Estructuras de Datos Lineales

MSc. Franci Suni Lopez
fsunilo@unsa.edu.pe

Ciencia de la Computación - UNSA

1

Objetivo de la clase

- Conocer, entender y diferenciar las principales estructuras de datos lineales.

2

2

Estructura de datos lineales

Elementos **accedidos de forma secuencial** pero no necesariamente almacenados en orden secuencial.

- Listas enlazadas
- Pilas
- Colas

3

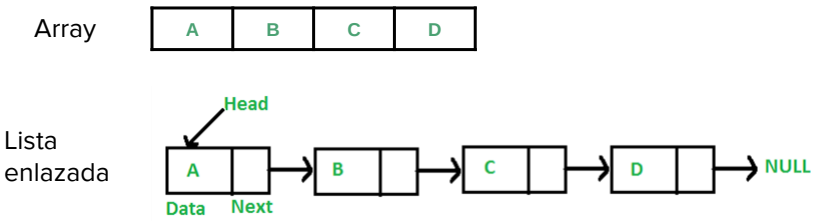
Listas enlazadas

4

Lista: definición

Una lista es una secuencia de cero o más elementos de un mismo tipo.

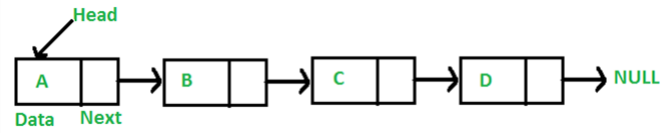
Digamos que el usuario ingresará una secuencia de letras, una por una. ¿Cómo almacenamos esas letras?



5

5

¿Qué es una lista enlazada?



Estructura de datos usada para almacenar un conjunto de datos del mismo tipo.

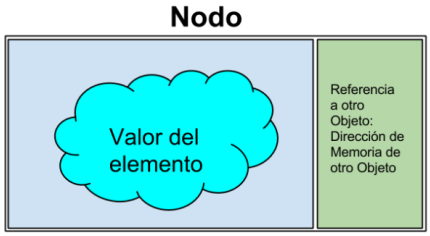
Propiedades

- Elementos sucesivos son conectados por punteros.
- El último elemento apunta a NULL.
- Incrementa/reduce su tamaño durante la ejecución.
- No desperdicia memoria.

6

6

Las listas enlazadas usan referencias



- **Valor del elemento:** información o dato que se desea almacenar en la lista.
- **Referencia:** puntero hacia otro objeto.

Por cada elemento nuevo es necesario crear un **nodo**

7

Lista: operaciones

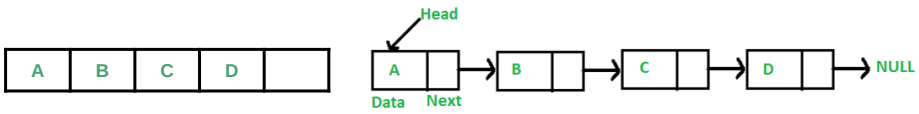
- **agregar(valor):** adiciona el valor al final de la lista.
- **insertar(valor, pos):** adiciona el valor en la posición pos en la lista.
- **remover(pos):** elimina el elemento que se encuentra en la posición indicada.
- **buscar(valor):** retorna la posición del elemento cuyo valor coincida con el especificado.

8

Operaciones: ejercicio

Supongamos que tenemos la secuencia de elementos:

A, B, C, D



Realice las siguientes operaciones:

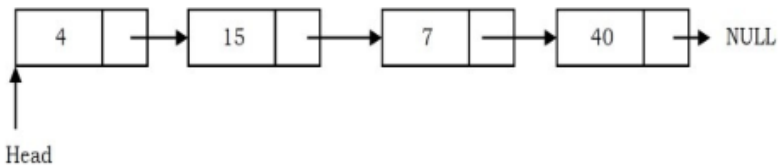
1) Agregar al final F.
2) Remover primer elemento.
3) Insertar al inicio X.
4) Remover elemento C.
5) Insertar H en la posición 2.
6) Remover último elemento.
7) Buscar D y Z.

