UNIVERSIDAD NACIONAL DEL ALTIPLANO DE PUNO

Facultad de ingeniería mecánica eléctrica, electrónica y sistemas

Escuela profesional de ingeniería de sistemas

Actividad 01 – Complemento

PRESENTADO POR:

CCALLO ARPITA JHAK ESNAYDER

CURSO:

ALGORITMOS Y ESTRUCTURAS DE DATOS

DOCENTE:

ZANABRIA GALVEZ ALDO HERNAN

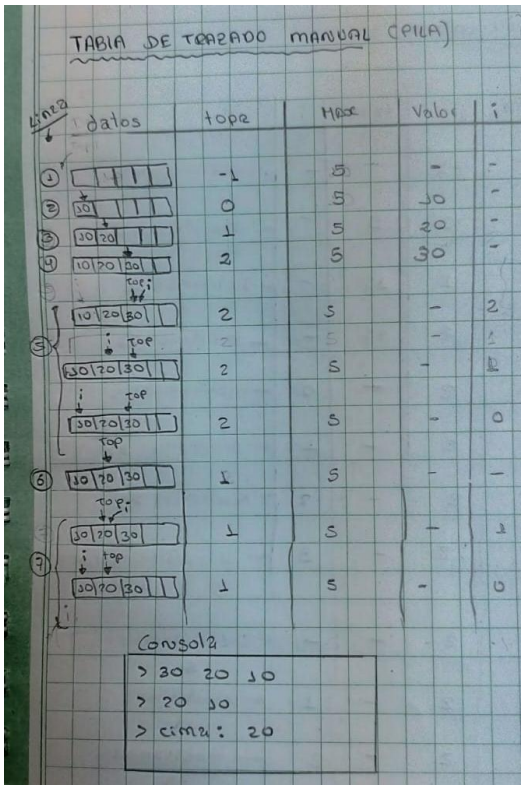
SEMESTRE IV

PUNO – PERÚ

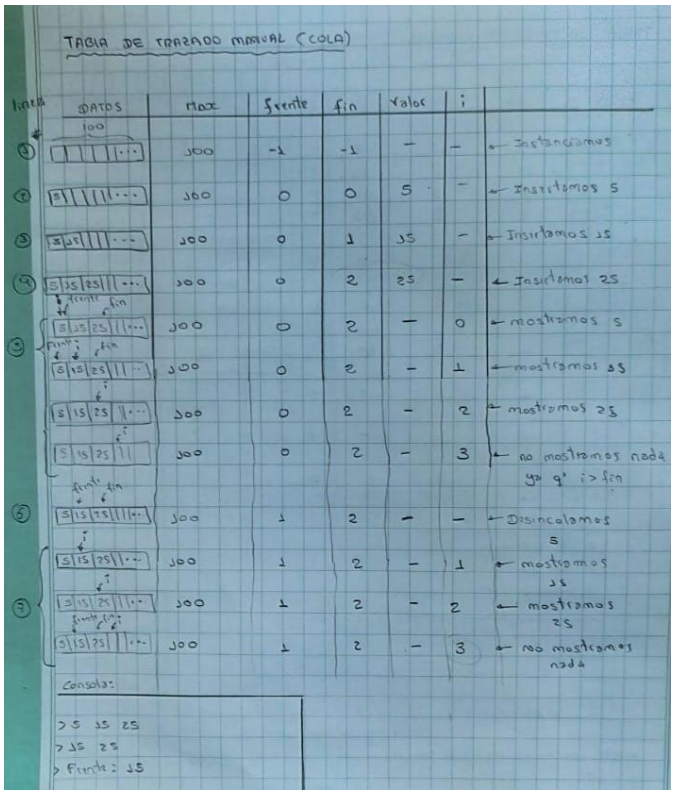
13 de Abril - 2025

1- Tabla de trazado manual:

