

Actividad 5.4

Lista contigua ordenada

Algoritmos y Estructuras de Datos

Tema 5: ordenación

Amplía la actividad 4.2 resuelta, programando una lista contigua redimensionable que se mantenga siempre ordenada.

Por lo tanto:

- No toques nada de la clase ListaContigua. En su lugar, crea una nueva clase llamada ListaContiguaOrdenada que sea hija de ListaContigua.
- La nueva clase permitirá insertar un elemento en la posición adecuada para que la lista contigua siga estando ordenada. El lugar adecuado para poner el elemento se buscará mediante la búsqueda binaria. Para recordar lo que es la búsqueda binaria recursiva, repasa la actividad 3.5.
- En ListaContiguaOrdenada programa un método privado recursivo que, mediante la búsqueda binaria, encuentre un elemento en la lista (devolviendo su posición) o, si no lo encuentra, devuelva la posición en que debería insertarse. El método devolverá una posición (entre 0 y n-1) y tendrá tres parámetros: el valor a buscar, la posición de inicio y la posición de fin. Se buscará ese valor en la lista actual, entre la posición de inicio y la posición de fin (ambas inclusive). Lógicamente, en cada sucesiva llamada recursiva se irá variando la posición de inicio y de fin.
- En la nueva clase redefiniremos el método público "buscar()" heredado de la ListaContigua, para que use internamente el método privado que implementa la búsqueda binaria recursiva. Igual que el "buscar()" de la clase padre, devolverá la posición en donde ha encontrado el elemento, o -1 si no lo ha encontrado.
- La nueva clase tiene que permitir también eliminar un elemento que se proporciona por parámetro (en vez de su posición, que es lo que ya permite ListaContigua). Internamente el elemento se localizará con el método privado que implementa la búsqueda binaria. Lógicamente, la precondition es que el elemento exista.

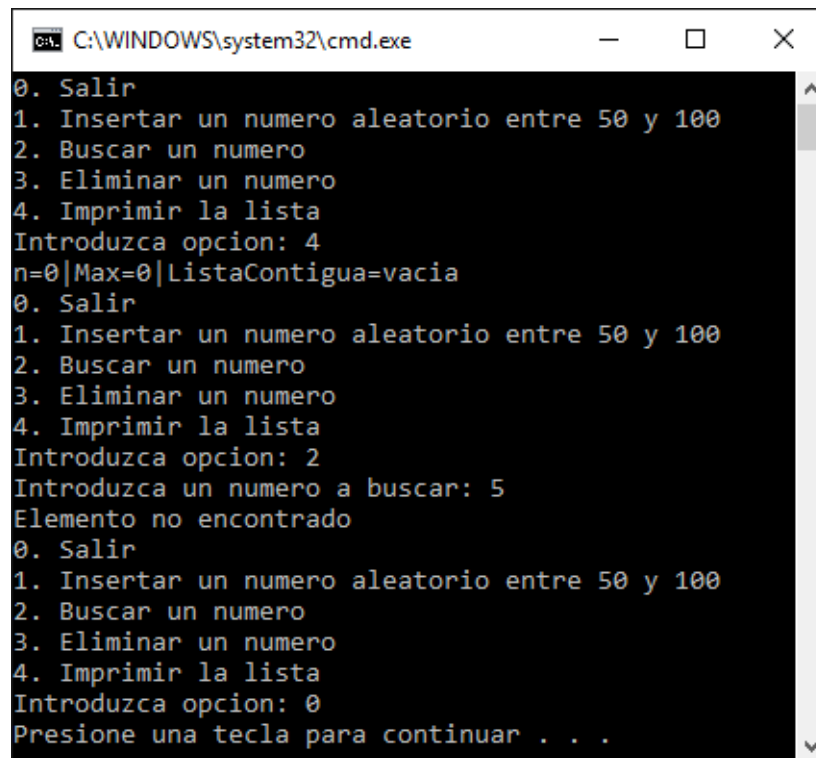
Realiza un main que pruebe la nueva clase. Tendrá el siguiente menú, el cual se repite constantemente:

1. Salir
2. Insertar un número aleatorio entre 50 y 100. Informa del número insertado.
3. Buscar un número que proporciona el usuario por teclado. Si lo encuentra, dirá su posición en la lista. Si no lo encuentra, informará de ello.
4. Eliminar un número que existe previamente. El número lo proporciona el usuario por teclado.

5. Imprimir la lista.

Prueba la aplicación como consideres más conveniente, según lo aprendido en el tema 4 sobre cómo hacer pruebas de caja negra. Pista: recuerda que tienes que probar todos los casos frontera, y finalmente todos los casos “normales”. Por lo tanto, una posible batería de pruebas de caja negra podría ser:

Con la lista vacía, prueba a imprimirla y buscar un número.



```
C:\WINDOWS\system32\cmd.exe
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 4
n=0|Max=0|ListaContigua=vacía
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 2
Introduzca un numero a buscar: 5
Elemento no encontrado
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 0
Presione una tecla para continuar . . .
```

Inserta un elemento. Con la lista con 1 elemento, prueba a imprimirla, buscar un número que no existe y que es menor que el que hay, buscar un número que no existe y que es mayor que el que hay, buscar el número que sí existe, eliminarlo, imprimir la lista.

```
C:\WINDOWS\system32\cmd.exe

0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 1
Numero 94 insertado correctamente
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 4
n=1|Max=2|ListaContigua=94
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 2
Introduzca un numero a buscar: 93
Elemento no encontrado
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 2
Introduzca un numero a buscar: 95
Elemento no encontrado
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 2
Introduzca un numero a buscar: 94
Elemento encontrado en la posicion 0
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 3
Introduzca un numero a eliminar (tiene que existir en la lista): 94
Elemento 94 eliminado correctamente
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 4
n=0|Max=2|ListaContigua=vacía
0. Salir
1. Insertar un numero aleatorio entre 50 y 100
2. Buscar un numero
3. Eliminar un numero
4. Imprimir la lista
Introduzca opcion: 0
Presione una tecla para continuar . . . █
```

Inserta dos elementos. Una lista con 2 elementos sigue considerándose un caso “frontera”, pues no tiene posiciones intermedias entre la primera y la última. Imprímela, busca un elemento que no existe y que es menor que el primero, busca un elemento que no existe y que está entre el primero y el segundo, busca un elemento que no existe y que es mayor que el último, busca el primer elemento, busca el segundo elemento, elimina el primer elemento, imprime, elimina el segundo elemento, imprime. Ya no se muestra el pantallazo, pues no se considera necesario.

Finalmente, inserta ocho elementos. Imprímela, busca un número que no existe, busca el primer número de la lista, elimínalo, imprime, busca un número intermedio de la lista, elimínalo, imprime, busca el último número de la lista, elimínalo, imprime. Tampoco se muestra el pantallazo.