

2022

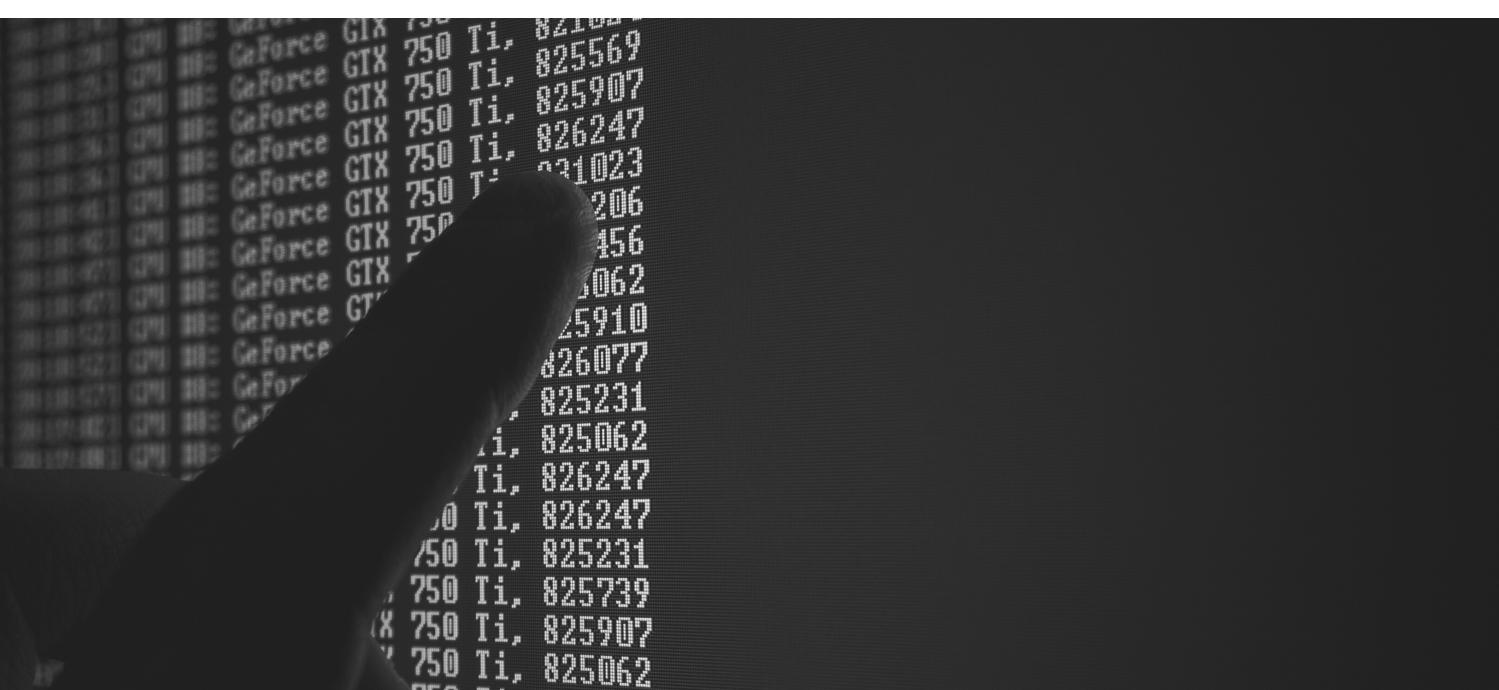
DRAWING PAPER



The image features a faint, diagonal watermark of a computer terminal window. The window displays C++ code related to matrix multiplication. The text includes prompts like "Enter rows and columns for first matrix:", "Enter elements of first matrix.", "Enter elements of matrix 1:", and "Enter elements of second matrix:". It also shows code snippets such as "cout << i + 1 << j + 1 << ":";" and "cin >> a[i][j];". The overall theme of the page is technical documentation, specifically a manual for drawing paper, with a subtle reference to programming and data structures.

INTRODUCCIÓN

Día con día se presentan diferentes tipos de problemas en el campo de la programación para los cuales se puede buscar la optimización de las soluciones planteando soluciones integrales que implementen tipos de datos abstractos para una mejor administración de memoria en el sistema y poder optimizar el desarrollo de la solución aplicando los conceptos de programación orientada a objetos y así poder llevar a cabo una mejor visualización de datos



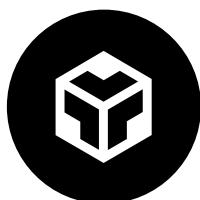
REQUERIMIENTOS

Estos son los requerimientos mínimos para correr el código desarrollado para la aplicación Drawing Paper.



