**Estructura del Código**

El programa implementa una **cola dinámica** utilizando una lista enlazada en C. Se compone de tres archivos:

1. **main.c** → Maneja la lógica principal y el menú de opciones.
2. **cola.c** → Contiene las funciones para manipular la cola.
3. **cola.h** → Define la estructura nodo y los prototipos de funciones.

**1. Estructura de Datos (Definida en cola.h)**

La cola se implementa con la siguiente estructura:

c

CopiarEditar

struct nodo {

int ID;

struct nodo \*siguiente;

};

Cada nodo contiene:

* ID: un entero que identifica al nodo.
* siguiente: un puntero al siguiente nodo en la cola.

**2. Funciones para Manejo de la Cola (cola.c)**

**a) crear\_nodo(int ID)**

* Reserva memoria (malloc) para un nuevo nodo.
* Asigna el ID proporcionado.
* Inicializa siguiente como NULL.

c

CopiarEditar

struct nodo \*crear\_nodo(int ID) {

struct nodo \*nodo = (struct nodo \*)malloc(sizeof(struct nodo));

if (nodo != NULL) {

nodo->ID = ID;

nodo->siguiente = NULL;

}

return nodo;

}

**b) insertar\_nodo(struct nodo \*\*cola, struct nodo \*nodo)**

* Si la cola está vacía, el nodo se convierte en el primer elemento.
* Si ya hay elementos, recorre la lista hasta el último nodo y enlaza el nuevo nodo.

c

CopiarEditar

void insertar\_nodo(struct nodo \*\*cola, struct nodo \*nodo) {

struct nodo \*actual;

if (\*cola == NULL) {

\*cola = nodo;

} else {

actual = \*cola;

while (actual->siguiente != NULL) {

actual = actual->siguiente;

}

actual->siguiente = nodo;

}

}

**c) extraer\_nodo(struct nodo \*\*cola)**

* Extrae el primer nodo de la cola (FIFO).
* Si la cola está vacía, devuelve NULL.
* Si hay elementos, avanza el puntero de la cola al siguiente nodo y retorna el nodo extraído.

c

CopiarEditar

struct nodo \*extraer\_nodo(struct nodo \*\*cola) {

struct nodo \*nodo;

if (\*cola == NULL) return NULL;

nodo = \*cola;

\*cola = (\*cola)->siguiente;

nodo->siguiente = NULL;

return nodo;

}

**d) eliminar\_nodo(struct nodo \*nodo)**

* Libera la memoria de un nodo eliminado.

c

CopiarEditar

void eliminar\_nodo(struct nodo \*nodo) {

free(nodo);

}

**e) imprimir\_cola(struct nodo \*cola)**

* Recorre la cola y muestra los ID de cada nodo.

c

CopiarEditar

void imprimir\_cola(struct nodo \*cola) {

if (cola == NULL) {

printf("La cola está vacía.\n");

} else {

printf("Cola: ");

while (cola != NULL) {

printf("%d -> ", cola->ID);

cola = cola->siguiente;

}

printf("NULL\n");

}

}

**f) vaciar\_cola(struct nodo \*\*cola)**

* Extrae y libera cada nodo de la cola hasta dejarla vacía.

c

CopiarEditar

void vaciar\_cola(struct nodo \*\*cola) {

struct nodo \*temp;

while (\*cola != NULL) {

temp = extraer\_nodo(cola);

printf("Nodo con ID %d eliminado.\n", temp->ID);

eliminar\_nodo(temp);

}

printf("Cola vaciada correctamente.\n");

}

**3. Lógica Principal (main.c)**

El programa tiene un menú que permite al usuario interactuar con la cola.

**Menú y Ciclo Principal**

Se ejecuta un bucle do-while hasta que el usuario elija salir (opción 4).

c

CopiarEditar

do {

printf("\n--- MENU ---\n");

printf("1. Agregar\n2. Quitar\n3. Imprimir\n4. Salir\n");

printf("Seleccione una opción: ");

scanf("%d", &opcion);

**Opciones del Menú**

**Opción 1: Agregar un nodo**

* Solicita un ID al usuario.
* Crea un nuevo nodo y lo inserta en la cola.

c

CopiarEditar

case 1:

printf("Ingrese ID: ");

scanf("%d", &ID);

nodo = crear\_nodo(ID);

if (nodo == NULL) {

printf("Error: Memoria insuficiente.\n");

} else {

insertar\_nodo(&cola, nodo);

printf("Nodo con ID %d agregado.\n", ID);

}

break;

**Opción 2: Extraer un nodo**

* Extrae el primer nodo de la cola e informa al usuario cuál se eliminó.

c

CopiarEditar

case 2:

nodo = extraer\_nodo(&cola);

if (nodo == NULL) {

printf("La cola está vacía. No hay nodos para extraer.\n");

} else {

printf("Nodo con ID %d extraído.\n", nodo->ID);

eliminar\_nodo(nodo);

}

break;

**Opción 3: Imprimir la cola**

* Muestra los elementos en orden de llegada.

c

CopiarEditar

case 3:

imprimir\_cola(cola);

break;

**Opción 4: Vaciar la cola y salir**

* Llama a vaciar\_cola() antes de finalizar el programa.

c

CopiarEditar

case 4:

vaciar\_cola(&cola);

printf("Saliendo...\n");

break;

**4. Ejemplo de Ejecución**

**Entrada:**

CopiarEditar

1

10

1

20

3

2

3

4

**Salida:**

yaml

CopiarEditar

Ingrese ID: 10

Nodo con ID 10 agregado.

Ingrese ID: 20

Nodo con ID 20 agregado.

Cola: 10 -> 20 -> NULL

Nodo con ID 10 extraído.

Cola: 20 -> NULL

Nodo con ID 20 eliminado.

Cola vaciada correctamente.

Saliendo...

**Conclusión**

Este código implementa una **cola dinámica** con operaciones eficientes de inserción y eliminación basadas en listas enlazadas. Maneja memoria dinámicamente (malloc y free), garantiza que la cola se vacíe antes de salir y tiene una interfaz simple para el usuario.