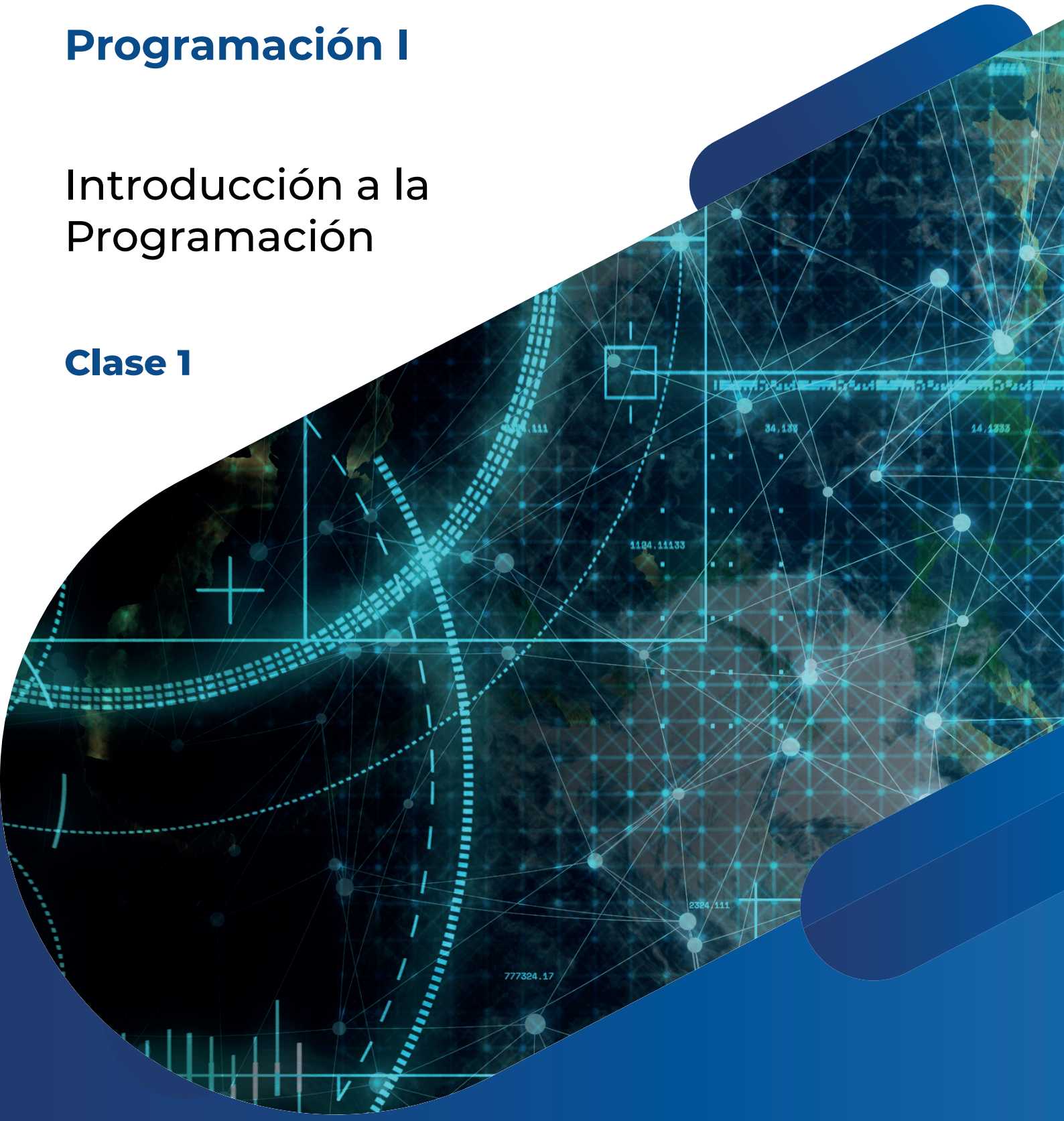
Programación I

Introducción a la Programación

Clase 1



Ingeniería en ciberseguridad

La excelencia no se improvisa

1. INTRODUCCIÓN DE LA CLASE

Esta primera clase se inicia con una cálida bienvenida al fascinante mundo de la programación, donde exploraremos los conceptos fundamentales que constituyen la base para desarrollar soluciones tecnológicas. En un diálogo cercano y motivador, conversaremos sobre qué es un algoritmo, qué implica tener un programa y la importancia del código fuente. Durante este recorrido, se plantearán ejemplos cotidianos y se realizará una analogía con procesos lógicos de la vida diaria para que el estudiante pueda identificar, de manera intuitiva, los elementos que componen la programación estructurada.