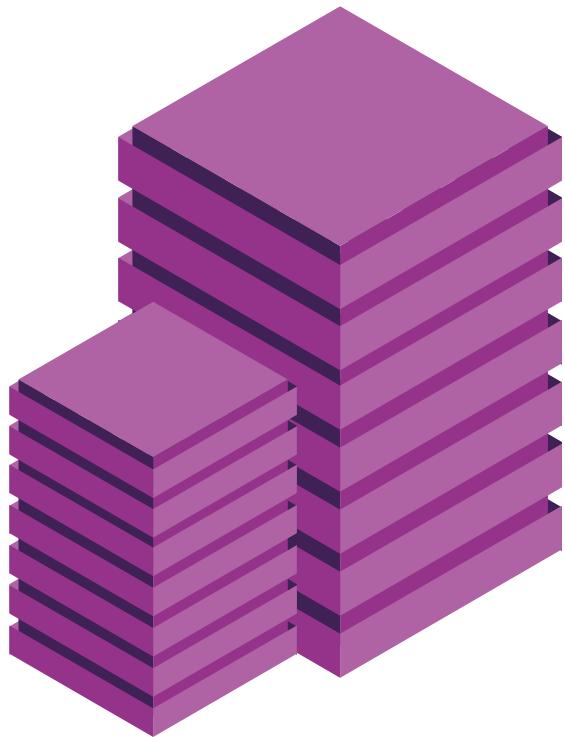
JAVA

- RAM: 128 MB
- Espacio en disco: 124 MB para JRE; 2 MB para Java Update
- Procesador: Mínimo Pentium 2 a 266 MHz



NETBEANS

- Intel Pentium III 800 MHz (800MHz Intel Pentium III or equivalent)
- RAM :512 MB
- Disco duro :750 MB



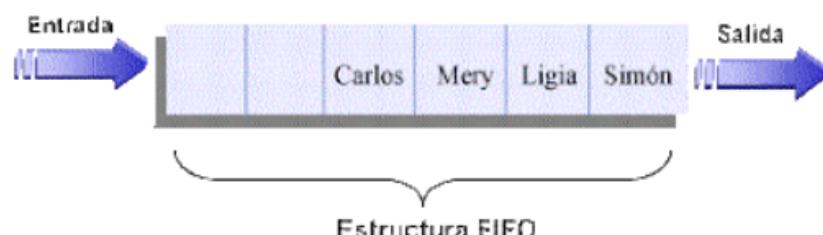
ESTRUCTURAS

COLA

Una cola 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 el frente (parte inicial, frente) de la lista. Las aplicaciones utilizan una cola para almacenar elementos en su orden de aparición o concurrencia.

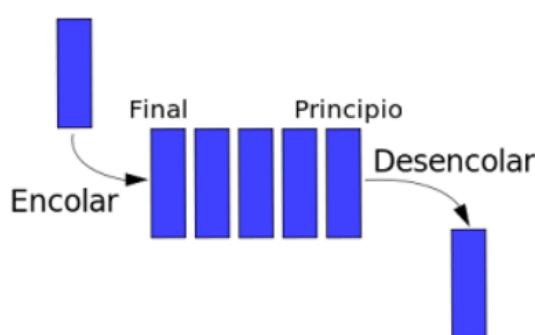


Los elementos se eliminan de la cola en el mismo orden en que se almacena, por ello una cola es una estructura de tipo FIFO (first-in/first-out) es decir que el dato primero en entrar será el primero en salir.



En resumen...

Una cola es una estructura de datos del tipo FIFO, con dos operaciones imprescindibles: encolar y desencolar, al primer elemento se lo conoce como "frente o principio" y al último como "final".



PILA

Una pila cuenta con 2 operaciones imprescindibles:

Apilar (push): Consiste en añadir un elemento a la pila. En la siguiente imagen se explica de una forma más abstracta en la que se puede ver que se está añadiendo el dato "7" sobre el dato "8", convirtiéndose éste en la "cima" de la pila.

