Clase de hoy

Practicas **Colas** implementadas **Dinámicamente**

Resumen de Clases **Colas con Implementación Dinámica**

Definición / Idea Básica de COLAS

Es una estructura lineal, similares a las pilas, pero con una diferencia en el modo de insertar y eliminar elementos. Es decir, se diferencian de sus "primas" (pilas) tanto en lo estructural como en los algoritmos de alta y baja. Justamente por esto se conocen también como estructuras FIFO, del acrónimo en ingles (First Input First Output), es decir el **primer elemento** *en entrar* será el **primero en salir**. Dicho de otra forma "no se pueden eliminar elementos del medio o del final".

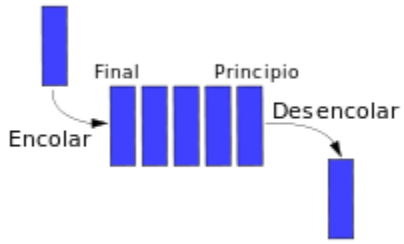
Uso: Se utilizan para almacenar datos que se deben procesar según el orden de llegada.

Ejemplos:

- *La cola de clientes en un banco o supermercado.*
- *La cola de paquetes en un Router o Encaminador de una red.*

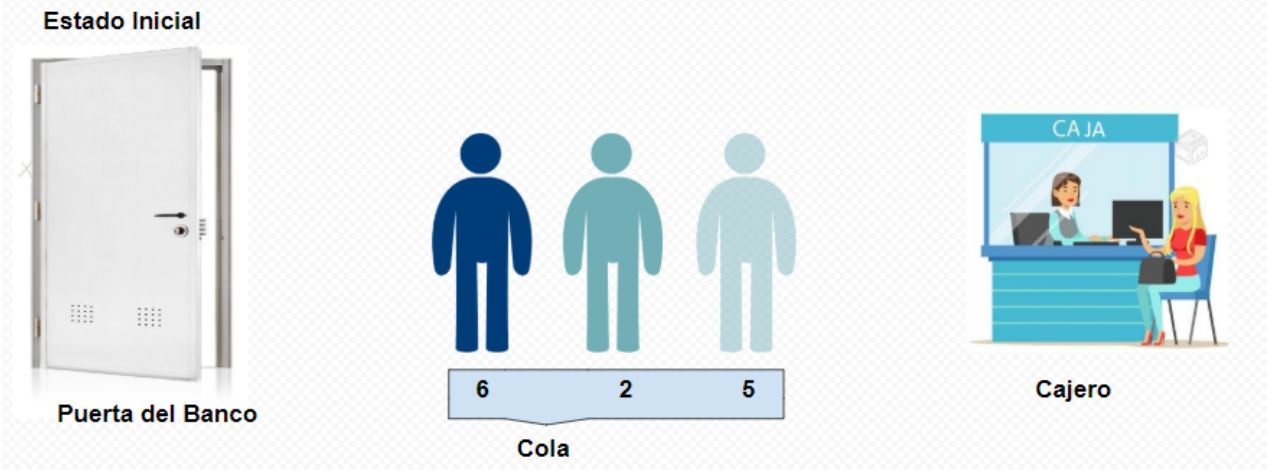
¿Como trabajan?

La forma de trabajo se puede ver en la gráfica simplificada siguiente:



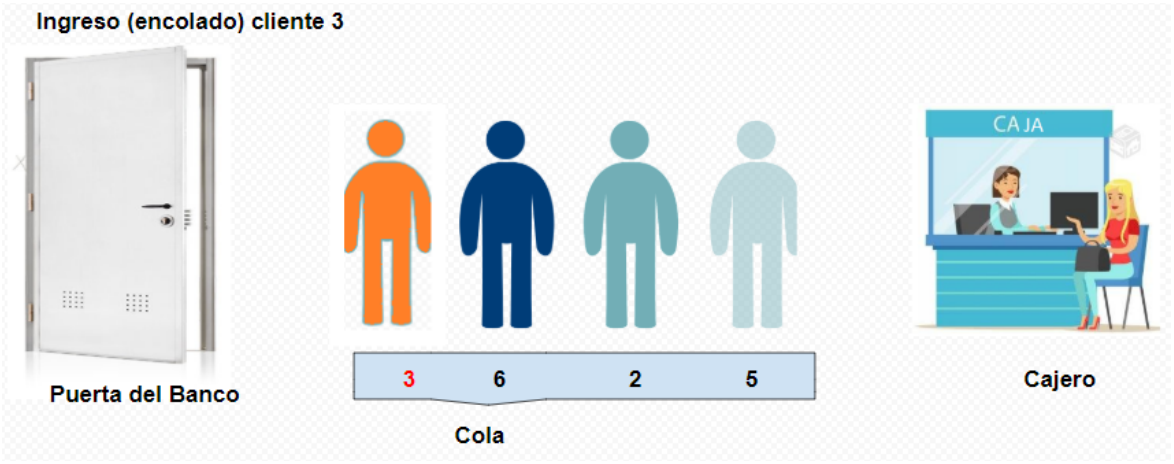
Ejemplo **Cola de Clientes de Banco**

Llevando al ejemplo de los clientes del banco, supongamos que en un momento determinado hay tres clientes, el número 5, el número 2 y el número 6, los cuales llegaron en ese orden:

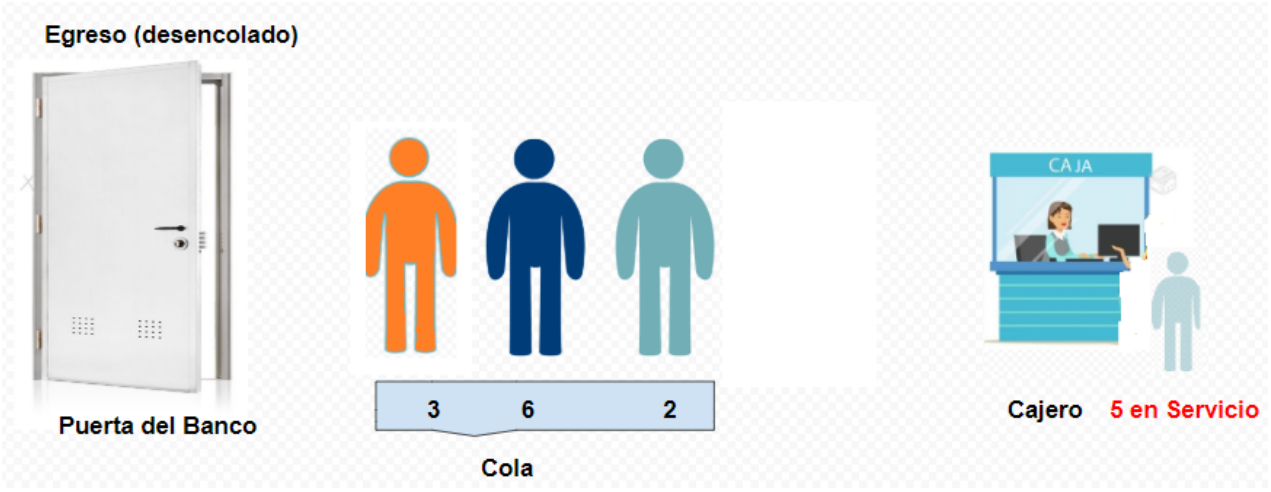


Nota: La chica que se está atendiendo (en servicio), no se considera que esta en cola.

Supongamos que **ingresa** el **cliente 3** (arribo de un nuevo cliente).



Ahora finaliza el servicio del cliente que está en la caja y **egresa** un cliente de cola.
¿Cuál egresará?



Tipos de Implementaciones

Hay dos formas de implementar una cola:

- una alternativa se basa en el uso de vectores (al que llamaremos implementación **estática**, ya que el dimensionado y alojamiento de memoria se define en tiempo de compilación).
- y la que trataremos seguidamente denominada, implementación **dinámica**, ya que no tienen tamaño fijo y el alojamiento se realiza cuando se necesita, en la ejecución en si del proceso de encolado.