Además, en esta sesión se destacará la relevancia de conocer las diferencias entre compiladores e intérpretes, y se presentarán algunos de los lenguajes de programación más populares (como Python, Java y C++), subrayando sus particularidades y ámbitos de aplicación. De esta forma, se busca no solo que el estudiante asocie los conceptos teóricos, sino que también se sienta estimulado a indagar más allá de la teoría y a experimentar con ejemplos prácticos que lo inviten a profundizar en el campo de la ingeniería en ciberseguridad y gestión de TI.

Clase 1: Introducción a la Programación (Parte 1)

Reto # 1

Contenido de la Clase:

1. **Tema: Introducción a la Programación**

En esta sección se aborda de manera detallada y ampliada el contenido fundamental de la clase, abarcando desde la definición y ejemplificación de los conceptos básicos en programación hasta la comparación de métodos de traducción del código (compiladores e intérpretes) y la introducción a lenguajes de programación populares. Cada subtema se expone con ejemplos, explicaciones teóricas y actividades prácticas, con el objetivo de que el estudiante adquiera una comprensión profunda y aplicable de los fundamentos de la programación.

1.1. Conceptos Básicos de Programación

El primer bloque temático aborda la esencia de la programación. Se parte de la definición y aplicación de conceptos que constituyen el esqueleto de cualquier lenguaje:

Algoritmo: Un algoritmo es, en esencia, una secuencia finita y ordenada de instrucciones diseñadas para resolver un problema o ejecutar una tarea específica. Se le puede considerar como la "receta" que, paso a paso, guía la solución de un problema concreto. Por ejemplo, para preparar un café, se siguen instrucciones precisas:

calentar agua, agregar café a una taza, verter el agua caliente, remover y servir. De manera similar, en programación, un algoritmo establece la secuencia lógica necesaria para transformar entradas en salidas deseadas.

Para ilustrar este concepto, imaginemos un algoritmo que determine si un número es par o impar. La secuencia de pasos sería la siguiente:

ι. Recibir el número ingresado por el usuario.
ιι. Dividir el número por 2 y determinar el residuo.
ιιι. Si el residuo es 0, entonces el número es par; de lo contrario, es impar.
ιϖ. Mostrar el resultado al usuario.

Esta representación secuencial enfatiza la importancia de descomponer un problema en pasos simples y lógicos. El algoritmo, por tanto, se puede expresar de forma gráfica mediante un diagrama de flujo, lo que ayuda a visualizar el recorrido lógico y a identificar posibles mejoras o errores en la secuencia.

Una definición clara y precisa de cada paso en un algoritmo es crucial para evitar ambigüedades y facilitar su posterior implementación en un lenguaje de programación.

