Universidad Nacional Autónoma de México
Facultad de Ingeniería
Algoritmos y estructuras de datos
Tema 3:
ESTRUCTURAS DE DATOS COMPUESTAS:
LISTAS LINEALES

3 Estructuras de datos compuestas: listas lineales.

Objetivo: Aplicar las formas de representar y operar en la computadora las principales listas lineales.

3 Estructuras de datos compuestas: listas lineales.

3.1 Generalidades.

3.1 Generalidades.

Los conjuntos son tan fundamentales para las ciencias de la computación como lo son para las matemáticas.

Los conjuntos manipulados por algoritmos pueden crecer, reducirse o cambiar en el tiempo, por lo tanto, se les denomina conjuntos dinámicos.

Dentro de los conjuntos dinámicos se pueden realizar diversas operaciones que se pueden agrupar en dos categorías: consultas y operaciones de modificación.

Las consultas simplemente regresan información del conjunto. Las operaciones de modificación pueden cambiar los elementos del conjunto.

A continuación se presenta una lista de operaciones típicas dentro de algún conjunto de elementos:

- **buscar (S, k): Dado un conjunto S, la consulta regresa el elemento x que coincida con conj[x] = k o nulo si no se encuentra coincidencia.**

- **insertar (S, x): es una operación de modificación que incrementa el conjunto S con el elemento x.**

- **eliminar (S, x): es una operación de modificación tal que dado un elemento x del conjunto S, se quita el elemento x del conjunto S.**

- **minimo (S): es una consulta dentro de un conjunto ordenado de elementos S que devuelve el elemento menor.**

- **máximo (S): es una consulta dentro de un conjunto ordenado de elementos S que devuelve el elemento mayor.**

- **sucesor (S, x): es una consulta que busca un elemento x dentro de un conjunto ordenado S y devuelve el elemento mayor a *x* o nulo si el elemento x es el mayor.**

- **predecesor (S, x): es una consulta que busca un elemento x dentro de un conjunto ordenado S y devuelve el elemento menor a *x* o nulo si el elemento x es el menor.**

El tiempo que toma ejecutar una operación en un conjunto S es proporcional al tamaño de entrada del conjunto.

Una estructura de datos consiste en una colección de nodos o registros que mantienen relaciones entre sí.

Un nodo es un elemento básico que mantiene la información en una estructura. A su vez, un nodo se puede dividir en campos para manipular con facilidad su información.

Se denominan estructuras lineales debido a que los elementos ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor.

Dentro de las estructuras de datos se pueden identificar fortalezas y debilidades.

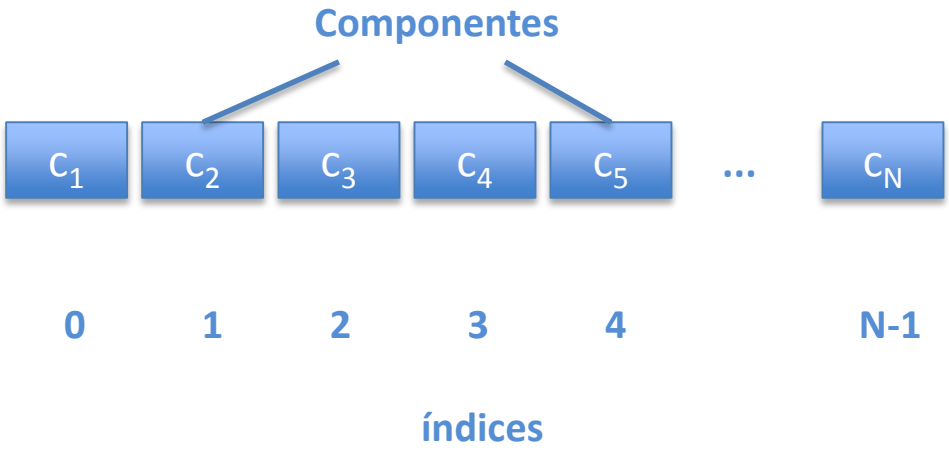
Estructura	Ventajas	Desventajas
Arreglo	Rápida inserción. Acceso rápido a la información si se conoce el índice.	Búsqueda lenta. Eliminación lenta. Tamaño fijo.
Arreglo ordenado	Búsqueda más rápida que el arreglo.	Inserción lenta. Eliminación lenta. Tamaño fijo.
Pila	Provee acceso directo al último elemento insertado.	Acceso lento a otros elementos.

Estructura	Ventajas	Desventajas
Cola	Provee acceso directo al primer elemento insertado.	Acceso lento a otros elementos.
Lista ligada	Rápida inserción. Rápida eliminación.	Búsqueda lenta.
Árbol binario	Búsqueda rápida. Inserción rápida. Eliminación rápida.	El algoritmo de eliminación es lento.
Árbol rojo-negro	Búsqueda rápida. Inserción rápida. Eliminación rápida.	Complejo.

Estructura	Ventajas	Desventajas
Árbol 2-3-4	Búsqueda rápida. Inserción rápida. Eliminación rápida. Árbol siempre balanceado.	Complejo.
Tabla Hash	Acceso rápido si se conoce la clave. Rápida inserción.	Eliminación lenta. Acceso lento si no se conoce la clave. Uso ineficiente de memoria.
Montículo (heap)	Inserción rápida. Eliminación rápida.	Acceso lento a otros elementos.
Gráfo o gráfica	Modela situaciones del mundo real.	Algunos algoritmos son lentos y complejos.

Arreglos

Un arreglo es la estructura de datos más elemental que existe. Consiste en un conjunto finito y homogéneo de nodos. Al conjunto de nodos se le identifica con un nombre y a cada nodo con un índice.

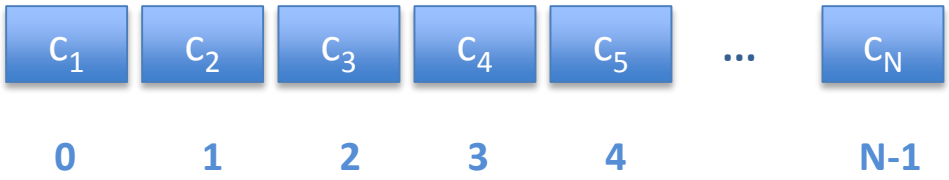


Un arreglo consta de un número finito de componentes así como de un valor o índice asignado a cada elemento.

Dentro de una estructura de datos tipo arreglo se pueden realizar las siguientes operaciones:

- **Lectura**
- **Asignación / Modificación**
- **Inserción**
- **Eliminación**
- **Ordenación**
- **Búsqueda**

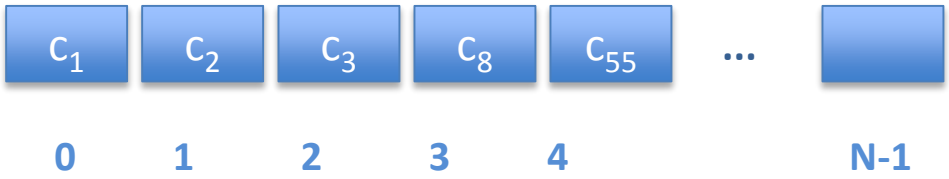
Para leer (obtener) un valor dentro de un arreglo solo es necesario especificar el índice asignado a dicho elemento. Por tanto, nombre[2] lee componente C_3.



Para asignar o modificar el valor de un nodo del arreglo es necesario indicar el índice del mismo y el valor que tomará el nodo:

nombre [3] ← c_{55}

Es posible insertar un valor dentro de un arreglo, sobre todo cuando los valores se encuentran ordenados. Por ejemplo, si se quiere insertar el elemento c_8 dentro del arreglo anterior se tiene que cambiar el elemento c_{55} a la posición próxima siguiente e insertar en su lugar el elemento c_8. El término insertar implica mover contenidos de los nodos, no insertar un nuevo nodo.



Para eliminar un valor simplemente se asigna un valor nulo al índice indicado. Es importante aclarar que no se elimina el nodo, simplemente su valor.

nombre [4] ← NULO

Es posible ordenar un arreglo tanto de manera ascendente como descendente (ordenación).

Así mismo, se puede buscar un valor dentro de los nodos del arreglo (búsqueda).

Las operaciones descritas (lectura, asignación o modificación, inserción, eliminación, ordenación, búsqueda) se pueden aplicar tanto a arreglos unidimensionales como a arreglos multidimensionales.

3.2 Pila.

3.2 Pila.

Una pila (o stack) es una estructura de datos lineal, la cual soporta las operaciones de agregar (push) y eliminar (pop) únicamente por uno de sus extremos.

El orden de eliminación es inverso al de agregación, es decir, el último elemento que se agregó es el primer que se elimina. A esta característica se le conoce como LIFO (Last-Input, First-Output).

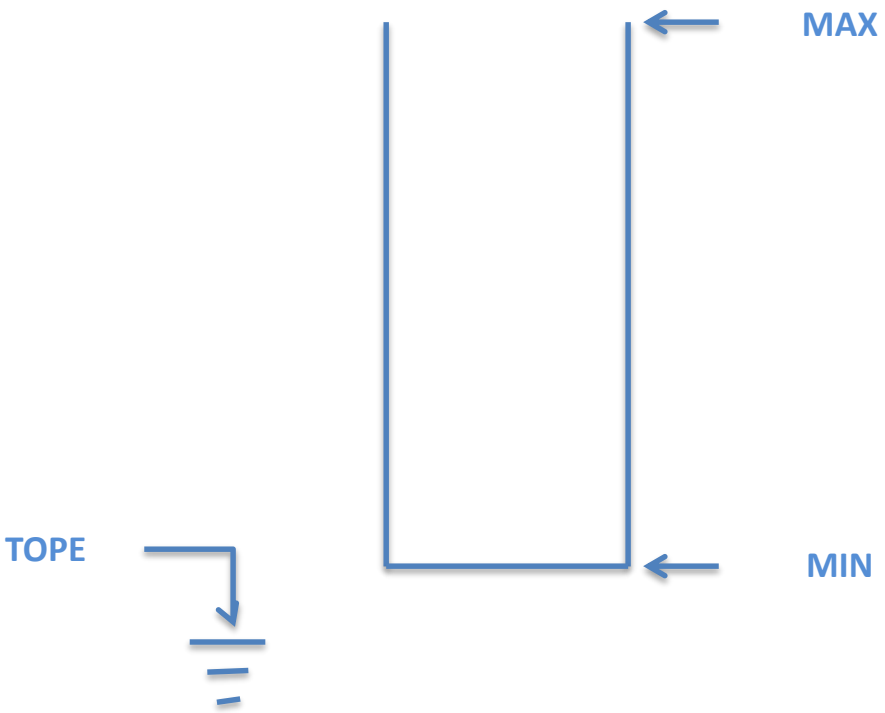
Las pilas son estructuras de datos lineales ya que cada cada nodo de la pila tiene un único predecesor y un único sucesor.

Por tanto, una pila es una colección de datos a los que se puede acceder mediante un extremo que se conoce como tope.

