Prácticas de Laboratorio

1. Verificador de Delimitadores (Pilas)

Realizar un programa para **verificar delimitadores**. Debe de verificar paréntesis, corchetes y llaves (emplear pilas). El programa debe de verificar el uso correcto de **( ), [ ]** y **{ }**, dentro de una expresión algebraica. Algunos ejemplos se muestran a continuación:

Entrada: **`[a+(b*c)*d+{e-(f*g)}]`**
Salida: **Expresión válida**

Entrada: **`[a+(b*c)*d+{e-(f*g))]`**
Salida: **Expresión inválida**

2. Evaluador de Expresiones Algebraicas (Pilas)

Realizar un programa para **evaluar una expresión algebraica**, que contenga las operaciones básicas: ***, /, +** y **-** (emplear pilas). Además debe de reconocer los paréntesis como delimitadores. Por ejemplo:

Entrada: **`2*(5+8)-5/2`**
Salida: `2*(5+8)-5/2 =` **`23.5`**

3. Seleccionar alguna de las siguientes dos opciones:

a) Verificador de expresiones Algebraicas (Recursión)

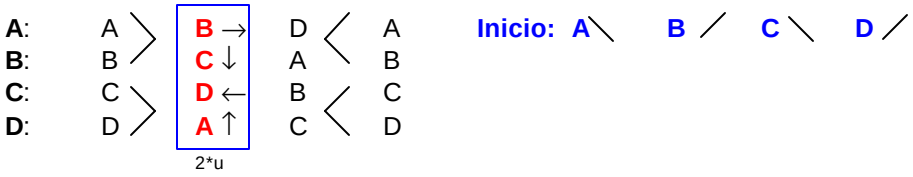
Realizar un programa recursivo, para **verificar una expresión algebraica** con las operaciones ***** y **+**; y que emplee paréntesis como delimitadores.

Entrada: **`(a*b+c*d)+(e*(f)+g)`**
Salida: **Expresión válida**

Entrada: **`(a*b+c*d)+(e*(f)+g`**
Salida: **Expresión inválida**

b) Curva de Sierpinski (Recursión)

Realizar un programa que dibuje la **curva de Sierpinski**. Su definición es:



Por ejemplo la curva **Sierpinski de orden 3** se muestra en la **Figura 1**.

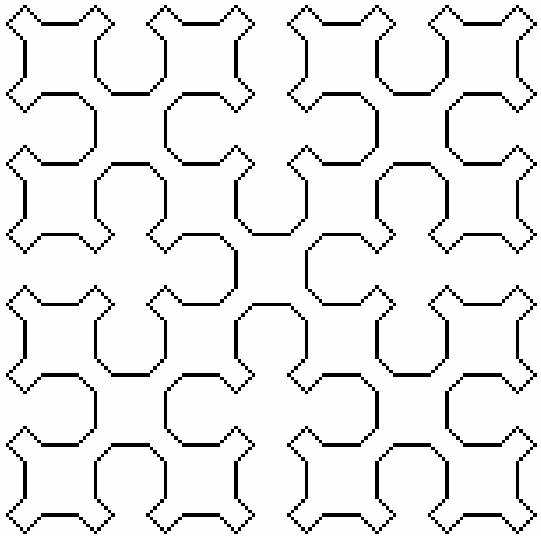


Figura 1: Curva **Sierpinski de orden 3**.

4. Agregación de Cadenas (Colas)

Realice un programa para **agregar cadenas** leídas desde un archivo, en una **cola de prioridad** y grabarlas en un archivo de salida. El programa debe de recibir parámetros de la línea de comandos y de no ser así, debe de tomar dichos parámetros por omisión.

Por ejemplo, tenemos: **`c:>cola_1 agda1.txt agda1.txt¿`**, y la salida del programa y el contenido se muestra en las **Figuras 2** y **3**.

```
        Colas de Prioridad
I = Insertar un Dato
S = Suprimir un Dato
V = Ver todos los elementos
Q = Salir
Elija una Opción:
> I
Introducir un Dato: Último elemento a insertar
> I
Introducir un Dato: Alguna cita por ejemplo
> I
Introducir un Dato: 2 Primer elemento
> V
Datos de la Cola
Elemento 0 = 2 Primer elemento
Elemento 1 = Alguna cita por ejemplo
Elemento 2 = Último elemento a insertar
```

Figura 2: Ejemplo del uso de colas de prioridad para agregar cadenas.

```
2 Primer elemento
Alguna cita por ejemplo
Último elemento a insertar
```

Figura 3: Contenido del archivo `agda1.txt`.

