TALLER – PILAS EN C#

**JIMENA SOFIA VALIENTE BLANDÓN**

# Invertir una cadena Descripción:

Pide al usuario una cadena y utiliza una pila para invertir el orden de los caracteres.

# Objetivo:

Practicar el uso básico de Push() y Pop().

# Pista:

Recorre la cadena, mete cada carácter en la pila y luego sácalos para formar la cadena invertida.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static void Main(){**

**Console.Write("Ingresa una cadena de texto: ");**

**string cadenaOriginal = Console.ReadLine();**

**Stack<char> pila = new Stack<char>();**

**foreach (char c in cadenaOriginal){**

**pila.Push(c);**

**}**

**string cadenaInvertida = "";**

**while (pila.Count > 0){**

**cadenaInvertida += pila.Pop();**

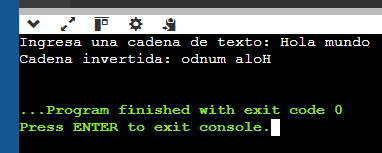
**}**

**Console.WriteLine("Cadena invertida: " + cadenaInvertida);**

**}**

**}**

**Resultado:**

****

# Verificar paréntesis balanceados Descripción:

Dada una expresión con paréntesis ()[]{}, determina si están correctamente balanceados.

# Objetivo:

Aplicar la pila para analizar estructuras anidadas.

# Pista:

Cada vez que se encuentre un símbolo de apertura, se apila; cuando se encuentre uno de cierre, se desapila y se verifica el par correspondiente.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static bool EstaBalanceada(string expresion){**

**Stack<char> pila = new Stack<char>();**

**foreach (char c in expresion){**

**// Apilar símbolos de apertura**

**if (c == '(' || c == '[' || c == '{'){**

**pila.Push(c);**

**}**

**// Verificar símbolos de cierre**

**else if (c == ')' || c == ']' || c == '}'){**

**if (pila.Count == 0) return false;**

**char tope = pila.Pop();**

**if ((c == ')' && tope != '(') ||**

**(c == ']' && tope != '[') ||**

**(c == '}' && tope != '{'))**

**{**

**return false;**

**}**

**}**

**}**

**return pila.Count == 0;**

**}**

**static void Main(){**

**Console.Write("Ingresa una expresión con paréntesis: ");**

**string expresion = Console.ReadLine();**

**if (EstaBalanceada(expresion)){**

**Console.WriteLine("La expresión está balanceada.");**

**}else{**

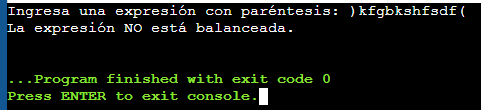
**Console.WriteLine("La expresión NO está balanceada.");**

**}**

**}**

**}**

**Resultado:**



# Convertir número decimal a binario

**Descripción:**

Convierte un número entero decimal a binario usando una pila.

# Objetivo:

Usar pilas para transformar datos paso a paso.

# Pista:

Divide el número entre 2, apila los restos, y luego sácalos para formar el número binario.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static string DecimalABinario(int numero){**

**Stack<int> pila = new Stack<int>();**

**while (numero > 0){**

**int residuo = numero % 2;**

**pila.Push(residuo);**

**numero /= 2;**

**}**

**string binario = "";**

**while (pila.Count > 0){**

**binario += pila.Pop();**

**}**

**return binario == "" ? "0" : binario;**

**}**

**static void Main(){**

**Console.Write("Ingresa un número decimal: ");**

**int numero = int.Parse(Console.ReadLine());**

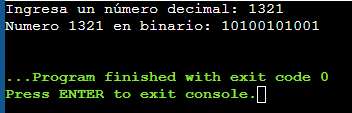
**string resultado = DecimalABinario(numero);**

**Console.WriteLine($"Numero {numero} en binario: {resultado}");**

**}**

**}**

**Resultado:**



# Evaluar expresión postfija (notación polaca inversa) Descripción:

Evalúa expresiones como "5 3 + 8 2 - \*" usando una pila.