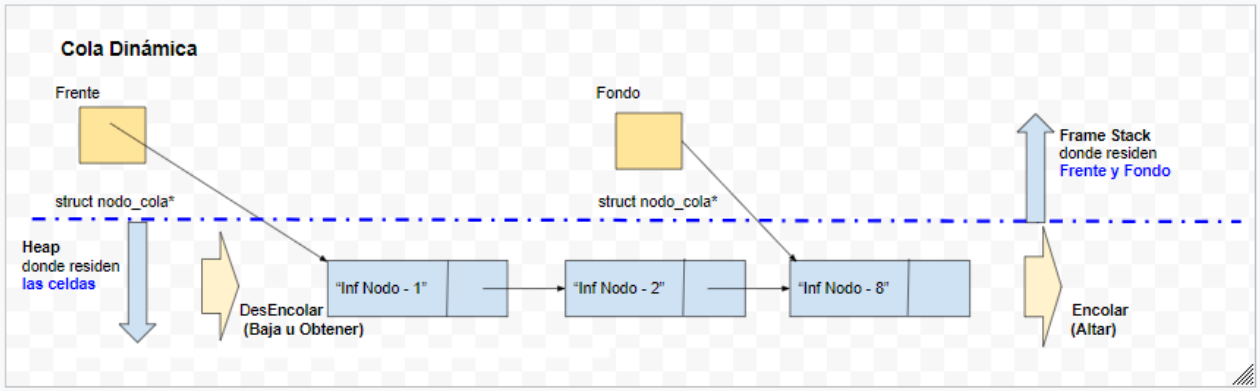
Implementación Dinámica

Una cola se puede implementar con una técnica similar a la que vimos con pilas, utilizando estructuras autoreferenciadas, y definiendo un TDA con su estructura y 3 funciones:

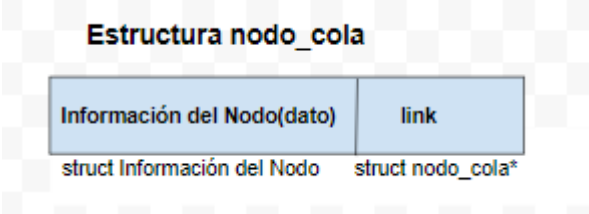
- cola_vacia: devuelve verdadero si la cola no tiene elementos, falso caso contrario.
- cola_agregar (**push**): para encolar en la estructura (altas).
- cola_obtener (**pop**): para recuperar el valor correspondiente según dinámica FIFO.

En cuanto a la referencia inicial, una **cola dinámica** se implementa con 2 punteros: uno apuntando al primer elemento que arribo a la cola (**frente**), y otro al último elemento (**fondo**).

Representación



Definición de Nodos



link en este caso apunta al próximo nodo de la cola. Como en todas las estructuras lineales dinámicas que vimos, cuando link == NULL → representa el fin de cola.

.

En C++

```
struct nodo_cola     {
      struct informacion del nodo….(dato);
      struct nodo_cola * link;
};
typedef struct nodo_cola NCola;

// dentro del espacio de memoria de definición de la cola
NCola * frente = NULL;
NCola * fondo = NULL;
```

Resumen de Características

- Son estructuras autoreferenciadas, como ya mencionamos, similar a todas las estructuras lineales (autoreferencias atributo link), que utiliza los mismos principios que todas las estructuras lineales que trataremos.

- Posee dos puntos de acceso, Único punto de obtención (frente), que es un puntero al primer nodo de la cola. Cuando este puntero tiene NULL significa que la cola está vacía. Fondo, único punto donde se incorporarán las altas.

- Son ideales para almacenar datos que deben ser procesados en el mismo orden que su llegada.

- No es válido pensar en recorrer todos los elementos en busca de alguna característica, no están pensadas para este fin.

Idea y Pseudocodigo de altas

Alta en Cola

```
1       void cola_agregar(NCola * & frente, NCola * &fondo, int ndato) {
2               NCola * nuevo;                          // def estática del puntero nuevo nodo

3               nuevo = (NCola*) new(NCola);                    // Reserva nodo
4               nuevo->dato                     = ndato;  // informacion del nodo….
5               nuevo->link                     = NULL;   // alta sin empleados al momento
6               if (cola_vacia(frente))
7                       frente          = nuevo;          // cola Vacia, se actualiza tambien frente
8                   else  fondo->link   = nuevo;          // insertamos al final con el puntero fondo
9               fondo           = nuevo;
10      }
```

Cola_Vacia

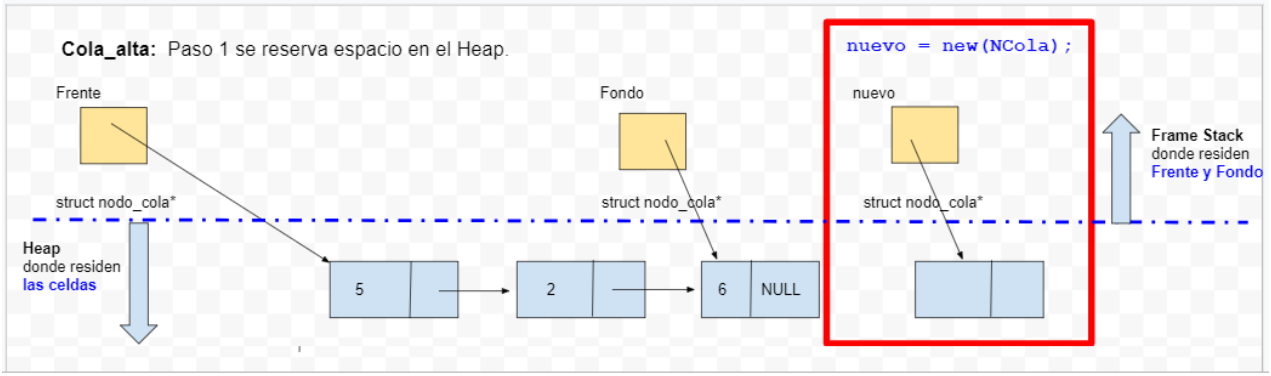
```
1       bool cola_vacia(NCola * frente) {
2           return (        );
3       }
```

