UNIVERSIDAD DE MÁLAGA
E.T.S.I. TELECOMUNICACIÓN

Estructuras Avanzadas de Datos

Programación Modular

(1º de Ingeniería de Telecomunicación)

1. Supongamos que tenemos un lenguaje de programación que tiene la clase Pila pero no el tipo de datos ARRAY. Un array se caracteriza por ser una lista secuencial de datos a los que se puede acceder de forma directa especificando el índice asociado. Las operaciones fundamentales son:

   - Creación: establece el tamaño del array, el tipo Indice y el tipo Base.
   - Asignación de un dato a una localización particular indicada con un índice.
   - Acceso a un dato de una localización determinada del array, indicada con un índice

   Consideremos las siguientes definiciones para estas operaciones:

```
INTERFAZ CLASE Array
TIPOS
    // Definicion del tipo TipoElem
METODOS
    Crear(E Z inf, sup, E TipoElem inic)
    Asignar(E Z pos, E TipoElem x)
    TipoElem Acceder(E Z pos)
FIN Array

IMPLEMENTACION CLASE Array
ATRIBUTOS
    TipoPila pila
    Z inf, sup
METODOS
    Crear(E Z inf, sup, E TipoElem inic)
    INICIO
        ......
    FIN Crear
    Asignar(E Z pos, E TipoElem x)
    INICIO
        ......
    FIN Asignar
    TipoElem Acceder(E Z pos)
    INICIO
        ......
    FIN Acceder
FIN Array
```

   Implementa estos métodos basándote en la definición previa.

2. Utilizando la clase Pila, diseña un algoritmo que determine si una cadena de caracteres de entrada es de la forma

   x y

   donde x es una cadena que consiste en caracteres arbitrarios e y es la inversa de x. Por ejemplo, si x = "ABBDYCA", entonces y = "ACYDBBA".

3. Añade los siguientes métodos a cada implementación de la clase Cola realizadas en clase:

```
TipoElem BasePila()
// Devuelve el valor que esta en el fondo de la pila sin modificarla
```

4. Supongamos que tenemos una máquina con un solo registro y seis instrucciones:

```
LD  A       //Situa el operando A en el registro
ST  A       //Situa el contenido del registro en A
AD  A       //Suma A al contenido del registro
SB  A       //Resta A del contenido del registro
ML  A       //Multiplica el contenido del registro por A
DV  A       //Divide el contenido del registro por A
```

   Escribe un programa que acepte una expresión en postfija que contenga operandos simples (formados por letras mayúsculas) y operadores (+,-,*,/), y que escriba en la pantalla la secuencia de instrucciones del tipo anterior para evaluar la expresión en dicha máquina, dejando el resultado en el registro.Por ejemplo, dada la expresión: ABC*+DE-/, el programa debería dar como resultado:

```
LD  B
ML  C
ST  Temp1
LD  A
AD  Temp1
ST  Temp2
LD  D
SB  E
ST  Temp3
LD  Temp2
DV  Temp3
ST  Temp4
LD  Temp4
```

5. Escribe un programa que lea expresiones que contengan paréntesis ( ), corchetes [ ], y llaves {}, y que compruebe que están perfectamente anidadas. Puedes suponer que cada línea contiene exactamente un expresión. Ejemplo:

```
A[I+X{Y-2}]
LEGAL
B+[3-{X/2})*19+2/(X-7)
ERROR, SE ESPERABA ]
```

6. Definir a nivel de utilización de los siguientes métodos:

```
B PilasIguales(E Pila p2)
// Devuelve VERDADERO si y solo si p2 es igual que la pila sobre la que se ejecuta el metodo
TipoPila CopiarPila()
// Devuelve una copia de la pila
TipoPila ConcatenarPilas(E TipoPila p2)
// Devuelve una pila con los elementos de p2 sobre los de la pila sobre la
// que se ejecuta el metodo.
```

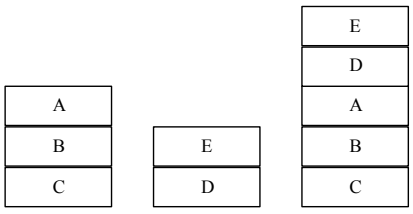

Figura 1: Concatenación de pilas

7. Añade los siguientes métodos a cada implementación de la clase Cola realizadas en clase:

```
TipoElem FinalCola()
// Devuelve el valor que esta en el Final de la cola sin modificarla
```

8. Una cola de doble entrada es una colección de elementos homogéneos en la que se permiten inserciones y extracciones en ambos extremos. Diseña la clase ColaDoble con las operaciones usuales.

