UNIVERSIDAD NACIONAL DE ASUNCIÓN

FACULTAD POLITÉCNICA

Problemas en Distintos Lenguajes de Programación

Alumno: Fernando David Fleitas Cáceres.

Materia: Estructura de los Lenguajes.

Carrera: Ingeniería en Informática.

Semestre: 6to.

AÑO: 2025.

San Lorenzo – Paraguay.

INDICE

SOLUCIÓN IMPLEMENTADA EN CADA LENGUAJE

Problema 1: Clasificación y filtrado de registros.

Para este problema utilizamos como ejemplo un conjunto de registros nombre, edad, carrera y promedio académico representados en la clase **Alumno,** utilizamos el algoritmo de **QuickSort** para ordenar por promedios en este caso utilizamos el atributo promedio del alumno. Se define una función que recorre la lista de alumnos utilizando un bucle For e imprime solo aquellos cuyo promedio es mayor o igual a un valor dado redefiniendo la función toString() correspondiente a cada lenguaje (por defecto imprime los de promedio mayor o igual a 4).

Lenguaje Python:

Python ya incluye funciones integradas que hacen lo mismo de forma más directa:

- **Para ordenar:** se puede usar `sort()` o `sorted()` con el parámetro key=lambda x: x.promedio.
- **Para filtrar:** se puede usar `filter()`, de esta forma para filtrar por promedio con una función lambda.

  ```
  list(filter(lambda alumno: alumno.promedio >= 4, listaAlumnos))
  ```

  Una función lambda en Python es una función anónima (sin nombre) que se define en una sola línea utilizando la palabra clave lambda.

Lenguaje JavaScript:

- JavaScript permite el ordenamiento por comparación:

  ```
  listaAlumnos.sort((a, b) => a.promedio - b.promedio);
  ```

- También permite el filtrado mediante la función:

  ```
  listaAlumnos.filter(alumno => alumno.promedio >= 4).forEach(a =>
  console.log(a.toString()));
  ```

Al igual que Python, JavaScript permite el uso de funciones lambda mediante las arrowFunction. Estas funciones se usan comúnmente en métodos como sort(), filter() y forEach() para simplificar el código.

Lenguaje C#:

Se implementó la clase Alumno, guardándolos en un List<T>, ordenamiento por QuickSort comparando por el atributo promedio. Recorrido lineal para imprimir la ocurrencia de un promedio dado.

Funciones integradas que se podrían haber usado:

- lista.OrderBy(alumno => alumno.Promedio) para ordenar.
- lista.Where(alumno => alumno.Promedio >= x) para filtrar.

Lenguaje Ruby:

Ruby ofrece métodos integrados muy potentes como:

- .sort_by o .sort para ordenar listas.
- .select para filtrar elementos.

Por ejemplo:

```
ordenados = lista_alumnos.sort_by(&:promedio)
filtrados = lista_alumnos.select { |a| a.promedio >= 4 }
```

Lenguaje Go:

- Se implementaron las listas mediante los slices, similar a los array en Python.
- Float64 para los promedios.
- En Go no hay clases, se utilizaron los structs.
- Se definen funciones sobre los structs.
- Go sí tiene el paquete sort para ordenar slices.

Problema 2: Búsqueda secuencial con validación en estructuras de datos.

Para este problema utilizamos como ejemplo un conjunto de registros con título, autor, año, precio y editorial representados en la clase **Libro**. Se definieron funciones que permiten realizar búsquedas secuenciales dentro de la lista de libros según distintos criterios: autor, título o rango de precios. Estas funciones recorren manualmente la lista comparando el campo correspondiente de cada objeto Libro con el criterio ingresado por el usuario. También se agregaron controles de errores, como mostrar un mensaje si la lista está vacía o si no se encuentra ninguna coincidencia. También se redefinieron los métodos toString() de cada lenguaje para dar una salida formateada de los datos.