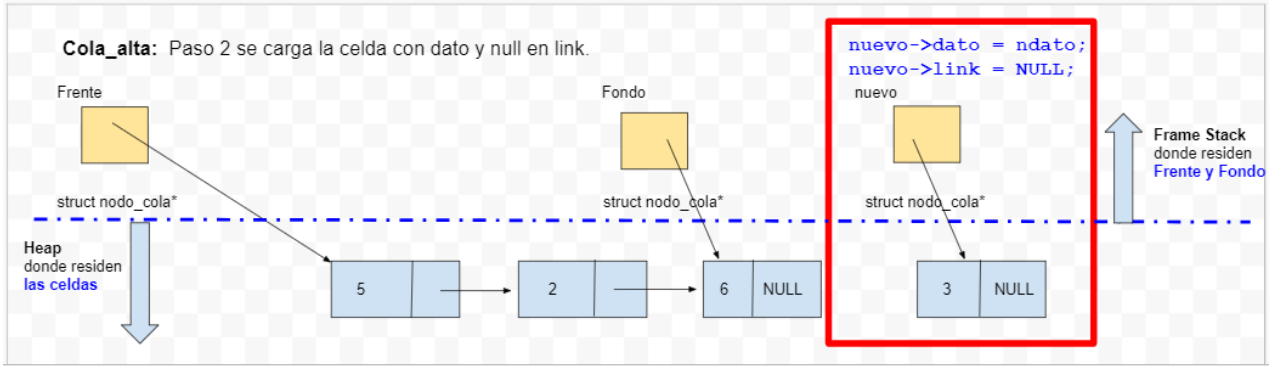
Baja en Cola

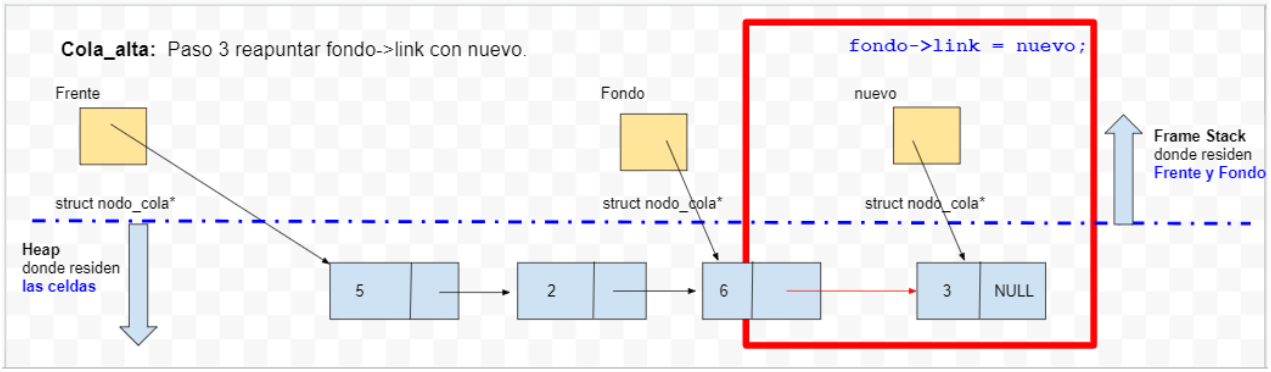
```
1       int cola_obtener(NCola * & frente, NCola * &fondo) {
2               if (!cola_vacia(frente)) {
3                   NCola *aux  = frente;         // def estatica del puntero aux  con frente
4                   int res             = aux->dato;        // carga de dato en el auxiliar n_dato
5                   frente              = frente->link;     // frente apunta al segundo
6                   delete aux;                             // libera el espacio desocupado
7                   if (cola_vacia(frente)) fondo = NULL;   // actualiza fondo si se recupero el ultimo
8                   return n_dato;                          // dato devuelto por la función
9                } else
10                      cout << “Cola Vacia, lanzar excepción”;
11      }
```

