Lenguaje de Programación 1

Colas - Queue

Algoritmos y Estructura de Datos.
Una perspectiva en C - Capitulo 11
Aguilar & Martínez

Colas - Queue

Conceptos claves

- Concepto de Cola
- Cola basada en array
- Cola basada en lista enlazada
- Estructura FIFO

Contenido

Concepto de cola

Cola implementada con array lineales

Cola implementada con array circulares

Cola implementada con listas enlazadas

Bicolas: colas de doble entrada

Introducción

- La estructura de datos es muy utilizada en la vida cotidiana y en computación.
- Esta estructura almacena y recupera sus elementos atendiendo a un estricto orden.
- Las colas se conocen como estructuras FIFO (*First-in, First-out*)
- Las colas tienen numerosas aplicaciones en el mundo de la computación: colas de mensajes, colas de tareas a realizar en una impresora, colas de prioridades.

Concepto de Cola

Una cola es una estructura de datos que almacena elementos en una lista y permite acceder a los datos por uno de los extremos de la lista.

Un elemento se inserta en la cola de la lista y se elimina por el frente de la lista.

Operaciones

Insertar - Encolar (Enqueue)

Añade un elemento por el extremo **final** de la cola

Quitar - Decolar (Dequeue)

Elimina o extrae un elemento por el extremo opuesto o **frente** de la cola.

Implementaciones

Colas mediante array

- Dimensión de la cola es fija
- La cola puede estar vacía y llena

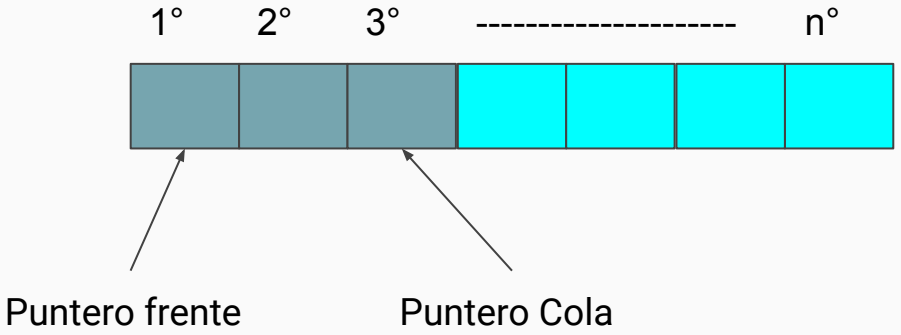
Cola mediante listas

- Dimensión de la cola es dinámica
- La cola puede estar vacía y llena según la capacidad de la memoria RAM