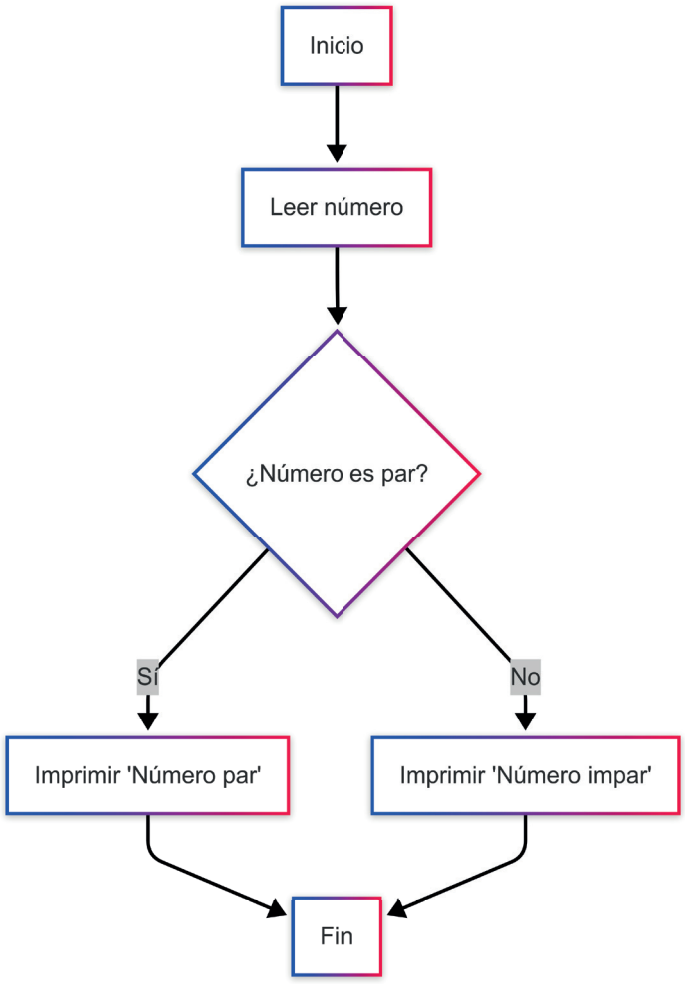


Imagen 1. Diagrama de flujo de un algoritmo que muestra la secuencia para determinar si un número es par o impar.

Damián Nicolalde Rodríguez. (2024). Diagrama de flujo de un algoritmo.

En el contexto de la programación, el algoritmo es el esqueleto que sustenta la solución de cualquier problema. La práctica de diseñar algoritmos fomenta el pensamiento crítico y el análisis lógico, habilidades esenciales para cualquier desarrollador. Se recomienda que los estudiantes practiquen la redacción de pseudocódigo, que es una representación informal del algoritmo en un lenguaje natural adaptado a estructuras de programación, lo que facilita la transición al código real.

Para una mejor comprensión revise el video:

- Título del enlace relacionado: Qué es un algoritmo y para qué se usa
- Descripción del enlace relacionado: Este video explica de manera sencilla y visual qué es un algoritmo y cómo se aplica en nuestra vida diaria
- Enlace: Qué es un algoritmo y para qué se usa | Computación y programación

Programa: Un programa es la implementación de uno o varios algoritmos utilizando un lenguaje de programación. Es un conjunto de instrucciones escritas que la computadora puede interpretar y ejecutar para realizar una tarea determinada. Por ejemplo, un programa que calcula el promedio de una serie de números implementa un algoritmo que suma los números y divide el total por la cantidad de elementos.

La estructura de un programa moderno se basa en la modularización, es decir, la división del código en partes o módulos que realizan tareas específicas. Esto permite que el programa sea más fácil de entender, mantener y escalar, ya que cada módulo puede desarrollarse y probarse de manera independiente. Además, un buen programa incorpora mecanismos de control de errores, entradas y salidas de datos y, en general, se organiza siguiendo principios de programación estructurada.

La modularización y la claridad en la estructura de un programa facilitan tanto la depuración como la futura ampliación del software, aspectos vitales en proyectos de gran escala.

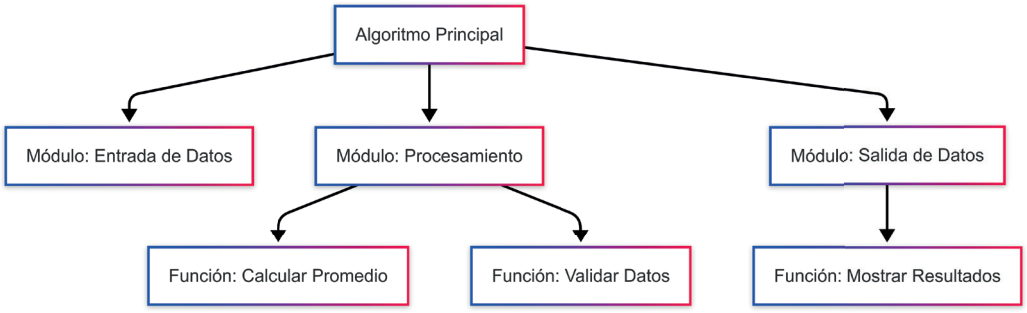


Imagen 2. Esquema gráfico que muestra la estructura de un programa dividido en módulos y funciones, destacando la relación entre el algoritmo principal y los submódulos.

