Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

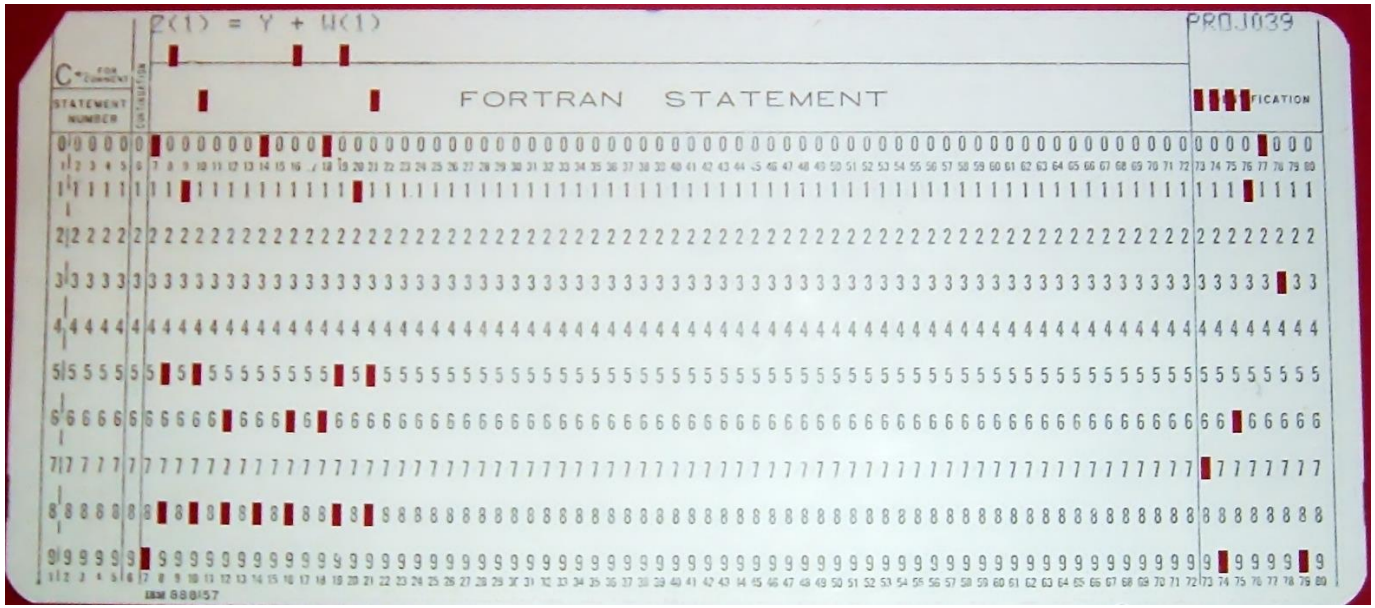
MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

Institución Universitaria de Envigado

Arrays: Eficiencia en operaciones matemáticas

Numpy utiliza implementaciones eficientes de operaciones matemáticas y algebraicas, escritas en **C** y **Fortran**, lo que permite realizar cálculos mucho más rápidos que usando bucles en Python puro.