# Objetivo:

Practicar manipulación avanzada de datos en la pila.

# Pista:

Cuando se lea un número, apilarlo; cuando se lea un operador, desapilar dos operandos, aplicar la operación y apilar el resultado.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static int EvaluarPostfija(string expresion){**

**Stack<int> pila = new Stack<int>();**

**string[] tokens = expresion.Split(' ');**

**foreach (string token in tokens){**

**if (int.TryParse(token, out int numero)){**

**pila.Push(numero);**

**}else{**

**if (pila.Count < 2){**

**throw new InvalidOperationException("Expresión inválida: faltan operandos.");**

**}**

**int b = pila.Pop();**

**int a = pila.Pop();**

**int resultado = 0;**

**switch (token){**

**case "+":**

**resultado = a + b;**

**break;**

**case "-":**

**resultado = a - b;**

**break;**

**case "\*":**

**resultado = a \* b;**

**break;**

**case "/":**

**resultado = a / b;**

**break;**

**default:**

**throw new InvalidOperationException("Operador desconocido: " + token);**

**}**

**pila.Push(resultado);**

**}**

**}**

**if (pila.Count != 1){**

**throw new InvalidOperationException("Expresión inválida: sobran operandos.");**

**}**

**return pila.Pop();**

**}**

**static void Main(){**

**Console.Write("Ingresa una expresión postfija (ej. 5 3 + 8 2 - \*): ");**

**string expresion = Console.ReadLine();**

**try{**

**int resultado = EvaluarPostfija(expresion);**

**Console.WriteLine("Resultado: " + resultado);**

**}**

**catch (Exception ex){**

**Console.WriteLine("Error: " + ex.Message);**

**}**

**}**

**}**

**Resultado:**



# Deshacer (Undo) en un editor de texto simulado Descripción:

Simula un editor con las operaciones: escribir, deshacer.

# Objetivo:

Aplicar pilas en el manejo de estados.

# Pista:

Cada acción de escritura se apila. Al hacer “deshacer”, se desapila la última acción.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static void Main(){**

**Stack<string> pila = new Stack<string>();**

**string textoActual = "";**

**while (true)**

**{**

**Console.WriteLine("\nTexto actual: \"" + textoActual + "\"");**

**Console.WriteLine("Opciones: [1] Escribir [2] Deshacer [3] Salir");**

**Console.Write("Selecciona una opción: ");**

**string opcion = Console.ReadLine();**

**if (opcion == "1")**

**{**

**Console.Write("Escribe algo: ");**

**string nuevoTexto = Console.ReadLine();**

**pila.Push(textoActual);**

**textoActual += nuevoTexto;**

**}**

**else if (opcion == "2")**

**{**

**if (pila.Count > 0)**

**{**

**textoActual = pila.Pop();**

**Console.WriteLine("Última acción deshecha.");**

**}**

**else**

**{**

**Console.WriteLine("No hay acciones para deshacer.");**

**}**

**}**

**else if (opcion == "3")**

**{**

**Console.WriteLine("Saliendo del editor...");**

**break;**

**}**

**else**

**{**

**Console.WriteLine("Opción inválida.");**

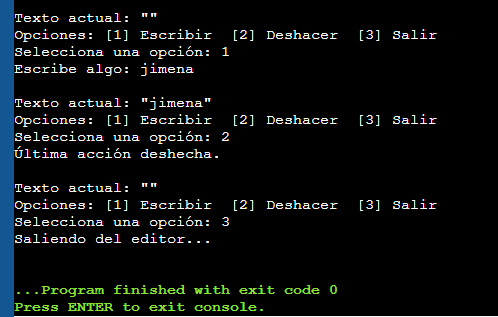
**}**

**}**

**}**

**}**

**Resultado:**



# Comprobar si una palabra es palíndroma Descripción:

Verifica si una palabra se lee igual al derecho y al revés usando una pila.