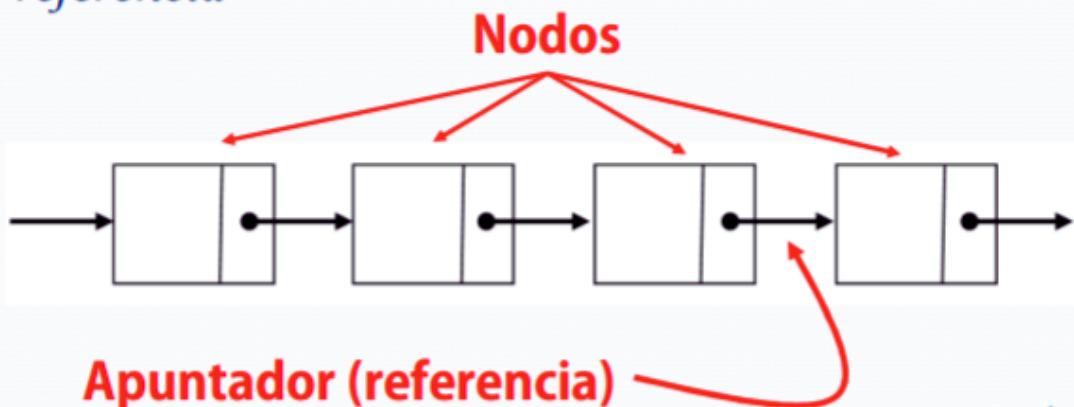


Desapilar (pop): Es la operación contraria de "apilar", es decir, en lugar de añadir elementos a la pila, se los va a quitar de la misma. En la siguiente imagen se explica de una forma más abstracta en la que se puede ver que se está quitando de la pila el dato "7" establecido anteriormente como la "cima".



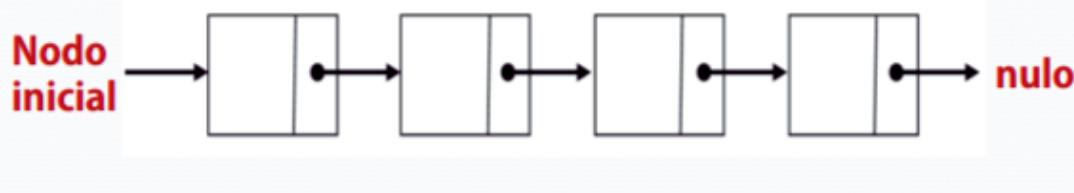
LISTA SIMPLE

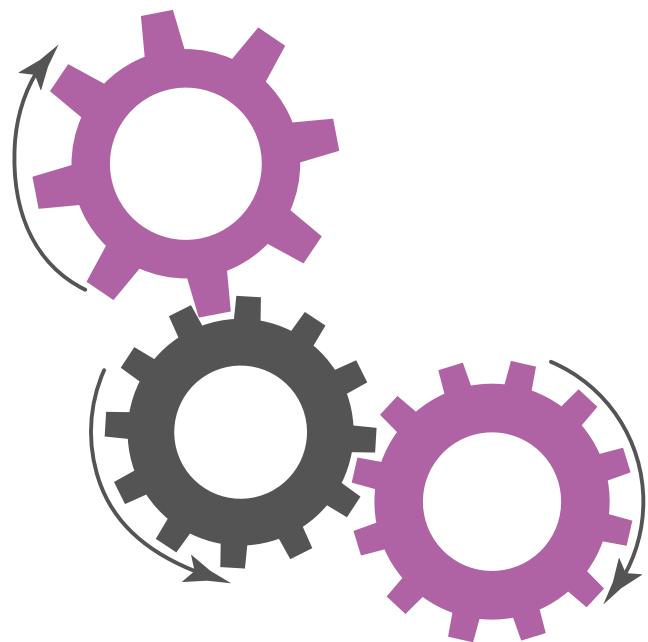
Es una estructura de datos lineal y dinámica que se compone de un conjunto de nodos en secuencia enlazados mediante un apuntador o referencia



4

- La lista simple tiene un apuntador inicial
- El último nodo de la lista apunta a nulo





