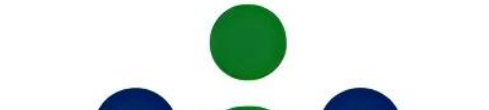

Ejercicio 3

Docente: Jimmy Nataniel Requena Llorentty

Materia: Programación III

Carrera: Ingeniería En Sistemas

Estudiantes: Joaquin Marcos Maita Flores

Santa Cruz – Bolivia

2025

Mini-Cadena

```
##include <iostream>


struct Nodo {
    int dato;
    Nodo* siguiente;

    Nodo(int valor_dato) : dato(valor_dato), siguiente(nullptr) {} // Constructor
conciso
};

int main() {
    // 1. Crear el primer nodo (cabeza de nuestra mini-lista)
    Nodo* cabeza = new Nodo(10); // Usamos 'new' porque queremos memoria dinámica
    std::cout << "Creado primer nodo (cabeza) con dato: " << cabeza->dato <<
std::endl;

    // 2. Crear un segundo nodo
    Nodo* segundoNodo = new Nodo(20);
    std::cout << "Creado segundo nodo con dato: " << segundoNodo->dato <<
std::endl;

    // 3. ¡ENLAZARLOS!
    // El puntero 'siguiente' del primer nodo (cabeza) ahora apunta al
segundoNodo
    cabeza->siguiente = segundoNodo;
    std::cout << "Enlazando cabeza->siguiente con segundoNodo." << std::endl;

    // 4. Crear un tercer nodo
    Nodo* tercerNodo = new Nodo(30);
    std::cout << "Creado tercer nodo con dato: " << tercerNodo->dato <<
std::endl;

    // 5. Enlazar el segundo nodo con el tercero
    segundoNodo->siguiente = tercerNodo; // O cabeza->siguiente->siguiente =
tercerNodo;
    std::cout << "Enlazando segundoNodo->siguiente con tercerNodo." << std::endl;

    // ¿Cómo accedemos a los datos ahora?
    std::cout << "\nRecorriendo la mini-lista:" << std::endl;
    std::cout << "Dato en cabeza: " << cabeza->dato << std::endl;
    std::cout << "Dato en el segundo nodo (via cabeza->siguiente): " << cabeza-
>siguiente->dato << std::endl;
```

```
    std::cout << "Dato en el tercer nodo (via cabeza->siguiente->siguiente): "
              << cabeza->siguiente->siguiente->dato << std::endl;

    std::cout<< "\nJoaquin Marcos Maita Flores : Primera Lista Simple 2025/06/10
hrs. 20:21"<<std::endl;

    // ¡IMPORTANTE! Liberar la memoria dinámica cuando ya no se necesite
    // Se debe hacer en orden inverso o con cuidado para no perder punteros
    std::cout << "\nLiberando memoria..." << std::endl;
    delete cabeza->siguiente->siguiente; // Borra el tercer nodo (tercerNodo)
    cabeza->siguiente->siguiente = nullptr; // Buena práctica
    std::cout << "Tercer nodo liberado." << std::endl;

    delete cabeza->siguiente; // Borra el segundo nodo (segundoNodo)
    cabeza->siguiente = nullptr; // Buena práctica
    std::cout << "Segundo nodo liberado." << std::endl;

    delete cabeza; // Borra el primer nodo
    cabeza = nullptr; // Buena práctica
    std::cout << "Primer nodo (cabeza) liberado." << std::endl;
    std::cout<< "\nJoaquin Marcos Maita Flores : Primera Lista Simple 2025/06/10
hrs. 20:21"<<std::endl;

    return 0;
}
```

CÓDIGO EJECUTADO

```
input
Creado primer nodo (cabeza) con dato: 10
Creado segundo nodo con dato: 20
Enlazando cabeza->siguiente con segundoNodo.
Creado tercer nodo con dato: 30
Enlazando segundoNodo->siguiente con tercerNodo.

Recorriendo la mini-lista:
Dato en cabeza: 10
Dato en el segundo nodo (via cabeza->siguiente): 20
Dato en el tercer nodo (via cabeza->siguiente->siguiente): 30

Joaquin Marcos Maita Flores : Primera Lista Simple 2025/06/10 hrs. 20:21

Liberando memoria...
Tercer nodo liberado.
Segundo nodo liberado.
Primer nodo (cabeza) liberado.

Joaquin Marcos Maita Flores : Primera Lista Simple 2025/06/10 hrs. 20:21


...Program finished with exit code 0
Press ENTER to exit console.
```

Este ejemplo enseña:

- **struct Nodo: Cómo se define un nodo (dato + puntero al siguiente).**
- **new / delete: Gestión de memoria dinámica.**
- **Enlazado: Cómo conectar nodos (nodo->sig = nuevo_nodo).**
- **Recorrido: Uso de un puntero temporal (temp) para moverse por la lista.**
- **Liberar memoria: Eliminar todos los nodos para evitar *memory leaks*.**