Ejercicios

1. Utilizar el TAD `pila` de caracteres para escribir un algoritmo que compruebe que una expresión aritmética está bien balanceada respecto a paréntesis, corchetes y llaves. La expresión se recibirá como parámetro de entrada de tipo cadena de caracteres, `TCadena[0..MaxCadena]`. Ejemplos:

`5*a(2+[{2*a-1}*b-1]/2)` ⟶ Bien

`5*a(2+[{2*a-1*b-1}/2)` ⟶ Mal, se espera un `]`, no un `)`

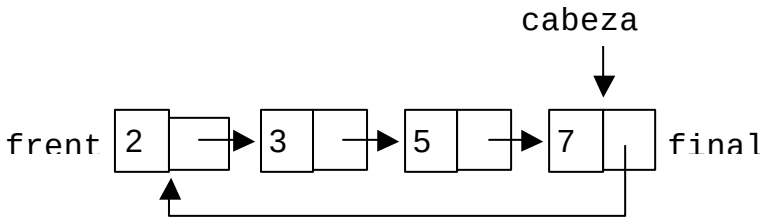
`5*a-2+{2*a-1*b-1}/2)` ⟶ Mal, `)` inesperado

2. Escribe un algoritmo que extraiga el último elemento de una pila y lo ponga en la cima, respetando el orden del resto de los elementos. Utiliza para ello una pila auxiliar.

3. Intercambiar los elementos de una pila y una cola de números enteros de tres maneras distintas:

a) Utilizando sólo un TAD pila auxiliar.
b) Utilizando sólo un TAD cola auxiliar.
c) Utilizando sólo variables de tipo simple.

4. Supón que se representa el TAD cola mediante una lista enlazada circular:




```
TIPOS
  Nodo *Cola
  REGISTRO Nodo
    TipoElemento dato
    Cola sig
  FINREGISTRO
```

Teniendo en cuenta que `cabeza` apunta al final y `cabeza->sig` al frente, implementa todas las operaciones con esta representación.

5. Al TAD `Lista` explicado en clase añade las siguientes operaciones implementadas dinámicamente:

```
B Esta(E TipoElemento elem)
// Devuelve VERDADERO si dato está en lista
InsertarTras(E TipoElemento anterior, nuevo)
// Inserta “nuevo” en la lista después de “anterior” si existe éste
Eliminar(E TipoElemento elem)
// Borra el elemento que sigue a “elem” en lista
```

6. Utiliza un TAD Lista cuyo `TipoElemento` es un tipo enumerado de los países europeos:

`ENUM {Alemania, Belgica, Bulgaria, España, Francia, Grecia,... Rusia} TPaisEuropeo`

Suponiendo que la presidencia de la Unión Europea cambia cada seis meses de forma rotatoria, escribe un algoritmo que reciba en un TAD `Lista` con la ordenación actual (que sólo contendrá países de la Unión) y calcule qué país presidirá la Unión Europea dentro de x meses. El primer país todavía no lleva un mes.

7. Escribe un algoritmo que reciba un TAD `Lista` con el mismo `TipoElemento` que el ejercicio anterior, que contenga los países de la UE; un TAD `Cola` del mismo `TipoElemento`, con los países a la espera de entrar en la UE; y otro TAD `Cola` de naturales con el número de parados en los países a la espera de entrar en la UE. Suponiendo que cada número de parados de una de las colas corresponde al país que está en la misma posición de la otra cola, el algoritmo deberá insertar en la lista de la UE sólo los países cuyas cifras de parados estén por debajo de un número que recibe como argumento.

8. Mediante el uso de clases, implementa un TAD **`Matriz`** de números reales de forma estática, siendo `MaxFila` y `MaxColumna` los límites de la misma. Su especificación incluirá las siguientes operaciones:

a) Un constructor que reciba dos argumentos, su número de filas y de columnas.
b) Un constructor de copia.
c) Un destructor.
d) **`Poner`**`(E N fila, columna; E R nuevo)`, que pone un valor en la posición indicada. Si `fila` y `columna` caen fuera de rango, no hace nada.
e) `R` **`Valor`**`(E N fila, columna; S B fuera_rango)`, que devuelve el valor si `fila` y `columna` caen en el rango de la matriz. Si no, `fuera_rango` se pondrá a `FALSO`.