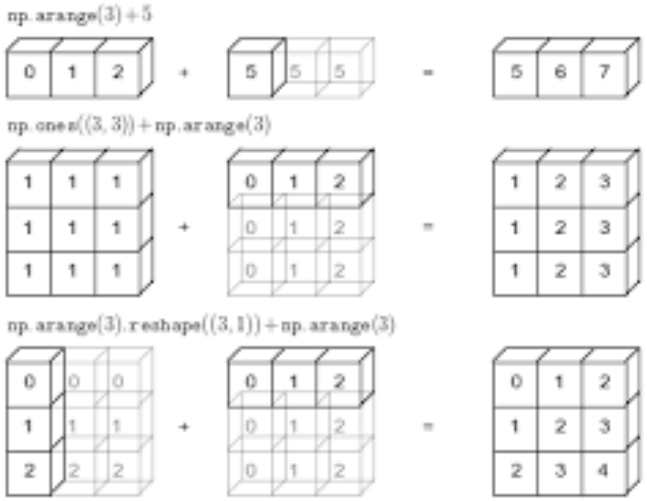
Vectorización de operaciones

Numpy permite realizar operaciones en arrays completos de una manera vectorizada, lo que significa que puedes aplicar una operación a todo el array de una vez, en lugar de iterar sobre cada elemento individualmente.

```
Console C:/Users/JTR/Desktop/R/ProyectosR/Vectores/
> #Creamos un vector numérico
> VectorNumerico <- c(1, 3, 5, 7, 9)
> #Las operaciones aritméticas, aplican a cada dato del vector
> VectorNumerico + 1
[1]  2  4  6  8 10
> VectorNumerico * 2
[1]  2  6 10 14 18
>
```

Broadcasting

Numpy realiza operaciones de broadcasting automáticamente, lo que significa que puede realizar operaciones entre arrays de diferentes formas y tamaños de manera eficiente, adaptando los valores automáticamente para que las operaciones sean válidas.



- **Dimensiones incompatibles:** Cuando las dimensiones de los arrays no se pueden alinear para realizar operaciones
- **Ambigüedad en las operaciones:** El broadcasting puede causar ambigüedad si las formas de los arrays no son claras y no se puede determinar cómo deben ser expandidos.
- **Resultados inesperados:** En algunos casos, el broadcasting puede dar como resultado resultados inesperados o incorrectos si no se tiene en cuenta cómo se expanden los arrays durante la operación.

Indexación y segmentación avanzadas

Numpy ofrece una amplia gama de técnicas de indexación y segmentación que permiten acceder y manipular partes específicas de los arrays de manera eficiente.

Herramientas para álgebra lineal y estadísticas

Numpy proporciona una amplia gama de funciones y herramientas para realizar operaciones de álgebra lineal y estadísticas de manera eficiente, como la descomposición de matrices, cálculos de promedios, desviaciones estándar, entre otros.

Ejemplos Listas vs Array

Listas	Arrays
Son ideales cuando necesitas una estructura de datos flexible y dinámica, donde los elementos pueden ser insertados y eliminados fácilmente en cualquier posición (O(1)).	Son más eficientes para acceder a elementos aleatorios debido a que proporcionan acceso indexado directo a sus elementos (O(1)).
Pueden implementarse fácilmente como pilas (LIFO) o colas (FIFO).	Son ideales cuando necesitas almacenar una gran cantidad de datos homogéneos.
Proporcionan una capacidad dinámica, lo que significa que pueden cambiar de tamaño automáticamente a medida que se agregan o eliminan elementos.	Son más eficientes en términos de tiempo y espacio para realizar operaciones en bloques de datos.

Ejemplos Listas vs Array

Flexibilidad vs. Eficiencia:

- Si la flexibilidad y la capacidad de cambiar el tamaño dinámicamente son prioritarias, las listas son preferibles.
- Si la eficiencia en el acceso y la manipulación de datos es crucial, **especialmente en aplicaciones que implican grandes conjuntos de datos homogéneos**, los arrays son más adecuados.

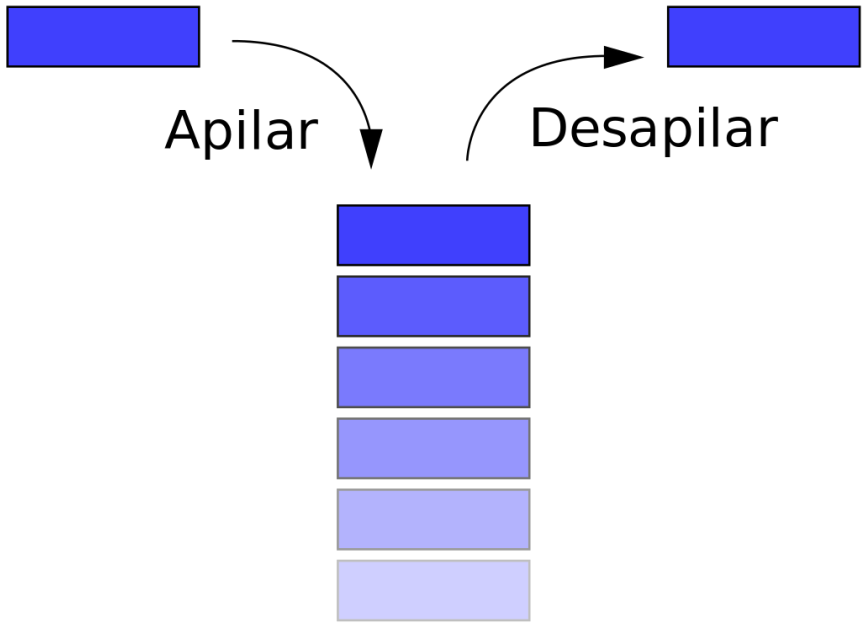
Uso de bibliotecas especializadas:

Existen bibliotecas especializadas como numpy que ofrecen arrays multidimensionales optimizados para operaciones matemáticas, lo que puede ser la mejor opción para tareas específicas como el procesamiento de datos científicos y numéricos.

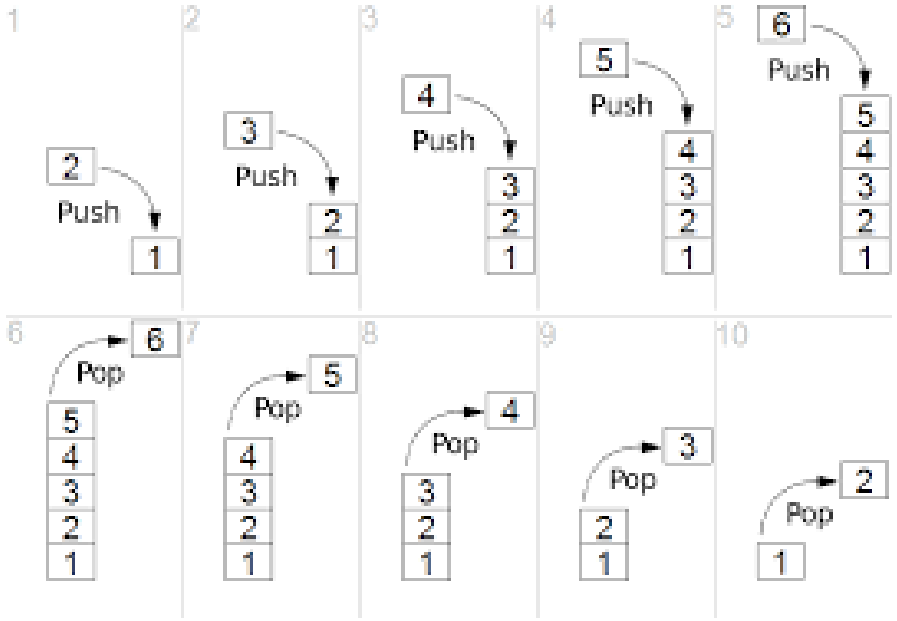
Pilas

Una pila es una lista ordenada o estructura de datos que permite almacenar y recuperar datos, siendo el modo de acceso a sus elementos de tipo **LIFO** (Last In, First Out).

Esta estructura se aplica en multitud de supuestos en el área de la informática debido a su simplicidad y capacidad de dar respuesta a numerosos procesos.

