Tarea de investigación

Julio A. López Álvarez

Luis Fernando Solano Montoya

Universidad Cenfotec

Estructura de datos I

Christian Sibaja Fernández

Fecha: Junio, 2021

Resumen Ejecutivo o Abstract

La presente investigación busca brindar un resumen y ejemplos programados sobre las listas circulares, doblemente enlazadas, circulares doblemente enlazadas y las bicolas en C++ .

Palabras clave: Programación, C++, IDE, Clases, estructura de datos, colas, bicolas, listas, enlaces.

Tabla de Contenido

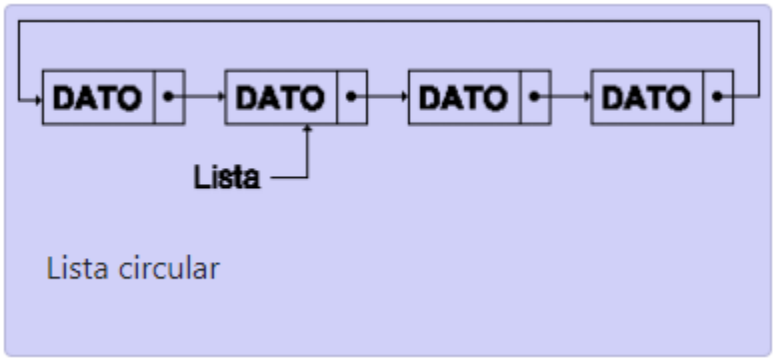
Capítulo 1 – Marco teórico

Lista circular

Definición

Esta estructura es una lista regular cuyo nodo final apunta al nodo inicial. Este tipo de listas permite evitar excepciones en las operaciones que se realicen sobre ellas. Cada nodo apunta al nodo siguiente como cualquier lista.

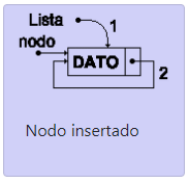
Seguidamente se muestra un ejemplo grafico de cómo funciona una lista circular:



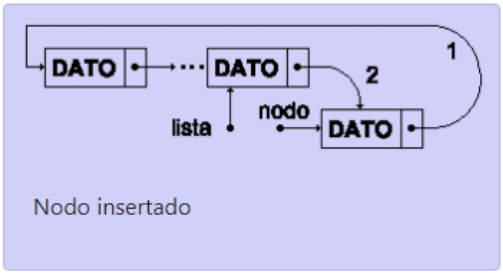
Operaciones básicas con listas circulares.

Añadir un elemento

Se puede realizar en una lista vacía o cuna con datos. En una lista vacía es muy sencillo, pues solo se debe crear el nodo que apunte a si mismo.



Si la lista posee datos, se parte de el nodo “cabecera” y se puede agregar al inicio de la lista (caso general) de este modo el nodo final apuntara al nodo nuevo y este apuntará al anterior nodo “cabecera”.



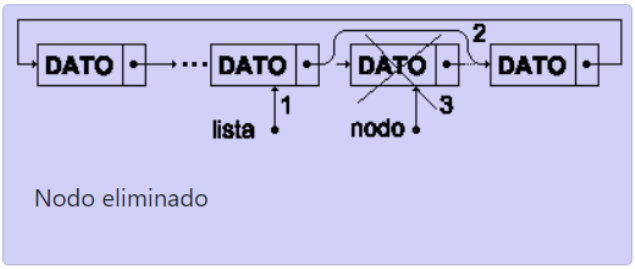
Esta función de puede mezclar con la función de “buscar” para poder insertar nodos en posiciones específicas, en listas ordenadas.

Buscar elementos

Funciona igual que la búsqueda en una lista regular, se debe almacenar el nodo “cabecera” en un auxiliar para poder recorrer la lista. Se debe tener precaución pues al ser circular, si no se controla, puede crear un ciclo de búsqueda sin fin, esto se puede lograr mediante un atributo “tamaño” que repita la validación de búsqueda solamente la cantidad de veces necesaria, dependiendo del tamaño de la lista. Se puede iniciar la búsqueda en cualquier punto de la lista pues al ser circular no hay inconvenientes.