Colas mediante array lineales

Componentes

1. Arreglo de n elementos
2. Puntero a cola
3. Puntero a frente

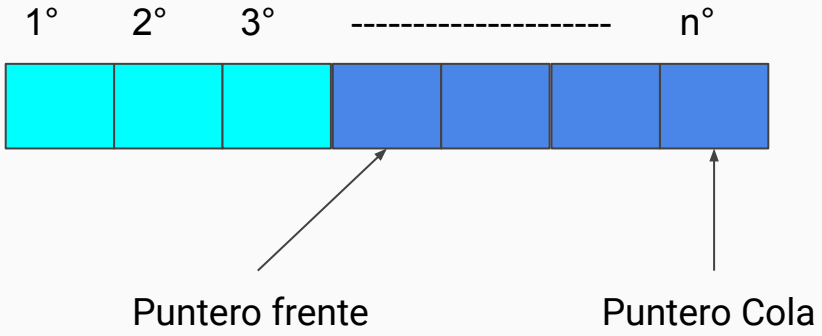


Componentes

1. Arreglo de n elementos
2. Puntero a cola
3. Puntero a frente

INCONVENIENTE

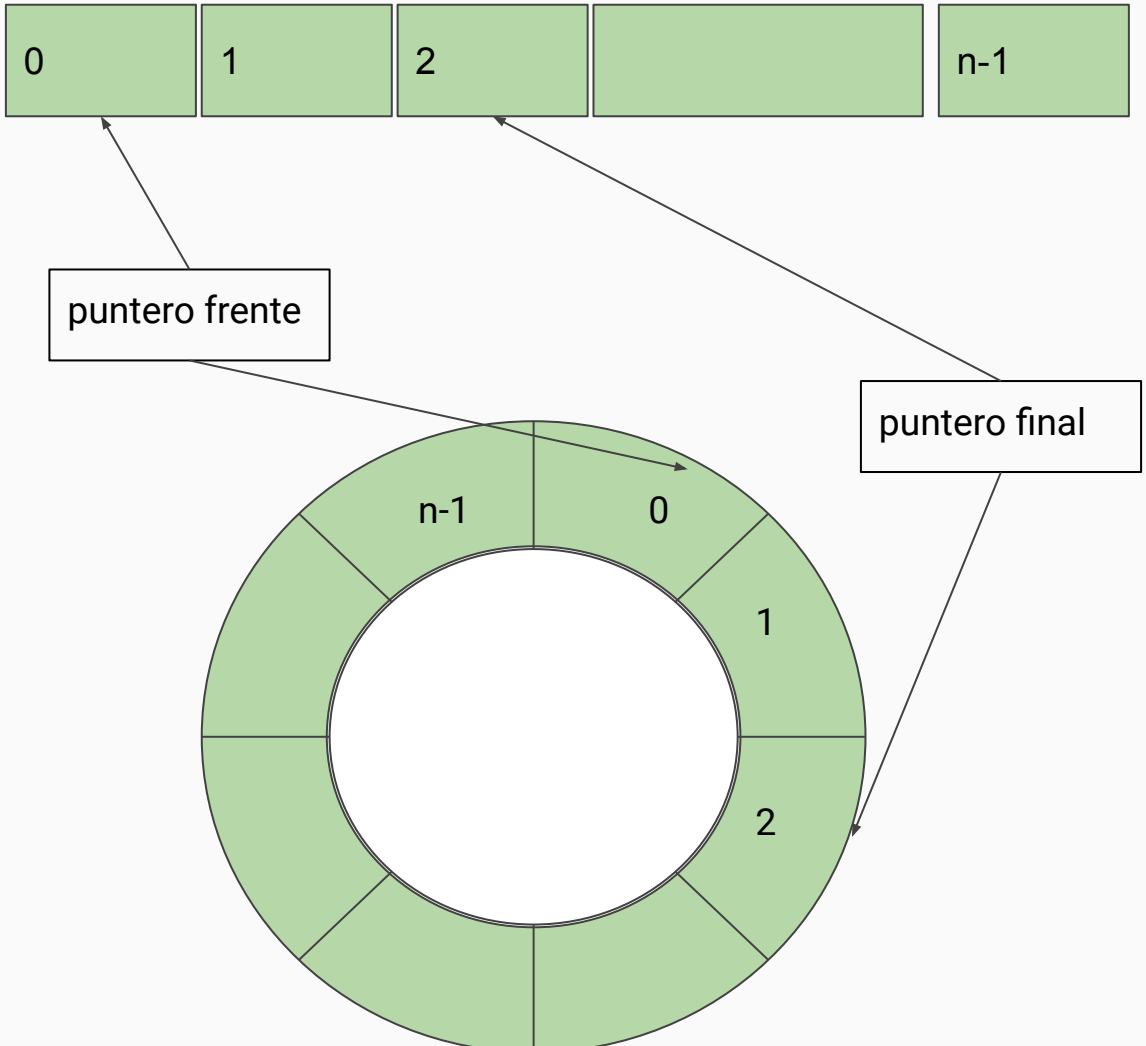
El avance puede dejar huecos en la izquierda, pudiendo no poder encolar nuevos elementos aunque haya lugar en la cola



Colas mediante arrays circulares

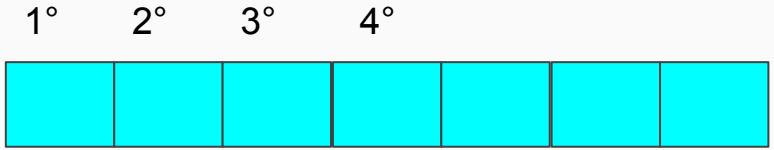
Componentes

1. Arreglo de n elementos
2. Puntero a final
3. Puntero a frente



Ejemplo básico de funcionamiento

Cola vacía
PunteroFrente = 0
PunteroFinal = -1



Encolar(50)
PunteroFrente = 0
PunteroFinal = 0

Encolar(25)
PunteroFrente = 0
PunteroFinal = 1

Decolar(50)
PunteroFrente = 1
PunteroFinal = 1

Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

```
/*archivo colaarray.h*/

#define MAXTAMQ 100

struct COLA{
    TipoDato listaCola[MAXTAMQ];
    int frente, final;
}

typedef struct COLA Cola;

/*Operaciones sobre la Cola*/
void crearCola(Cola * cola);
void insertarCola(Cola * cola, TipoDato elemento);
TipoDato quitarCola(Cola * cola);
void limpiarPila(Cola * cola);
TipoDato frenteCola(Cola * cola);
int colaLlena(Cola cola);
int colaVacia(Cola cola);
```