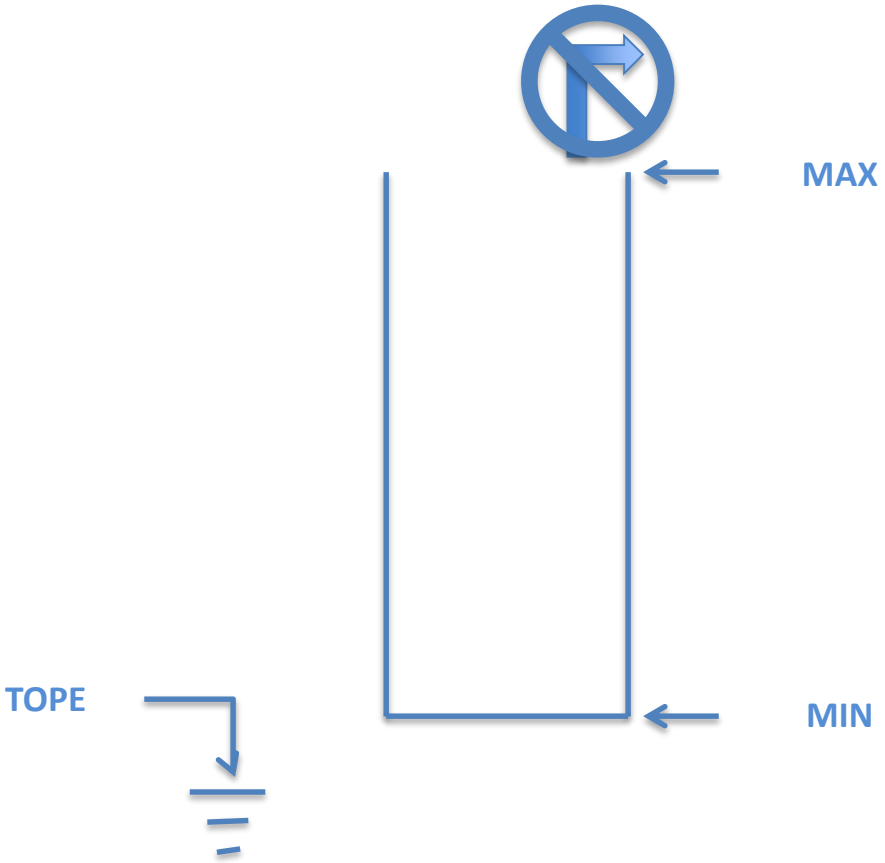
Las operaciones que se pueden realizar sobre una pila son insertar un nuevo elemento (push) y obtener el elemento que se encuentra hasta arriba de la pila (pop).

Para poder diseñar un algoritmo que defina el comportamiento de una pila se deben considerar 3 casos: cuando la pila está vacía, cuando la pila está llena y cuando la pila tiene algún(os) elemento.