Borrar elementos

Para esto, se debe controlar el nodo que se desea eliminar, esto se puede hacer mediante el método de búsqueda visto en la sección anterior. Para eliminar el nodo identificado, se debe almacenar el nodo anterior y el siguiente en auxiliares que permitan cambiar la referencia del nodo anterior al siguiente del nodo a eliminar



Lista doblemente enlazada

Definición

Se trata de una lista lineal, es decir en un solo sentido o no circular, en la cual cada nodo tiene 2 enlaces: uno al siguiente y otro al anterior. Estas se pueden recorrer en ambos sentidos desde cualquier nodo, el nodo “cabecera” tendrá como enlace anterior NULL. Un ejemplo de su instancia en C++ puede ser la siguiente:

Tarea II
6

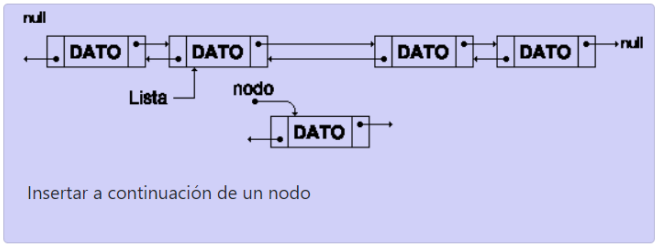
```
class Nodo {

   int dato;

   Nodo *siguiente;

   Nodo *anterior;

};
```

Operaciones básicas con listas doblemente enlazadas

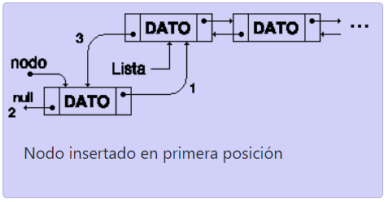
Añadir un elemento

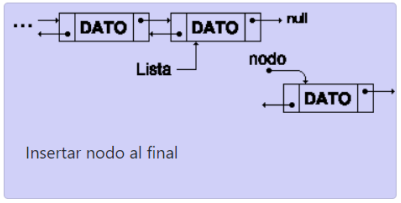
Para el proceso de agregar elementos se puede hacer de distintas maneras dependiendo de donde se quiera agregar el elemento, así, si es una lista ordenada puede utilizarse una función para ubicar la posición correcta y una vez ahí:

1. El nodo anterior debe apuntar como siguiente al nuevo elemento.
2. El nodo siguiente debe apuntar como anterior al nuevo elemento.



En caso de que se desee insertar al inicio, seria mas sencillo pues el nodo “cabecera” apuntaría como anterior al nuevo elemento, quien pasaría a ser el nuevo nodo “cabecera”. Caso contrario y aun más sencillo sería en la posición final, en cuyo caso el nodo final debe apuntar como siguiente al nuevo elemento.





Buscar elementos

Lo ideal para controlar y simplificar la búsqueda en listas doblemente enlazadas es manejar un Nodo “cabecera” de manera que la búsqueda pueda empezar en este y a partir de ahí implementar un método de búsqueda regular en listas sencillas. Caso contrario se debe:

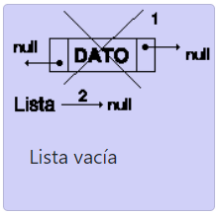
1. Recorrer la lista hasta llegar al inicio de la misma. Esto se logra obteniendo el nodo de enlace anterior del nodo de referencia hasta que encuentre el que tiene NULL como anterior.
2. Luego, se parte por el recorrido hacia delante mediante un bucle de búsqueda que compare el valor de cada nodo hasta encontrar el valor correcto.