9

9

Listas simplemente enlazadas

Secuencia de nodos, en los que se guardan campos de datos arbitrarios y una referencia al **siguiente nodo** (dirección de memoria).



Si el nodo no se encuentra enlazado a un nodo, el valor de la referencia será **NULL**.

Para tener acceso a todos los elementos de la lista sólo se requiere conocer la referencia al **primer nodo** (Cabeza).

10

10

Listas simplemente enlazadas

```
struct ListNode {
    int data;
    struct ListNode *next;
};
```

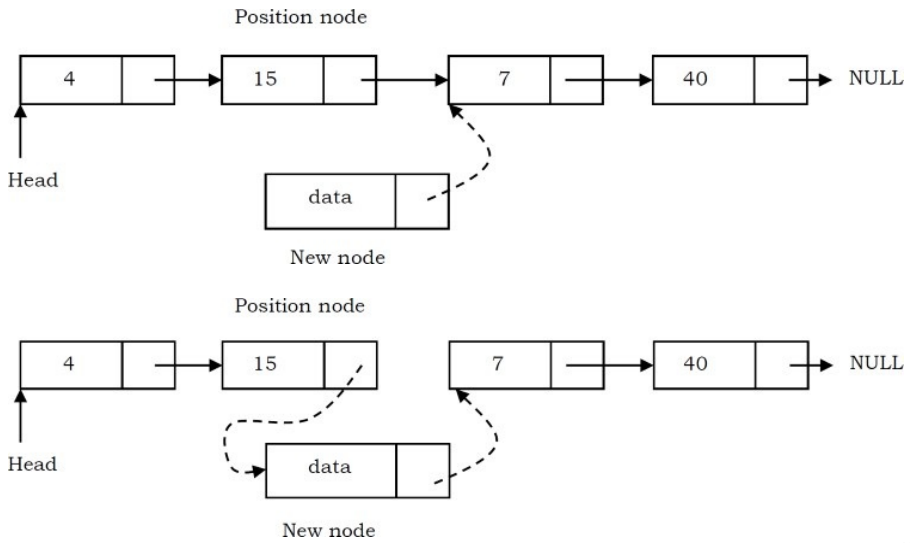
```
int ListLength(struct ListNode *head) {
    struct ListNode *current = head;
    int count = 0;

    while (current != NULL) {
        count++;
        current = current→next;
    }
    return count;
}
```

11

11

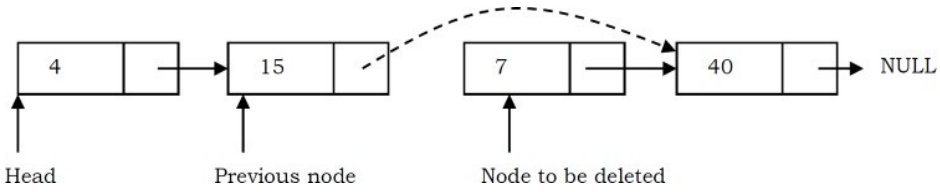
Listas simplemente enlazadas: insert

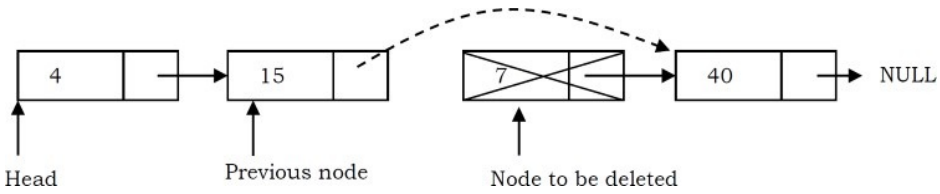


12

12

Listas simplemente enlazadas: delete





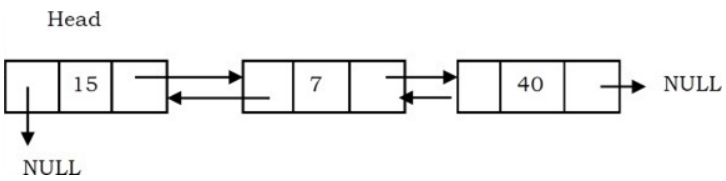
13

13

Listas doblemente enlazadas

Ventaja: podemos navegar en ambas direcciones.