# Objetivo:

Usar pilas para comparar secuencias.

# Pista:

Apila los caracteres, luego desapílalos y compara con la cadena original.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static bool EsPalindroma(string palabra){**

**Stack<char> pila = new Stack<char>();**

**foreach (char c in palabra){**

**pila.Push(c);**

**}**

**foreach (char c in palabra){**

**if (c != pila.Pop()){**

**return false;**

**}**

**}**

**return true;**

**}**

**static void Main(){**

**Console.Write("Ingresa una palabra: ");**

**string palabra = Console.ReadLine();**

**if (EsPalindroma(palabra)){**

**Console.WriteLine("La palabra es palíndroma.");**

**}else{**

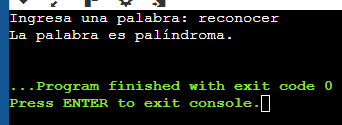
**Console.WriteLine("La palabra NO es palíndroma.");**

**}**

**}**

**}**

**Resultado:**



# Revertir el orden de palabras en una oración Descripción:

Invierte el orden de las palabras en una oración sin invertir las letras de cada palabra.

# Ejemplo:

Entrada: "C# es genial" → Salida: "genial es C#"

# Pista:

Divide la oración en palabras, apílalas y sácalas en orden inverso.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static string InvertirPalabrasConPila(string oracion){**

**Stack<char> pilaCaracteres = new Stack<char>();**

**Stack<string> pilaPalabras = new Stack<string>();**

**string palabra = "";**

**foreach (char c in oracion){**

**if (c != ' '){**

**pilaCaracteres.Push(c);**

**}else{**

**palabra = "";**

**while (pilaCaracteres.Count > 0)**

**{**

**palabra = pilaCaracteres.Pop() + palabra;**

**}**

**pilaPalabras.Push(palabra);**

**}**

**}**

**palabra = "";**

**while (pilaCaracteres.Count > 0){**

**palabra = pilaCaracteres.Pop() + palabra;**

**}**

**if (palabra != "") pilaPalabras.Push(palabra);**

**string resultado = "";**

**while (pilaPalabras.Count > 0){**

**resultado += pilaPalabras.Pop();**

**if (pilaPalabras.Count > 0)**

**resultado += " ";**

**}**

**return resultado;**

**}**

**static void Main(){**

**Console.Write("Ingresa una oración: ");**

**string oracion = Console.ReadLine();**

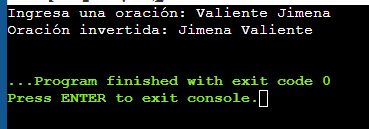
**string invertida = InvertirPalabrasConPila(oracion);**

**Console.WriteLine("Oración invertida: " + invertida);**

**}**

**}**

**Resultado:**



# Evaluar expresión infija simple Descripción:

Evalúa una expresión aritmética infija con paréntesis, como "3 + (2 \* 4)".

# Objetivo:

Usar dos pilas: una para operandos y otra para operadores.

# Pista:

Implementa el algoritmo del “Shunting Yard” (Dijkstra).