Estas listas pueden implementar búsquedas en ambos sentidos porque también se podría agregar una validación del valor en tanto se encuentra el valor inicial y aprovechar así ese recorrido. Otro atributo que facilita la recursividad e implementación de métodos de búsqueda es el tamaño del arreglo.

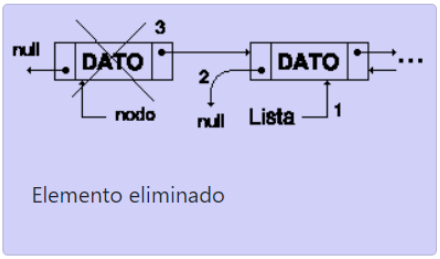
Borrar elementos

Este proceso puede presentar distintos escenarios:

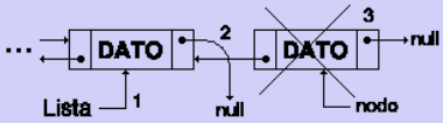
1. Eliminar el único nodo de la lista: Para esto solo se debe hacer que la lista apunte a NULL.



2. Eliminar el primer nodo: en este caso se debe asignar como nodo “cabecera” al nodo siguiente y hacer que este último apunte a NULL como anterior.

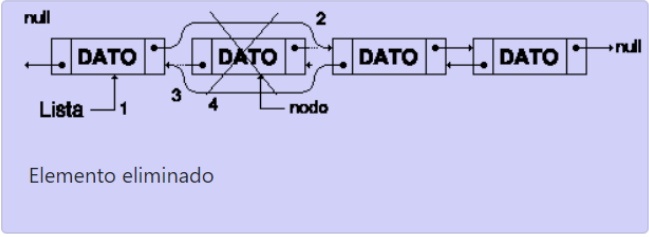


3. Eliminar el ultimo nodo: se debe ubicar el ultimo nodo, mediante una búsqueda y luego hacer que su anterior apunte como siguiente a NULL.



Elemento eliminado

4. Eliminar un nodo intermedio: en este se alarga un poco mas el proceso:
    a. Se debe ubicar el nodo a eliminar mediante una búsqueda.
    b. El nodo anterior debe apuntar como siguiente al nodo siguiente al elemento a eliminar y viceversa con el nodo siguiente.



Lista circular doblemente enlazada

Definición

En una lista enlazada doblemente circular, cada nodo tiene dos enlaces, similares a los de la lista doblemente enlazada, excepto que el enlace anterior del primer nodo apunta al último y el enlace siguiente del último nodo, apunta al primero. Como en una lista doblemente enlazada, las inserciones y eliminaciones pueden ser hechas desde cualquier punto con acceso a algún nodo cercano. Aunque estructuralmente una lista circular doblemente enlazada no tiene ni principio ni fin, un puntero de acceso externo puede establecer el nodo apuntado que está en la cabeza o al nodo cola, y así mantener el orden tan bien como en una lista doblemente enlazada.

Las listas circulares son más útiles para describir estructuras circulares y tienen la ventaja de poder recorrer la lista desde cualquier punto. También permiten el acceso rápido al primer y último elemento por medio de un puntero simple.

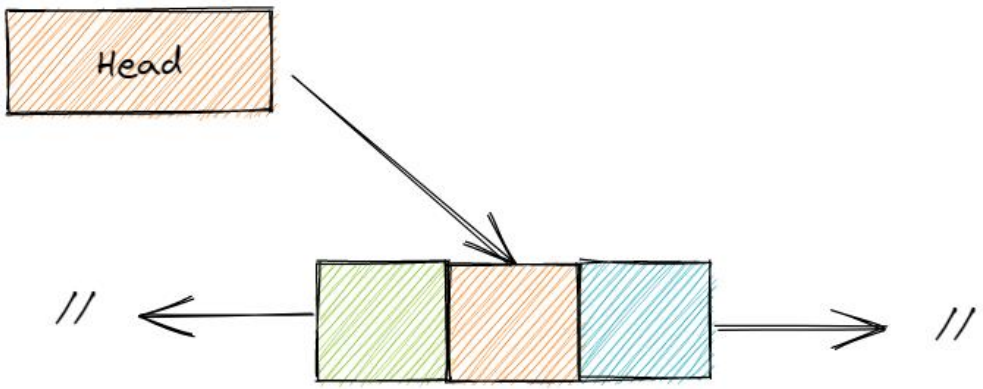
Operaciones básicas con listas doblemente enlazadas