Desventaja: mayor espacio por los punteros.



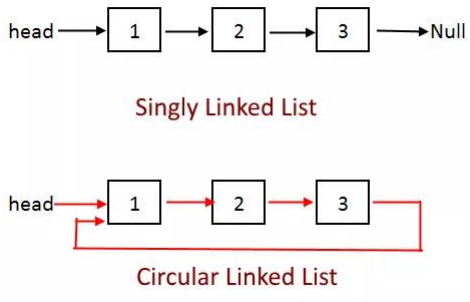

```
struct DLLNode {
    int data;
    struct DLLNode *next;
    struct DLLNode *prev;
};
```

14

14

Lista enlazada circular

Es una lista lineal en la que el último nodo apunta al primero.

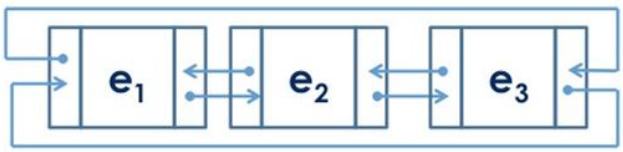


15

15

Lista doblemente enlazada circular

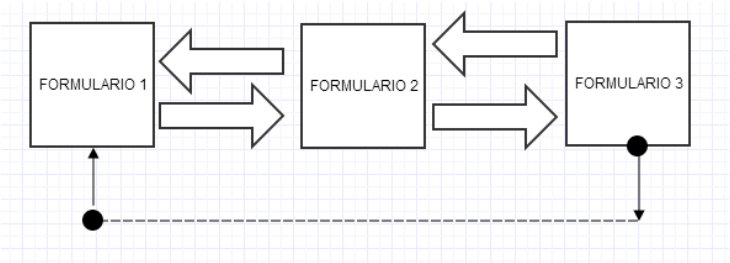
Permite recorrer la estructura en ambas direcciones.



16

16

- Un formulario electrónico al finalizar cada pagina te da la opción, atrás y siguiente que te permite rectificar cualquier error regresando atrás y adelante y al finalizar comienza nuevamente el formulario para el siguiente usuario



17

17

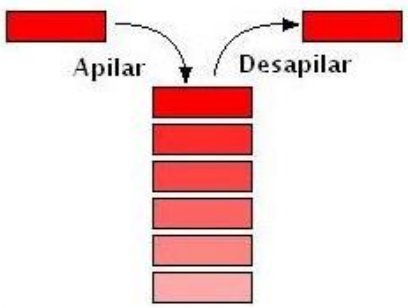
Pilas

18

18

Definición

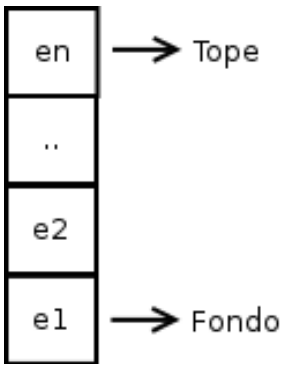
Una pila o *stack* es una lista ordinal en la que el modo de acceso a sus elementos es de tipo **LIFO** (Last In First Out) que permite almacenar y recuperar datos.



19

Operaciones

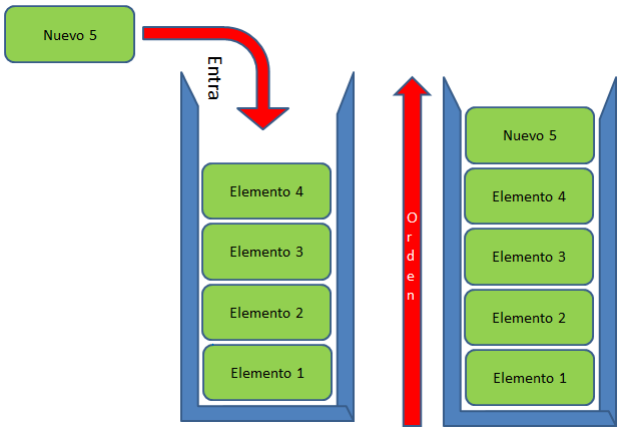
- **push(item):** coloca un elemento en la pila.
- **pop():** retira y devuelve el último elemento apilado.
- **top():** devuelve el elemento que está en la cima de la pila.