Damián Nicolalde Rodríguez. (2024). Estructura de un programa.

En el diseño de un programa, es importante seguir un proceso iterativo que incluya la planificación, codificación, pruebas y optimización. Este proceso no solo garantiza que el programa funcione correctamente, sino que también promueve buenas prácticas de desarrollo y mantenimiento del código. El uso de comentarios y la adopción de estándares de codificación son aspectos complementarios que ayudan a que el programa sea comprensible para otros desarrolladores y para el propio autor en futuras revisiones.

Código fuente: El código fuente es la versión original del programa, escrita en un lenguaje de programación. Este código es legible por los humanos y sirve como base para la creación del programa ejecutable, ya sea a través de la compilación o la interpretación. La calidad del código fuente es determinante, ya que un código bien estructurado y documentado reduce la probabilidad de errores y facilita el mantenimiento a lo largo del tiempo.

Consideremos el siguiente ejemplo en Python, donde se imprime un mensaje en la consola:

```
# Ejemplo de código fuente en Python para imprimir “Hola Mundo”

print(“Hola Mundo”)
```

Este breve fragmento ilustra cómo una instrucción puede transformar la idea en una acción concreta. Sin embargo, en aplicaciones más complejas, el código fuente debe estar organizado en funciones, clases y módulos, siguiendo principios de diseño que favorezcan la reutilización y la legibilidad.

La calidad del código fuente se refleja en la claridad, consistencia y documentación del mismo, factores que reducen significativamente la aparición de errores.

