# Práctica de Recursividad de Pila con Arrays 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 arrays constituye una poderosa herramienta para manipular arreglos de manera elegante y eficiente. Esta práctica explora diferentes estrategias de procesamiento recursivo.

## 1. Ejercicios de Programación

1. **Reemplazo de Primera Ocurrencia** Escriba un método recursivo ReplaceFirstOccurrence(array, element) que reciba una lista y un elemento y cambie por un cero la primera aparición de ese elemento en la lista:
   - ReplaceFirstOccurrence([0,5,1,2,5,3,4], 5) → [0,0,1,2,5,3,4]
   - ReplaceFirstOccurrence([0,5,1,2,5,3,4], 8) → [0,5,1,2,5,3,4]
2. **Reemplazo de Todas las Ocurrencias** Escriba un método recursivo ReplaceAllOccurrences(array, element) que reciba una lista y un elemento y cambie por un cero todas las apariciones de ese elemento en la lista:
   - ReplaceAllOccurrences([0,5,1,2,5,3,4], 5) → [0,0,1,2,0,3,4]
   - ReplaceAllOccurrences([0,5,1,2,5,3,4], 8) → [0,5,1,2,5,3,4]
3. **Separación por Paridad** Escriba un método recursivo SeparateByParity(array) que reciba una lista no nula y devuelva otra compuesta por dos sublistas, la primera con elementos pares y la segunda con impares:
   - SeparateByParity([12,14,16,20,30]) → [[12,14,16,20,30], []]
   - SeparateByParity([12,1,20,3]) → [[12,20],[1,3]]
4. **Búsqueda de Patrón Binario** Escriba un método recursivo FindBinaryPattern(array) que reciba una lista y retorne true si encuentra el patrón binario de 10 (1010) en secuencia (no necesariamente consecutiva) en el arreglo:
   - FindBinaryPattern([2,34,2,12,10,0,3,10,10]) → false
   - FindBinaryPattern([2,45,1,5,2,5,1,1,0,1,0,3]) → true
5. **Reemplazo por Paridad** Escriba un método recursivo ReplacePairOdd(array) que reciba un array y reemplace los números pares por 1 y los impares por 0:
   - ReplacePairOdd([1,2,3,4,5]) → [0,1,0,1,0]
   - ReplacePairOdd([2,4,6,8,10]) → [1,1,1,1,1]

## 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.