Añadir un elemento

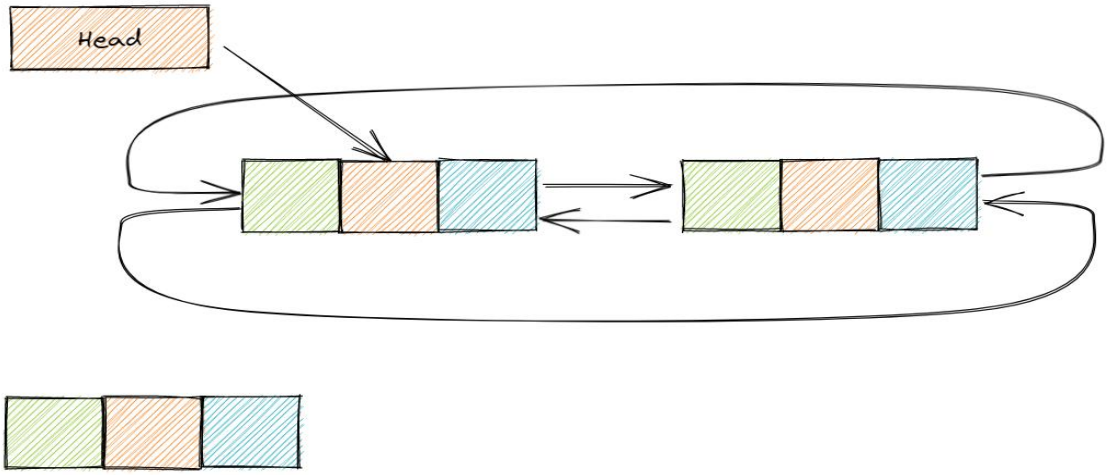
Lo que se hace es verificar que la cabeza no esté nula, si la cabeza esta nula insertamos el nodo en la cabeza, si la cabeza no está nula, el ultimo nodo en el siguiente empieza a apuntar al último, y la cabeza en el anterior apunta a este nodo.