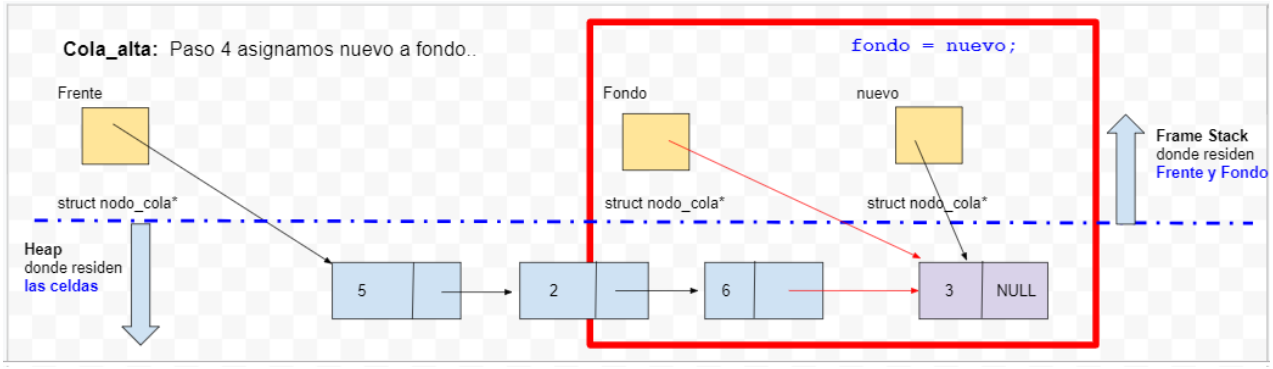
Interpretando el Alta: Ingreso de dato = 3

Nota: nuevo pertenece a otro espacio de direcciones propio del frame stack donde reside cola_agregar.

