

Nombre : Lilibeth Tatiana Gomez Mantilla

## TALLER – PILAS EN C#

### 1. 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().

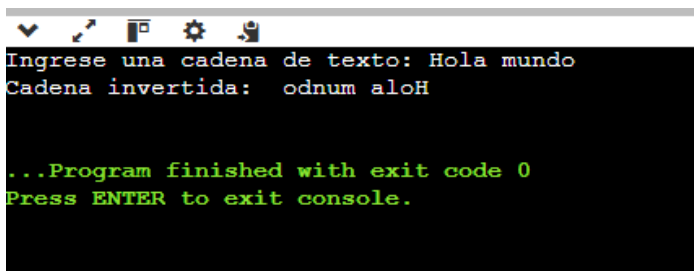
Pista: Recorre la cadena, mete cada carácter en la pila y luego sácalos para formar la cadena invertida.

```
using System;
using System.Collections.Generic;
class Program
{
    static void Main()
    {
        Console.Write("Ingrese una cadena de texto: ");
        string texto = Console.ReadLine();

        Stack<char> caracter = new Stack<char>();

        foreach (char c in texto)
        {
            caracter.Push(c);
        }

        string texto_Invertido = string.Empty;
        while (caracter.Count > 0)
        {
            texto_Invertido += caracter.Pop();
        }
        Console.WriteLine("Cadena invertida: " + texto_Invertido);
    }
}
```



```
▼ ↗ 📄 ⚙️ 🐞
Ingrese una cadena de texto: Hola mundo
Cadena invertida:  odnum aloH

...Program finished with exit code 0
Press ENTER to exit console.
```

## 2. 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.

```
using System;
using System.Collections.Generic;
class ProgramaVerificador
{
    static bool Parentesis(string expresion)
    {
        Stack<char> pilaSimbolos = new Stack<char>();

        foreach (char simbolo in expresion)
        {
            if (simbolo == '(' || simbolo == '[' || simbolo == '{')
            {
                pilaSimbolos.Push(simbolo);
            }
            else if (simbolo == ')' || simbolo == ']' || simbolo == '}')
            {
                if (pilaSimbolos.Count == 0)
                    return false;

                char simboloApertura = pilaSimbolos.Pop();

                if ((simbolo == ')' && simboloApertura != '(') ||
                    (simbolo == ']' && simboloApertura != '[') ||
                    (simbolo == '}' && simboloApertura != '{'))
                {
                    return false;
                }
            }
        }
        return pilaSimbolos.Count == 0;
    }
    static void Main()
    {
        Console.WriteLine("Escribe una expresión con paréntesis: ");
        string expresionUsuario = Console.ReadLine();

        if (Parentesis(expresionUsuario))
            Console.WriteLine("Los paréntesis están balanceados.");
        else
            Console.WriteLine("Los paréntesis no están balanceados.");
    }
}
```

```
Escribe una expresión con paréntesis: ()
Los paréntesis están balanceados.
```

### 3. 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.

```
using System;
using System.Collections.Generic;
class Program
{
    static void Main()
    {
        Console.Write("Ingresa un numero decimal: ");
        int numeroDecimal;
        while (!int.TryParse(Console.ReadLine(), out numeroDecimal) || numeroDecimal < 0)
        {
            Console.Write("Entrada invalida. Ingresa un numero entero positivo: ");
        }

        Stack<int> pilaBinaria = new Stack<int>();
        int numero = numeroDecimal;
        do
        {
            int residuo = numero % 2;
            pilaBinaria.Push(residuo);
            numero /= 2;
        } while (numero > 0);
        Console.Write($"El numero {numeroDecimal} en binario es: ");
        while (pilaBinaria.Count > 0)
        {
            Console.Write(pilaBinaria.Pop());
        }
        Console.WriteLine();
    }
}
```

```
Ingresa un numero decimal: 25
El numero 25 en binario es: 11001

...Program finished with exit code 0
Press ENTER to exit console.□
```

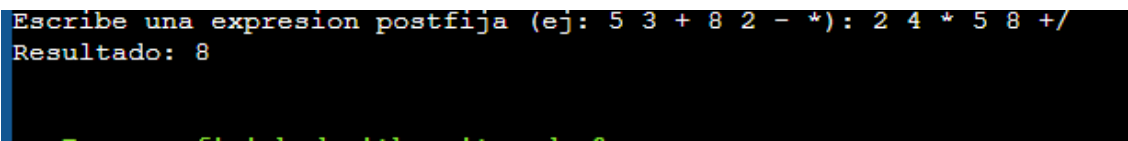
#### 4. Evaluar expresión postfija (notación polaca inversa)