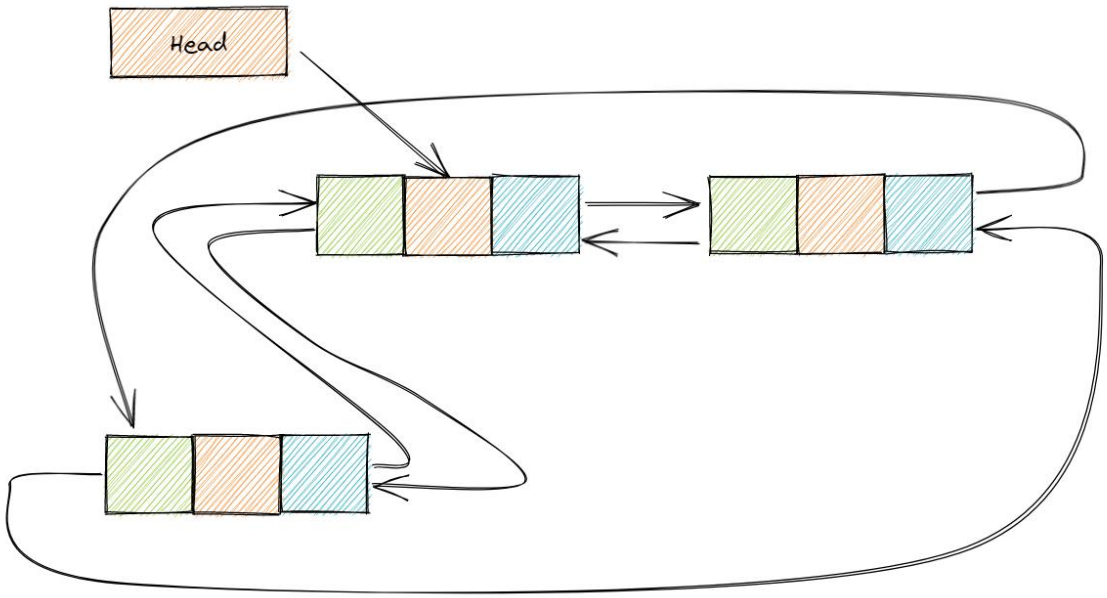
Escenario 1

Tarea II
10



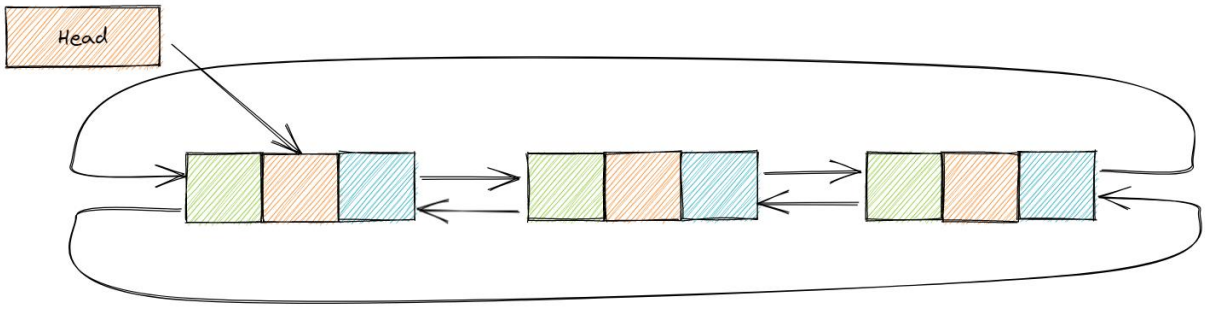
Escenario 2

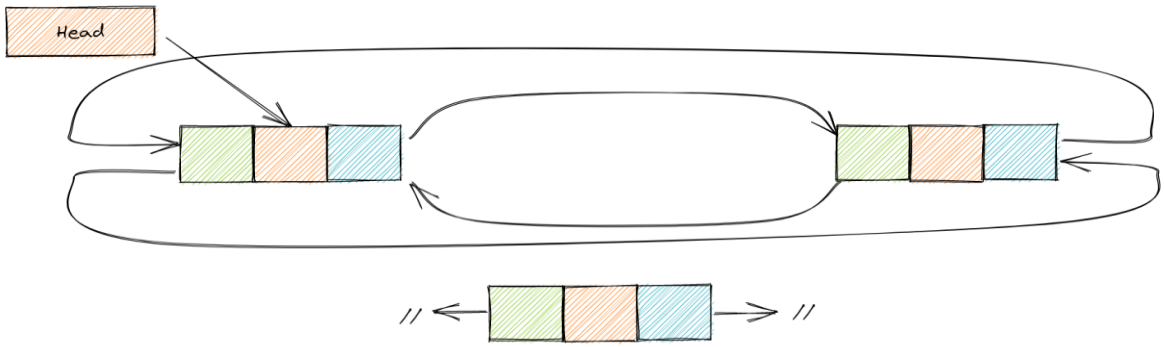




Borrar un elemento

Para borrar, se encuentra el elemento deseado, y si es cabeza se pone a apuntar la cabeza al siguiente y el ultimo empieza a apuntar a la nueva cabeza, la nueva cabeza empieza a apuntar al último elemento. Para elementos intermedios el siguiente y le anterior empiezan a apuntarse entre ellos y el nodo actual se elimina a las referencias.