20

Operación: push

Los nuevos elementos son colocados en la cima de la pila.

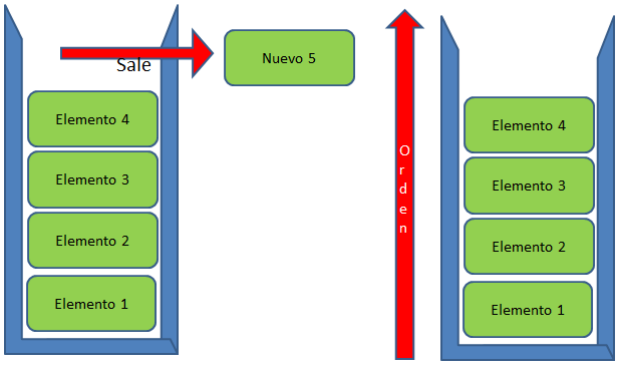


21

21

Operación: pop

El elemento que se retira es el que se encuentra en la cima de la pila.



22

22

Aplicaciones

- Implementación de llamadas a funciones, incluyendo recursividad.
- Análisis de expresiones en compiladores.
- Secuencia de “deshacer” en un editor de texto.

Implementación de una pila

Existen varias formas de implementar una pila, siendo los más comunes:

- Usando arrays simples.
- Usando arrays dinámicos.
- Usando listas enlazadas.

Implementación: usando array simple

Los elementos son agregados de izquierda a derecha.

Usamos una variable para mantener el índice del elemento **top**.

Excepciones: cuando la pila esta full o vacía.

25

Implementación: usando array dinámicos

¿Qué hacer cuando todos los espacios de una pila que usa un array de tamaño fijo están ocupados?

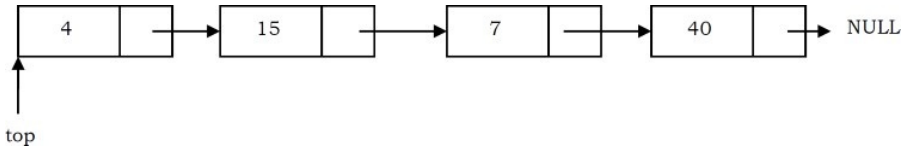
- Incrementar el tamaño del array en 1 cada vez que la pila esté full.

  Sería muy costoso por el número de copias que implica.

- **Duplicación de array:** cuando el array este full se crea un nuevo array con el doble de tamaño y se copian los elementos.

26

Implementación: usando listas enlazadas



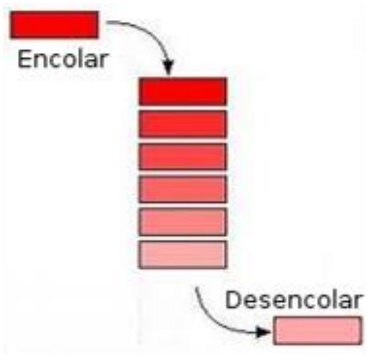
- PUSH inserta el elemento al comienzo de la lista.
- POP: elimina el elemento del inicio (nodo head/top).

Colas

Definición

Una cola o *queue* es una estructura de datos donde el modo de acceso a sus elementos es de tipo **FIFO** (First In First Out).

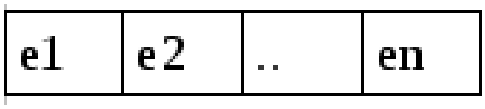
La **inserción** se realiza por un extremo y la **extracción** por el otro.