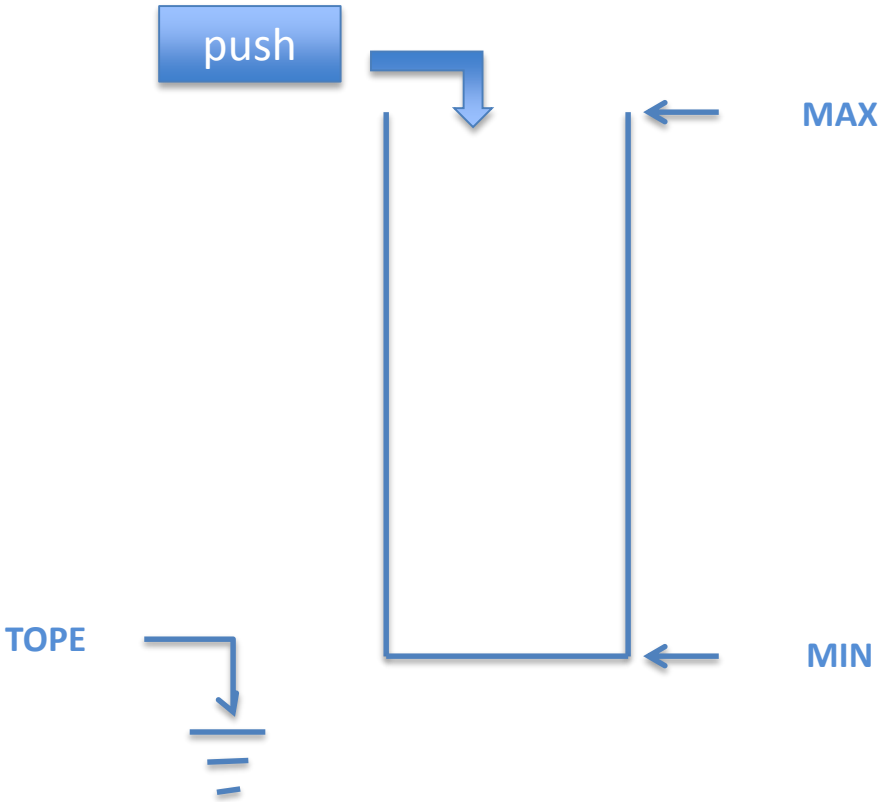
Pila vacía



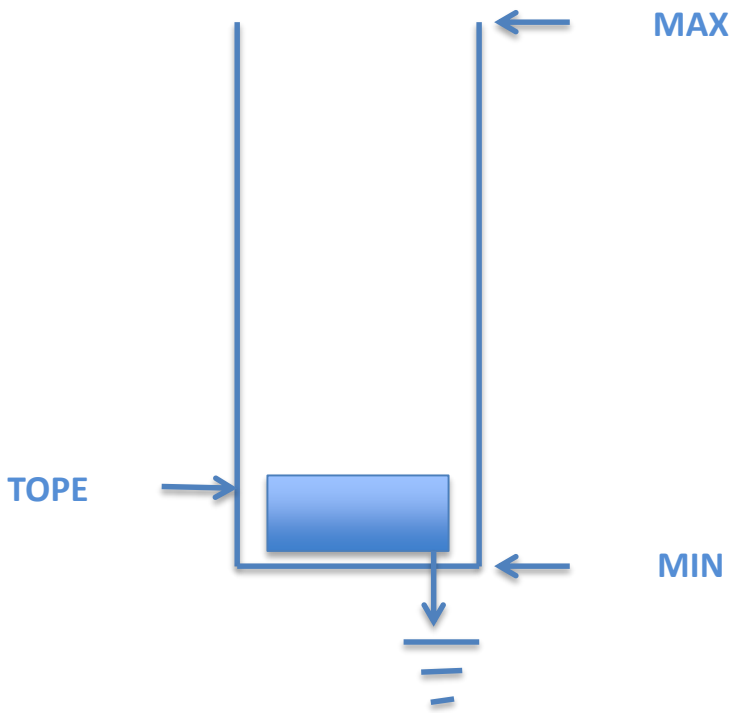
Pila vacía



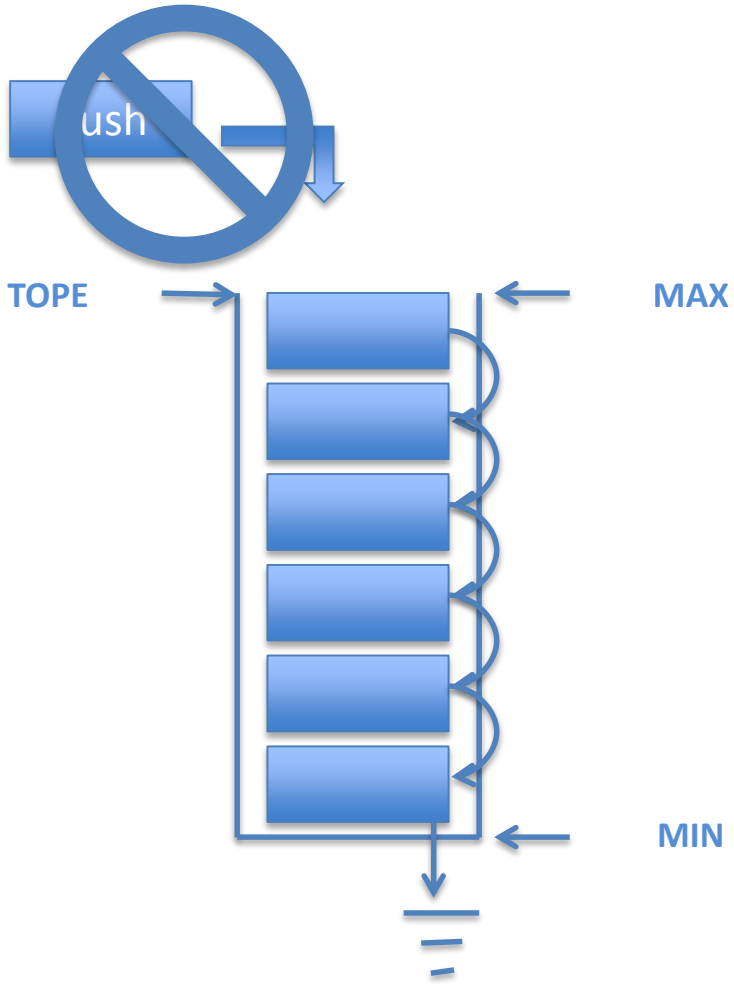
Pila vacía



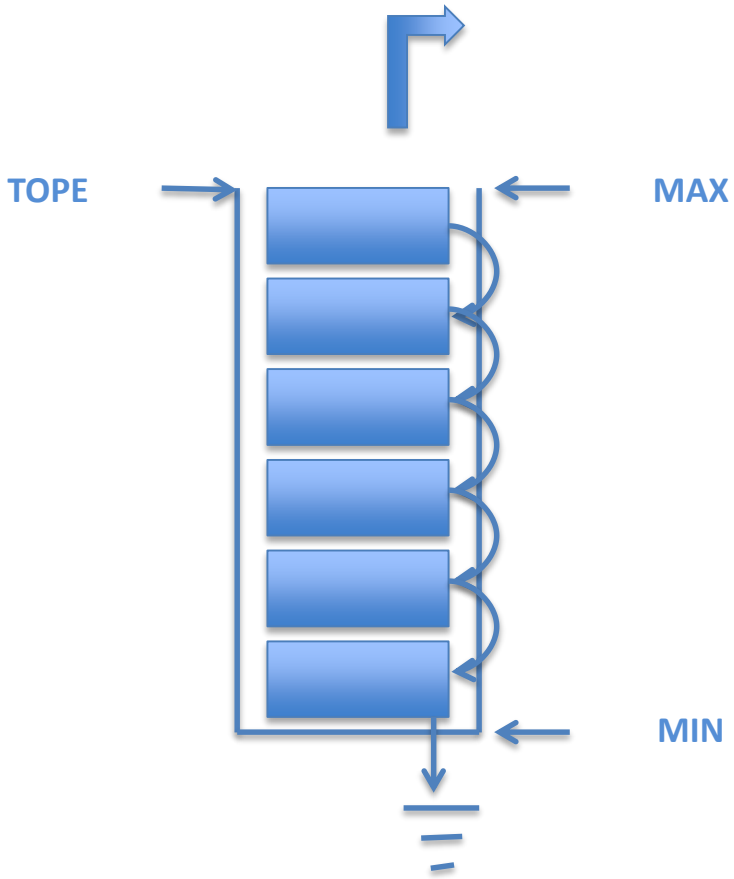
Pila vacía



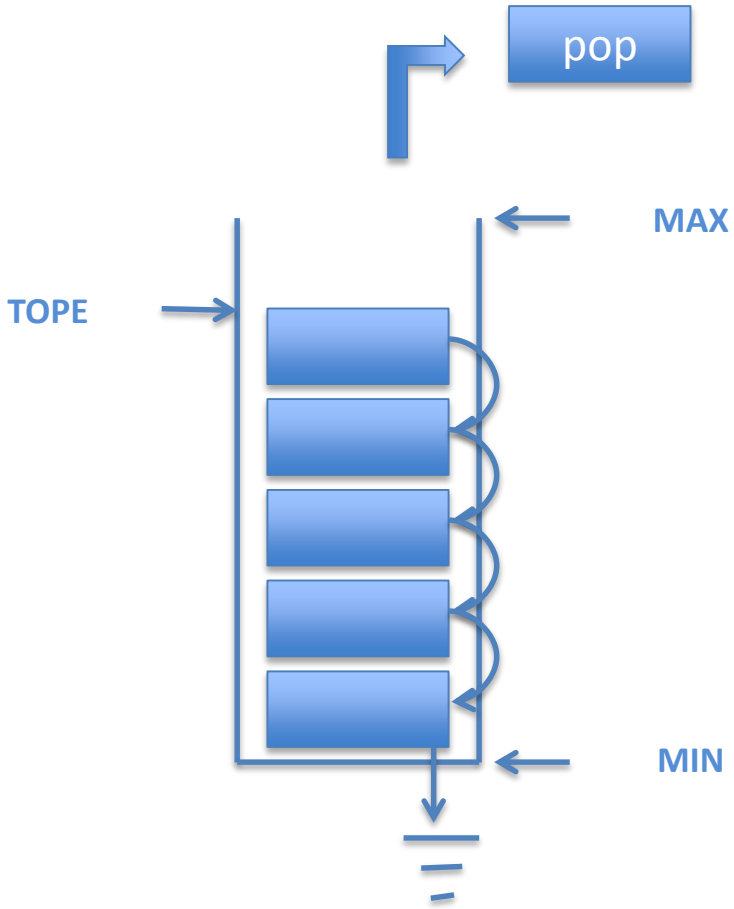
Pila llena



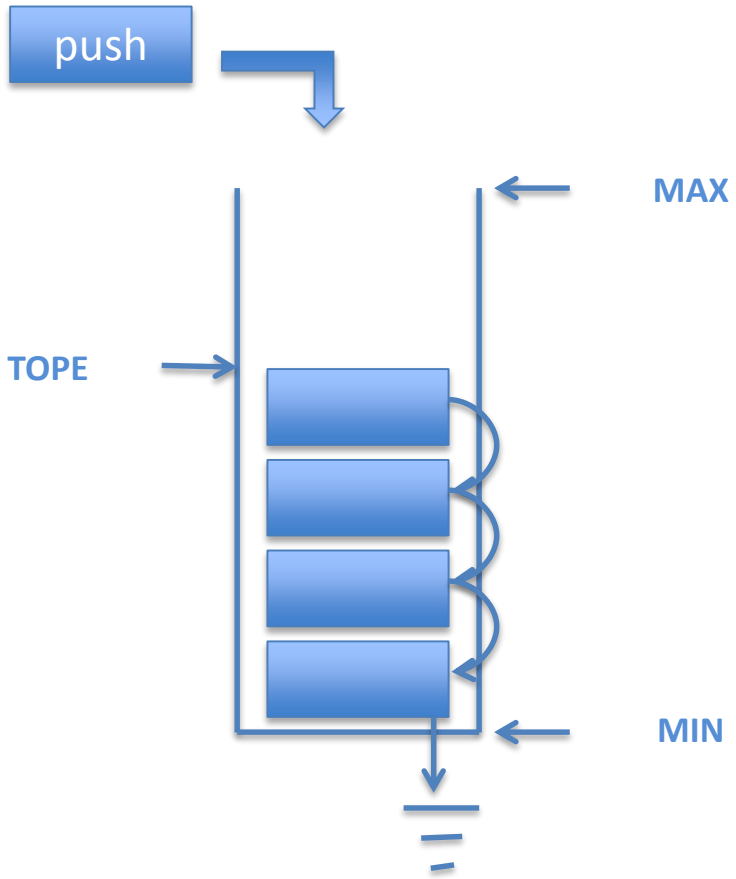
Pila llena



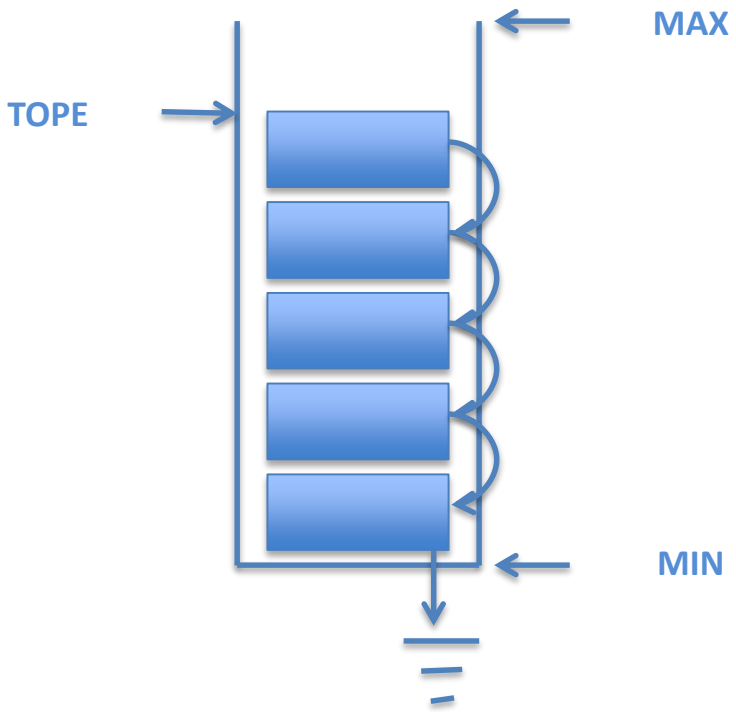
Pila llena



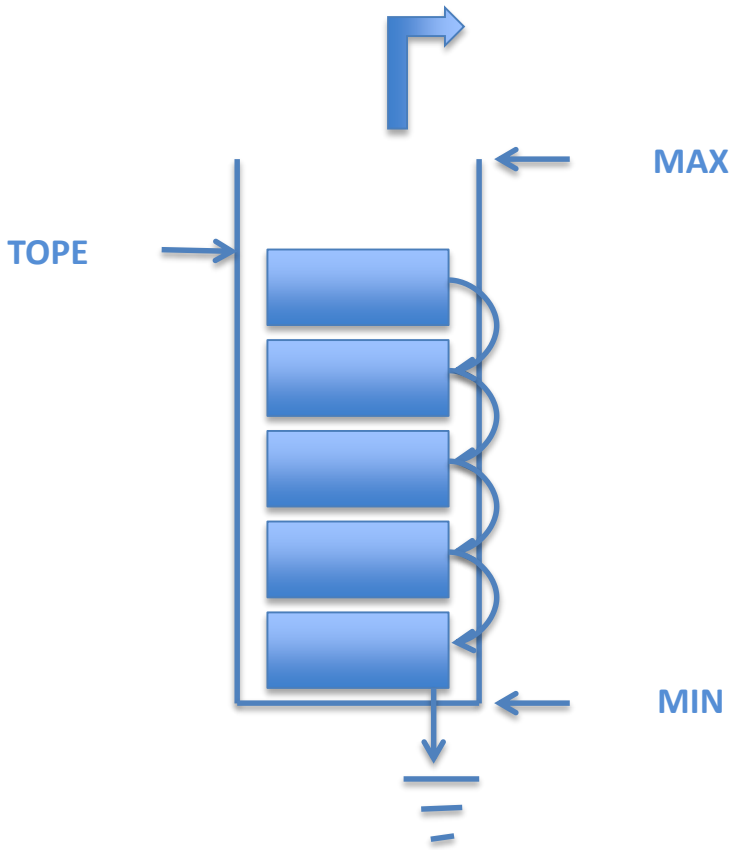
Pila con elementos



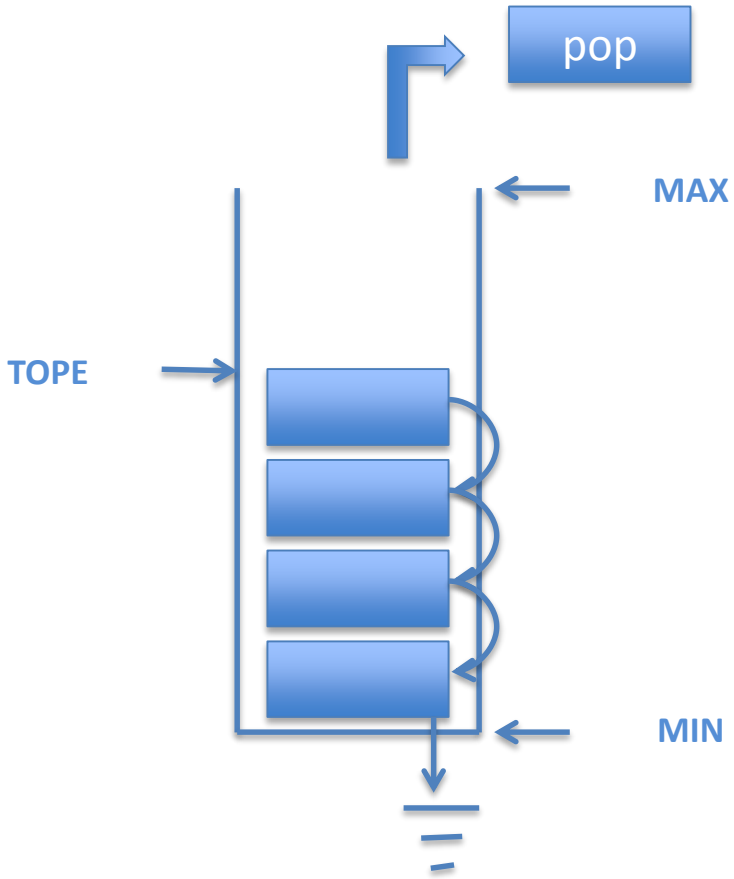
Pila con elementos



Pila con elementos



Pila con elementos



Tarea 1

Programar una estructura de datos tipo pila con las funciones POP, PUSH y MOSTRAR.

El programa que se debe entregar es una aplicación de la estructura.

3.3 Cola.

3.3 Cola.

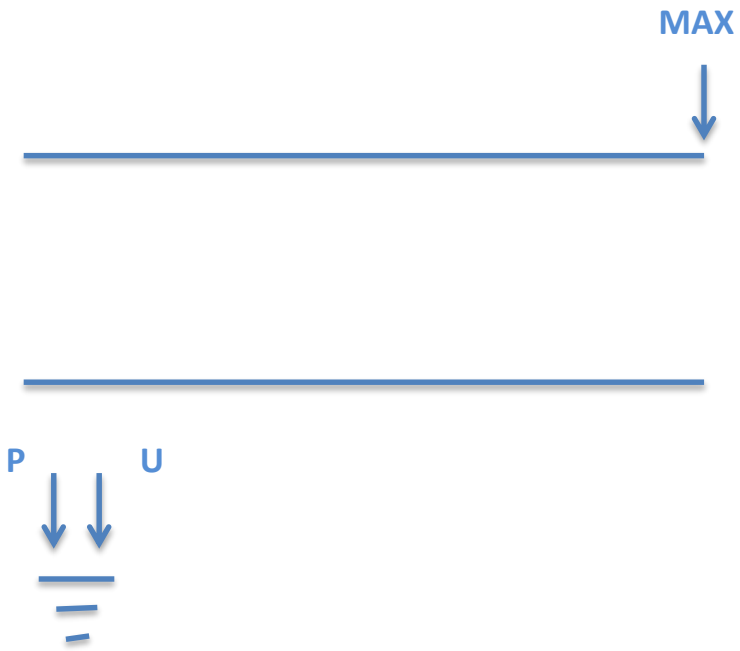
Una cola es una estructura de datos lineal en la que las operaciones se realizan por ambos extremos. Permite introducir elementos al final de la estructura y los ya existentes se obtienen por el inicio de la estructura.

Una estructura obtiene o elimina elementos en el mismo orden en el que los insertó, es decir, es una estructura FIFO (First-In, First-Out).

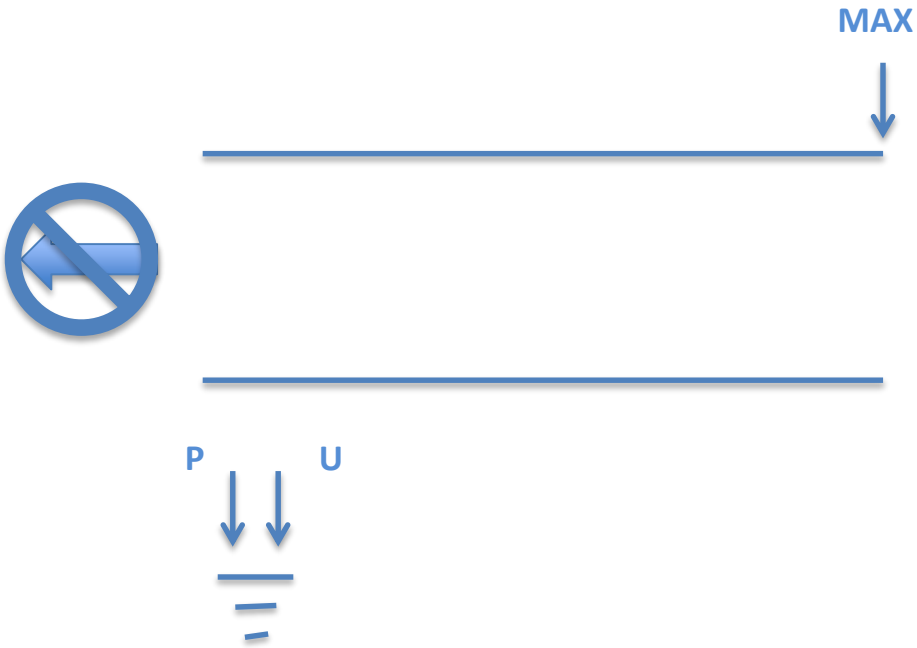
Las colas son estructuras de datos de tamaño fijo. Para su diseño hay que considerar los casos extremos así como el caso normal.