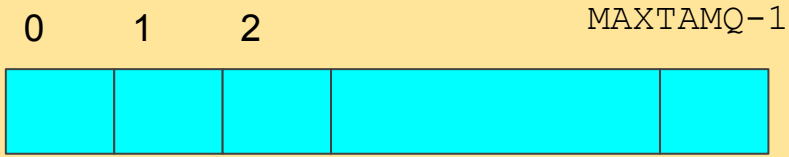
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Crea una cola nueva*/
void crearCola(Cola * cola){
    cola -> frente = 0;
    cola -> final  = MAXTAMQ - 1;
}
```

Cola vacía
Frente = 0
Final = MAXTAMQ-1



Funciones

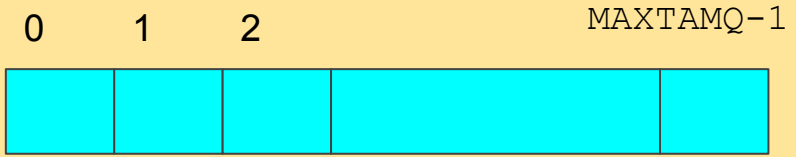
- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Devuelve la posición del siguiente elemento*/
int siguiente(int n){
     return ((n + 1) % MAXTAMQ);
}
```

Si n = 1
retorna 2

Si n = MAXTAMQ-1
retorna 0

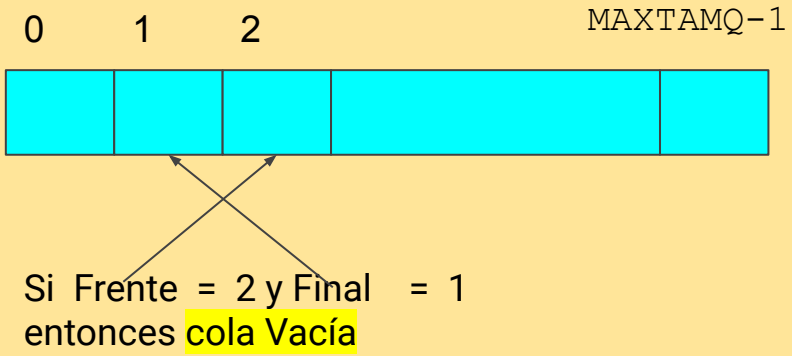


Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Verifica si la Cola está vacía*/
int colaVacia(Cola pila){
    return (cola.frente == siguiente(cola.final));
}
```

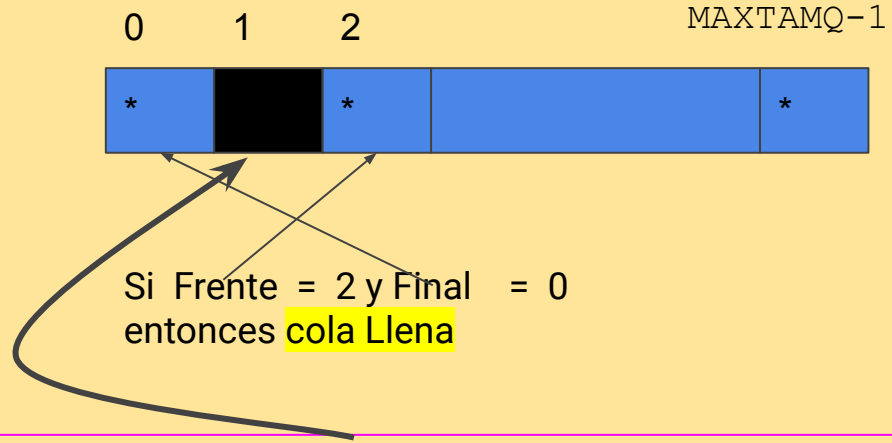


Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Verifica si la cola está llena*/
int colaLlena(Cola cola){
     return (cola.frente == siguiente(siguiente(cola.final)));
}
```



Note que para distinguir una cola llena de una vacía se deja una posición no ocupada