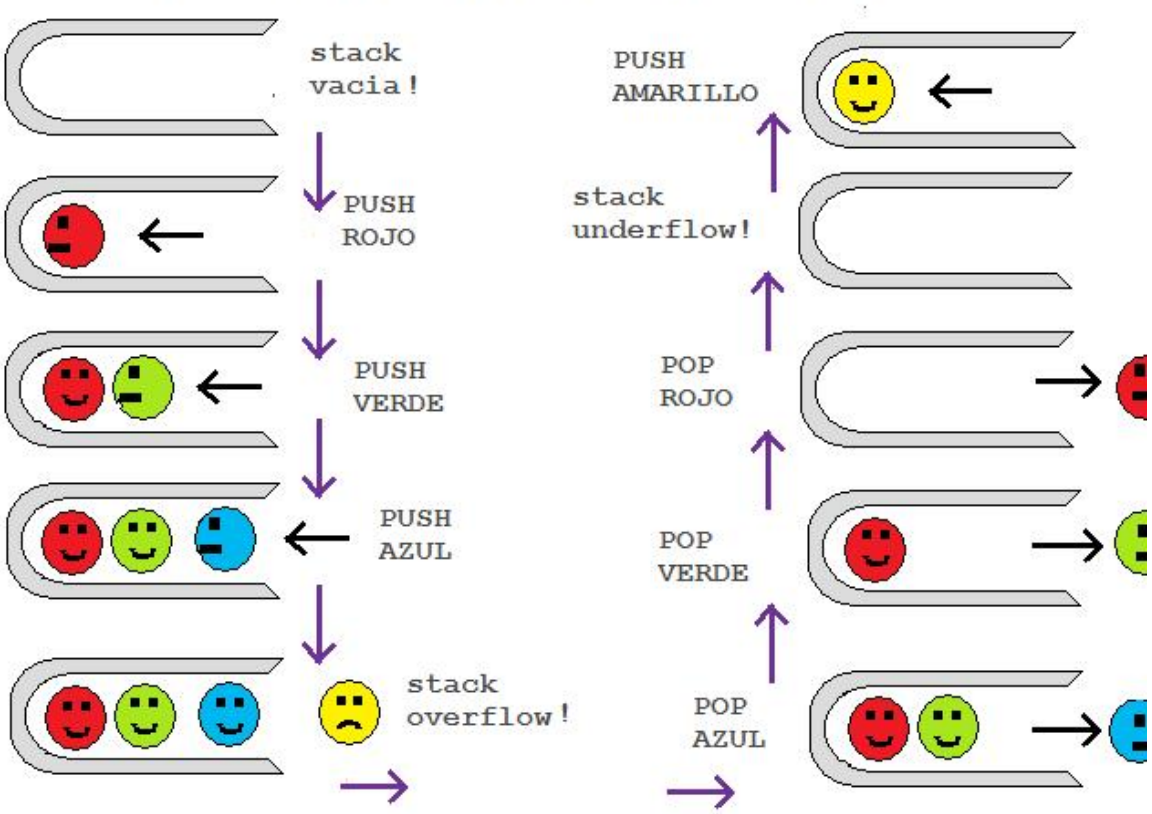


Pilas



Para el manejo de los datos cuenta con dos operaciones básicas: apilar (**push**), que coloca un objeto en la pila, y su operación inversa, retirar (**pop**), que retira el último elemento apilado.

Pilas


stack vacia!
PUSH ROJO
PUSH VERDE
PUSH AZUL
stack overflow!
PUSH AMARILLO
stack underflow!
POP ROJO
POP VERDE
POP AZUL

Pilas

Una pila de libros que se exhiben en una librería o una pila de platos. Es de suponer que, si el cocinero necesita un plato limpio, tomará el que está encima de todos, que es el último que se colocó en la pila.

Pilas

Las pilas son estructuras de datos lineales, como los arreglos, ya que los componentes ocupan lugares sucesivos en la estructura y cada uno de ellos tiene un único sucesor y un único predecesor, con excepción del último y del primero, respectivamente

Una pila se define formalmente como una colección de datos a los cuales se puede acceder mediante un extremo, que se conoce generalmente como tope.

Las pilas no son estructuras fundamentales de datos Para su representación requieren el uso de otras estructuras de datos, como arreglos o listas.

Pilas: Operaciones

- **Crear (constructor):** crea la pila vacía.
- **Tamaño (size):** regresa el número de elementos de la pila.
- **Apilar (push):** añade un elemento a la pila.
- **Desapilar (pop):** lee y retira el elemento superior de la pila.
- **Leer último (top o peek):** lee el elemento superior de la pila sin retirarlo.
- **Vacía (empty):** devuelve cierto si la pila está sin elementos o falso en caso de que contenga alguno.