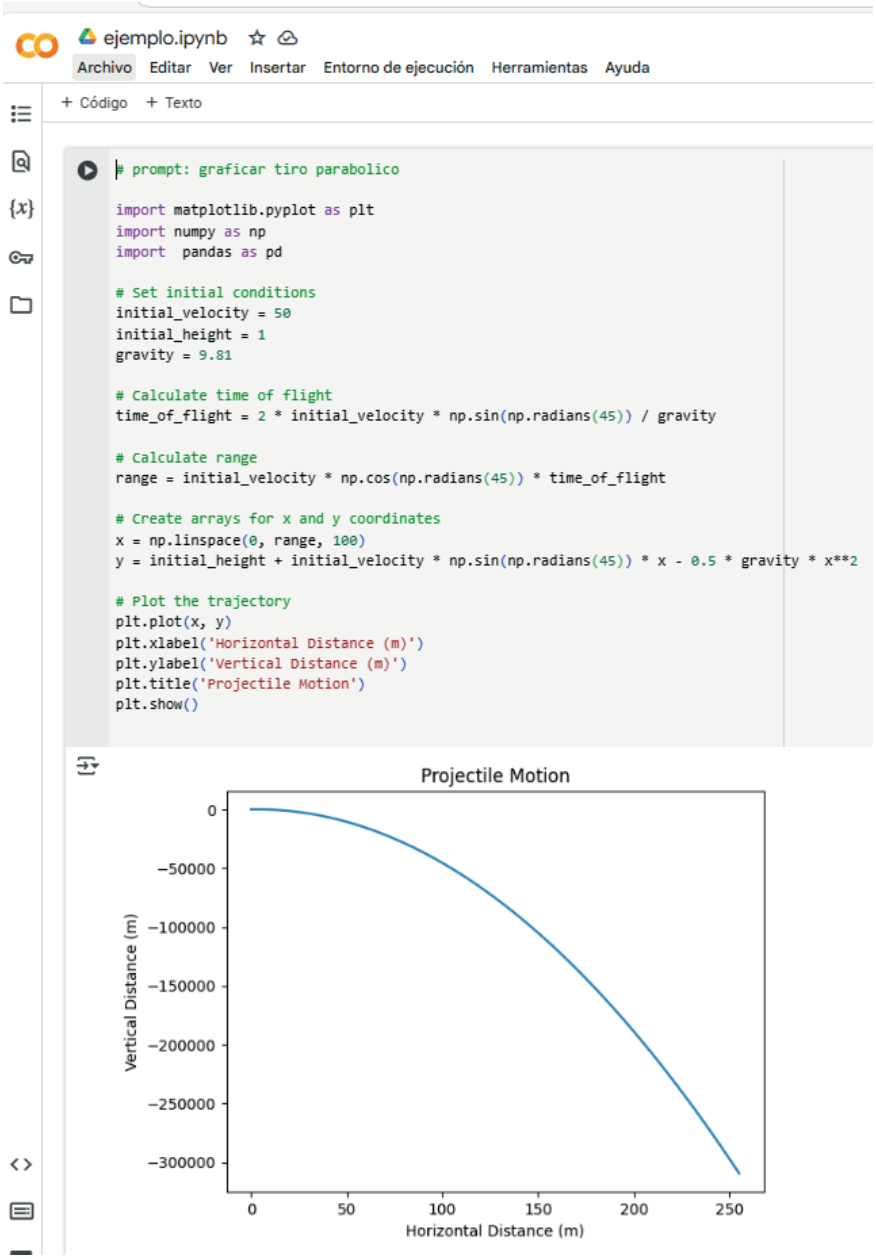


Imagen 3. Captura de pantalla de un entorno de desarrollo integrado (IDE) mostrando un ejemplo de código fuente en Python.

Damián Nicolalde Rodríguez. (2024). Código fuente en Python.

Además, el código fuente es el punto de partida para la optimización de la seguridad y eficiencia del programa, aspectos especialmente críticos en contextos de ciberseguridad. La adopción de buenas prácticas de programación, como el uso de nombres descriptivos para variables y funciones, la organización modular y la inclusión de comentarios explicativos, son esenciales para que el código pueda ser comprendido y mejorado por otros desarrolladores.

Para una mejor comprensión revise el video:

- Título del enlace relacionado: Qué es código fuente en el lenguaje de programación
- Descripción del enlace relacionado: Este video explica el concepto de código fuente en la programación y su importancia en el desarrollo de software.
- Enlace: 💥 QUE ES CODIGO FUENTE EN EL LENGUAJE DE PROGRAMACION 📚 DICCIONARIO DE ALGORITMOS ✅

1.2. Diferencias entre Compiladores e Intérpretes

En el proceso de transformación del código fuente a un programa ejecutable, existen dos métodos fundamentales: la compilación y la interpretación. Cada método tiene características propias que influyen en el rendimiento, la flexibilidad y la facilidad de depuración del software.

Compiladores: Un compilador es un programa especializado que traduce el código fuente completo a un lenguaje máquina, generando un archivo ejecutable independiente. Este proceso se realiza antes de la ejecución del programa, de manera que el archivo resultante puede ser distribuido y ejecutado sin necesidad del código fuente original. Entre las ventajas de los compiladores se encuentran la optimización del código, que mejora la velocidad de ejecución, y una mayor seguridad, puesto que el código binario resultante es menos susceptible a modificaciones no autorizadas.

Por ejemplo, en lenguajes como C++ o Java (en el caso de la compilación previa a la ejecución en la JVM), la compilación permite transformar programas complejos en aplicaciones eficientes y robustas. No obstante, un inconveniente es que cualquier modificación en el código fuente requiere una recompilación completa, lo que puede ralentizar el ciclo de desarrollo en proyectos de gran envergadura.

La compilación es fundamental en entornos donde el rendimiento es prioritario, aunque puede resultar menos flexible durante la fase de desarrollo.