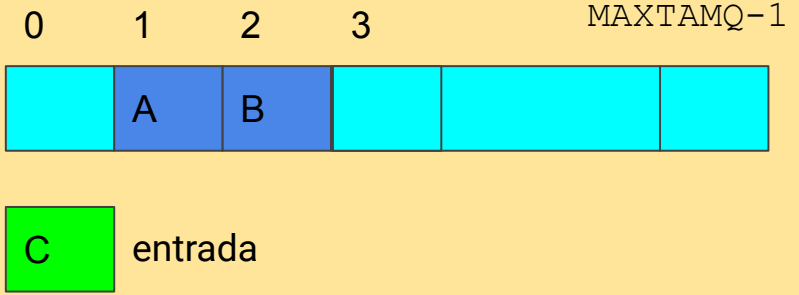
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Inserta en la cola*/
void insertarCola(Cola * cola, TipoDato entrada){
    if(colaLlena(*cola)){
        puts("Desbordamiento de cola\n");
        exit (1);
    }
    /*avance circular al siguiente del final*/

    cola -> final = siguiente(cola -> final);
    cola -> listaCola[cola -> final] = entrada;
}
```

Frente = 1
Final = 2



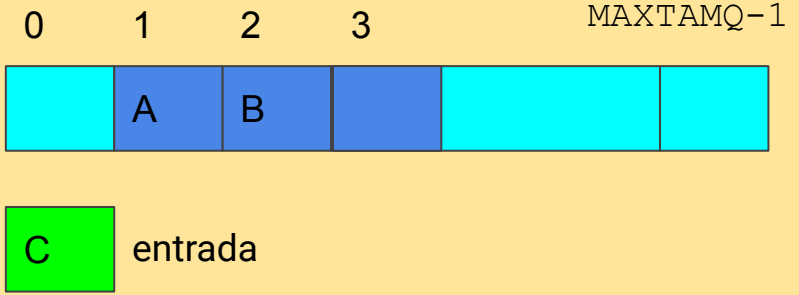
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Inserta en la cola*/
void insertarCola(Cola * cola, TipoDato entrada){
    if(colaLlena(*cola)){
        puts("Desbordamiento de cola\n");
        exit (1);
    }
    /*avance circular al siguiente del final*/

    cola -> final = siguiente(cola -> final);
    cola -> listaCola[cola -> final] = entrada;
}
```

Frente = 1
Final = 3



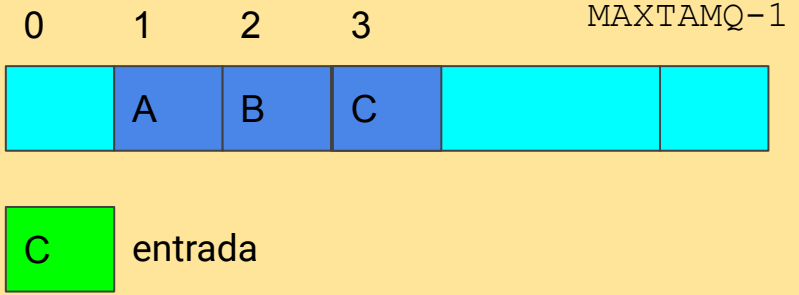
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Inserta en la cola*/
void insertarCola(Cola * cola, TipoDato entrada){
    if(colaLlena(*cola)){
        puts("Desbordamiento de cola\n");
        exit (1);
    }
    /*avance circular al siguiente del final*/

    cola -> final = siguiente(cola -> final);
    cola -> listaCola[cola -> final] = entrada;
}
```

Frente = 1
Final = 3



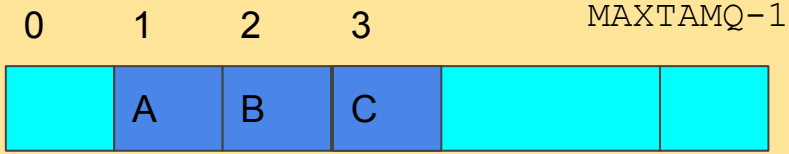
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Decola*/
TipoDato quitarCola(Cola * cola){
    TipoDato tem;
    if(colaVacia(*cola)){
        puts("Cola Vacía\n");
        exit (1);
    }
    tem = cola -> listaCola[cola -> frente];

    /*avance circularmente final*/
    cola -> frente = siguiente(cola -> frente);
    return tem;
}
```