Pilas (Ejemplos)

Navegador Web

- Se almacenan los sitios previamente visitados
- Cuando el usuario quiere regresar (presiona el botón de retroceso), simplemente se extrae la última dirección (pop) de la pila de sitios visitados.

Editores de texto

- Los cambios efectuados se almacenan en una pila
- Usualmente implementada como arreglo
- Usuario puede deshacer los cambios mediante la operación “undo”, la cual extraer el estado del texto antes del último cambio realizado.

Pilas estáticas

Una pila estática se implementa utilizando un arreglo de tamaño fijo. Las operaciones de inserción (push) y eliminación (pop) se realizan en el tope de la pila, que es el extremo del arreglo. La ventaja principal de una pila estática es que es simple y eficiente en términos de acceso y uso de memoria. Sin embargo, la desventaja es que su tamaño está limitado por el tamaño del arreglo, lo que significa que no puede crecer dinámicamente más allá de su tamaño inicial.

Pilas dinámicas

Una pila dinámica se implementa utilizando estructuras de nodos enlazados. Cada nodo contiene un elemento y un enlace al siguiente nodo en la pila. A diferencia de una pila estática, una pila dinámica no tiene un límite fijo en su tamaño, ya que puede crecer o reducirse dinámicamente según sea necesario. La principal ventaja de una pila dinámica es su flexibilidad para manejar una cantidad variable de elementos. Sin embargo, puede tener una sobrecarga adicional debido a la gestión de memoria dinámica.

Pilas

Pilas Estáticas	Pilas Dinámicas
Eficiencia de memoria	Flexibilidad en el tamaño
Simplicidad de implementación	No hay límite en el tamaño
Rendimiento predecible	Manejo eficiente de la memoria

Pilas

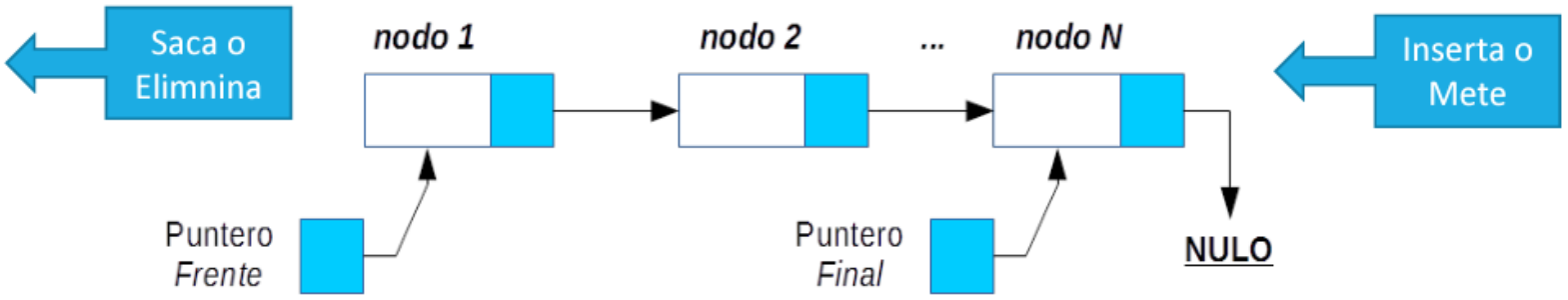
Estás desarrollando un editor de texto simple. Una de las funcionalidades que quieres implementar es la capacidad de "Deshacer" y "Rehacer" las acciones del usuario. Para esto, necesitarás utilizar dos pilas: una para las operaciones de deshacer y otra para las operaciones de rehacer. Cada vez que el usuario realiza una acción, esa acción se almacena en una de las pilas. Cuando el usuario decide deshacer una acción, la acción más reciente se mueve a la segunda pila y se revierte en el editor.