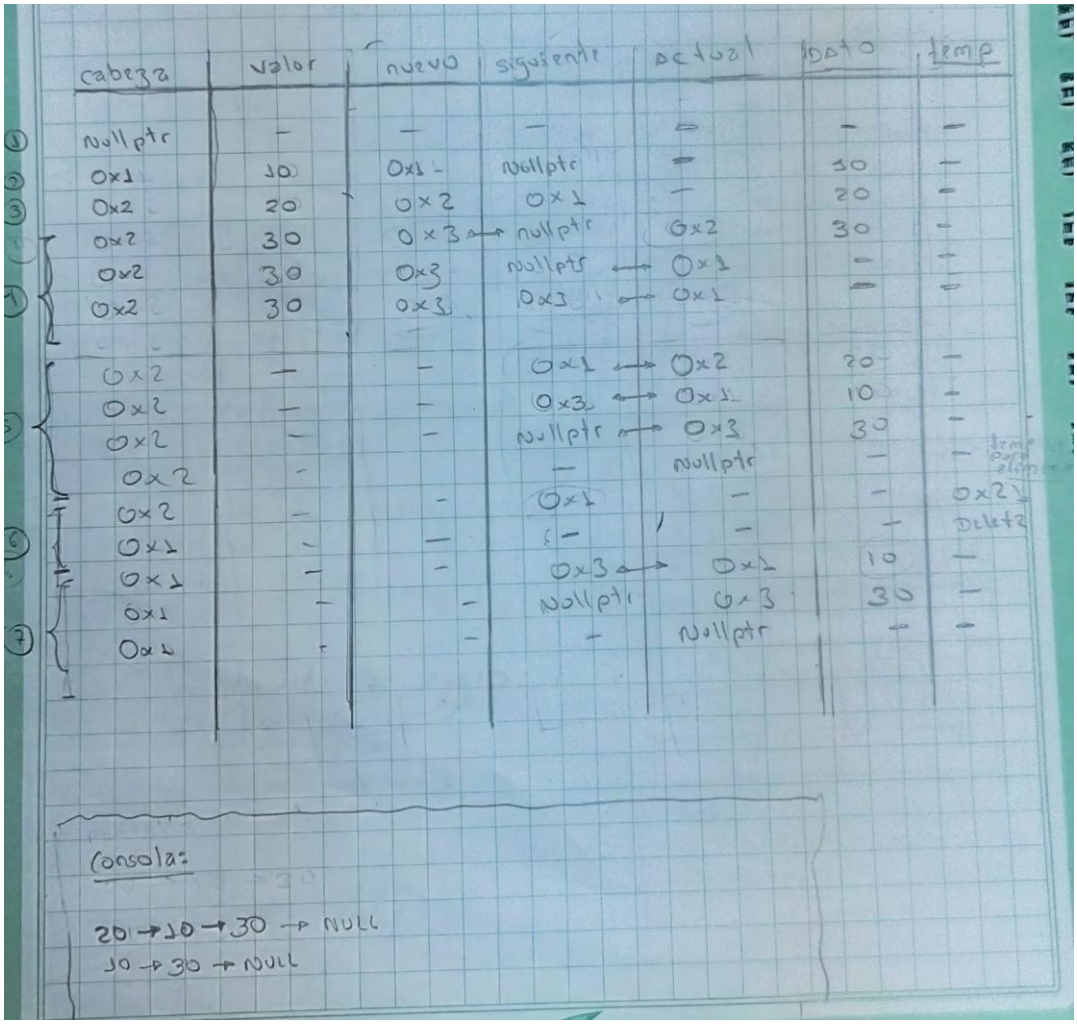
Pilas:



Colas:



Listas:


cabeza
valor
nuevo
siguiente
actual
dato
temp
Nullptr
0x1
10
0x1
Nullptr
0x2
0x1
20
0x2
0x3
Nullptr
0x3
30
0x2
Nullptr
0x1
0x3
0x1
Consola:
20 → 10 → 30 → NULL
10 → 30 → NULL

2- Capturas del código en ejecución en OnlineGDB

Pilas:

main.cpp

```cpp
#include <iostream>
using namespace std;
#define MAX 10

class Pila {
private:
    int datos[MAX];
    int tope;

public:
    Pila() {
        tope = -1;
    }

    bool estaVacia() {
        return tope == -1;
    }

    bool estaLlena() {
        return tope == MAX - 1;
    }

    void apilar(int valor) {
        if (estaLlena()) {
            cout << "Pila llena\n";
            return;
        }
        datos[++tope] = valor;
```

```
30 20 10
20 10
Cima: 20
```

Colas:

main.cpp

```cpp
#include <iostream>
#define MAX 100
using namespace std;

class Cola {
private:
    int datos[MAX];
    int frente, fin;

public:
    Cola() {
        frente = fin = -1;
    }

    bool estaVacia() {
        return frente == -1;
    }

    bool estaLlena() {
        return fin == MAX - 1;
    }

    void encolar(int valor) {
        if (estaLlena()) {
            cout << "Cola llena\n";
            return;
        }
        if (estaVacia())
```

```
5 15 25
15 25
Frente: 15
```