ALGORITMOS

ORDENAMIENTO

```
public void ORDENAR() {
    //APLICANDO BUBBLE SORT
    NodoC aux_i;
    NodoC aux_j;
    NodoC auxValor;
    NodoC auxAnterior_i = null;
    NodoC auxAnterior_j = null;

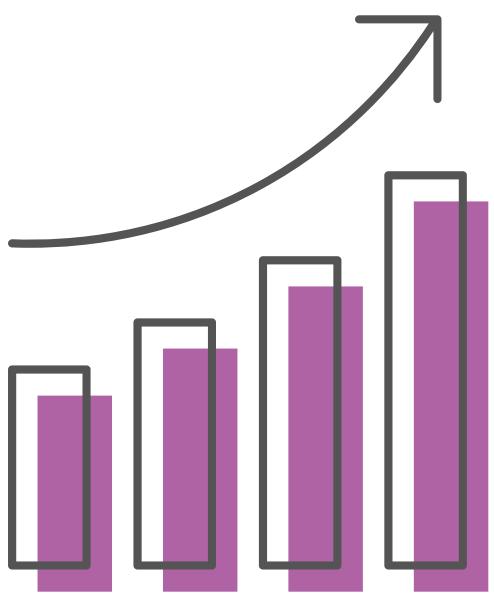
    aux_i = this.inicio;

    while (aux_i != null) {
        aux_j = aux_i.next;
        while (aux_j != null) {
            if (aux_i.getId_cliente() > aux_j.getId_cliente()) {
                //System.out.println("\tcambiar: " + aux_i.getId_cliente() + " " + aux_j.getId_cliente());
                auxValor = aux_i.next;
                if (auxValor.hashCode() != aux_j.hashCode()) {

                    aux_i.next = aux_j.next;
                    aux_j.next = auxValor;
                    auxAnterior_j.next = aux_i;
                } else {
                    //intercambiar las posiciones de los nodos [i],[j]
                    auxValor = aux_i;
                    aux_i = aux_j;
                    aux_j = auxValor;

                    //Al cambiar la posicion de un nodo el nodo anterior de [i] debe apuntar a [j]
                    if (auxAnterior_i != null) {
                        auxAnterior_i.next = aux_i;
                    }

                    //si cambiamos la posicion del primer nodo de la lista, debemos cambiar el inicio
                    if (aux_j.hashCode() == inicio.hashCode()) {
                        inicio = aux_i;
                    }
                }
            }
            auxAnterior_j = aux_j;
            aux_j = aux_j.next;
        }
        auxAnterior_i = aux_i;
        aux_i = aux_i.next;
    }
}
```



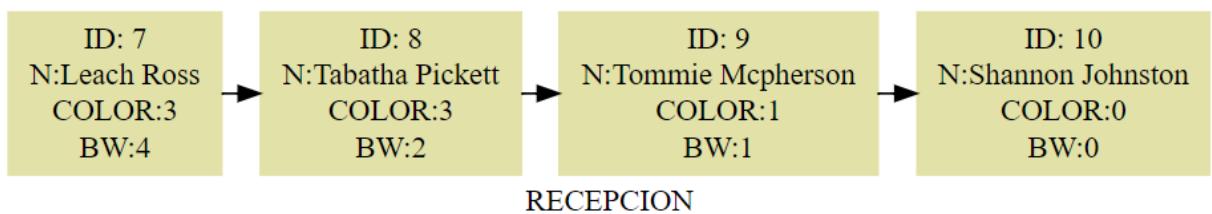
GRAFOS

GRAPHVIZ

Código para generar grafo de la lista de recepción (Cola)

```
1 digraph G [{}  
2 node[ style=filled ,color="#E1E1A8", shape=box];label="RECEPCION";  
3 //agregar nodos  
4 N363771819[label="ID: 7  
5 N:Leach Ross  
6 COLOR:3  
7 BW:4"];  
8 N668386784[label="ID: 8  
9 N:Tabatha Pickett  
10 COLOR:3  
11 BW:2"];  
12 N1329552164[label="ID: 9  
13 N:Tommie Mcpherson  
14 COLOR:1  
15 BW:1"];  
16 N189568618[label="ID: 10  
17 N:Shannon Johnston  
18 COLOR:0  
19 BW:0"];  
20 //enlazar nodos  
21 {rank=same;  
22 N363771819->N668386784;  
23 N668386784->N1329552164;  
24 N1329552164->N189568618;  
25 }  
26 }
```

Grafo resultante



CONCLUSIONES

- La implementación de POO facilita el desarrollo de la solución para proyectos.
- La utilización de estructuras de programación secuenciales, cíclicas y condicionales para un mejor desarrollo.
- Para una mejor visualización de los TDA's se utilizó un software externo llamado Graphviz.