Los casos extremos se presentan cuando la estructura está vacía o cuando está llena. El caso normal es cuando se ingresa un nuevo elemento atrás de otro cuando todavía queda espacio.

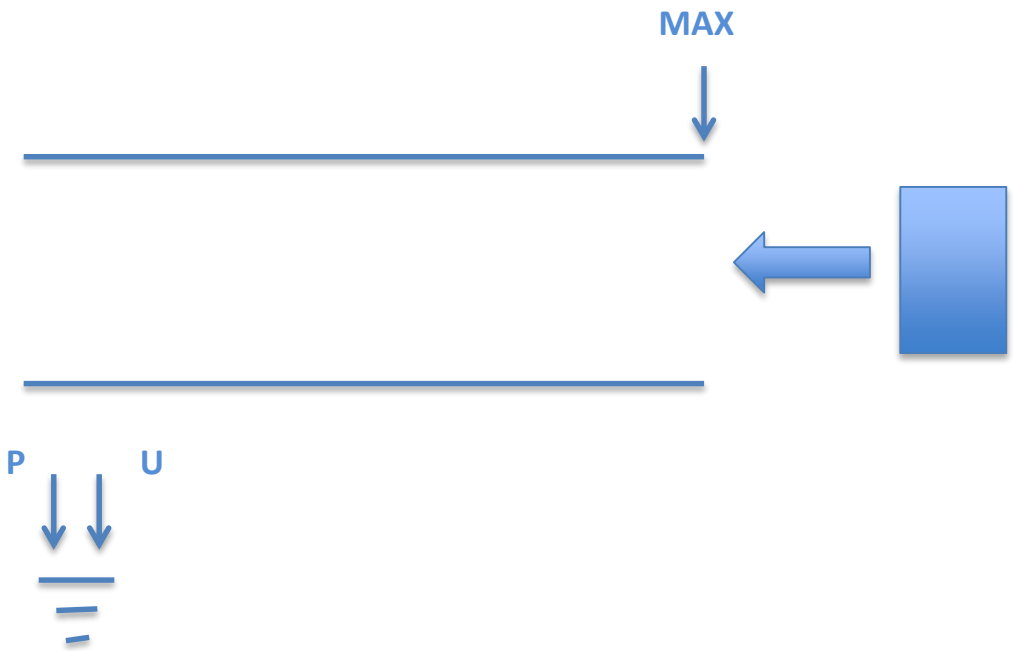
En una cola vacía tanto el apuntador al primer elemento como el apuntador al último elemento apuntan a nulo.



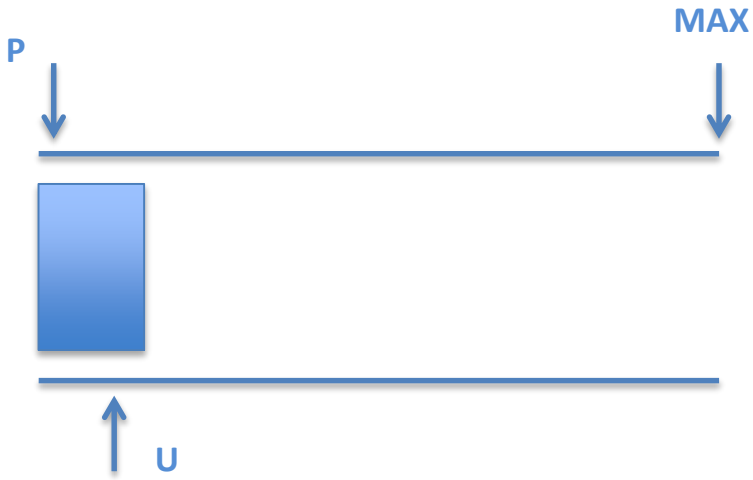
En una cola vacía no es posible extraer elemento alguno, por lo tanto, la función obtener debe validar que la cola no esté vacía.



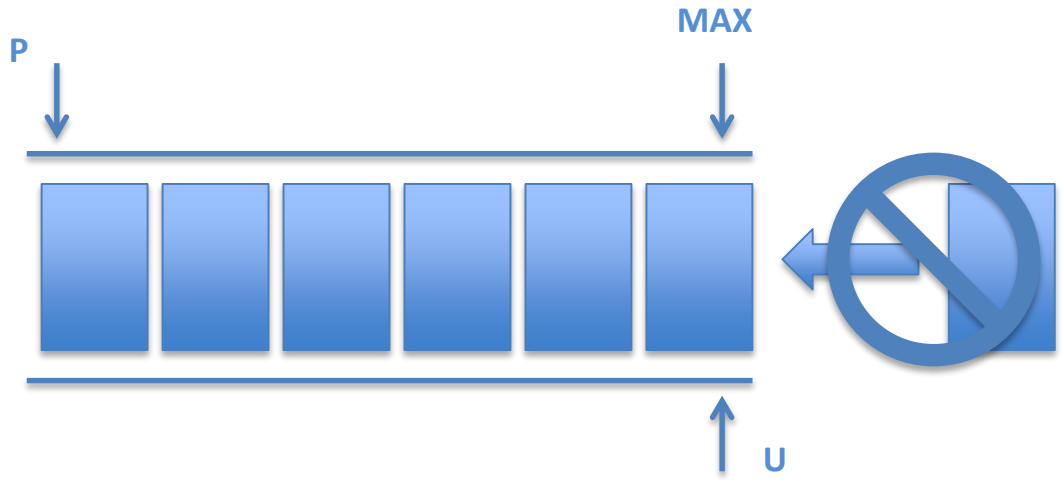
Si una cola está vacía es posible insertar un elemento. En este caso, los apuntadores primero y último apuntan al mismo elemento.



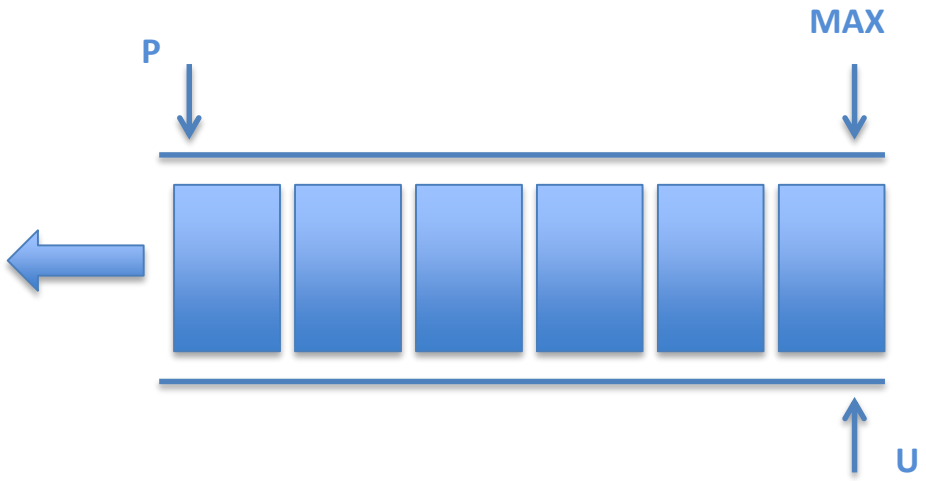
Si una cola está vacía es posible insertar un elemento. En este caso, los apuntadores primero y último apuntan al mismo elemento.



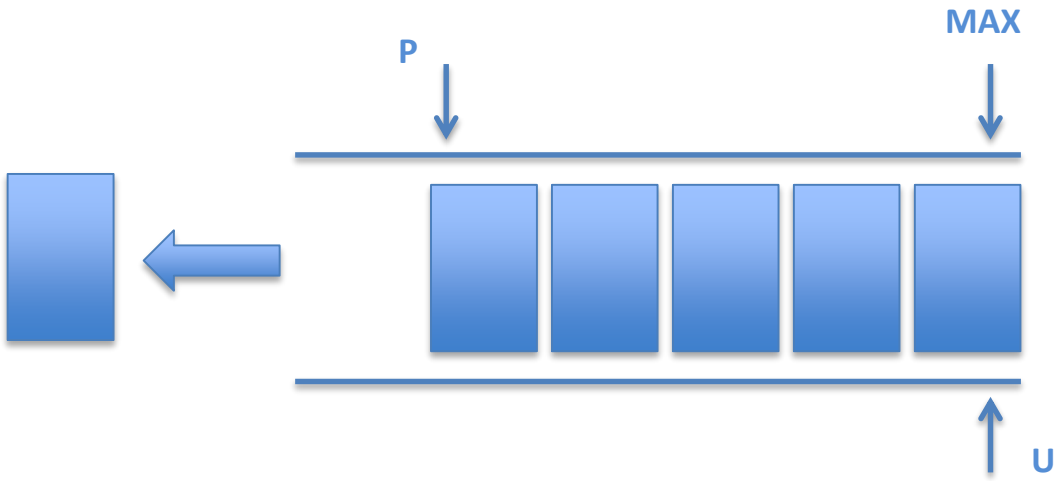
Cuando la cola se encuentra llena no es posible insertar un nuevo elemento.



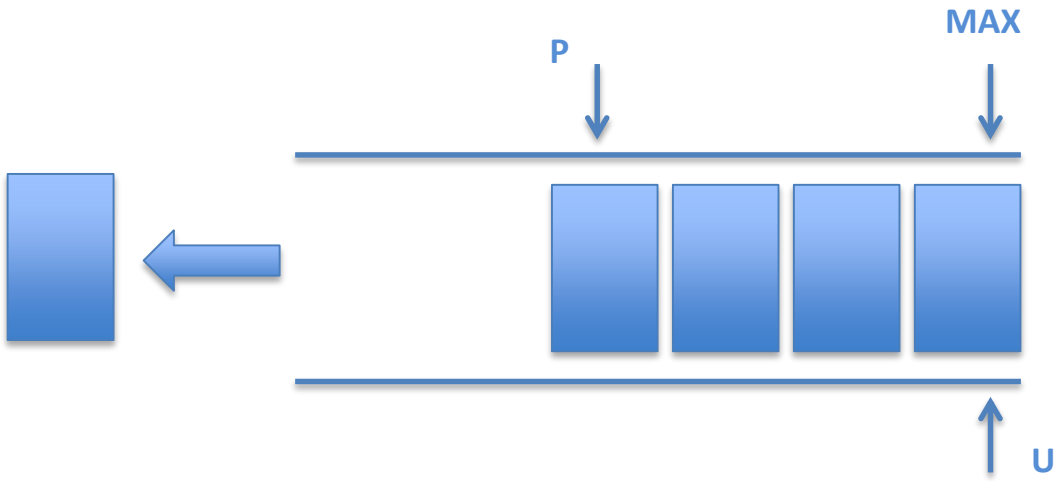
Cuando la cola se encuentra llena y se extrae un elemento, el apuntador del primer elemento se mueve a siguiente elemento.



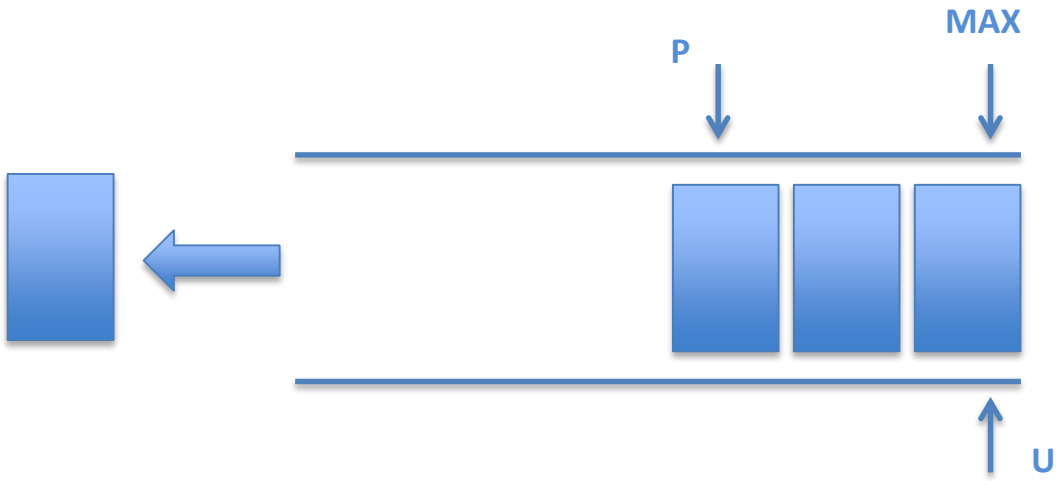
Cuando la cola se encuentra llena y se extrae un elemento, el apuntador del primer elemento se mueve a siguiente elemento.



