11 | Cola en arreglo

Meta

Que el alumno domine el manejo de información almacenada en una *Cola*.

Objetivos

Al finalizar la práctica el alumno será capaz de:

- Implementar el tipo de dato abstracto *Cola* utilizando un arrelgo.

Código Auxiliar 11.1: Cola en arreglo

https://github.com/computacion-ciencias/ed-cola-en-arreglo-cs

Antecedentes

Arreglo

Para programar una cola en un arreglo es necesario utilizar algunos trucos:

1. Se deben tener dos enteros indicando las posiciones del primer elemento (cabeza) y último elemento en la cola.

2. Los elementos se colocan a la derecha del último elemento, módulo la longitud del arreglo, siempre que haya espacios disponibles. Si no hay espacios, se debe cambiar el arreglo por uno más grande. Al hacer este cambio la cabeza quedará en la posición cero del arreglo nuevo.

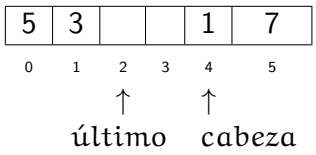

Figura 11.1 Cola en un arreglo, cuando ya se han eliminado elementos de la cabeza y se han formado elementos más allá de la longitud del buffer.

3. Los elementos se remueven de la posición de la cabeza y el indicador de esta se recorre a la casilla siguiente, a la derecha, módulo la longitud del arreglo. Un ejemplo se muestra en la Figura 11.1.

Iterador

La idea para implementar el iterador es semejante a la de la pila en arreglo, con las siguientes diferencias:

- El atributo que indica el siguiente elemento a devolver se inicializa en la cabeza de la cola, cuando el iterador es creado, cuidado porque ahora este valor no siempre es cero.
- Al recorrerse al elemento siguiente el indicador se incrementa en uno módulo la longitud del arreglo, pues el elemento siguiente al que está almacenado en la última casilla del arreglo se encontraría en la casilla cero.
- Para determinar si hay un elemento siguiente debe considerarse que el final de la cola puede estar antes o después de la cabeza. Hay varias formas de hacerlo, una es comparar el índice siguiente con el índice de la cabeza.

Desarrollo

En esta práctica se implementará la *Cola* utilizando arreglos. Por razones didácticas, no se permite el uso de ninguna clase que se encuentre en el API de C#, por ejemplo clases como `List<T>`, `ArrayList<T>` o cualquiera que haga el manejo de arreglos dinámicos.

1. Crea la clase `ColaEnArreglo<T>` en el proyecto `Cola` espacio de nombres

   `ED.Estructuras.Lineales.Cola`,

   que implemente la interfaz `ICola<T>`. Agrega los atributos y propiedades necesarias.

2. Crea un constructor que reciba como parámetro de tipo entero con el tamaño inicial desado para el buffer. El encabezado de tu constructor quedará:

```
1 public ColaEnArreglo(int tamInicial);
```

3. Crea otro constructor que no reciba parámetros. Asignarás un valor inicial para el buffer con un tamaño por defecto que tú puedes elegir; como se trata de un valor ligeramente arbitrario, la usanza es crear un atributo `static readonly`, es decir, una constante con el valor que hayas elegido de modo que, si luego quieres cambiar el valor, tu código no se vea afectado[1]. Puedes llamar a tu otro constructor para no repetir el trabajo:

```
1 public ColaEnArreglo() : this(TamInicial) {}
```

4. Programa el método `EstáVacía`, te ayudará a que el código de los métodos siguientes sea más legible.

5. Programa el método `Mira` suponiendo que la cola puede estar vacía o ya tener elementos. Lanza `InvalidOperationException` si la cola está vacía.

6. Programa el método `Forma` para agregar un elemento. Esta implementación no permitirá `null`, por lo que deberás lanzar `ArgumentNullException` si el argumento es `null`. Cuida el caso en que se inserta el dato a una cola vacía. Ojo: en este caso las dimensiones del arreglo no cambian si se llega al final, es posible que haya espacios vacíos al inicio del arreglo y deberás reutilizarlos antes que cambiar el tamaño del arreglo. Así puedes ahorrar tiempo, al no copiar a todos al nuevo arreglo. Verifica que funcione.

7. Programa el método `Atiende` para sacar un elemento. Deberás ir recorriendo la cabeza según sea necesario, dejando el hueco a su izquierda módulo la longitud del arreglo. Cuida el caso en que se saco el último elemento que estaba en la cola. Lanza `InvalidOperationException` si la cola está vacía. Verifica que funcione.

8. Agrega los métodos faltantes de IColección<T> y ICollection<T>:

   - `public boolean Add(E e)`, será sinónimo de `Forma`.
   - `public void Clear()`. Tendrás que borrar todos los elementos en el buffer, no dejes basura en él (objetos en el buffer que ya no están en la cola). Hay varias formas de hacerlo, puedes hacerlo a mano o usar el método `Atiende` aunque no hagas nada con el objeto devuelto; no vayas a recorrer todo el arreglo, sólo las casillas ocupadas.

[1] Por convención los nombres de las constantes se escriben con *Pascal case*, cada palabra inicia con mayúsculas.

- `public boolean Remove(T item)` Compara `item` sólo con el objeto devuelto por `Mira()`, si son iguales lo remueve, si no devuelve `false`. Si `o` es `null` lanza `ArgumentNullException`.

9. Programa el iterador implementando la interfaz `IEnumerator<T>`. Aunque en una cola sólo se pueden agregar y remover elementos en un extremo, necesitaremos un iterador que permitar ver todos los elementos en la cola, desde la cabeza hasta el último.

10. Implementa la versión no genérica del iterador utilizando `yield return` para ir devolviendo los elementos formados en la cola en el orden en que se encuetran.

11. Haciendo uso del iterador, es posible programar los métodos siguientes para cualquier colección en general:

    - `public bool Contains(T item)`. Puedes utilizar el iterador para ver si el elemento está en algún lugar de la cola.
    - `public void CopyTo(T[] array, int arrayIndex)`. Copia los elementos de la cola en el arreglo indicado.

Preguntas

1. ¿Qué técnica utilizas para detectar cuando la cola está vacía?

2. ¿Qué fórmula utilizas para detectar cuando el buffer de la cola está lleno?

3. ¿Cuál es la complejidad para el mejor y peor caso de los métodos `Mira`, `Forma` y `Atiende`? Justifica.