9. Supongamos que una Cola va a representarse mediante una estructura dinámica circular utilizando las siguientes declaraciones:

```
INTERFAZ CLASE CCOLA
METODOS
    Crear()
    Destruir()
    B EstaVacia()
    B EstaLlena()
    Encolar(TipoElemento elem)
    Desencolar()
    TipoElemento Frente()
FIN CCOLA

IMPLEMENTACION CLASE CCOLA
    TIPOS
        REGISTRO Nodo
              TipoElem valor
              PtrCola sig
        FINREGISTRO
    ATRIBUTOS
        Nodo *c
    METODOS
             ........
FIN CCOLA%
```

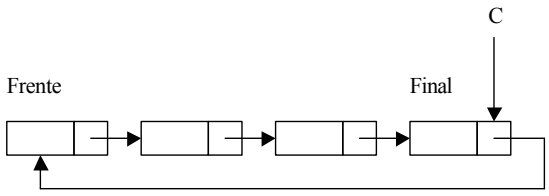

Figura 2: Cola circular

Como se ve en la figura 2, c apunta al dato del Final de la misma, y c→sig apunta al Frente.

a) Con esta representación implementa todos los métodos de la clase Cola.

b) Repetir el ejercicio suponiendo que el puntero c apunta al nodo etiquetado como Frente.

10. Diseña la clase "ColaConPrioridad". Así mismo, diseña un módulo de utilización simple que utilice la clase diseñada. Una cola con prioridad es aquélla en la que se introducen valores con una prioridad especificada, y cuando se sacan valores de ella, se sacarán los valores introducidos con mayor prioridad. Para valores con igual prioridad el orden de salida es como una cola normal FIFO. Los elementos base de la cola serán números. Las operaciones que ofrece la clase son:

```
INTERFAZ CLASE CCOLA_PRIO
CONST
    Min_Prioridad = 1
    Max_Prioridad = 5
METODOS
    CrearCola()
    DestruirCola()
    MeterElem(E TipoElem el, E N prio)
    SacarElemMaximaPrio(S TipoElem elem)
    SacarElemMinimaPrio(S TipoElem elem)
    B ColaVacia()
    B Colallena()
FIN CCOLA_PRIO
```

Sigue la estructura de la figura 3.

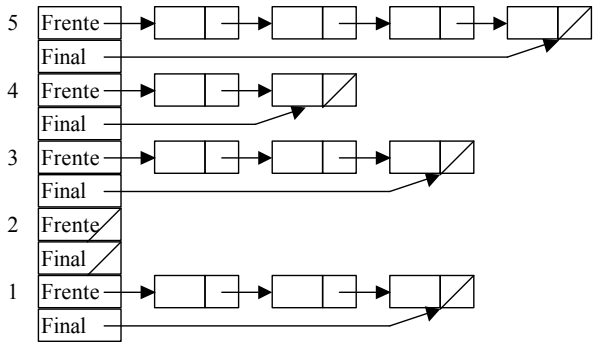

Figura 3: Cola de prioridad

11. A la clase `ListaPosicional` explicado en clase añade los siguientes métodos.

```
B EnLista(E TipoElem dato)
// Devuelve VERDADERO si dato esta en lista, FALSO en caso contrario
InsertaDesp(E TipoElem dato_ant, nuevo_dato)
// Inserta nuevo_dato en la lista despues de dato_ant
BorraDesp(E TipoElem dato)
// Borra el elemento que sigue a dato en lista
```

12. Se entiende por rotación hacia adelante de un paso, que la cabeza de una lista se saque de ésta y se inserte como un nuevo dato al final. La rotación hacia atrás es la inversa de la anterior. Añade las siguientes operaciones a la clase `ListaPosicional`. Implementarlas desde el punto de vista del usuario.

    a) Adelante1, Atrás1 que roten la lista en un dato, en la dirección indicada.

    b) Cambia las definiciones para rotar en k datos.