Pilas

- Implementar dos pilas (undo_stack y redo_stack) usando listas en Python.
- Crear funciones para realizar las siguientes operaciones:
  - hacer_acción(accion): Realiza una acción y la guarda en la pila de undo_stack
  - deshacer(): Deshace la última acción (mueve la acción de undo_stack a redo_stack)
  - rehacer(): Rehace la última acción deshecha (mueve la acción de redo_stack a undo_stack).

Colas

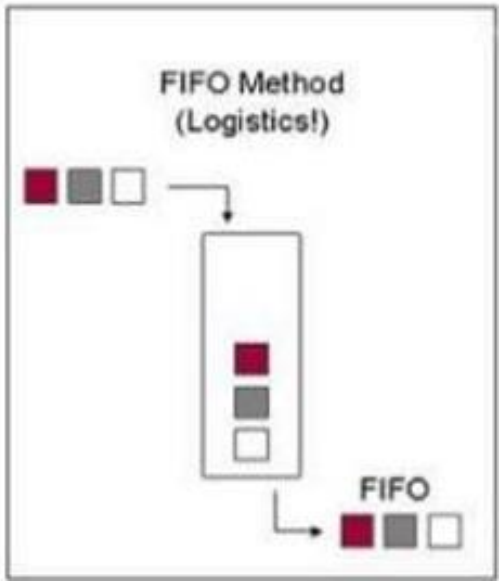
Es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los dos extremos de la lista. Un elemento se inserta en la cola (parte final) de la lista y se suprime o elimina por la frente (parte inicial, cabeza) de la lista.



Colas

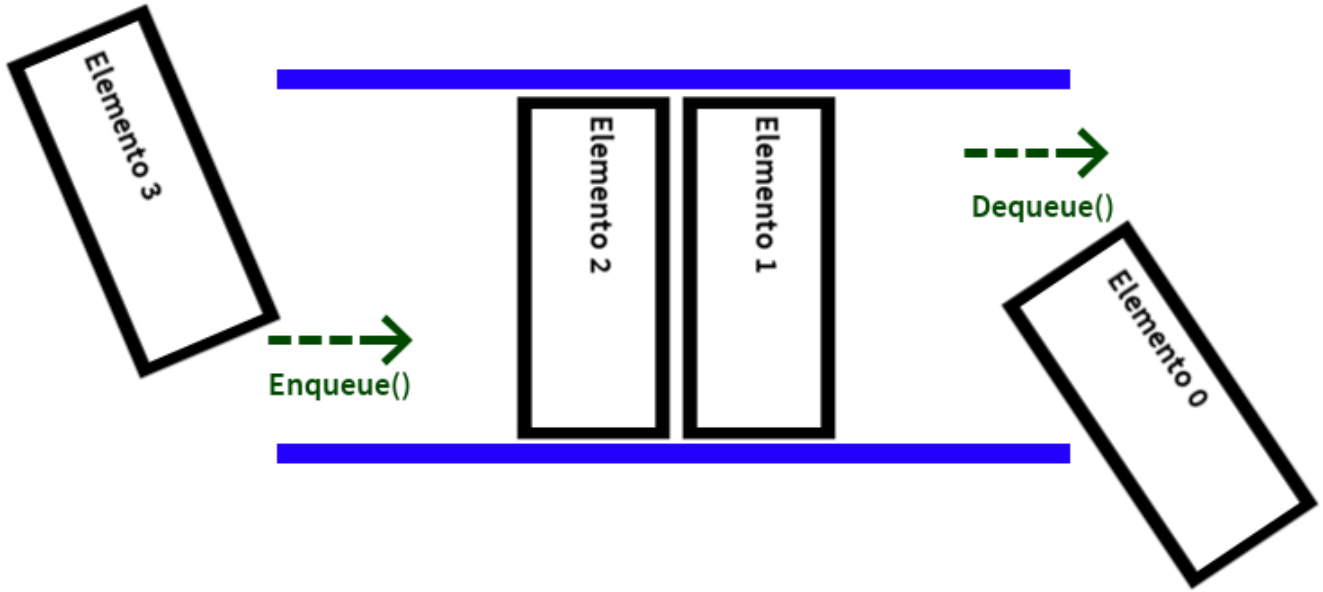
Los elementos se eliminan (se quitan) de la cola en el mismo orden en que se almacenan y, por consiguiente, una cola es una estructura de tipo FIFO (first-in-first-out).