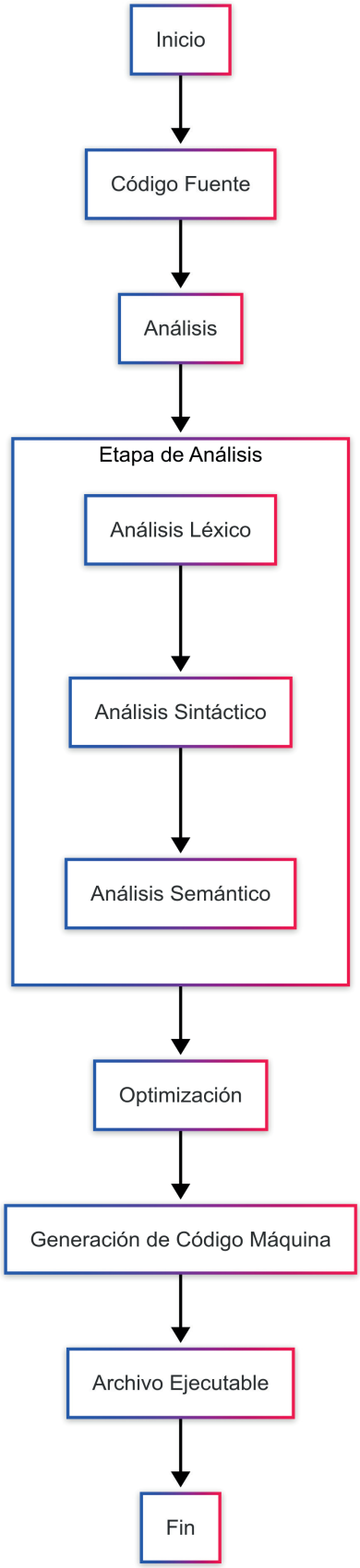


Imagen 4: Diagrama que ilustra el proceso de compilación, mostrando la conversión del código fuente en un

archivo ejecutable, pasando por etapas de análisis, optimización y generación de código máquina.

Damián Nicolalde Rodríguez. (2024). Proceso de compilación.

El proceso de compilación también permite detectar errores de sintaxis y optimizar el uso de recursos, lo cual es crucial para aplicaciones de misión crítica. Los compiladores modernos incorporan técnicas avanzadas de análisis y optimización que pueden transformar un código poco eficiente en una aplicación de alto rendimiento, aprovechando al máximo las capacidades del hardware.

Intérpretes: A diferencia de los compiladores, un intérprete traduce y ejecuta el código fuente línea por línea en tiempo real. Este método permite que el desarrollador observe de forma inmediata los resultados de cada instrucción, facilitando la identificación y corrección de errores durante el desarrollo. Los lenguajes interpretados, como Python, se benefician de esta característica, ya que permiten una mayor flexibilidad y rapidez en la fase de prototipado.

La ejecución interpretada ofrece la ventaja de poder modificar el código y ver los resultados sin necesidad de un proceso de compilación completo, lo que acelera el ciclo de pruebas y desarrollo. Sin embargo, la ejecución en tiempo real puede conllevar una penalización en el rendimiento, ya que cada línea debe ser traducida al momento de su ejecución.

La interpretación es ideal para entornos de desarrollo y enseñanza, donde la rapidez en la retroalimentación y la facilidad para corregir errores son factores determinantes.