Cuando la cola se encuentra llena y se extrae un elemento, el apuntador del primer elemento se mueve a siguiente elemento.



Cuando la cola se encuentra llena y se extrae un elemento, el apuntador del primer elemento se mueve a siguiente elemento.



Tarea 2

Programar una estructura de datos tipo cola con las funciones INSERTAR, EXTRAER y MOSTRAR.

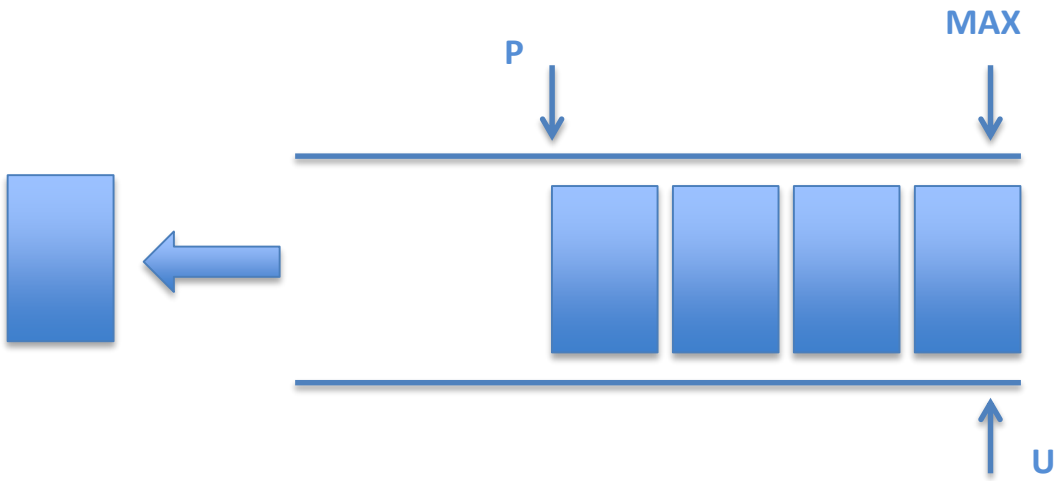
El programa que se debe entregar es una aplicación de la estructura.

Cola circular.

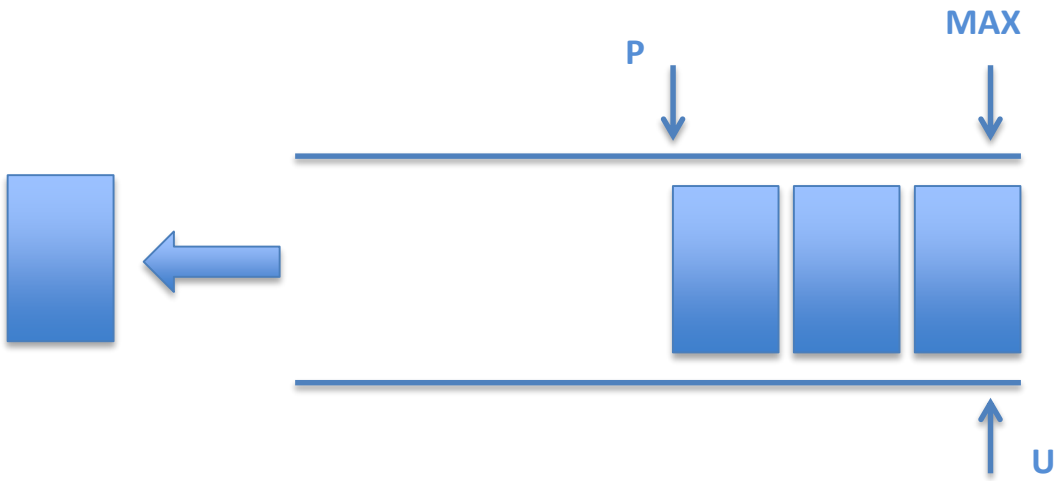
Una cola circular constituye una estructura de datos lineal en la cual el siguiente elemento del último es, en realidad, el primero.

Una estructura tipo cola circular utiliza de manera más eficiente la memoria que una cola simple.