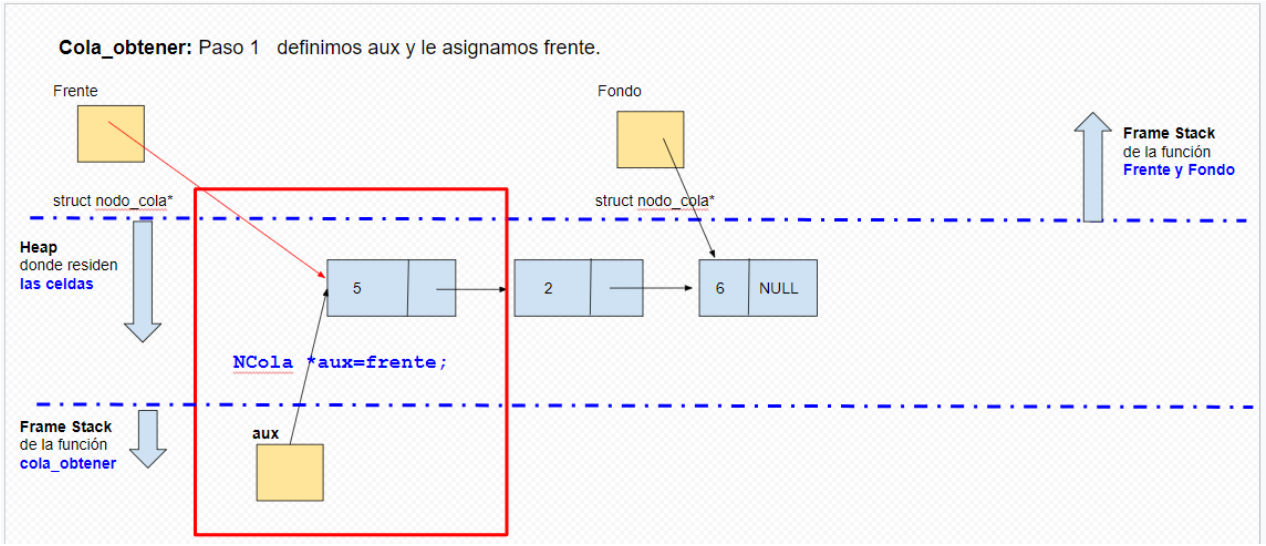


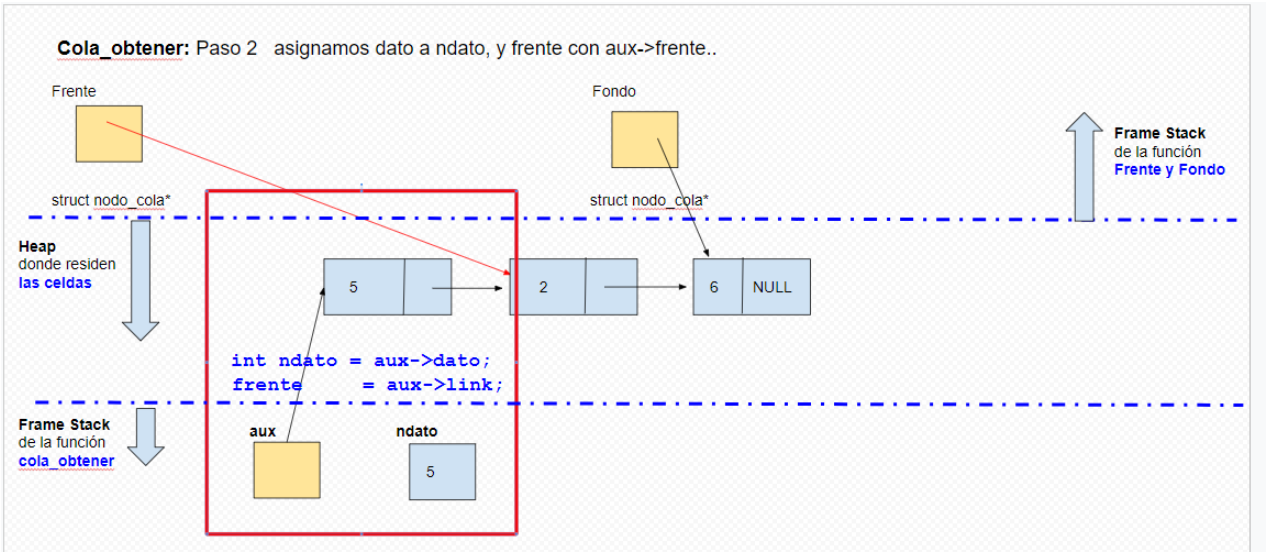