    Implementarlas también accediendo directamente a la estructura con variables dinámicas.

13. Implementar la clase `POLI`, que recoge la estructura de datos polinomio, basándose en la clase `ListaPosicional`:

```
INTERFAZ CLASE POLI
TIPOS
   TYPE COEFICIENTE = REAL;
METODOS
    Crear()
    CopiaPoli(POLI ori)
    N ObtenerGradoPoli()
```

```
    R ObtenerMono(POLI p, N grado)
    SumaEnPoliMono(COEFICIENTE a; N n)
    SumaEnPoliPoli(VAR a : POLI; b : POLI)
    SumaPoliMono(VAR res : POLI; p : POLI; a : COEFICIENTE; n : CARDINAL)
    SumaPoliPoli(VAR res : POLI; a, b : POLI)
    Destruir()
FIN POLI
```

14. Diseña un algoritmo que escriba todos los nodos de un árbol binario por niveles a partir del nivel 0. Por ejemplo, el árbol de la figura 4 debería escribirse 4 2 5 1 3 6.

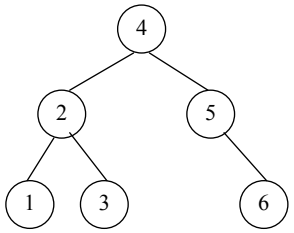

Figura 4: Árbol binario

15. Diseña una función recursiva que compruebe si una cadena de caracteres representa una expresión en posfija válida.
16. Un árbol binario puede utilizarse para almacenar una expresión algebraica, siendo las hojas los operandos y los demás nodos los correspondientes operadores.El árbol de la figura 5 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.

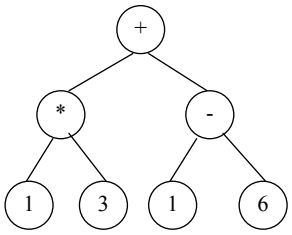

Figura 5: Expresión aritmética

17. Construir un algoritmo B LogEval(E ArbolBin e) que, aplicado a un árbol binario sobre caracteres (su tipo base es C) que contiene expresiones lógicas (sin variables) bien formadas como las del árbol de la figura 6, lo evalúe teniendo en cuenta posibles çortocircuitos". Esto es, si una de las ramas de una operación Y ($\wedge$) es ya falsa (F), no es necesario evaluar la otra rama. Así mismo si una de las ramas de una expresión O ($\vee$) es cierta (V), no es necesario evaluar la otra. El árbol sólo contiene los caracteres ($\wedge$, $\vee$, V, F) y se supone correcto, esto es, todos los operadores $\vee$ y $\wedge$ están en nodos internos y tienen dos descendientes mientras que los valores V y F están en hojas.
18. Desarrollar los siguientes métodos para árboles binarios:

```
N NumEles ()
// devuelve el numero de elementos del arbol
B EsHoja?()
// devuelve VERDADERO si el arbol es un nodo hoja, FALSO en caso contrario
```