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static void Main(){**

**Console.WriteLine("Evaluador de expresión infija con paréntesis");**

**Console.WriteLine("Ejemplo: 3 + (2 \* 4)");**

**Console.Write("Ingresa la expresión: ");**

**string expresion = Console.ReadLine();**

**try{**

**double resultado = EvaluarInfija(expresion);**

**Console.WriteLine("El resultado es: " + resultado);**

**}**

**catch (Exception e){**

**Console.WriteLine("Error: " + e.Message);**

**}**

**}**

**static double EvaluarInfija(string expresion){**

**Stack<double> operandos = new Stack<double>();**

**Stack<char> operadores = new Stack<char>();**

**for (int i = 0; i < expresion.Length; i++){**

**char c = expresion[i];**

**// Ignorar espacios**

**if (c == ' ')**

**continue;**

**// Si es número, leerlo completo**

**if (char.IsDigit(c)){**

**string num = "";**

**while (i < expresion.Length && (char.IsDigit(expresion[i]) || expresion[i] == '.')){**

**num += expresion[i];**

**i++;**

**}**

**i--;**

**operandos.Push(double.Parse(num));**

**}**

**// Si es paréntesis de apertura**

**else if (c == '('){**

**operadores.Push(c);**

**}**

**// Si es paréntesis de cierre**

**else if (c == ')'){**

**while (operadores.Peek() != '('){**

**double resultado = AplicarOperacion(operandos, operadores);**

**operandos.Push(resultado);**

**}**

**operadores.Pop(); // quitar '('**

**}**

**else if (EsOperador(c)){**

**while (operadores.Count > 0 && Prioridad(operadores.Peek()) >= Prioridad(c)){**

**double resultado = AplicarOperacion(operandos, operadores);**

**operandos.Push(resultado);**

**}**

**operadores.Push(c);**

**}**

**}**

**while (operadores.Count > 0){**

**double resultado = AplicarOperacion(operandos, operadores);**

**operandos.Push(resultado);**

**}**

**return operandos.Pop();**

**}**

**static bool EsOperador(char c){**

**return c == '+' || c == '-' || c == '\*' || c == '/';**

**}**

**static int Prioridad(char op){**

**if (op == '+' || op == '-') return 1;**

**if (op == '\*' || op == '/') return 2;**

**return 0;**

**}**

**static double AplicarOperacion(Stack<double> operandos, Stack<char> operadores){**

**if (operandos.Count < 2)**

**throw new Exception("Faltan operandos.");**

**double b = operandos.Pop();**

**double a = operandos.Pop();**

**char op = operadores.Pop();**

**switch (op){**

**case '+': return a + b;**

**case '-': return a - b;**

**case '\*': return a \* b;**

**case '/': return a / b;**

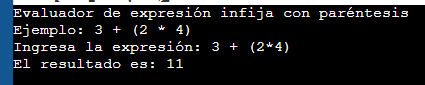
**default: throw new Exception("Operador no válido.");**

**}**

**}**

**}**

**Resultado:**



# Historial de navegación

**Descripción:**

Simula un navegador con “atrás” y “adelante”.

# Objetivo:

Usar dos pilas: una para el historial anterior y otra para las páginas a las que se puede volver.

# Pista:

Al visitar una nueva página, apílala en “atrás” y limpia la pila “adelante”.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static void Main(){**