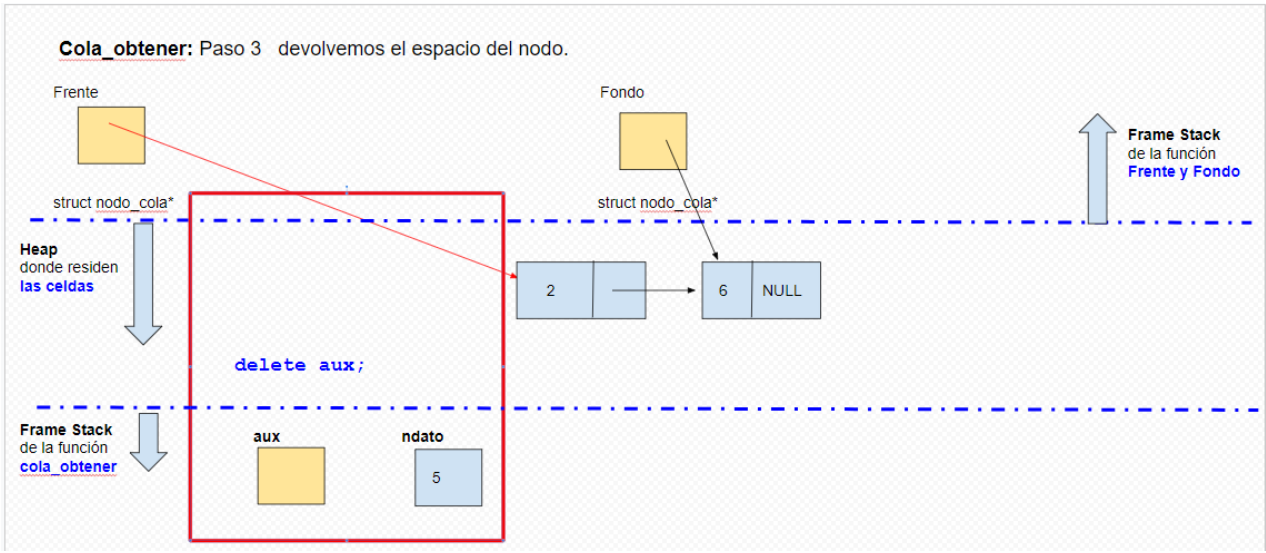


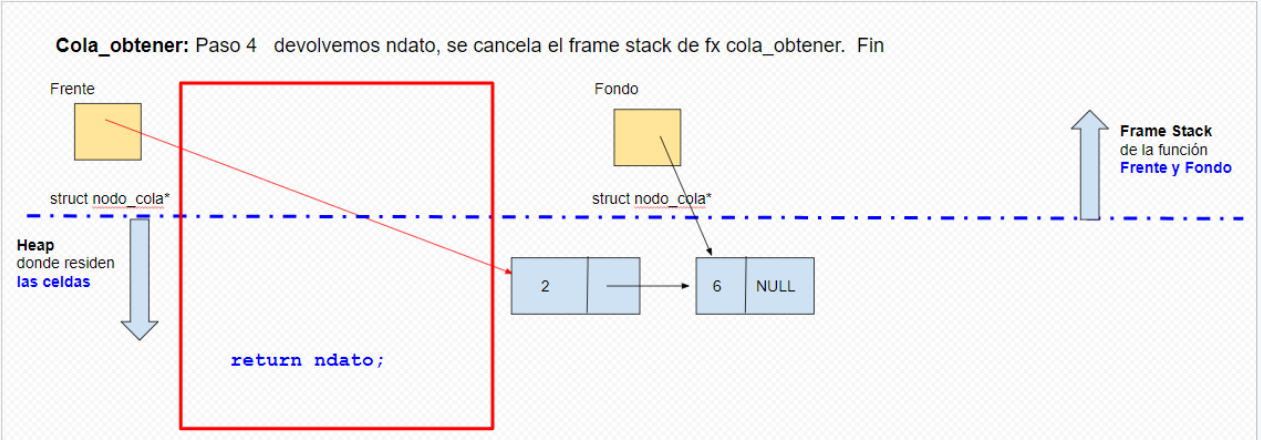


Interpretando el Baja (Obtener):