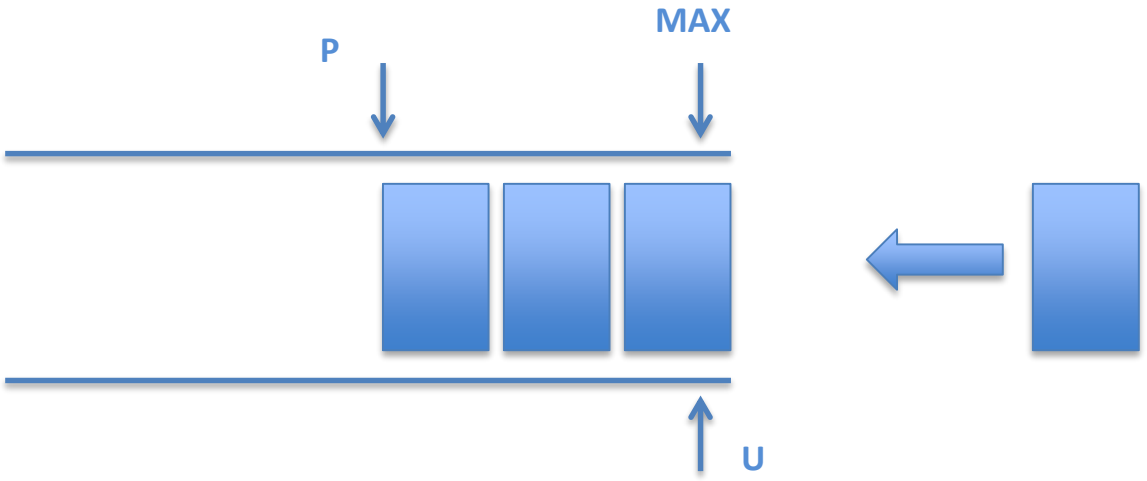
Obtener



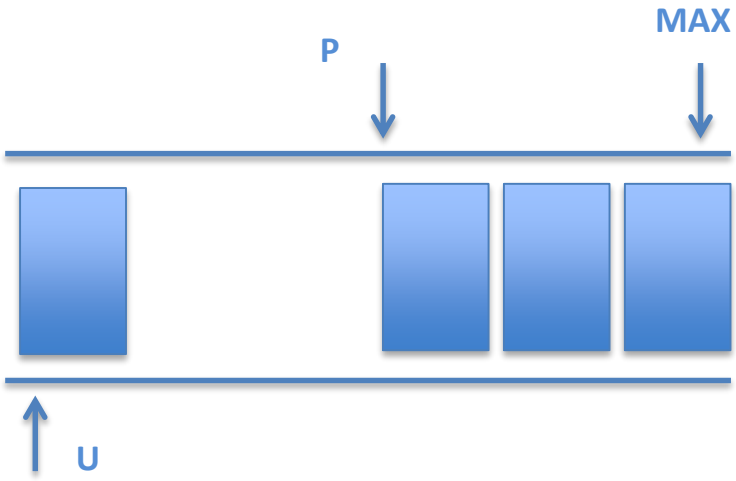
Obtener



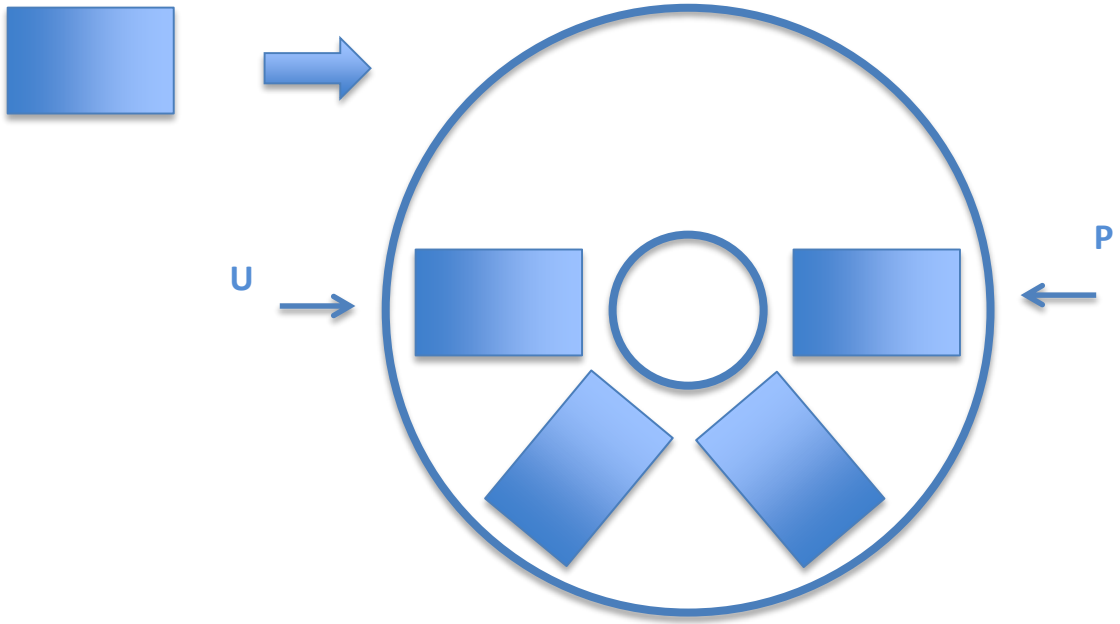
Insertar



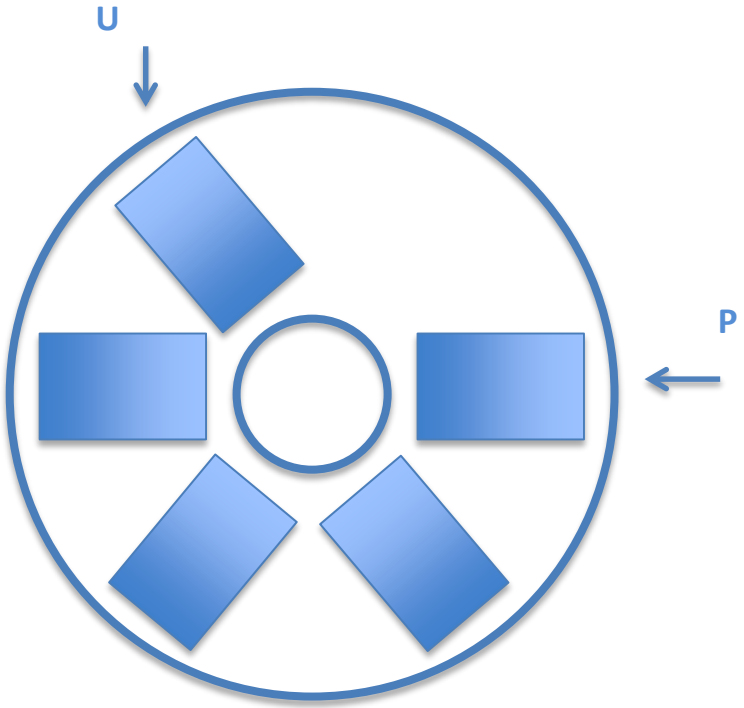
Insertar



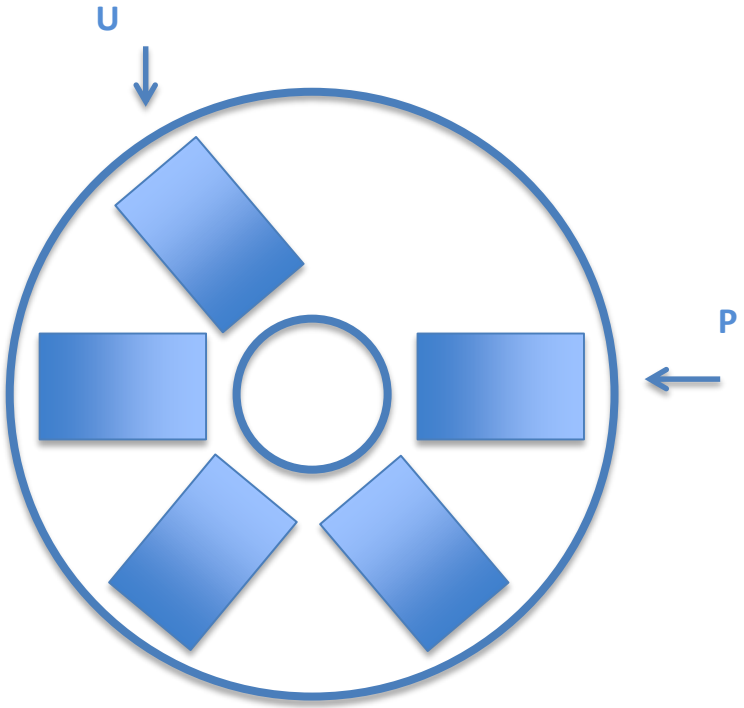
Insertar



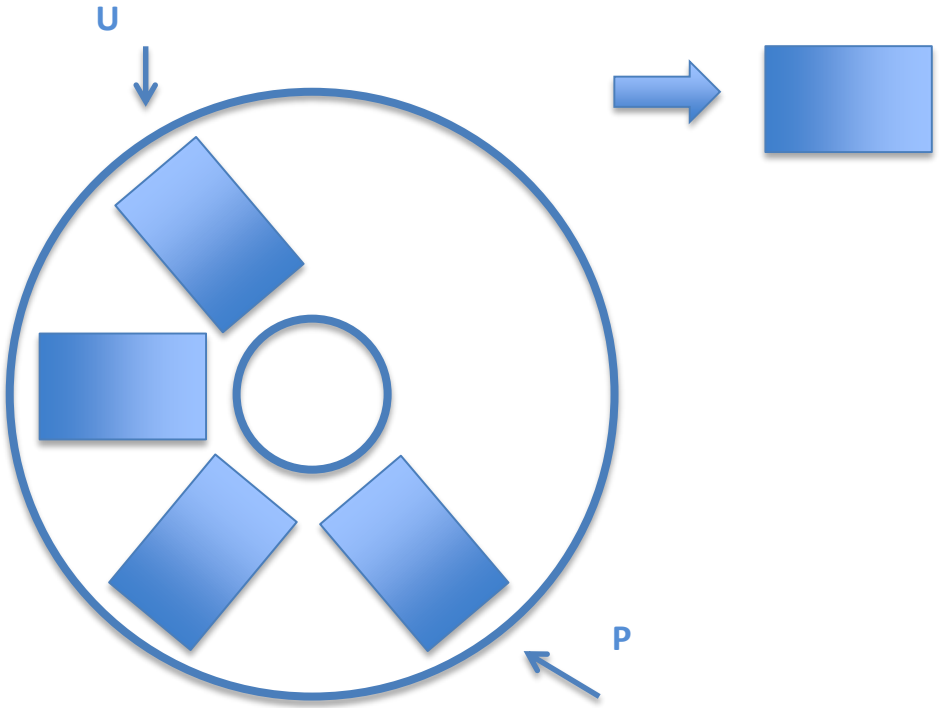
Insertar



Obtener



Obtener



Tarea 3

Programar una estructura de datos tipo cola circular con las funciones INSERTAR, OBTENER y MOSTRAR.

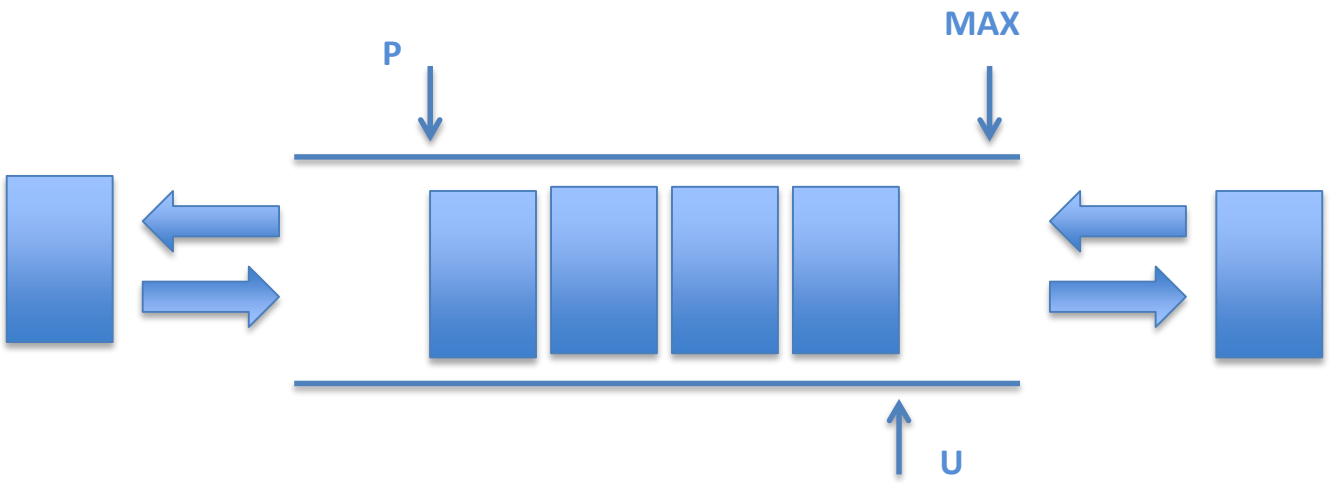
El programa que se debe entregar es una aplicación de la estructura.

3.4 Cola doble.

3.4 Cola doble.

Una cola doble es una estructura de datos tipo cola en la cual las operaciones agregar y eliminar se pueden realizar por ambos extremos.

Dentro de una cola doble es posible insertar y agregar por ambos lados:



Por lo tanto, dentro de una cola doble se deben crear 4 algoritmos diferentes:

- **Agregar por P**
- **Retirar por P**
- **Agregar por U**
- **Retirar por U**

Tarea 4

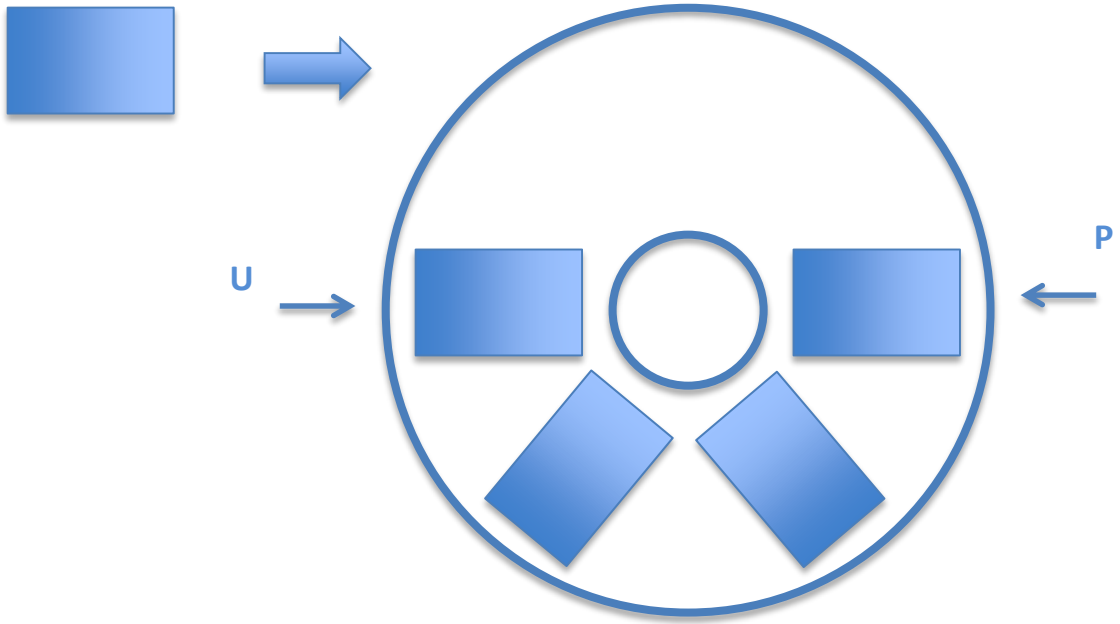
Programar una estructura de datos tipo cola doble con las funciones AGREGAR (por P y por U), RETIRAR (por P y por U) y MOSTRAR.

El programa que se debe entregar es una aplicación de la estructura.

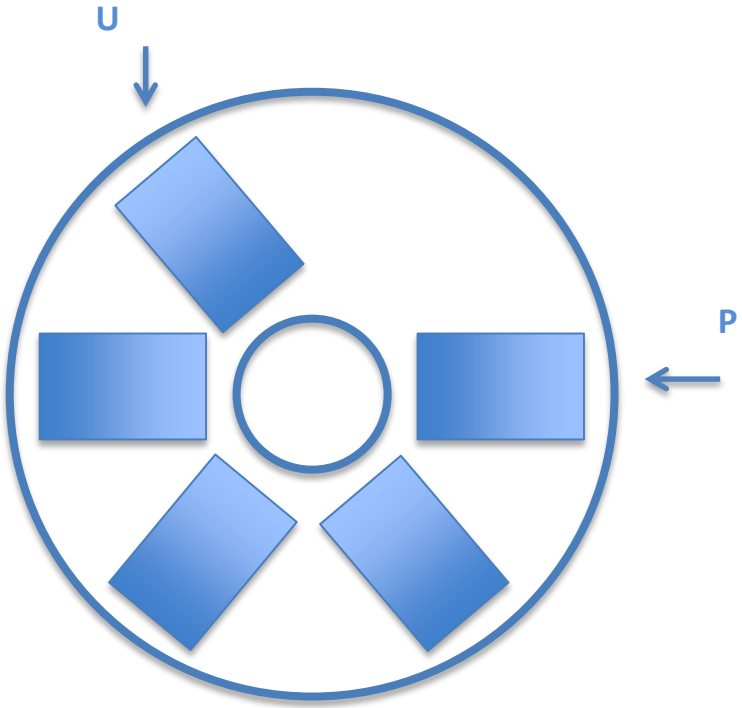
Cola doble circular.

Una cola doble circular constituye una estructura de datos lineal en la cual el siguiente elemento del último es, en realidad, el primero y, además, las operaciones agregar y eliminar se pueden realizar por ambos extremos.