Frente = 1
Final = 3



tem

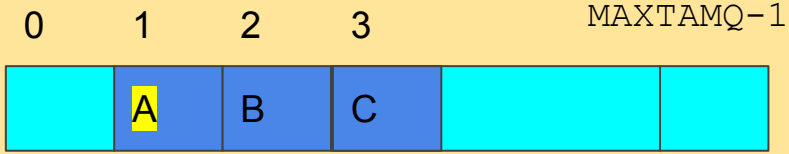
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Decola*/
TipoDato quitarCola(Cola * cola){
    TipoDato tem;
    if(colaVacia(*cola)){
        puts("Cola Vacía\n");
        exit (1);
    }
    tem = cola -> listaCola[cola -> frente];

    /*avance circularmente final*/
    cola -> frente = siguiente(cola -> frente);
    return tem;
}
```

Frente = 1
Final = 3



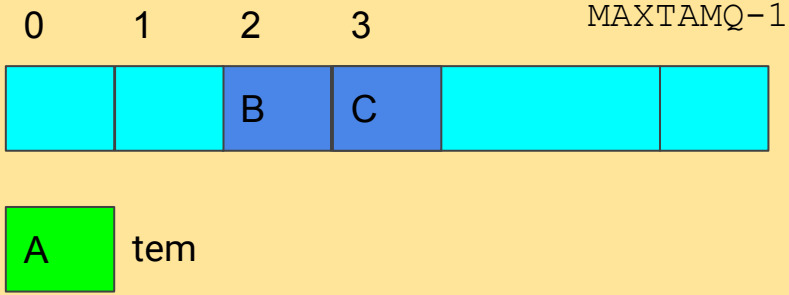
Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

/*archivo colaarray.c*/

```
/*Decola*/
TipoDato quitarCola(Cola * cola){
    TipoDato tem;
    if(colaVacia(*cola)){
        puts("Cola Vacía\n");
        exit (1);
    }
    tem = cola -> listaCola[cola -> frente];

    /*avance circularmente final*/
    cola -> frente = siguiente(cola -> frente);
    return tem;
}
```

Frente = 2
Final = 3



Funciones

- crearCola
- siguiente
- colaVacia
- colaLlena
- insertarCola
- quitarCola
- frenteCola

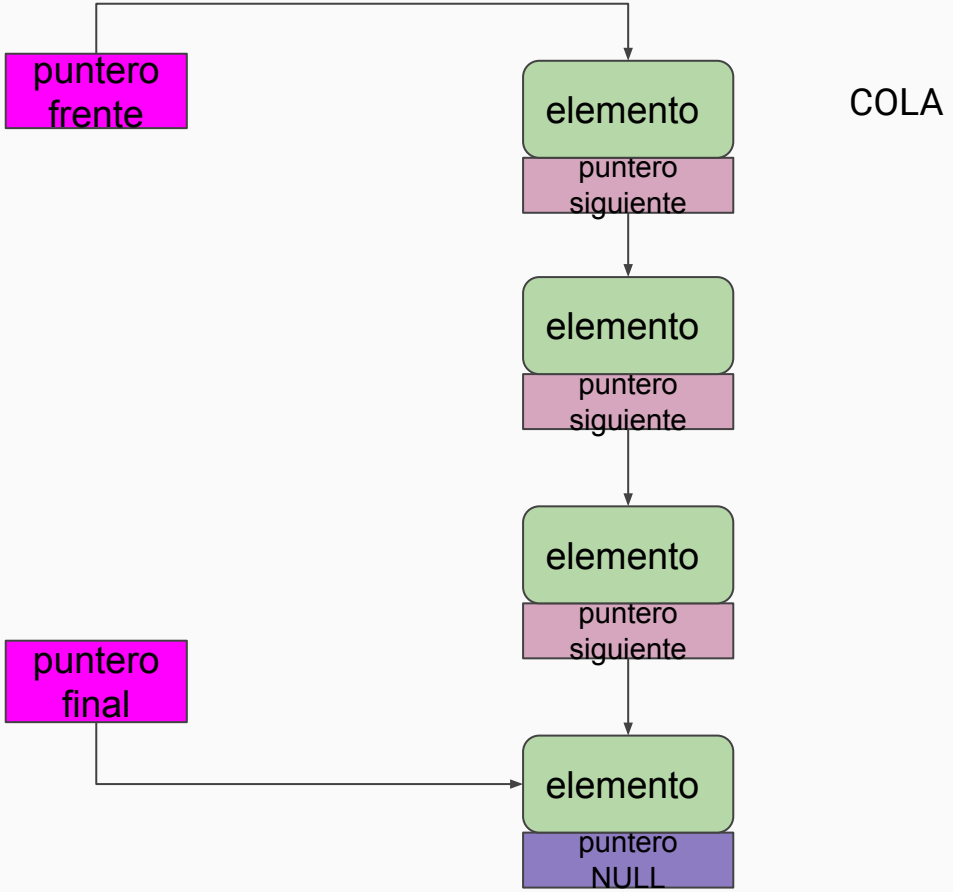
/*archivo colaarray.c*/