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.

```
using System;
using System.Collections.Generic;
class Program
{
    static void Main()
    {
        Console.Write("Escribe una expresion postfija (ej: 5 3 + 8 2 - *): ");
        string entrada = Console.ReadLine();
        string[] partes = entrada.Split(' ');
        Stack<double> pila = new Stack<double>();
        foreach (string parte in partes)
        {
            if (double.TryParse(parte, out double numero))
            {
                pila.Push(numero);
            }
            else
            {
                double b = pila.Pop();
                double a = pila.Pop();
                switch (parte)
                {
                    case "+": pila.Push(a + b); break;
                    case "-": pila.Push(a - b); break;
                    case "*": pila.Push(a * b); break;
                    case "/": pila.Push(a / b); break;
                }
            }
        }
        Console.WriteLine("Resultado: " + pila.Pop());
    }
}
```



```
Escribe una expresion postfija (ej: 5 3 + 8 2 - *): 2 4 * 5 8 +/
Resultado: 8
```



```
}
```

```
Texto actual:
1. Escribir
2. Deshacer
3. Salir
Elige una opcion (1-3): 1
Escribe algo: Hola mundo

Texto actual: Hola mundo
1. Escribir
2. Deshacer
3. Salir
Elige una opcion (1-3): 2

Texto actual:
1. Escribir
2. Deshacer
3. Salir
Elige una opcion (1-3): 3

Texto final:
```

## 6. 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.

```
using System;
using System.Collections.Generic;
class Program{
    static void Main(){
        Console.Write("Escribe una palabra: ");
        string palabra = Console.ReadLine().ToLower();
        Stack<char> pila = new Stack<char>();
        foreach (char letra in palabra)
        {
            pila.Push(letra);
        }
        string palabraInvertida = "";
        while (pila.Count > 0)
        {
            palabraInvertida += pila.Pop();
        }

        if (palabra == palabraInvertida)
        {
            Console.WriteLine("La palabra es palindroma.");
        }
        else
        {
            Console.WriteLine("La palabra no es palindroma.");
        }
    }
}
```

```
Escribe una palabra: radar
La palabra es palindroma.

...Program finished with exit code 0
Press ENTER to exit console.
```

## 7.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.

```
using System;
using System.Collections.Generic;
class Program{
    static void Main(){
        Console.Write("Escribe una oracion: ");
        string oracion = Console.ReadLine();
        string[] palabras = oracion.Split(' ', StringSplitOptions.RemoveEmptyEntries);
        Stack<string> pila = new Stack<string>();
        foreach (string palabra in palabras)
        {
            pila.Push(palabra);
        }
        string oracionInvertida = "";
        while (pila.Count > 0)
        {
            oracionInvertida += pila.Pop() + " ";
        }
        Console.WriteLine("Oracion invertida: " + oracionInvertida.Trim());
    }
}
```

```
Escribe una oracion: Hello World c:
Oracion invertida: c: World Hello

...Program finished with exit code 0
Press ENTER to exit console.□
```

## 8. 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).

```
using System;
using System.Collections.Generic;
class Program{
    static void Main(){
        Console.WriteLine("Escribe una expresion infija (ej: 36+(8-7)): ");
        string expresion = Console.ReadLine().Replace(" ", "");

        Stack<double> operandos = new Stack<double>();
        Stack<char> operadores = new Stack<char>();

        for (int i = 0; i < expresion.Length; i++){
            char c = expresion[i];

            if (char.IsDigit(c))
            {
                string numero = "";

                // Leer todos los digitos seguidos
                while (i < expresion.Length && char.IsDigit(expresion[i]))
                {
                    numero += expresion[i];
                    i++;
                }

                i--; // retroceder porque el for tambien avanza
                operandos.Push(double.Parse(numero));
            }
            else if (c == '(')
            {
                operadores.Push(c);
            }
            else if (c == ')')
            {
                while (operadores.Peek() != '(')
                {
                    Evaluar(operandos, operadores);
                }
                operadores.Pop(); // quitar el '('
            }
            else if (EsOperador(c))
            {
                while (operadores.Count > 0 && Prioridad(operadores.Peek()) >=
Prioridad(c))
                {
```



```

        Evaluar(operandos, operadores);
    }
    operadores.Push(c);
}
}
while (operadores.Count > 0)
{
    Evaluar(operandos, operadores);
}

Console.WriteLine("Resultado: " + operandos.Pop());
}
static bool EsOperador(char c)
{
    return c == '+' || c == '-' || c == '*' || c == '/';
}
static int Prioridad(char operador)
{
    if (operador == '+' || operador == '-') return 1;
    if (operador == '*' || operador == '/') return 2;
    return 0;
}
static void Evaluar(Stack<double> operandos, Stack<char> operadores)
{
    double b = operandos.Pop();
    double a = operandos.Pop();
    char op = operadores.Pop();
    switch (op)
    {
        case '+': operandos.Push(a + b); break;
        case '-': operandos.Push(a - b); break;
        case '*': operandos.Push(a * b); break;
        case '/': operandos.Push(a / b); break;
    }
}
}
}

```

```

Escribe una expresion infija (ej: 36+(8-7)): 3+(4-8)
Resultado: -1

...Program finished with exit code 0
Press ENTER to exit console.