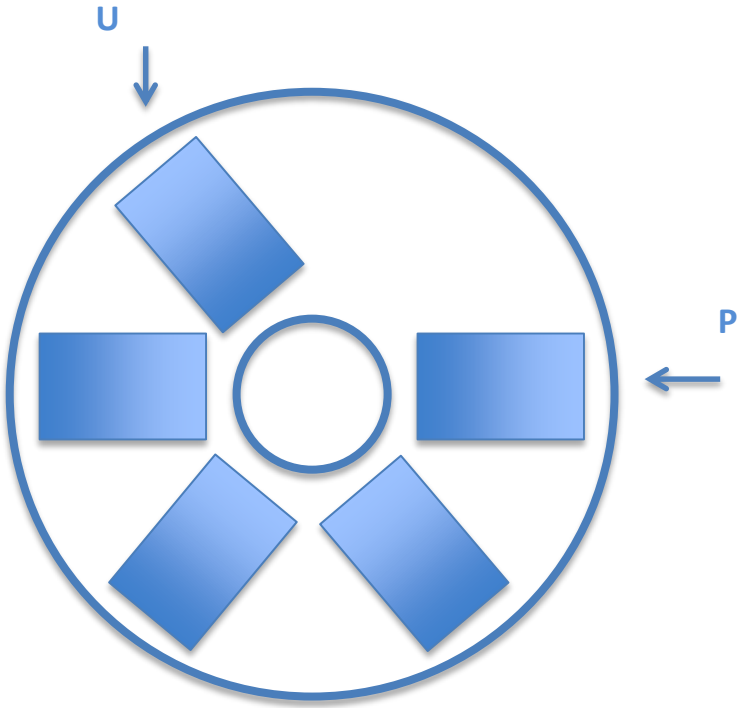
Insertar por U



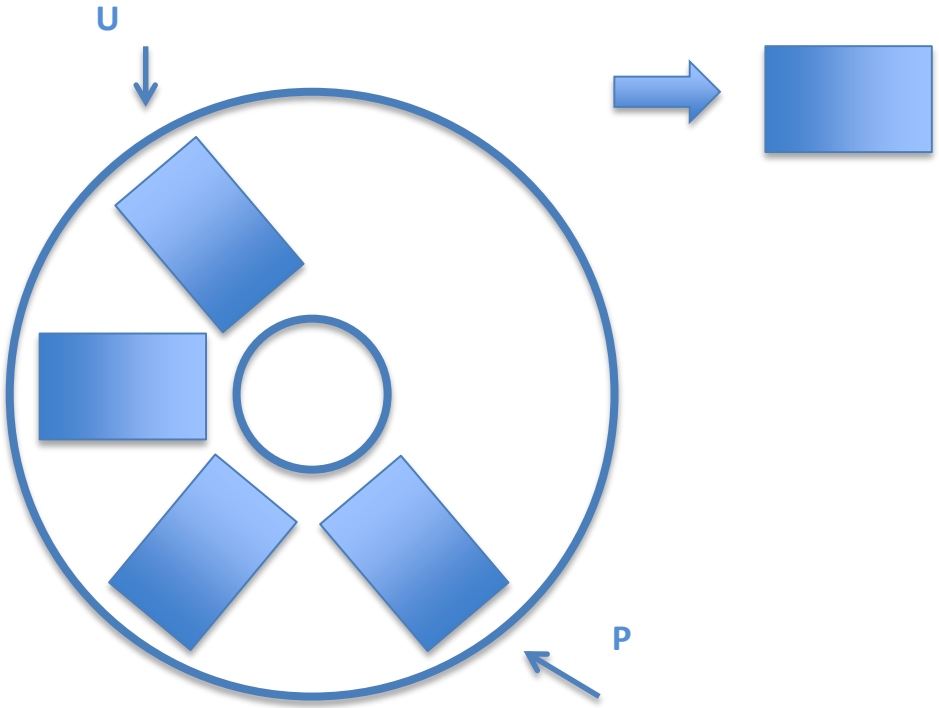
Insertar por U



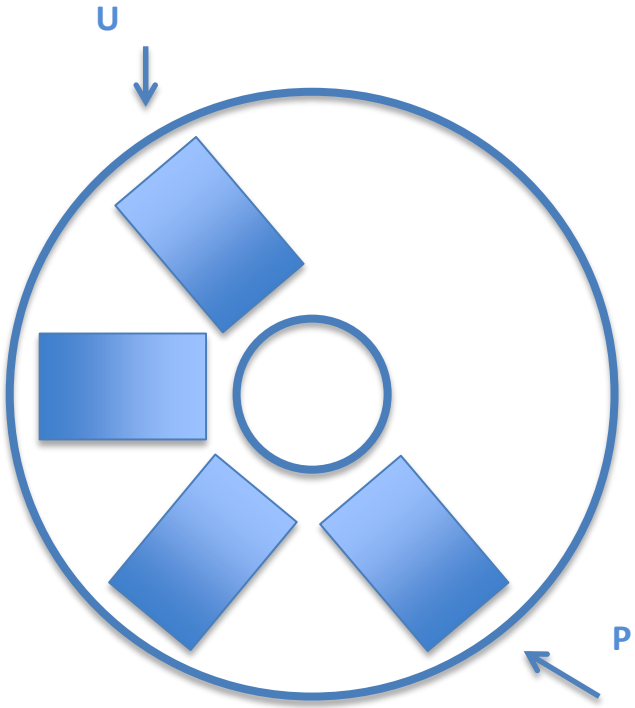
Eliminar por P



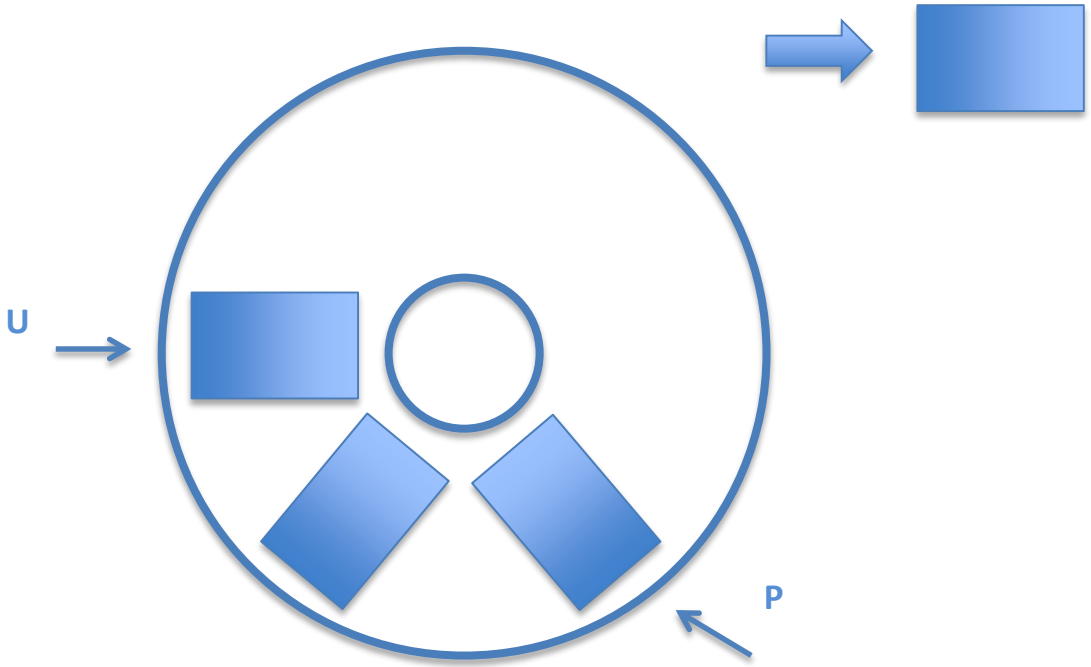
Eliminar por P



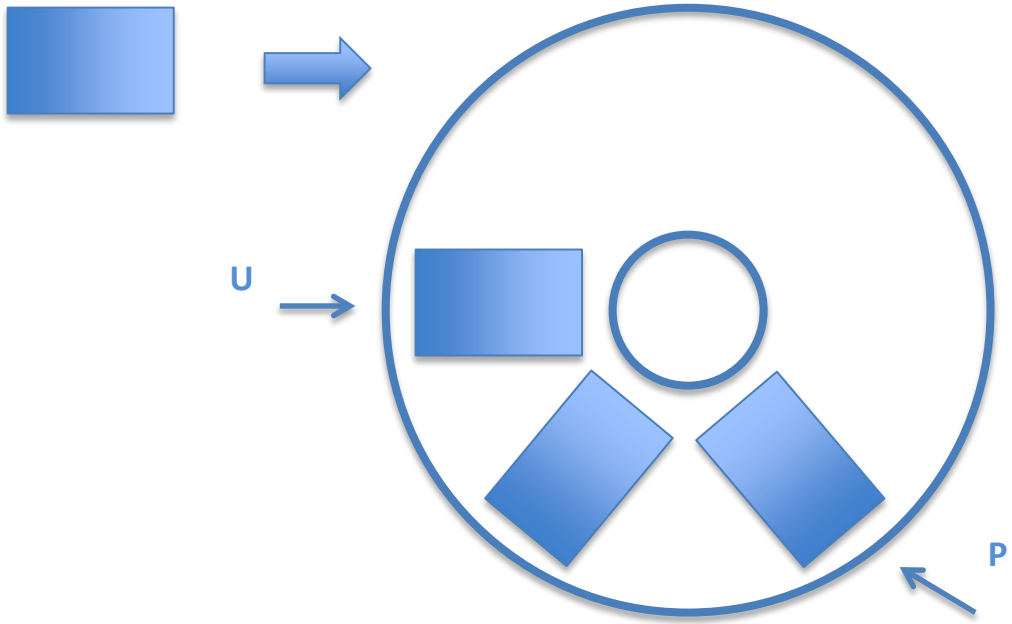
Eliminar por U



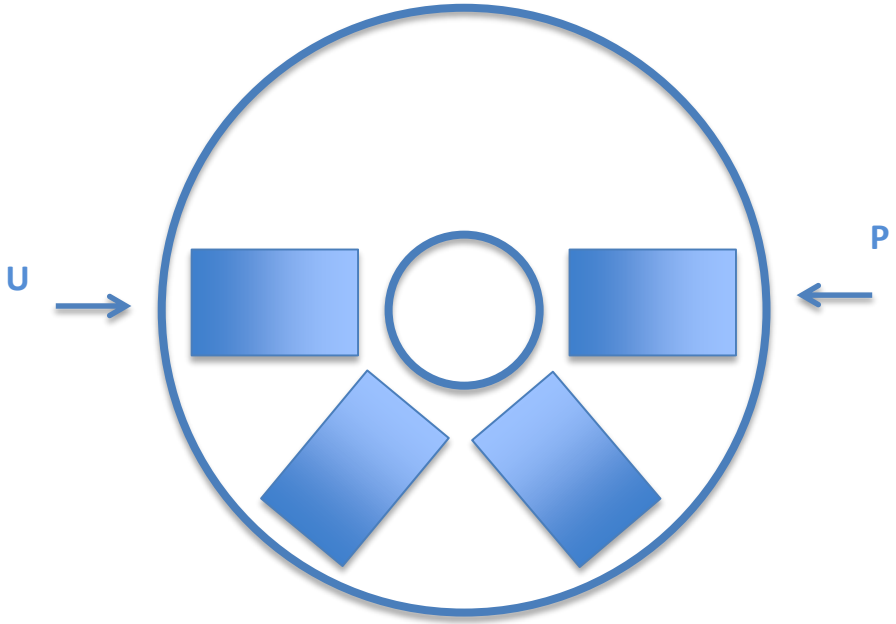
Eliminar por U



Insertar por P



Insertar por P



3.5 Lista circular.

3.5 Lista circular.

Todas las estructuras de datos presentadas hasta el momento son estructuras estáticas, debido a que se les asigna un espacio en tiempo de compilación y éste permanece fijo durante todo el tiempo de ejecución.

Las listas son un tipo de estructura de datos lineal y dinámica. Es lineal porque a cada elemento le sigue solo un elemento. Es dinámica porque su tamaño no es fijo y se puede definir al momento de utilizarse (en tiempo de ejecución).

Lista simplemente ligada

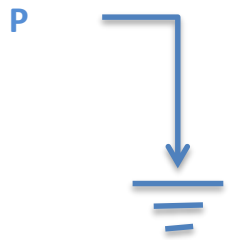
El tipo de lista más sencilla se conoce como lista simplemente ligada y está constituida por un conjunto de nodos ligados entre sí por una referencia.

Las operaciones que se pueden efectuar dentro de una lista son: recorrido, inserción, eliminación y búsqueda.

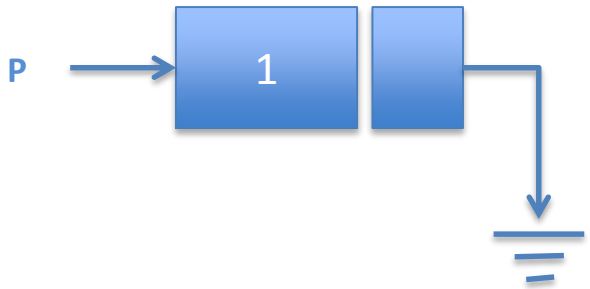
Un elemento o nodo de la lista consta de un espacio reservado para la información y una referencia hacia el siguiente elemento, es decir:

Una lista vacía no ocupa lugar en la memoria porque los nodos se crean en tiempo de ejecución.

Sin embargo, se debe tener un apuntador que se encarga de mantener la referencia al primer nodo, de otro modo sería imposible recuperar la lista. Por tanto, la liga al primer elemento de la lista apunta a nulo cuando ésta está vacía.

