class ArrayList : TDAList{

private:

object array

int n //size

int capacity

public:

object anterior( posición )

return recupera(posición-1)

object anterior( posición ){--O(1)

si n>1 //validar rango

Si pos>=2 y pos<=n

return array[posición-2]

else

return null

else

return null

}

//mas elegante y sencillo

object anterior( posición ){--O(1)

Si pos>=2 y pos<=n

return array[posición-2]

else

return null

object siguiente( posición ){--O(1)

si n>=1 //validar rango

si posición >= n

return null

else

return array[posición]

else

return null

}

object siguiente(posicion){

return recupera(posicion+1)

}

bool suprime(posicion){

Si posicion >= 1 y <=n {

for i = posicion to n i++{

Si i == n {

Array[i-1] = null

n--;

Return true;

}else{

array[i-1] = array[i];

}

}

}else{

Return false;

}

bool append(item) --O(1)

return inserta(item,n+1)

bool append(item) – O(1)

//validar si hay capacidad en el arreglo

//y la lista puede seguir creciendo

array[n+1]=item

n++

return true

bool inserta(item, posicion){ --O(n)

si posicion>=1 and posicion <= n+1 //validar rango

si posicion == n + 1

array[posicion-1]=item

Else

for int i = n ; i > posicion-1; i--

array[i] = array[i-1]

array[posicion-1] = item

n++

return true

else

return false

}

object primero(){--O(1)

si n>=1 //validar rango

return array[0]

else

return null

}

object recupera(posicion){ --O(1)

//validar el rango

Si posicion >= 1 y <=n

return array[posicion-1]

else

return null

}

}

posicion localiza (item){

For i=1 to n i++{

Si item == array[i-1]

Return i

}

Return -1

}

posicion localiza (item){

For i=0 to n i++{

Si item == array[i]

Return i + 1

}

Return -1

}

void imprime\_lista()

for i = 0 to n

print(array[i]) //la función print muestra en pantalla el dato de la casilla, de //forma amigable para el usuario

void anula()

for i = 0 to n

array[i]=null

n=0