Trabajo Práctico

Identificación

Curso: Estructura de los Lenguajes

Objetivos

Este trabajo práctico tiene como objetivo que los estudiantes se familiaricen con diferentes lenguajes de programación. Utilizarán **Python, R, Ruby, C#** y un lenguaje adicional a elección para implementar soluciones a problemas que permitan comparar cómo se manejan conceptos como sentencias de asignación, control de flujo, tipos de datos, subrutinas y control de errores en cada lenguaje. Los resultados de aprendizaje esperados son:

- Explicar los conceptos y usos de los diferentes lenguajes de programación.
- Facilitar la comprensión de los temas relacionados con la implementación de los lenguajes de programación.
- Aplicar criterios y características para evaluar lenguajes de programación.
- Enunciar y describir características y propiedades de los lenguajes de programación.
- Comparar estructuras de diferentes lenguajes de programación.
- Identificar características de los lenguajes que los hacen especiales

Lenguajes a utilizar:

- Lenguajes obligatorios: Python, R, Ruby, C#.
- Lenguaje adicional: un lenguaje a elección del estudiante.

Problemas a resolver

Los estudiantes implementarán los siguientes problemas de programación:

Problema 1: Ordenamiento sobre Registros

Descripción: Implementar un algoritmo de ordenamiento, como **QuickSort** o **MergeSort**, pero aplicado a un conjunto de registros u objetos que contengan múltiples campos (por ejemplo, una lista de estudiantes con atributos como nombre, edad, y calificación).

- **Objetivo**: Observar cómo cada lenguaje maneja el ordenamiento de estructuras de datos más complejas, que contienen varios campos, sin depender de funciones predefinidas.
- **Instrucciones**:
  - Define un conjunto de registros o estructuras de datos con múltiples atributos. Por ejemplo, puedes crear una lista de personas con nombre, edad, y calificación.
  - Implementa el algoritmo de ordenamiento de manera manual para ordenar los registros en función de uno de sus atributos (por ejemplo, ordenar por edad o calificación).
  - No se deben utilizar funciones integradas de ordenamiento. Si el lenguaje cuenta con estas funciones, menciona su existencia en la documentación, explicando cómo podrían ser útiles y por qué representan una ventaja en el desarrollo práctico.
  - Observa cómo se maneja el acceso a los campos dentro de cada registro y el control de flujo en cada lenguaje.

Problema 2: Búsqueda Binaria en Estructuras de Datos Conglomeradas

Descripción: Implementar el algoritmo de **búsqueda binaria** para encontrar un registro específico en una lista ordenada de estructuras de datos complejas.

- **Objetivo**: Analizar cómo cada lenguaje maneja el control de flujo, subrutinas y tipos de datos más complejos sin recurrir a funciones preexistentes. Además, explora el control de errores en este contexto.
- **Instrucciones**:
  - Define una lista de registros u objetos con múltiples campos (por ejemplo, productos con nombre, precio, y cantidad), asegurándote de que la lista esté ordenada por uno de los atributos.
  - Implementa la búsqueda binaria para localizar un registro en función de uno de sus atributos (por ejemplo, buscar un producto específico por nombre).
  - La implementación debe ser manual, usando bucles o recursión según sea necesario.
  - Si el lenguaje ofrece funciones predefinidas para realizar búsquedas en estructuras complejas, explícalas en la documentación y analiza cómo facilitan el proceso en aplicaciones reales.
  - Considera agregar control de errores para gestionar casos en los que el registro no se encuentre o la lista esté vacía.

Problema 3: Multiplicación de Matrices

Descripción: Implementar la **multiplicación de dos matrices** de dimensiones compatibles.

- **Objetivo**: Examinar cómo cada lenguaje maneja estructuras de datos complejas y evaluar la habilidad de los estudiantes para implementar este algoritmo desde cero.
- **Instrucciones**:
  - Realiza la multiplicación de matrices manualmente, sin utilizar funciones integradas de matrices o álgebra lineal.
  - Si el lenguaje cuenta con funciones predefinidas para operaciones matriciales, menciona estas funciones en la documentación, indicando cómo podrían mejorar la eficiencia del desarrollo y por qué representan una ventaja.

- Asegúrate de incluir control de errores para verificar la compatibilidad de las dimensiones de las matrices.

Requerimientos de entrega

1. **Código**: El código fuente de cada tarea, con el nombre ProblemaX_Lenguaje.ext (por ejemplo, Problema1_Python.py).
2. **La entrega se hace a través de la plataforma**
3. **Documentación**: Incluir un informe (mínimo 5 páginas) en formato PDF con:
   - Descripción de la solución implementada en cada lenguaje.
   - Comparación entre los lenguajes, basada en los **criterios de Sebesta**: facilidad de uso, eficiencia, expresividad, etc.
   - Reflexión final sobre los lenguajes usados, indicando cuál resultó más interesante y por qué.
   - Usar una tipografía de tamaño 10, Utiliza una fuente clara y formal como **Times New Roman**, **Arial**, o **Calibri**. Alineación **justificada** para el texto principal. Interlineado de 1.5 líneas para el cuerpo del texto, lo que mejora la legibilidad.
   - Mantén un estilo de citas coherente en las referencias del trabajo (APA, MLA, IEEE)

ATENCION:

1. **El trabajo es individual**, deberá ser realizado por el Alumno (de principio a fin). Se permite hacer referencias a páginas o tutoriales utilizados, pero no un directo *copy paste* porque se espera una propia codificación y redacción.

2. **Copiar será castigado con la pérdida total de los trabajos/actividades. Además, se pasarán los antecedentes a la instancia correspondiente.**

3. La documentación del trabajo deberá estar en formato .PDF

4. Enviar los códigos fuentes de sus ejercicios en archivos leíbles .txt .ipynb (cuaderno de Jupyter) .py . java .c u otra. Nombre correctamente sus archivos. Además, debe acompañar un archivo leame.txt con las indicaciones necesarias para ejecutar los programas.
   - Preparar y enviar todos los archivos del trabajo en una carpeta comprimida y nombrarlo: **NOMBRE_APELLIDO**.

5. **Entrega intermedia: 30 de octubre** está planificada para dar un seguimiento del trabajo a ser presentado a fin de tomar medidas correctivas y recibir sugerencias.
6. Las horas de clases podrán ser utilizadas para hacer el trabajo práctico y realizar consultas.

Plazo

- **Entregas intermedias** de avances del trabajo, 30 de octubre
- **Entrega del trabajo finalizado,** 20 de noviembre

Evaluación

Criterios de Evaluación

- **Funcionalidad (20%)**: Solución funcional de los problemas.
- **Comparación y Reflexión (40%)**: Análisis crítico usando los criterios de Sebesta.
- **Documentación y Presentación (30%)**: Claridad del informe y presentación del código.
- **Uso de Paradigmas (10%)**: Observación de conceptos clave (asignación, control, tipos de datos, subrutinas, y errores) en cada lenguaje.