**Stack<string> atras = new Stack<string>();**

**Stack<string> adelante = new Stack<string>();**

**string paginaActual = "Inicio";**

**int opcion;**

**do{**

**Console.WriteLine();**

**Console.WriteLine("===== SIMULADOR DE NAVEGADOR =====");**

**Console.WriteLine("Página actual: " + paginaActual);**

**Console.WriteLine("1. Visitar nueva página");**

**Console.WriteLine("2. Atrás");**

**Console.WriteLine("3. Adelante");**

**Console.WriteLine("4. Salir");**

**Console.Write("Elige una opción: ");**

**if (!int.TryParse(Console.ReadLine(), out opcion)){**

**Console.WriteLine("Por favor, ingresa un número válido.");**

**continue;**

**}**

**switch (opcion){**

**case 1:**

**Console.Write("Ingresa la nueva página: ");**

**string nueva = Console.ReadLine();**

**atras.Push(paginaActual);**

**paginaActual = nueva;**

**adelante.Clear();**

**Console.WriteLine("Has visitado: " + paginaActual);**

**break;**

**case 2:**

**if (atras.Count > 0){**

**adelante.Push(paginaActual);**

**paginaActual = atras.Pop();**

**Console.WriteLine("Retrocediste a: " + paginaActual);**

**}else{**

**Console.WriteLine("No hay páginas anteriores.");**

**}**

**break;**

**case 3:**

**if (adelante.Count > 0){**

**atras.Push(paginaActual);**

**paginaActual = adelante.Pop();**

**Console.WriteLine("Avanzaste a: " + paginaActual);**

**}else{**

**Console.WriteLine("No hay páginas siguientes.");**

**}**

**break;**

**case 4:**

**Console.WriteLine("Saliendo del navegador...");**

**break;**

**default:**

**Console.WriteLine("Opción no válida.");**

**break;**

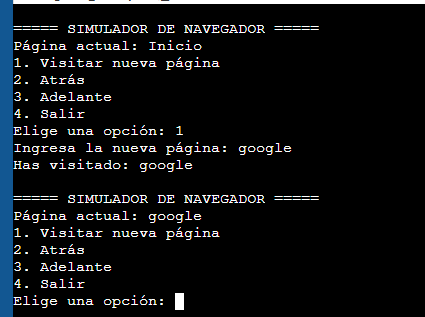
**}**

**} while (opcion != 4);**

**}**

**}**

**Resultado:**

****

# Convertir expresión infija a postfija Descripción:

Convierte expresiones infijas como "A + B \* C" a postfijas "A B C \* +".

# Objetivo:

Practicar el control de prioridad de operadores con pilas.

# Pista:

Usa una pila para los operadores y una lista de salida para los operandos.

**Script:**

**using System;**

**using System.Collections.Generic;**

**class EjerciciosPilas{**

**static int Precedencia(char operador){**

**if (operador == '+' || operador == '-') return 1;**

**if (operador == '\*' || operador == '/') return 2;**

**return 0;**

**}**

**static string InfijoAPostfijo(string expresion){**

**Stack<char> pilaOperadores = new Stack<char>();**

**List<string> salida = new List<string>();**

**for (int i = 0; i < expresion.Length; i++){**

**char c = expresion[i];**

**if (char.IsWhiteSpace(c)) continue;**

**if (char.IsLetterOrDigit(c)){**

**salida.Add(c.ToString());**

**}else if (c == '('){**

**pilaOperadores.Push(c);**

**}else if (c == ')'){**

**while (pilaOperadores.Count > 0 && pilaOperadores.Peek() != '('){**

**salida.Add(pilaOperadores.Pop().ToString());**

**}**

**if (pilaOperadores.Count > 0 && pilaOperadores.Peek() == '('){**

**pilaOperadores.Pop(); // quitar el paréntesis izquierdo**

**}**

**}else if ("+-\*/".Contains(c)){**

**while (pilaOperadores.Count > 0 && Precedencia(pilaOperadores.Peek()) >= Precedencia(c)){**

**salida.Add(pilaOperadores.Pop().ToString());**

**}**

**pilaOperadores.Push(c);**

**}**

**}**

**while (pilaOperadores.Count > 0){**

**salida.Add(pilaOperadores.Pop().ToString());**

**}**

**return string.Join(" ", salida);**

**}**

**static void Main(){**

**Console.Write("Ingresa una expresión infija (ej. A + B \* C): ");**

**string expresion = Console.ReadLine();**

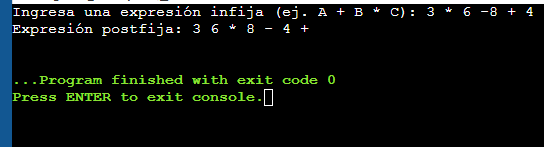
**string postfija = InfijoAPostfijo(expresion);**

**Console.WriteLine("Expresión postfija: " + postfija);**

**}**

**}**

**Resultado:**



**Vigencia:**

28/10/2025 hasta las 23:59 Hr

**Ing. Juan Carlos Arbeláez**