5. Seleccionar alguna de las siguientes tres opciones:

a) Derivación e Integración de Polinomios (Listas)

Encontrar la **derivada** y la **integral** de un polinomio con **listas enlazadas**.

Polinomio: `-1 +6x^2 -5x^4`
Derivada: **`+12x -20x^3`**
Integral: **`-1x +2x^3 -1x^5`**

b) Suma y resta de polinomios (Listas)

Dados dos polinomios **A** y **B** encontrar: **A + B**, **A - B** y **B – A**.

Polinomio P1: `+10 -1x +2x^4`
Polinomio P2: `+5 -1x +1x^3`
Polinomio SUMA: P1 + P2 **`+15 -2x +1x^3 +2x^4`**
Polinomio RESTA: P2 - P1 **`-5 +1x^3 -2x^4`**
Polinomio RESTA: P1 – P2 **`+5 -1x^3 +2x^4`**

c) Suma de números enteros positivos grandes (Listas)

Sumar dos **enteros positivos grandes** con **listas enlazadas**.

Entero A: `9012 5678 1234`
Entero B: `5412 2536`
SUMA: **`9013 1090 3770`**

6. Código de Huffman (Árboles Binarios)

Realizar un programa que realice:

- Leer un archivo de texto, el cual contenga palabras de hasta 10 caracteres diferentes que formen cadenas, por ejemplo `aaa, aba, bbb, cccaab,` etc.
- Con un árbol binario determine la cantidad y la frecuencia de cada uno de los caracteres.
- Determine el **árbol de Huffman** y lo muestre.
- Muestre el **código de Huffman** para cada uno de los caracteres contenidos en el archivo de texto.

7. Creación de un índice (Árboles AVL)

Realizar un programa que permita **realizar un índice** a partir de un archivo de texto empleando **árboles AVL**.

8. Complejidad

Elegir tres de los siguientes programas y comprobar su complejidad de forma práctica:

a) Evaluación de **expresiones postfijas**: `push, pop` y `evaluación`.
b) **Multiplicación** recursiva e iterativa
c) **Factorial** recursiva e iterativa
d) Secuencia de **Fibonacci** recursiva e iterativa
e) Torres de **Hanoi**
f) **Búsqueda Binaria**
g) Definición recursiva de **expresiones algebraicas**
h) Fractal de **Hilbert**
i) Fractal de **Sierpinski**

j) **Agregación de cadenas**: `insertar, suprimir, ver, guardar` y `leer`.
k) **Derivada de un polinomio** con listas lineales: `insertar, ver` y `buscar`.
l) **Problema de Josephus** con listas circulares: `insertar, suprimir` y `ver`.
m) Operaciones básicas con un **árbol binario**: `insertar, suprimir, buscar, ver` y `recorridos`.

9. Métodos de ordenamiento

De los siguientes métodos de ordenamiento elegir tres:

a) Burbuja
b) Selección
c) Inserción
d) Heap Sort
e) Merge Sort
f) Shell Sort
g) Quick Sort
h) Ripple Sort
i) Shell Sort

De cada uno de ellos generar una tabla comparativa de los tiempos de ejecución, para cada uno de los siguientes casos:

a) Arreglo en orden ascendente: `1,2,3,4,5,…,n`
b) Arreglo en orden descendente: `n, n-1,…2,1`
c) Arreglo con datos aleatorios entre `1 a n`

Probar y medir los tiempos de ejecución para tamaños de arreglo de `n = 500, 1,000, 2,000, 3,000, 4,000 y 5,000.`

Realizar las graficas de los tiempos de procesamiento **medido** y **teórico** para cada uno de los métodos elegidos y una gráfica comparativa de cada uno de ellos.

10. Inserción de Cadenas (Tablas Hash)

Realizar un programa que permita agregar cadenas en una **tabla hash**.

11. Manejo de Archivos (Árboles B)

Implementar un programa, el cual emplee **árboles B** para manejar un archivo.

12. Árboles B+ (Organización de Archivos)

Implementar un programa, el cual emplee **árboles B+** para manejar un archivo.