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

Atención: Si el alumno desea puede elegir los 5 lenguajes de su preferencia.

Lenguajes a utilizar:

- Lenguajes sugeridos: Python, R, Ruby, C# (pueden cambiar por otros).
- Lenguaje adicional: un lenguaje a elección del estudiante.

Problemas a resolver

Los estudiantes implementarán los siguientes problemas de programación:

Problema 1: Clasificación y filtrado de registros

Descripción:

a. Crear un conjunto de registros con al menos cuatro atributos (por ejemplo: nombre, edad, carrera, promedio académico).
b. Implementar un algoritmo de ordenamiento desde cero basado en uno de los atributos (por ejemplo, ordenar por promedio).
c. Luego, permitir filtrar los registros que cumplan una condición específica (por ejemplo, promedio mayor a un valor dado).

d. No utilizar funciones integradas de ordenamiento ni de filtrado.

e. Mencionar en la documentación si el lenguaje tiene funciones que podrían hacer estas tareas y su posible ventaja en desarrollo práctico.

f. Mostrar todos los registros ordenados y luego filtrados.

Objetivo: Observar cómo se manejan estructuras de datos compuestas, el acceso a atributos múltiples, y el control de flujo en distintos lenguajes.

Problema 2: Búsqueda secuencial con validación en estructuras de datos

Descripción:

a) Definir una lista de objetos con varios campos (por ejemplo, libros con título, autor, año y precio).

b) Implementar una búsqueda secuencial para encontrar registros que coincidan con un criterio de búsqueda flexible (por ejemplo: buscar por nombre o por rango de precio).

c) Agregar control de errores:

- Si no se encuentra el registro, mostrar mensaje adecuado.
- Si la lista está vacía, gestionar correctamente la situación.

d) No usar funciones integradas de búsqueda.

e) Documentar si el lenguaje ofrece mecanismos de búsqueda más eficientes o específicos para estructuras complejas.

Objetivo: Analizar el uso de control de flujo, funciones, tipos de datos complejos y manejo de errores.

Problema 3: Operaciones sobre matrices de registros numéricos

Descripción:

- Crear dos matrices de dimensiones compatibles (por ejemplo, 3x2 y 2x3) donde cada elemento es un número decimal. Las matrices pueden tener cualquier tamaño.
- Implementar manualmente:
  - La suma de dos matrices (de la misma dimensión).
  - La multiplicación de matrices (de dimensiones compatibles).
- No se permite el uso de funciones automáticas de matrices o álgebra lineal de librerías externas.
- Implementar verificación de compatibilidad de dimensiones antes de realizar operaciones.
- Documentar si existen funciones integradas en el lenguaje para este tipo de operaciones y su ventaja.

Objetivo: Evaluar la implementación de estructuras de datos complejas y el uso adecuado de control de flujo en operaciones matemáticas manuales.

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.
- Otros elementos para la comparación, referenciar correctamente el trabajo.
- 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 codificación y redacción propia.

2. **Copiar será castigado con la pérdida total de los trabajos. 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: 2 de Junio**.
6. Las horas de clases podrán ser utilizadas para hacer el trabajo práctico y realizar consultas.

Plazo

- **Entrega del trabajo finalizado**: 2 de Junio

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.