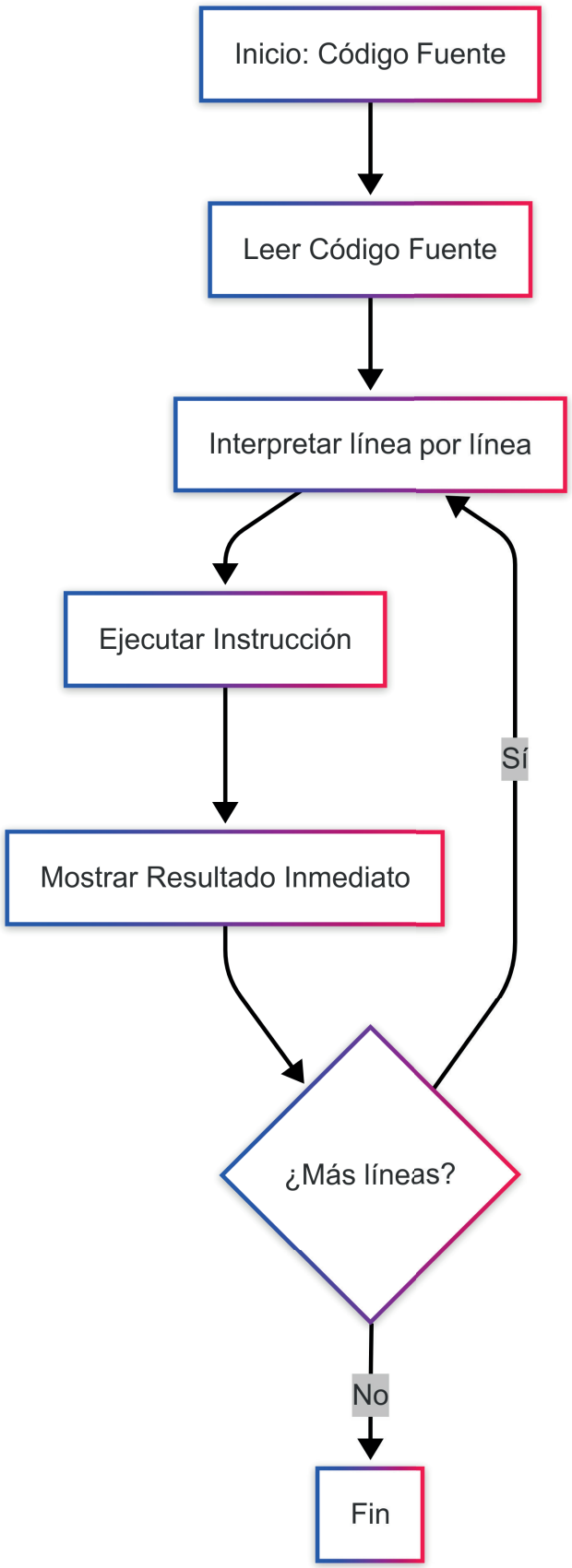


Imagen 5: Esquema ilustrativo del proceso de interpretación, en el que se muestra cómo el código fuente es

leído y ejecutado de manera secuencial, permitiendo la interacción inmediata con el entorno de desarrollo.

Damián Nicolalde Rodríguez. (2024). Proceso de interpretación. Creación de autor.

El uso de intérpretes favorece la experimentación y el aprendizaje, ya que los estudiantes pueden modificar y ejecutar fragmentos de código sin esperar largos tiempos de compilación. Esta característica es especialmente útil en un entorno educativo, donde la retroalimentación inmediata permite consolidar el aprendizaje de manera más efectiva.

1.3. Lenguajes de Programación Populares (Python, Java, C++)

Esta parte del contenido se centra en la presentación y análisis de tres lenguajes de programación que han tenido un impacto significativo en la industria y la educación. Cada uno de ellos posee características únicas que los hacen idóneos para diferentes tipos de aplicaciones y contextos de desarrollo.

Python: es un lenguaje de programación interpretado reconocido por su sintaxis limpia y legible. Su diseño favorece la productividad y la claridad, lo que lo convierte en una opción privilegiada tanto para principiantes como para expertos. Python se utiliza en múltiples áreas, tales como análisis de datos, inteligencia artificial, desarrollo web, automatización y scripting. Su extenso ecosistema de librerías y frameworks lo hace extremadamente versátil.

Ejemplo de código en Python:

```
# Ejemplo simple en Python para imprimir “Hola Mundo”

print(“Hola Mundo”)
```

La sencillez de Python permite a los desarrolladores concentrarse en la lógica del problema en lugar de en la sintaxis compleja, lo que resulta en un desarrollo más rápido y menos propenso a errores. Además, el soporte comunitario y la abundancia de recursos educativos hacen de Python una herramienta ideal para introducir a los estudiantes en la programación.

Imagen 6: Logo oficial de Python, ampliamente reconocido en la industria del software.

Python Software Foundation. (n.d.). Python Software Foundation. https://www.python.org/psf-landing/.

Durante el desarrollo de este curso, se explorarán ejemplos prácticos en los que se utilicen librerías populares como NumPy para cálculos numéricos y Pandas para la manipulación de datos, demostrando cómo Python puede resolver problemas complejos de manera eficiente y con un código legible.

Java: es un lenguaje de programación compilado que se ejecuta sobre la Máquina Virtual de Java (JVM), lo que le confiere una gran portabilidad y robustez. Su diseño orientado a objetos fomenta la reutilización del código y la implementación de soluciones escalables. Java es ampliamente utilizado en el desarrollo de aplicaciones empresariales, sistemas móviles (especialmente en Android) y aplicaciones de gran escala.

Ejemplo de código en Java:

```
// Ejemplo simple en Java para imprimir “Hola Mundo”
public class HolaMundo {
        public static void main(String[] args) {
                System.out.println(“Hola Mundo”);
        }
}
```

La fortaleza de Java reside en su capacidad para ofrecer un entorno seguro y gestionado, donde aspectos como la gestión de memoria y la seguridad son manejados automáticamente por la JVM. Este enfoque reduce la carga del desarrollador y permite centrarse en la lógica de la aplicación. Java permite aplicar conceptos fundamentales de la programación orientada a objetos, tales como la encapsulación, la herencia y el polimorfismo, demostrando cómo estos principios se implementan para crear aplicaciones robustas y mantenibles.