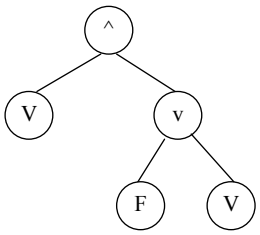

Figura 6: Expresión lógica

```
N Profundidad()
// devuelve la profundidad del \'{a}rbol
N CaminoMinimo()
// devuelve el valor del camino con valor minimo en el arbol.
// el valor de un camino es la suma de los valores de sus nodos.
BARBOL Copia()
// devuelve una copia del \'{a}rbol
B SonIguales?(E BARBOL s)
// devuelve VERDADERO si son iguales, FALSO en caso contrario
BASE MaxNodo()
// devuelve el mayor valor de los nodos del \'{a}rbol
BASE MinNodo()
// devuelve el menor valor de los nodos del \'{a}rbol
BASE MaxHoja()
// devuelve el mayor valor de las hojas del \'{a}rbol
BASE MinHoja()
// devuelve el menor valor de los hojas del \'{a}rbol
BASE MaxDelNivel(E N nivel)
// devuelve el mayor valor de las nodos de un nivel dado del \'{a}rbol
BASE MinDelNivel(E N nivel)
// devuelve el menor valor de las nodos de un nivel dado del \'{a}rbol
B EstaBalanceado?()
// devuelve VERDADERO si el arbol esta balanceado, FALSO en caso contrario
B EstaPerfectamenteBalanceado?()
// devuelve VERDADERO si el arbol esta perfectamente balanceado,
// FALSO en caso contrario
N MaximoNivelLleno()
// devuelve la profundidad del nivel lleno mas profundo
B EsDeBusqueda?()
// devuelve VERDADERO si el arbol cumple la condicion de ser de busqueda.
// FALSO en caso contrario
B EstaEn(E BASE e)
// devuelve VERDADERO si el valor de e coincide con el de algun nodo del
// arbol, FALSO en caso contrario.
```

19. Desarrollar el algoritmo Algoritmo BARBOL InPre_AB(ListaPosicional inOrder, preOrder), que toma como parámetros los recorridos en inorden y preorden de un árbol binario y devuelve el árbol que representan. Se suponen correctos tales recorridos y que el árbol no tiene elementos repetidos.
20. Implementar CrossRef, un gestor de listas múltiples: Un procedimiento Añade(l, palabra) insertará/añadirá a una lista de palabras l la palabra teniendo cada palabra de la lista enlazada una listilla de números con las posiciones en que entraron. Si la palabra ya estaba en la lista tan sólo añade a la listilla la posición en la que aparece. La lista de palabras se mantiene ordenada. Ir pidiendo palabras al usuario añadiéndolas a la lista, hasta que meta una palabra vacía; mostrar entonces la lista ordenada de palabras y al lado de cada palabra las posiciones en que se insertaron.

21. Un mapa es una función de un conjunto llamado dominio en otro conjunto llamado rango. Así que un mapa define una colección dinámica de vinculaciones del dominio en el rango. A cada vinculación le llamamos lazo. Un número arbitrario de lazos puede crearse, modificarse y destruirse durante la vida de un mapa. Si no hay lazos, consideramos que el mapa está vacío.

    a) Diseña un TAD Mapa donde el dominio es un tipo enumerado (e1, e2, ..., en) cualquiera y el rango es cualquier tipo de datos. Los métodos a implementar son:

```
CrearMapa()%
// crea un mapa vacio
AgregarLazo(E Dominio d, E Rango r)
// Agrega un lazo al mapa
EliminarLazo(E Dominio d)
// Elimina un lazo del mapa
EstaAsociado(E Dominio d, S B conectados, S Rango r)
// pone lazo a VERDADERO y r al valor del Rango asociado a d si existe
// tal lazo. En otro caso pone lazo a FALSO
B Algoritmo MapasIguales(E Mapa mp2)
// devuelve VERDADERO si son iguales, FALSO en caso contrario
```

    b) Si en la clase Mapa Dominio = Color = (blanca, roja, negra) y Rango = CARDINAL, podemos implementar la clase Urna. Un objeto de la clase Urna representa una urna con un número cualquiera de bolas blancas, rojas y negras. Este número es el elemento del rango asociado al color correspondiente en el dominio. Por ejemplo, una urna con dos bolas negras y una blanca estaría representada por un mapa con los lazos (negra,2) y (blanca,1).

    Utilizando los métodos ya implementados en la clase mapa, diseña:

```
A\~{n}adirBola(E Color c)
// A\~{n}ade una bola de color c a la urna
Algoritmo SacarBola(S Color c)
// Extrae una bola de la urna. Sup\'{o}n que existe una funcion
// Color Aleatorio()que devuelve un color de forma aleatoria al invocarla%
```