Lenguaje Python:

- Para filtrar elementos en vez de hacerlo así:

  ```
  if(minimo <= libro.precio and libro.precio <= maximo):
  ```

  se pueden definir de vuelta con una función filter:

  ```
  list(filter(lambda libro: 50000 <= libro.precio <= 70000, listaLibros))
  ```

Lenguaje JavaScript:

- Se puede usar la función filter para simplificar el código:

  ```
  listaLibros.filter(libro=>
  libro.autor.toLowerCase().includes("sabato"));
  ```

Lenguaje C#:

- List<T>: Lista genérica para almacenar objetos Libro.
- ToString() override: Para imprimir el libro.
- IndexOf con StringComparison.OrdinalIgnoreCase: Comparación insensible a mayúsculas/minúsculas.
- En C#, existen mecanismos más eficientes para búsqueda:

  ```
  var resultados = listaLibros.Where(l => l.Autor.Contains("sabato",
  StringComparison.OrdinalIgnoreCase));
  ```

Lenguaje Ruby:

- Array de objetos (Array<Libro>): Lista genérica para almacenar instancias de la clase Libro.
- Se sobrescribe el método to_s para mostrar la información del libro de forma legible al imprimirlo.

- .downcase.include?: Comparación insensible a mayúsculas/minúsculas al buscar texto dentro de cadenas.

Lenguaje Go:

- Slice de structs ([]Libro): Uso de un slice para almacenar y manejar múltiples libros dinámicamente.
- Uso de strings.ToLower() junto con strings.Contains() para realizar búsquedas case-insensitive.

Problema 3: Operaciones sobre matrices de registros numéricos.

Se implementaron manualmente funciones para:

- Crear matrices con valores iniciales en cero.
- Imprimir matrices de forma legible.
- Sumar matrices elemento por elemento.
- Multiplicar matrices, de la forma clásica de multiplicación matricial.

Estas operaciones permiten manipular matrices sin usar funciones integradas, y sirven para entender el manejo de listas anidadas, ciclos anidados y validación de dimensiones compatibles.

Lenguaje Python:

Se definieron una lista de listas para representar matrices y funciones de crearMatriz(), imprimirMatriz(), sumarMatrices(), multiplicarMatrices().

Python ofrece formas más directas para trabajar con matrices mediante el uso de librerías especializadas como NumPy, que permite operar con matrices de forma más directa y rápida:

- Crear matrices en NumPy:

```
import numpy as np
A = np.zeros((2, 3))  # Crea una matriz de 2x3 con ceros
```

- Sumar matrices:

```
C = A + B  # Suma directa de matrices
```

- Multiplicación:

```
C = np.dot(A, B)  # Producto matricial
```

Estas funciones permiten la operación de matrices de forma más eficiente, segura y permiten simplificar el código.

Lenguaje JavaScript:

- Creación dinámica de matrices (arreglos bidimensionales) con dimensiones arbitrarias, una lista con arrays dentro.
- Impresión formateada de matrices en consola.
- Verificación de compatibilidad dimensional para suma y multiplicación.
- Suma de matrices de igual tamaño, recorriendo cada elemento y sumándolo.
- Multiplicación de matrices compatibles usando tres bucles anidados para calcular cada elemento del resultado.
- Manejo de error cuando las dimensiones no son compatibles para multiplicación.

JavaScript tiene algunas librerías que facilitan el manejo de matrices como por ejemplo:

- **math.js:** Biblioteca muy popular que incluye operaciones con matrices, incluyendo suma, multiplicación, transposición, determinantes, etc. Ejemplo: math.add(A, B), math.multiply(A, B).

Lenguaje C#:

- Creación de matrices dinámicas con arreglos multidimensionales (double[,]).
- Función para imprimir matrices con formato decimal fijo.

Existen librerías que facilitan el manejo de operaciones matriciales:
https://numerics.mathdotnet.com/

- **Librería Math.NET Numerics:** Una de las librerías más completas para álgebra lineal y matemáticas en C#, que ofrece tipos de matrices y operaciones como suma, multiplicación, transposición, inversa, descomposiciones, etc. Ejemplo: Matrix<double>.Build.DenseOfArray(...) y luego usar .Add(), .Multiply().

Lenguaje Ruby:

- En ruby se pude usar la clase Matrix (incluida en la biblioteca estándar) para operaciones con matrices.

```
require 'matrix'
a = Matrix[[1.0, 2.0], [3.0, 4.0]]
b = Matrix[[5.0, 6.0], [7.0, 8.0]]
puts a + b
puts a * b
```

Lenguaje Go:

Las matrices están representadas como slices de slices ([][]float64), y se realizan operaciones de:

- Creación de matriz vacía con ceros (crearMatriz).
- Impresión de matrices (imprimirMatriz).
- Suma de matrices del mismo tamaño (sumarMatrices).
- Multiplicación de matrices compatibles en dimensión (multiplicarMatrices).