Listas:

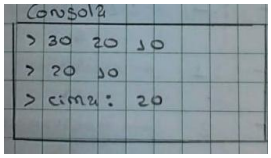
```cpp
#include <iostream>
using namespace std;

struct Nodo {
 int dato;
 Nodo* siguiente;
 };


 class Lista
 {
 private:
      Nodo* cabeza;

 public:
       Lista()
       {
       cabeza = NULL;
          }

    void insertarInicio(int valor) {
    Nodo* nuevo = new Nodo();
    nuevo->dato = valor;
    nuevo->siguiente = cabeza;
    cabeza = nuevo;
        }
```

```
20 -> 10 -> 30 -> NULL
10 -> 30 -> NULL
10 -> 15 -> 30 -> NULL
```

3- Comparación entre salida real y teórica

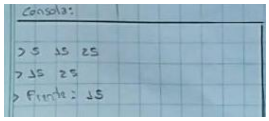
Pilas:

```
30 20 10
20 10
Cima: 20
```



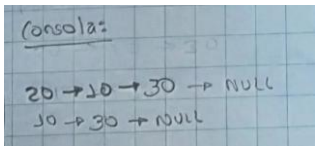
Colas:

```
5 15 25
15 25
Frente: 15
```



Listas:

```
20 -> 10 -> 30 -> NULL
10 -> 30 -> NULL
```



La salida es la misma para todas las estructuras de datos

4- Lista de errores o comportamientos inesperados

El único error que se encontró fue al compilar el código de la estructura de la pila ya que en este no se definía el valor de MAX que representa el tamaño del array.

5- Propuesta de mejora en el código o el control de errores

Añadir una verificación en la función mostrar que diga explícitamente que no hay elementos en la estructura, ya sea de la pila o la cola.

Implementar una **cola circular**, donde frente y fin se muevan en forma circular esto con el fin de no desperdiciar espacio una vez que la cola se llene y se hayan desencolado algunos elementos.

Usar nodos en lugar de arrays para las pilas y colas, esto evitara el desperdicio de memoria cuando no se utiliza todos los espacios de los arrays.

6- Conclusiones personales (mínimo 3)

- Implementar una pila o una cola estáticas es sencillo y comprensible, pero estas estructuras son limitadas y poco eficientes cuando se quiere trabajar con un numero de datos variables.

- El código proporcionado además de funcionar correctamente, es robusto ya que proporciona mensajes cuando cuando algo sale “mal” como cuando tratamos de desapilar una pila que ya esta vacía o cuando apilamos un numero de elementos que supera el máximo posible.

- Al analizar el código y seguir el flujo de datos en las tres estructuras (lista enlazada, pila y cola), se puede comprender con mayor claridad la diferencia entre las estructuras estáticas y dinámicas. Esto no solo facilita entender cómo funcionan internamente, sino que también permite identificar los beneficios y limitaciones de cada tipo de implementación.

Link a los códigos en online GDB:

Pila: https://onlinegdb.com/TMKb59ZwY

Cola: https://onlinegdb.com/qLnHDxMPP

Lista: https://onlinegdb.com/X5_8z4vgt

Función de lista que permite insertar un nuevo nodo en cualquier nodo de la lista.

```
void insertarEn(int valor, int vNodo){

    Nodo* nuevo= new Nodo();
    nuevo->dato =valor;
    nuevo->siguiente=NULL;

    if(cabeza==NULL){
        cout<<"La lista esta vacia, no pudes insertar el nodo despues de "<<vNodo<<endl;
        delete nuevo;
    }

    else{
        Nodo* temp = cabeza;//creamos un temporal para poder recorrer la lista

        while(temp!=NULL ){//revisamos hasta el ultimo nodo

            if(temp->dato==vNodo ){
                Nodo* temp2 = temp->siguiente;//creamos otro temporal para no perder al resto de la lista
                temp->siguiente=nuevo;
                nuevo->siguiente=temp2;
                return;//una vez insertado salimos de la funcion
            }

            temp=temp->siguiente;
        }
        cout<<"el nodo "<<vNodo<<" no existe no puede insertar un nuevo nodo"<<" "<<endl;
    }
}
```