El servicio de atención a clientes es un ejemplo típico de cola o el cajero de un banco.



Colas

Las colas no existen como estructuras de datos estándar en los lenguajes de programación. Este tipo de estructura de datos se puede representar mediante el uso de arreglos o listas.



Colas

Los datos se almacenan de un modo lineal y el acceso a los datos solo está permitido en los extremos de la cola. Los nodos de una cola se eliminan solo desde el principio (cabeza) de la misma y se insertan sólo al final (cola) de ésta.

15	20	9	.....	18	19

1.Ejemplo de Cola

15	20	9	.....	18	19

2.Vamos a Insertar el 13 en la Cola.

15	20	9	.....	18	19	13

3.Sacamos el frente de la Cola (15)

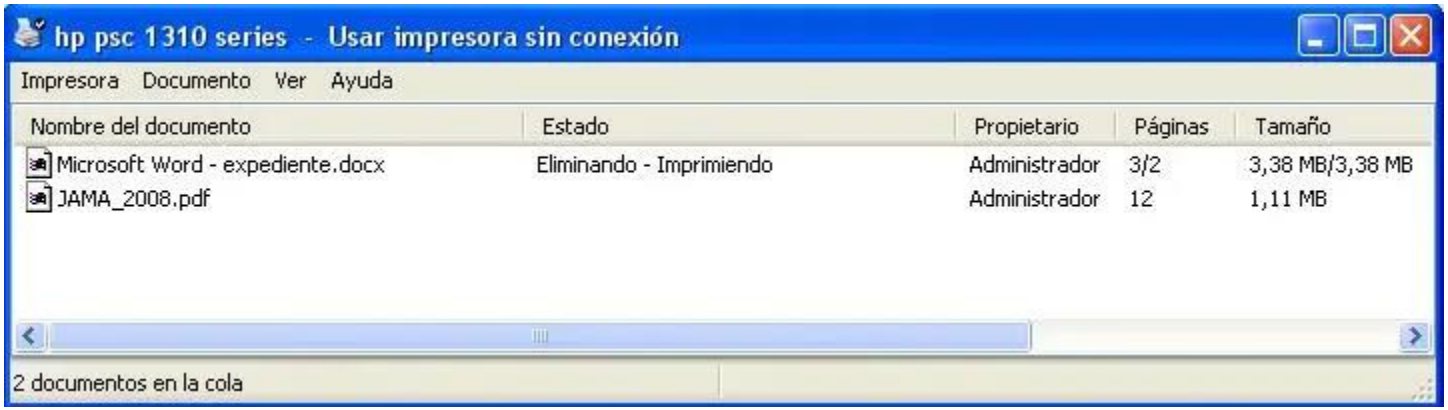
20	9	.....	18	19	13

Colas

Constructor
Está vacía
Agregar elemento
Eliminar elemento
Mostrar el primer elemento (Frente)
Mostrar el último elemento (Final)
Mostrar los datos de la cola
Destructor