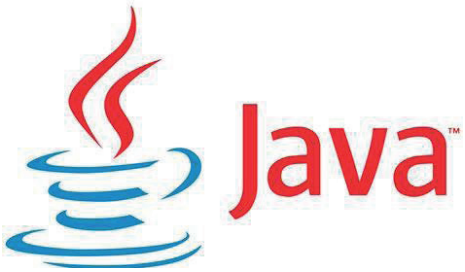


Imagen 7. Logo oficial de Java, representativo de su robustez y versatilidad.

Oracle Corporation. (n.d.). Java. https://www.java.com/es/.

Además, Java al igual que otros lenguajes de programación permite la gestión de excepciones y el manejo de errores, elementos cruciales en el desarrollo de aplicaciones críticas. Java ha sido utilizado para desarrollar soluciones empresariales de alto rendimiento.

C++: es un lenguaje de programación compilado que se distingue por su alto rendimiento y su capacidad para gestionar recursos a bajo nivel. Este lenguaje es ampliamente utilizado en el desarrollo de sistemas operativos, videojuegos y aplicaciones que requieren un control preciso sobre el hardware. C++ combina características de la programación estructurada con las ventajas de la programación orientada a objetos, lo que lo convierte en una herramienta poderosa para la creación de software de alto rendimiento.

Ejemplo de código en C++:

```
// Ejemplo simple en C++ para imprimir “Hola Mundo”
#include <iostream>
using namespace std;
int main() {
        cout << “Hola Mundo” << endl;
        return 0;
}
```

La capacidad de C++ para manipular directamente la memoria y optimizar el uso de recursos lo hace indispensable en aplicaciones donde el rendimiento es crítico. En próximas asignaturas, se explicarán conceptos clave como la gestión dinámica de memoria, punteros y referencias, así como la importancia de la eficiencia en el desarrollo de software. Los estudiantes aprenderán a identificar cuándo es apropiado utilizar C++ en función de las necesidades del proyecto, comparándolo con otros lenguajes de programación.

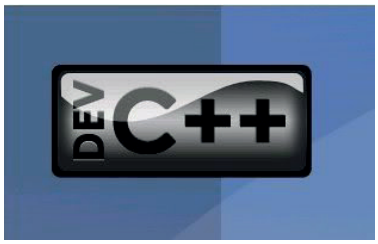


Imagen 8: Logo oficial de C++, símbolo que representa la eficiencia y el control de bajo nivel en el desarrollo de aplicaciones.

Bloodshed Software. (n.d.). Bloodshed Software. https://bloodshed.net/.

Se hará hincapié en que la elección del lenguaje de programación depende del contexto y los requisitos del proyecto. Mientras Python es ideal para prototipos rápidos y análisis de datos, Java y C++ ofrecen ventajas en términos de rendimiento y robustez en aplicaciones de gran escala. La discusión se complementará con ejemplos de proyectos reales y estudios de caso que demuestren cómo cada lenguaje aporta soluciones a problemas específicos en el ámbito de la ingeniería en ciberseguridad y la gestión de TI.

Referencias citadas en la Clase 1.

- https://elibro.puce.elogim.com/es/ereader/puce/230298
- https://puce.odilo.us/info/facil-aprendizaje-estructuras-de-datos-algoritmos-c-aprenda-facilmente-estructuras-de-datos-graficamente-03127232
- https://puce.odilo.us/info/aprende-c-en-un-fin-de-semana-03105596

Definición de los términos citados en la Clase 1.

Compilador: Herramienta que traduce el código fuente completo a lenguaje máquina, generando un archivo ejecutable independiente del código original.

Intérprete: Herramienta que traduce y ejecuta el código fuente línea por línea en tiempo real, facilitando la depuración y la experimentación durante el desarrollo.

Profundización Clase 1.

Recurso_profundizacion_clase1.docx

PUCE Virtual

La excelencia no se improvisa

síguenos