COMPARACIÓN DE LENGUAJES SEGÚN CRITERIOS DE EVALUACIÓN DEL SEBESTA

1. **Legibilidad (Readability)**
   - **Python**: Excelente. Sintaxis limpia, clara y muy parecida al lenguaje natural.
   - **C#**: Buena. Sintaxis estructurada, aunque más verbosa.
   - **JavaScript**: Aceptable. Flexible, pero la libertad excesiva puede generar confusión.
   - **Go**: Buena. Sintaxis simple y consistente, aunque menos expresiva.
   - **Ruby**: Muy buena. Sintaxis concisa y legible, similar a Python.
2. **Facilidad de escritura (Writability)**
   - **Python**: Muy alta. Requiere pocas líneas de código para hacer tareas comunes.
   - **JavaScript**: Alta. Es flexible y tiene muchas librerías disponibles.
   - **Ruby**: Alta. Expresiva y fácil de escribir.
   - **C#**: Media. Estructurado, pero algo más extenso por su formalidad.
   - **Go**: Media. Menos expresivo, pero muy claro y consistente.
3. **Confiabilidad**
   - **Go**: Alta. Uso estricto de tipos y manejo explícito de errores.
   - **C#**: Alta. Tipado fuerte y herramientas robustas.
   - **Python**: Media. El tipado dinámico puede generar errores inesperados.
   - **JavaScript**: Media a baja. El tipado débil facilita errores difíciles de detectar.
   - **Ruby**: Media. Similar a Python en cuanto a errores en tiempo de ejecución.
4. **Costo**
   - **Python, Go, JavaScript y Ruby**: Gratuitos, de código abierto y multiplataforma.
   - **C#**: Gratuito si se usa .NET Core, pero algunas herramientas profesionales son pagas (como algunas ediciones de Visual Studio).
5. **Eficiencia (tiempo de ejecución)**
   - **Go**: Muy eficiente. Compilado, con excelente manejo de concurrencia.
   - **C#**: Eficiente. Gracias al compilador JIT y al ecosistema .NET.
   - **JavaScript**: Eficiencia media. Mejorada por motores como V8.
   - **Python**: Baja eficiencia. Interpretado y más lento en tareas exigentes.
   - **Ruby**: Menor eficiencia. También interpretado y más lento que Python en muchos casos.
6. **Portabilidad**

   Todos los lenguajes mencionados son portables, aunque Go y JavaScript requieren menos configuración en general.
7. **Generalidad**

   Python, JavaScript, C# y Go son de propósito general. Ruby también, aunque hoy en día se usa más en web.

REFLEXIÓN FINAL

Realizar la implementación de un mismo problema en cinco lenguajes diferentes fue un desafío interesante que me permitió comprender cómo cada lenguaje aborda los mismos problemas desde distintas perspectivas. Aunque comparten estructuras similares, cada uno tiene su propia forma de expresar las soluciones, lo que exige adaptarse constantemente a su sintaxis, estilo y características.

De todos los lenguajes que utilicé, JavaScript fue el que más me gustó. Su flexibilidad para escribir código de forma rápida y su presencia tanto en el frontend como en el backend lo hacen muy atractivo. A pesar de no ser el más estricto, me pareció interesante por la libertad que ofrece a la hora de programar.

Sin embargo, reconozco que Python fue el más fácil de aprender y usar. Su sintaxis clara y directa facilitó mucho las implementaciones y me permitió enfocarme más en la lógica que en la estructura del lenguaje. En resumen, ambos lenguajes fueron destacables a su manera: JavaScript por su versatilidad, y Python por su simplicidad.

REFERENCIAS

- Sebesta, R. W. (n.d.). *Concepts of programming languages*. [Libro].
- Microsoft. (n.d.). *C# documentation*. Microsoft Learn. https://learn.microsoft.com/en-us/dotnet/csharp/
- Go. (n.d.). *The Go programming language*. https://go.dev/
- Python Software Foundation. (n.d.). *Python 3 documentation*. https://docs.python.org/3/
- Mozilla. (n.d.). *JavaScript guide*. MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript
- Ruby Language. (n.d.). *Documentation*. https://www.ruby-lang.org/en/documentation/
- OnlineGDB. (n.d.). *Online compiler and debugger*. https://www.onlinegdb.com/
- Node.js Foundation. (n.d.). *Node.js*. https://nodejs.org/en
- TutorialsPoint. (n.d.). *TutorialsPoint – Learn programming online*. https://www.tutorialspoint.com/
- W3Schools. (n.d.). *W3Schools online web tutorials*. https://www.w3schools.com/