29

29

Operaciones



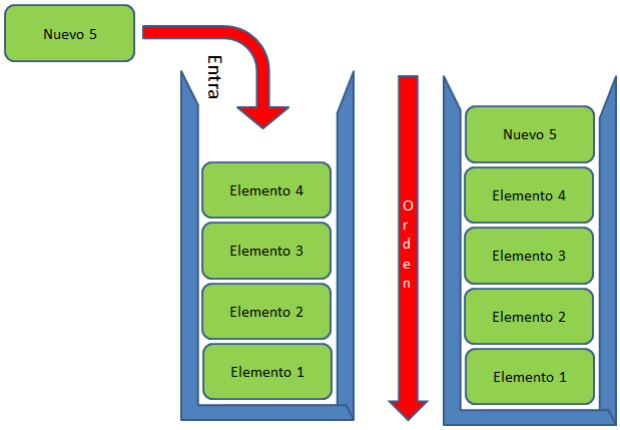
- **enqueue(item):** añade un elemento al final de la cola.
- **dequeue():** elimina y devuelve el elemento frontal de la cola (el primero que entró).
- **front():** devuelve el primer elemento de la cola sin removerlo.

30

30

Operaciones: enqueue

Añade el nuevo elemento al final de la cola.

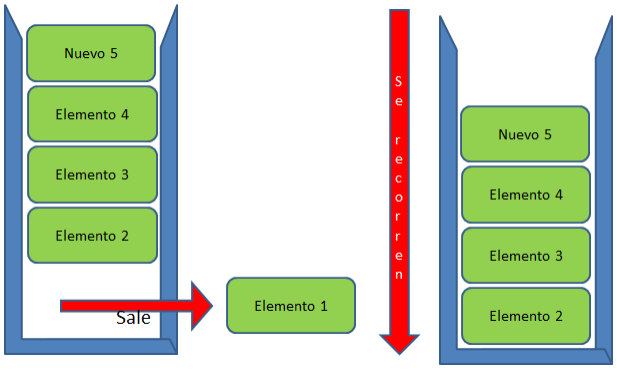


31

31

Operaciones: dequeue

Retira el primer elemento que ingresó a la cola.



32

32

Aplicaciones

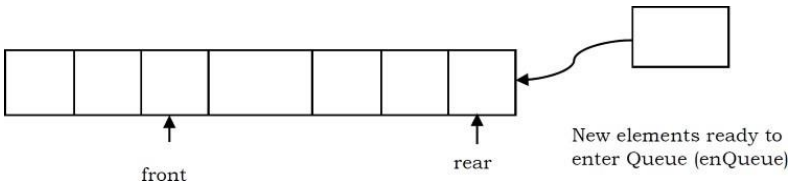
- Planificación de trabajos en los sistemas operativos.
- Transferencia de datos asíncronos.
- Simulación de colas del mundo real.

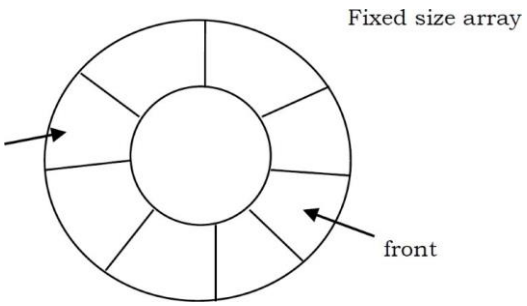
Implementación de una cola

Existen varias formas de implementar una cola, siendo los más comunes:

- Usando arrays circulares simples.
- Usando arrays circulares dinámicos.
- Usando listas enlazadas.

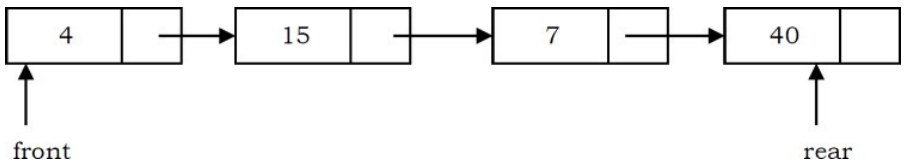
Implementación: usando array circular simple/dinámico