```

## 9. 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, apíjala en “atrás” y limpia la pila “adelante”.

```
using System;
using System.Collections.Generic;
class Program{
    static void Main(){
        Stack<string> pilaAtras = new Stack<string>();
        Stack<string> pilaAdelante = new Stack<string>();
        string paginaActual = "Inicio";

        while (true)
        {
            Console.WriteLine("\nPagina actual: " + paginaActual);
            Console.WriteLine("1. Visitar nueva pagina");
            Console.WriteLine("2. Atras");
            Console.WriteLine("3. Adelante");
            Console.WriteLine("4. Salir");
            Console.Write("Elige una opcion (1-4): ");
            string opcion = Console.ReadLine();
            switch (opcion)
            {
                case "1":
                    Console.Write("Nombre de la nueva pagina: ");
                    string nuevaPagina = Console.ReadLine();
                    pilaAtras.Push(paginaActual);
                    paginaActual = nuevaPagina;
                    pilaAdelante.Clear();
                    break;

                case "2":
                    if (pilaAtras.Count > 0)
                    {
                        pilaAdelante.Push(paginaActual);
                        paginaActual = pilaAtras.Pop();
                    }
                    else
                    {
                        Console.WriteLine("No hay paginas anteriores.");
                    }
                    break;

                case "3":
                    if (pilaAdelante.Count > 0)
                    {
                        pilaAtras.Push(paginaActual);
                        paginaActual = pilaAdelante.Pop();
                    }
                    else
                    {
                        Console.WriteLine("No hay paginas posteriores.");
                    }
                    break;

                case "4":
                    Console.WriteLine("Salir");
                    return;
            }
        }
    }
}
```

```

    }
    else
    {
        Console.WriteLine("No hay paginas siguientes.");
    }
    break;

case "4":
    Console.WriteLine("Saliendo del navegador...");
    return;

default:
    Console.WriteLine("Opcion no valida.");
    break;
}
}
}
}
}

```

```

Pagina actual: Inicio
1. Visitar nueva pagina
2. Atras
3. Adelante
4. Salir
Elige una opcion (1-4): 1
Nombre de la nueva pagina: youtube

Pagina actual: youtube
1. Visitar nueva pagina
2. Atras
3. Adelante
4. Salir
Elige una opcion (1-4): 1
Nombre de la nueva pagina: wikipedia

Pagina actual: wikipedia
1. Visitar nueva pagina
2. Atras
3. Adelante
4. Salir
Elige una opcion (1-4): 2

Pagina actual: youtube
1. Visitar nueva pagina
2. Atras
3. Adelante
4. Salir
Elige una opcion (1-4): 3

Pagina actual: wikipedia
1. Visitar nueva pagina
2. Atras
3. Adelante
4. Salir
Elige una opcion (1-4): 4
Saliendo del navegador...

```

## 10. 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.

```
using System;
using System.Collections.Generic;
class Program{
    static void Main() {
        Console.Write("Escribe una expresion infija (ej: A + B * C): ");
        string expresion = Console.ReadLine().Replace(" ", "");

        Stack<char> operadores = new Stack<char>();
        List<char> salida = new List<char>();

        foreach (char c in expresion)
        {
            if (char.IsLetter(c))
            {
                salida.Add(c);
            }
            else if (c == '(')
            {
                operadores.Push(c);
            }
            else if (c == ')')
            {
                while (operadores.Count > 0 && operadores.Peek() != '(')
                {
                    salida.Add(operadores.Pop());
                }
                if (operadores.Count > 0 && operadores.Peek() == '(')
                {
                    operadores.Pop();
                }
            }
            else if (EsOperador(c))
            {
                while (operadores.Count > 0 && Prioridad(operadores.Peek()) >=
Prioridad(c))
                {
                    salida.Add(operadores.Pop());
                }
                operadores.Push(c);
            }
        }

        while (operadores.Count > 0)
        {
            salida.Add(operadores.Pop());
        }
    }
}
```

```

    }

    Console.Write("Expresion postfija: ");
    foreach (char s in salida)
    {
        Console.Write(s + " ");
    }
    Console.WriteLine();
}

static bool EsOperador(char c)
{
    return c == '+' || c == '-' || c == '*' || c == '/';
}

static int Prioridad(char operador)
{
    if (operador == '+' || operador == '-') return 1;
    if (operador == '*' || operador == '/') return 2;
    return 0;
}
}

```

```

Escribe una expresion infija (ej: A + B * C): a+b*c
Expresion postfija: a b c * +

```