Buscar un elemento

Para buscar el elemento solo se encuentra se empieza en la cabeza y se recorre hasta cuando volvamos a llegar a la cabeza para evitar un loop infinito.

Bicola

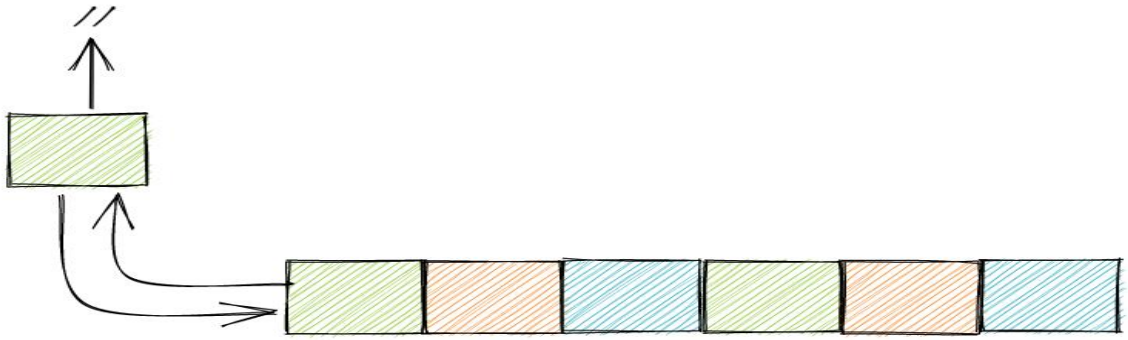
Definición

Una bicola es una cola bidimensional en la que las inserciones y eliminaciones se pueden realizar en cualquiera de los dos extremos de la lista, pero no por la mitad.

Agregar un elemento por delante

Para agregar adelante se revisa si la cabeza es nula, si es cierto se hace que la cabeza y el último elemento sea este nuevo nodo.

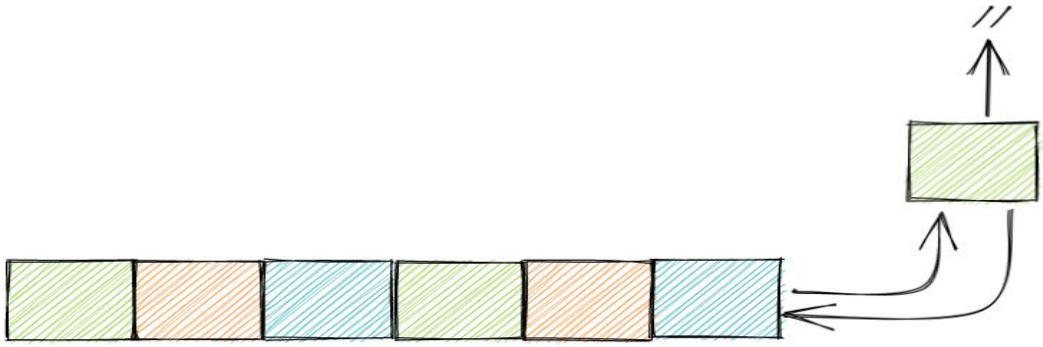
Si no, se hace que el anterior a la cabeza apunte a el nuevo elemento y se hace que el siguiente del nuevo elemento sea la cabeza. Luego se hace que la nueva cabeza sea el nuevo elemento.