Una vez implementado, escribe un algoritmo que reciba dos matrices y las multiplique, comprobando previamente que las dimensiones sean compatibles.

9. Mediante el uso de clases, implementa dinámicamente un TAD **`Conjunto`** de letras. Contendrá las operaciones:

a) Un constructor que no recibe argumentos, que creará un conjunto vacío.
b) Un constructor de copia.
c) Un destructor.

```
d) Incluir(E C letra)          // si no está ya en el conjunto
e) B Pertenece(E C letra)      // V si letra en el conjunto
f) B Vacio()                   // V si el conjunto está vacío
g) Union(E Conjunto a)         // unión con "a"
h) Interseccion(E Conjunto b)  // intersección con "a"
```

Una vez implementado, escribe algoritmos externos para realizar las operaciones siguientes:

```
a) Conjunto Diferencia(E Conjunto a, b)
   // (a-b): elementos de "a" que no pertenecen a "b"
b) Conjunto DifSimetrica(E Conjunto a, b)
   // unión de (a-b) y (b-a)
```

10. Mediante el uso de clases, implementa (dinámica o estáticamente) un TAD **`Cadena`** de caracteres. Además, supóngase definido el tipo `TCadena[0..MaxCadena]`. El TAD **`Cadena`** contendrá las operaciones:

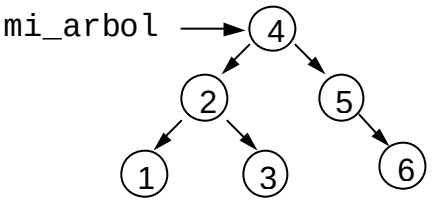
a) Un constructor que no recibe argumentos, que creará un cadena vacía.
b) Un constructor que recibe como argumento una cadena inicial como array de caracteres, de tipo `TCadena`.
c) Un constructor de copia.
d) Un destructor.

```
e) N Longitud()            // número de caracteres
f) Concatenar(E Cadena s)  // añade los caracteres de "s"
```

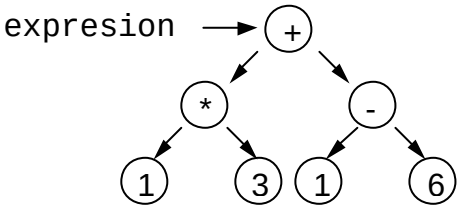
g) `Escibir()` `// escribe la cadena en pantalla`
h) `LeerLinea(E Fichero f)` `// lee caracteres de la posición de e/s actual de "f" hasta un '\n' y los asigna a la cadena`

Escribe un algoritmo que vaya leyendo las líneas de un fichero y se vaya quedando la más larga, que al final escribirá en pantalla. Para ello utiliza un subprograma `LineaSiguiente()`, que declare localmente una Cadena, la lea del fichero y la devuelva (al ser local, cada llamada supondrá la creación y destrucción de un objeto de tipo `Cadena`).

11. Diseña un algoritmo que escriba todos los nodos de un árbol (con la definición usual) por niveles a partir del nivel 0. Por ejemplo, para el árbol adjunto debería escribirse 4 2 5 1 3 6. (Sugerencia: utilizar el TAD `Cola`).



12. Un árbol binario puede utilizarse para almacenar una expresión algebraica, siendo las hojas los operandos y los demás nodos los correspondientes operadores. Por ejemplo el árbol adjunto representa representa a la expresión en notación prefija + * 1 3 - 1 6. Diseña un procedimiento recursivo que lleve una cadena de caracteres que representa una expresión en prefija a un árbol binario. Para simplificar, se considera que los operandos tienen una única cifra y el `TipoElemento` del árbol es de caracteres.



13. Muchos algoritmos sobre árboles binarios reflejan la naturaleza recursiva de éstos. Implementa métodos recursivos que realicen las siguientes tareas sobre árboles. Se usa una implementación dinámica de árboles.

a) Calcular la altura del árbol (el nivel mayor que contiene, siendo el 0 el de la raíz).
c) Encontrar el elemento mayor.
d) Calcular la suma de los elementos.
d) Encontrar un elemento dado.
f) Determinar el mayor nivel que está completo.
g) Borrar todos los nodos.
h) Insertar un elemento en el árbol.
i) Devolver el número de nodos
j) Devolver el número de hojas