Práctica de Recursividad de Pila con Listas en C#

Escuela de Ingeniería en Computadores
Instituto Tecnológico de Costa Rica

Algoritmos y Estructuras de Datos I

Profesores: M.Sc. Jeff Schmidt Peralta y M.Sc. Jason Leitón Jiménez y M.Sc. Leonardo Araya Martinez
Estudiante: Steven Aguilar Alvarez

Introducción

La recursividad de pila aplicada a listas representa una técnica fundamental en programación que permite manipular estructuras de datos de manera elegante y concisa. Esta práctica explora diferentes estrategias para procesar listas utilizando llamadas recursivas.

1. Ejercicios de Programación

1. **División por Umbral** Escriba un método recursivo SplitListByThreshold(threshold, list) que reciba un dígito y una lista y obtenga dos listas, la primera compuesta por los dígitos mayores o iguales al dígito dado y la segunda compuesta por los dígitos menores al dígito dado:
   - SplitListByThreshold(5,[1,0,2,9,0,9,9]) → ([9,9,9], [1,0,2,0])
   - SplitListByThreshold(8,[1,0,2,6,0]) → ([], [1,0,2,6,0])
2. **Calificación de Competencia** Escriba un método recursivo CalculateScore(scores) que reciba una lista con calificaciones de 10 jueces (escala 1-100), elimine la más alta y la más baja y retorne el promedio de las restantes:
   - CalculateScore([6,8,8,8,8,8,8,10,8,8]) → 8
3. **Agrupación por Paridad** Escriba un método recursivo SplitByParity(list) que reciba una lista de números enteros y retorne dos listas, una con los números pares y otra con los impares, manteniendo el orden original:
   - SplitByParity([1,4,6,7,3,8]) → ([4,6,8], [1,7,3])
   - SplitByParity([2,2,2,1,1]) → ([2,2,2], [1,1])
4. **Duplicados Consecutivos** Escriba un método recursivo CountConsecutiveDuplicates(list) que reciba una lista de números y cuente cuántas veces aparecen elementos duplicados de manera consecutiva:
   - CountConsecutiveDuplicates([1,2,2,3,3,3,4,4,2]) → 3
   - CountConsecutiveDuplicates([1,2,3,4,5]) → 0
5. **Sublistas Alternadas** Escriba un método recursivo AlternateSublists(list) que reciba una lista y retorne dos sublistas, tomando elementos de manera alternada. La primera sublista debe contener los elementos en posiciones pares (empezando desde 0) y la segunda los elementos en posiciones impares:
   - AlternateSublists([1,2,3,4,5,6]) → ([1,3,5], [2,4,6])
   - AlternateSublists([8,4,2,1]) → ([8,2], [4,1])

Observaciones Finales

Los ejercicios deben implementarse utilizando exclusivamente recursividad de pila en C#. Es fundamental mantener un control adecuado de la pila de ejecución mediante el uso apropiado de casos base y llamadas recursivas.