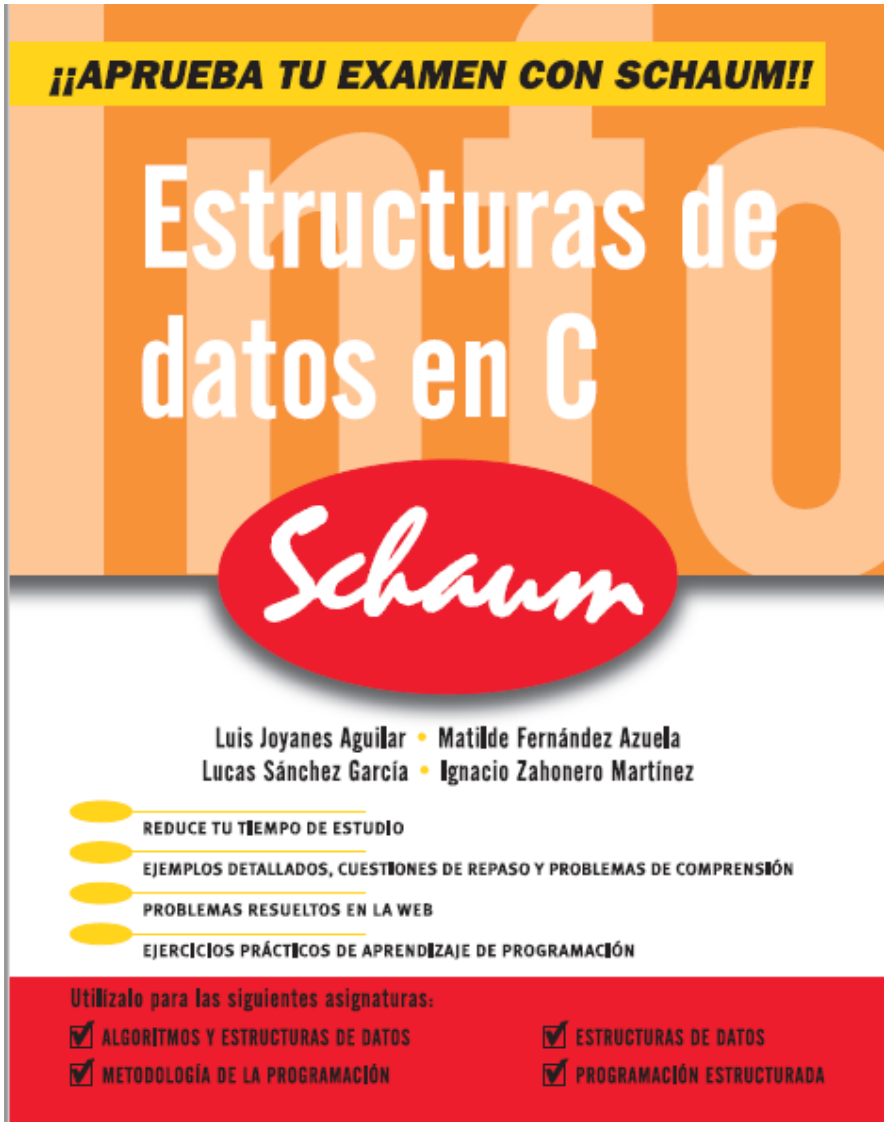








Bibliografía de base



Anexo para probar las funciones

```
#include <iostream>
#include <stdlib.h>

using namespace std;

// Version: 20190404
// DEFINICI?N DE TIPOS.

struct nodo_pila
{
	int dato;
	struct nodo_pila* link;
};
typedef struct nodo_pila NPila;

struct nodo_cola
{
	int dato;
	struct nodo_cola* link;
};
typedef struct nodo_pila NCola;



// DECLARACI?N DE FUNCIONES.


void cola_agregar (NCola* &frete, NCola* &fondo, int ndato);
int cola_obtener (NCola* &frete, NCola* &fondo);
bool cola_vacia (NCola* frete, NCola* fondo);
void cola_debug (NCola* frete, NCola *fondo);                    // usada solo para mostrar el
contenido - NO USAR PARA OTRO FIN


void menu_opcion1 (NCola* &frente, NCola* &fondo);
void menu_opcion2 (NCola* &frente, NCola* &fondo);
void menu_opcion3 (NCola* frente, NCola* fondo);


// DEFINICI?N DE FUNCIONES.

int main (void)
{
	//	NPila* pila = NULL;
	//	NCola* cola_frente = NULL, cola_fondo = NULL;
	NCola* frente=NULL;
	NCola* fondo=NULL;

	int opcion = 0;
	do {
		cout << "************Menu de Opciones************\n";
		cout << endl;
		cout << "****** Lista Simplemente Enlazada ******\n";
		cout << endl;
		cout << "1- Cola agregar.\n";
		cout << "2- Cola obtener.\n";
		cout << "3- Cola debug.\n";
		cout << endl;
		cout << "    0- Salir\n";
		cout << endl;
		cout << "                         Ingrese opcion: ";
		cin >> opcion;
```

```
                cout << endl;
                cout << endl;

                switch(opcion)
                {
                case 1:
                        menu_opcion1 (frente, fondo);
                        break;
                case 2:
                        menu_opcion2 (frente, fondo);
                        break;
                case 3:
                        menu_opcion3 (frente,fondo);
                        break;
                }
        } while ( opcion != 0);

        return 0;
}


void menu_opcion1 (NCola* &frente, NCola* &fondo)
{
        int dato_agregar;

        cout << "Que insertara en la cola?: ";
        cin >> dato_agregar;

    cola_agregar(frente, fondo, dato_agregar);

        cout << endl;
        cout << endl;
}

void menu_opcion2 (NCola* &frente, NCola* &fondo)
{
        int dato;


        dato = cola_obtener(frente, fondo);

        cout << "devuelto: " << dato << endl;

        cout << endl;
        cout << endl;
}

void menu_opcion3 (NCola* frente, NCola* fondo)
{

        cola_debug (frente, fondo);

        cout << endl;
        cout << endl;
}




void cola_agregar (NCola* &frente, NCola* &fondo, int ndato) {
                        NCola * nuevo;        // def estática del puntero nuevo nodo

                        nuevo = new(NCola);                     // Reserva nodo
                        nuevo->dato           = ndato;   // informacion del nodo….
```

```
                        nuevo->link                  = NULL;    // alta sin empleados al momento
                        if (frente != NULL)
                                          fondo->link        = nuevo;       // insertamos al final
con el puntero fondo
                  else    frente = nuevo;              // cola Vacia, se actualiza tambien frente
                fondo             = nuevo;
                        cola_debug(frente,fondo);
}

void cola_debug (NCola* frente, NCola * fondo)
{
        cout << "Cola Debug\n";
        cout << "------------------------\nfrente -> ";
        while (frente != NULL)
        {
                cout << frente->dato << " -> ";
                frente = frente->link;
        }
        cout << "NULL";
        cout << endl;
        cout << "fondo ->";
        if (fondo != NULL)
                  cout << fondo->dato; else cout << "NULL";
        cout << endl;
}

bool cola_vacia(NCola * frente) {
    return (frente == NULL);
}


int cola_obtener(NCola * & frente, NCola * &fondo) {
        if (!cola_vacia(frente)) {
        NCola *aux  = frente;             // def estatica del puntero aux  con frente
        int n_dato  = aux->dato;            // carga de dato en el auxiliar n_dato
        frente      = aux->link;            // frente apunta al segundo
           delete aux;                                          // libera el espacio desocupado
        if (cola_vacia(frente)) {
                   fondo = NULL;
           }
           return n_dato;                           // dato devuelto por la función
        } else
               cout << "Cola Vacia, lanzar excepción";
}

//Versión 2019/04/01
```