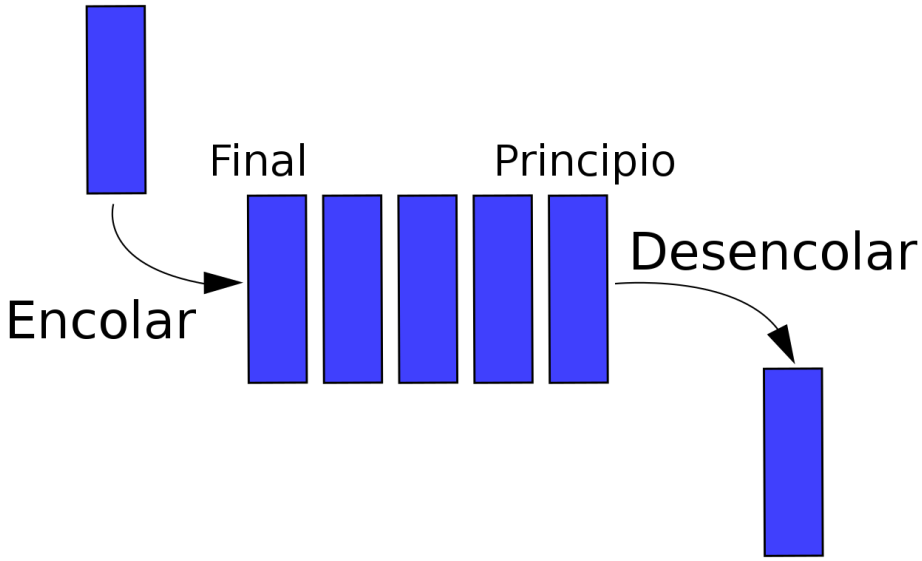
Colas

Una aplicación común es el envío para imprimir un documento en las colas de impresión. Cuando hay una sola impresora para atender a varios usuarios; suele suceder que algunos de ellos solicitan el servicio de impresión al mismo tiempo o mientras está ocupado el dispositivo. Se forma una cola con los trabajos que esperan para ser impresos y se procesarán en el orden en el que fueron introducidos a la cola.



Colas

Otra aplicación es la presentada en los sistemas de tiempo compartido. Varios usuarios comparten ciertos recursos como CPU y memoria de la computadora. Los recursos se asignan a los procesos que están en la cola de espera, en el orden en el cual fueron introducidos a la cola.



Cola Estática

Una cola estática se implementa utilizando un arreglo de tamaño fijo. Los elementos se insertan al final de la cola y se eliminan del principio. Una vez que se llena la cola, no se pueden agregar más elementos hasta que se eliminen algunos. La implementación estática tiene la ventaja de ser simple y eficiente en términos de acceso y uso de memoria.

Cola Dinámica

Una cola dinámica se implementa utilizando una estructura de nodos enlazados. Cada nodo contiene un elemento y un enlace al siguiente nodo en la cola. Al igual que las pilas dinámicas, las colas dinámicas pueden crecer o reducirse dinámicamente según sea necesario, ya que no tienen un tamaño fijo predefinido. La principal ventaja de una cola dinámica es su flexibilidad para manejar una cantidad variable de elementos.

Colas

Colas Estáticas	Colas Dinámicas
Eficiencia de memoria	Flexibilidad en el tamaño
Simplicidad de implementación	No hay límite en el tamaño
Rendimiento predecible	Manejo eficiente de la memoria

Ejercicios

1. En un banco, los clientes llegan y se forman en una fila para ser atendidos por los cajeros. Cada cliente tiene un tiempo estimado de atención. Simula el proceso de atención de clientes en un banco donde hay 3 cajeros. Implementa una cola para los clientes y, a medida que llegan, agrégalos a la cola. Cuando un cajero está disponible, atiende al siguiente cliente en la fila.
   - Crear una cola que represente la fila de clientes.
   - Simular la llegada de clientes y el tiempo de atención.
   - Mostrar el estado de la fila en todo momento.
   - Mostrar cuál cliente está siendo atendido por qué cajero y cuánto tiempo falta para que termine.

Ejercicios Listas – Pilas

- Implementar una función que recibe una lista de enteros L y un número entero n de forma que modifique la lista mediante el borrado de todos los elementos de la lista que tengan este valor.
- Escribir una función Reemplazar que tenga como argumentos una pila con tipo de elemento int y dos valores int: nuevo y viejo de forma que, si el segundo valor aparece en algún lugar de la pila,sea reemplazado por el segundo.

Ejercicios Listas – Pilas -

- Implementar una función Mezcla2 que tenga como parámetros dos listas de enteros ordenados de menor a mayor y que devuelva una nueva lista como unión de ambas con sus elementos ordenados de la misma forma.
- Construir una función que sume los elementos de una lista de enteros recursivamente.
- Construir una función imprimeInverso que imprima los elementos de una lista enlazada de enteros en orden inverso a partir de una posición p