35

Implementación: usando listas enlazadas



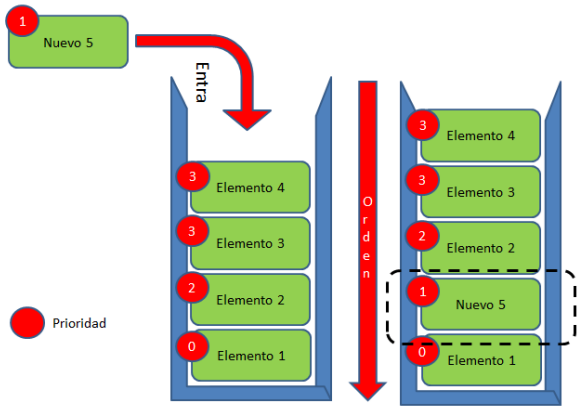
- ENQUEUE: inserta el elemento al final de la lista.
- DEQUEUE: elimina el elemento del inicio de la lista..

36

Colas de prioridad

Cuando un nuevo elemento entra en la cola, no siempre se coloca al final.

La cola **se mantiene ordenada** por orden de prioridad.

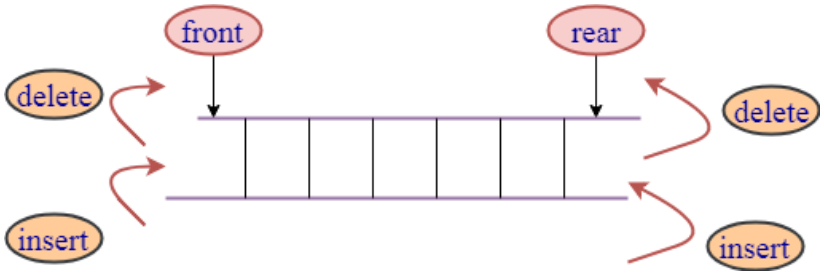


37

37

Deque (double-ended queue)

Estructura lineal en la cual los elementos pueden ser adicionados, consultados y eliminados por cualquiera de sus dos extremos.



38

38

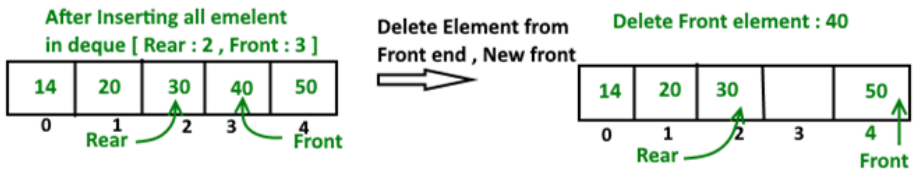
Deque: operaciones

- **addFront:** agrega un nuevo elemento al frente del deque.
- **addRear:** agrega un nuevo elemento al final del deque.
- **removeFront:** elimina el elemento del frente del deque y lo retorna.
- **removeRear:** elimina el elemento al final del deque y lo retorna.
- **isEmpty:** verifica si el deque está vacío.

Implementación de un deque

- Usando arrays circulares simples.
- Usando arrays circulares dinámicos.
- Usando listas doblemente enlazadas.

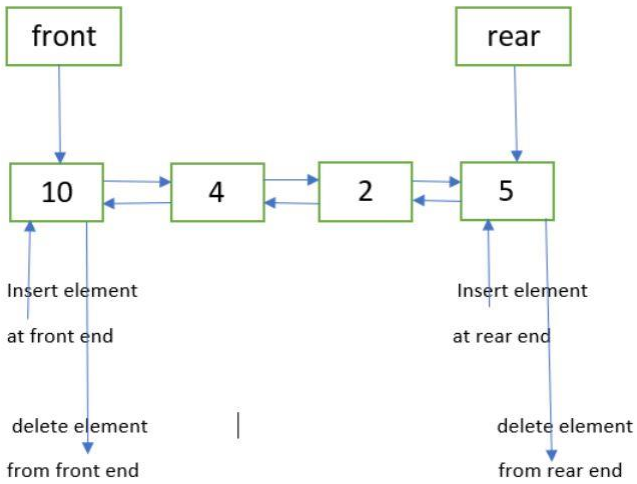
Deque: usando array circular simple/dinámico



41

41

Deque: usando lista doblemente enlazada



42

42