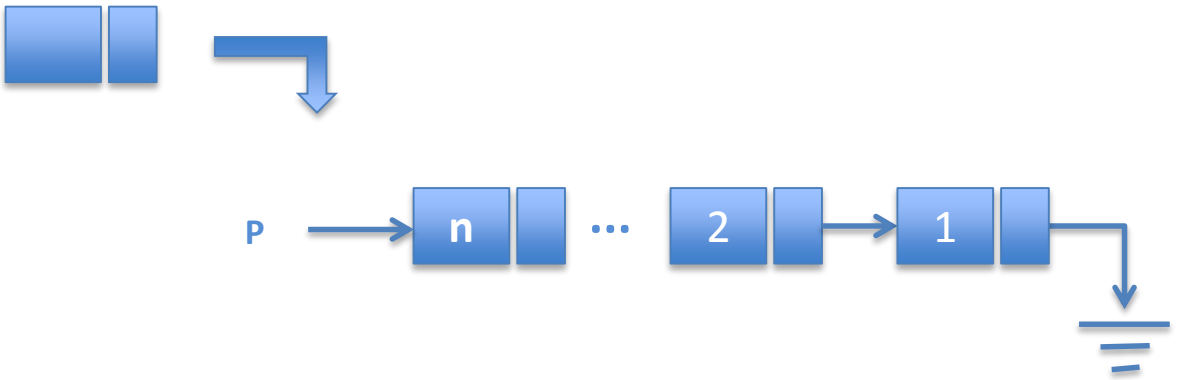


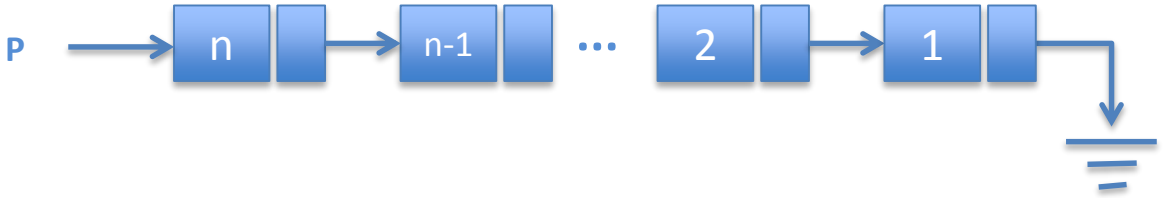
Por otro lado, cuando se crea un nodo en la lista, la liga del elemento no apunta a ningún otro nodo, pero la liga que mantiene la referencia del primer elemento debe apuntar a dicho elemento.



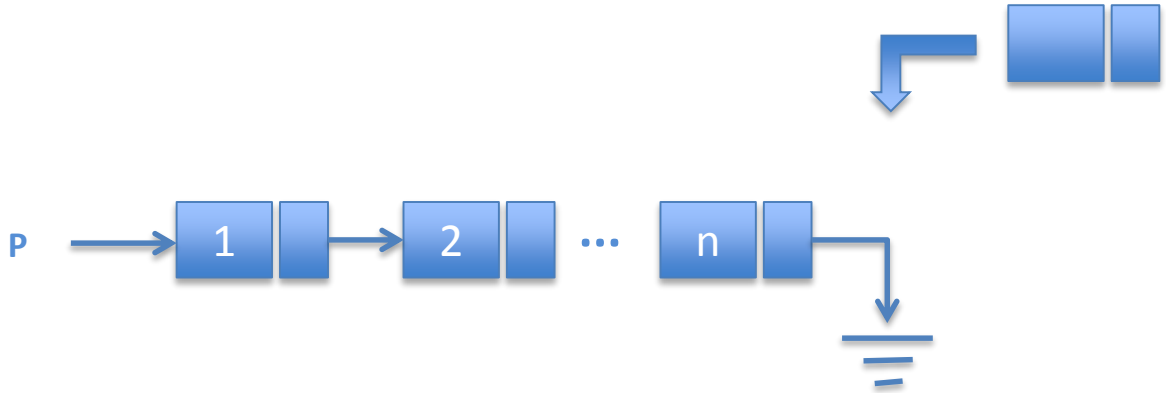
El caso general se presenta cuando la lista posee un conjunto de nodos y se inserta uno nuevo. La inserción en una lista se puede realizar al inicio o al final de la misma.

Insertar al inicio





Insertar al final



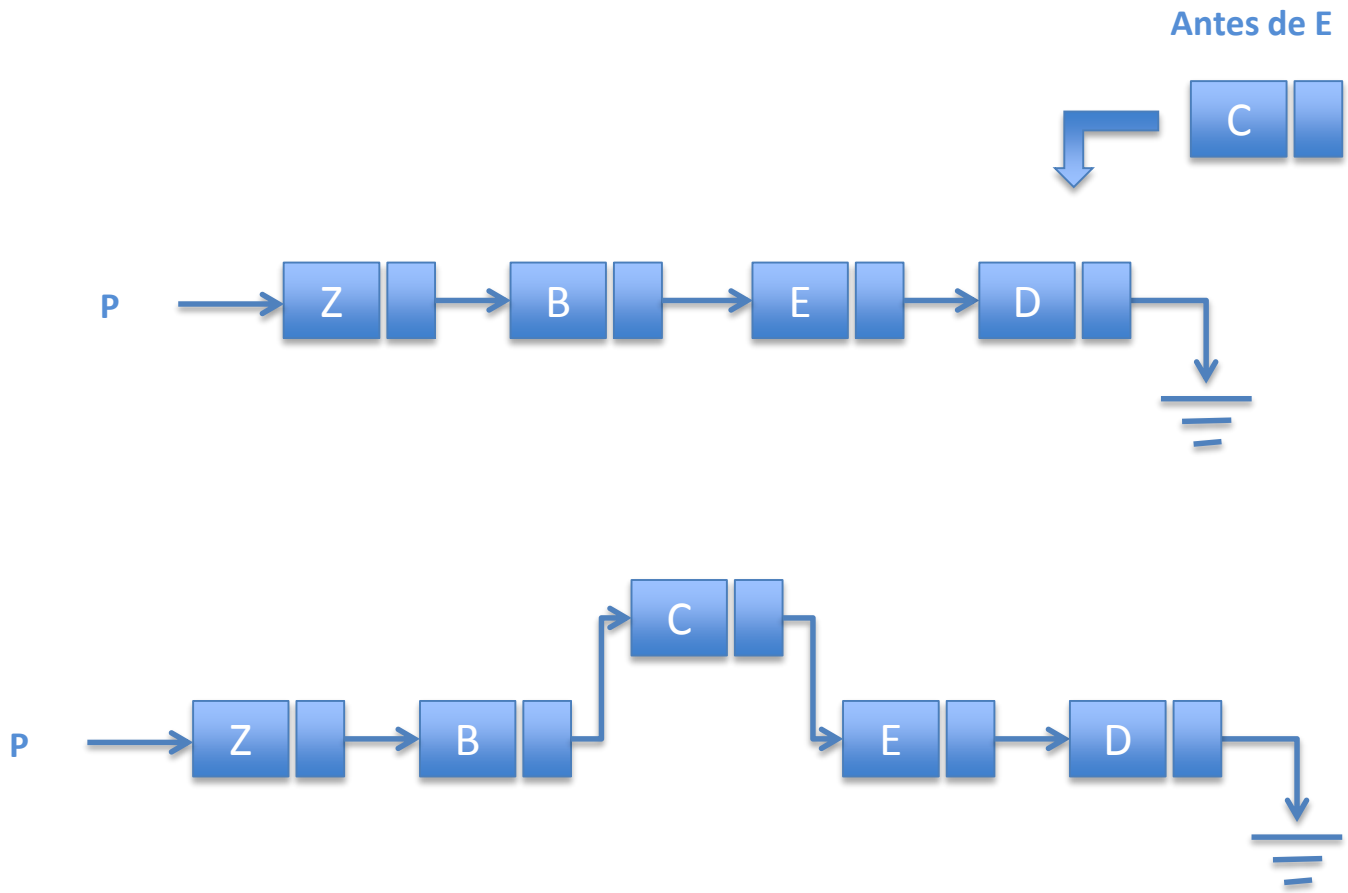
En una lista simplemente ligada también es posible insertar un nodo antes (o después) de otro nodo dado como referencia. En este tipo de inserción se pueden presentar varios casos.

Si la lista está vacía, el nodo se inserta como el único elemento de la lista.

Si el nodo buscado no existe, el elemento se inserta al final de la lista.

Si el nodo buscado sí existe se inserta el nodo antes o después (dependiendo de la condición) de dicho elemento.

Insertar antes de un elemento



También se puede eliminar algún nodo de la lista. Los pasos para borrar elementos son similares a los pasos para insertar y, por ende, hay que tener las mismas consideraciones al eliminar.

Los métodos de búsqueda e inserción ordenada se tocarán en temas posteriores.

Insertar

```
FUNC insertar (Nodo n) DEV booleano
    SI primero = nulo ENTONCES
        lista = agregaElemento()          ** reserva memoria
        SI lista <> nulo ENTONCES
            lista+numElems ← n
            lista->ap ← nulo
            primero ← lista
            numElem ← numElem + 1
            DEV verdadero
        FIN_SI
         EN_CASO_CONTRARIO
            DEV falso
        FIN_ECC
    FIN_SI
```

Insertar

```
    DE_LO_CONTRARIO
        lista = agregaElemento()          ** reserva memoria
        SI lista <> nulo ENTONCES
            lista+numElems ← n
            lista.elemento->ap ← nulo
            lista.elementoAnt->ap ← lista.elemento
            numElem ← numElem + 1
            DEV verdadero
        FIN_SI
         EN_CASO_CONTRARIO
            DEV falso
        FIN_ECC
    FIN_ECC
    DEV falso
FIN_FUNC
```

Eliminar

```
FUNC eliminar () DEV nodo
    SI primero = nulo ENTONCES
        DEV nulo
    FIN_SI
    DE_LO_CONTRARIO
        SI lista[numElem-1]->ap = nulo ENTONCES
            numElem ← numElem – 1
            nodo tmp = lista.elemento
            lista ← nulo
            primero ← nulo
            DEV tmp
        FIN_SI
        DE_LO_CONTRARIO
            numElem ← numElem - 1
            nodo tmp ← lista.elemento
            lista.elementoAnt->ap ← nulo
            DEV tmp
        FIN_DLC
    FIN_DLC
    DEV nulo
FIN_FUNC
```

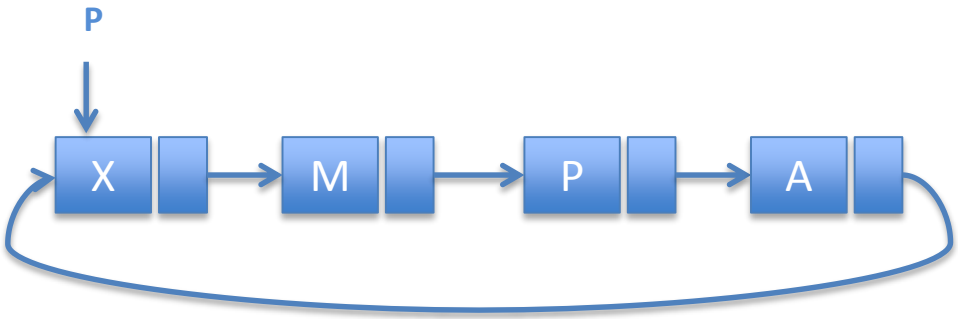
Tarea 5

Programar una estructura de datos tipo lista simplemente ligada con las funciones INSERTAR, ELIMINAR y MOSTRAR, utilizando memoria dinámica.

El programa que se debe entregar es una aplicación de la estructura.

Lista circular

Una lista circular es similar a una lista simplemente ligada (conjunto de nodos) con la diferencia de que el último elemento de la lista en lugar de apuntar a nulo apunta al primer nodo de la lista.



Las operaciones que se pueden realizar en una lista circular son las mismas que en una lista simplemente ligada, con la diferencia de la condición que se debe crear para evitar caer en ciclos infinitos al recorrer la lista.

Tarea 6

Programar una estructura de datos tipo lista circular con las funciones INSERTAR, ELIMINAR y MOSTRAR, utilizando memoria dinámica.

El programa que se debe entregar es una aplicación de la estructura.