Agregar un elemento por atrás

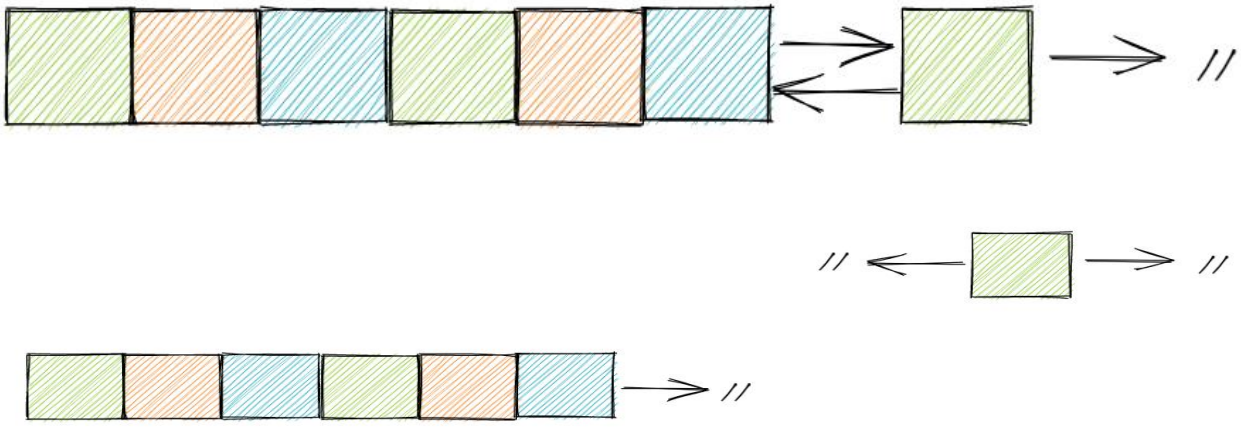
Para agregar adelante se revisa si la cabeza es nula, si es cierto se hace que la cabeza y el último elemento sea este nuevo nodo.

Si no, se hace que el siguiente del último elemento apunte a el nuevo elemento y se hace que el anterior del nuevo elemento apunte al último elemento. Luego se hace que el último elemento sea el nuevo elemento.



Sacar un elemento por atrás

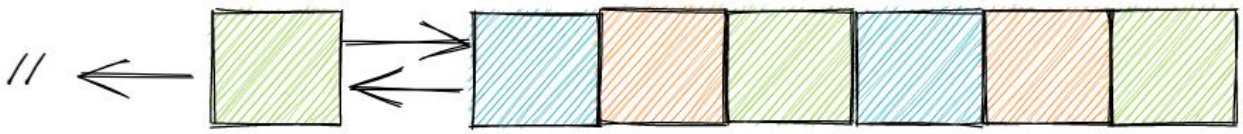
Se obtiene el último elemento, y se hace que el anterior del último elemento sea el último elemento. Luego se hace que este nuevo último elemento, el siguiente, se apunte a nulo, y se hace que el viejo último elemento, anterior, apunte a nulo, de esa manera se saca de la cola.

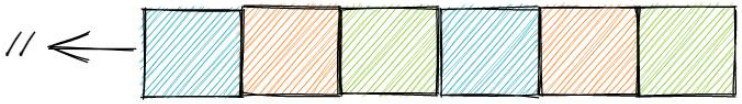


Sacar un elemento por delante

Se obtiene la cabeza, y se hace que el siguiente de la cabeza sea la nueva cabeza. Luego se hace que esta nueva cabeza, en el anterior, se apunte a nulo, y se hace que la vieja cabeza, siguiente, apunte a nulo, de esa manera se saca de la cola.

Tarea II
14





Capítulo 2 – Conclusiones

- Se determina que las listas son una estructura muy versátil y que permite aprovechar mucho la memoria del computador para almacenar de forma ordenada información. Permite generar mucha recursividad y reutilización del código en estructuras orientadas a objetos inclusive.

- Gracias a esta estructura es posible generar sistemas complejos que almacenan la información con diferentes tipos de enlace como analizo en las listas circulares y de doble enlace y la bicola.

Capítulo 3 - Lista de Referencias

Con Clase. (n.d.). *Con Clase*. Retrieved from http://conclase.net/c/edd/cap4

Cruz, M. (2011, 5 12). *https://blog.martincruz.me/2012/11/colas-dobles-en-c.html*.

JavaTpoint. (2021, 5 5). *https://www.javatpoint.com/circular-doubly-linked-list*.