```
/*Retorna elemento de cola sin decolar*/
TipoDato frenteCola(Cola cola){
    if(colaVacia(cola)){
        puts("Cola Vacía\n");
        exit (1);
    }

    return cola.listaCola[cola.frente];
}
```

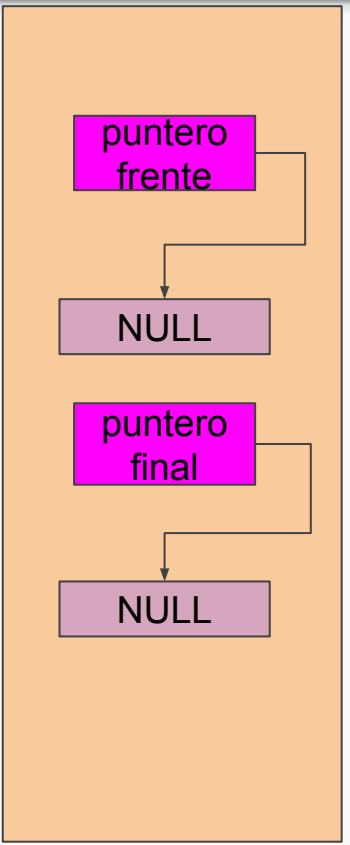
Cola mediante lista enlazada

Componentes

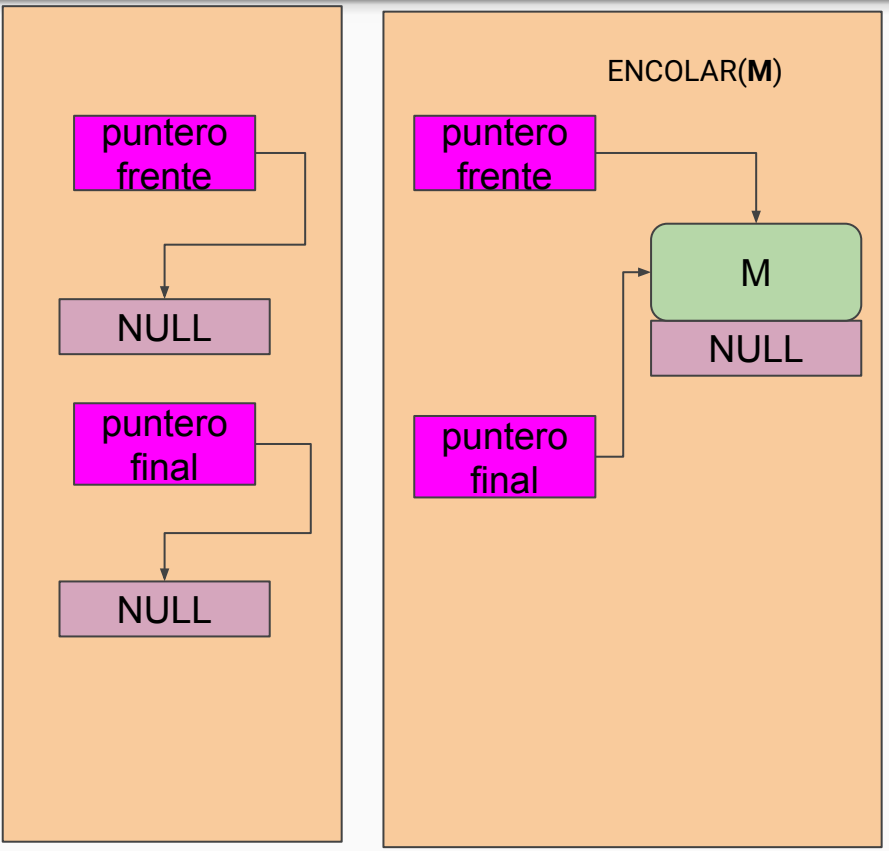
1. Lista simplemente enlazada
2. Dos punteros:
   a. frente (decolar)
   b. final (encolar)



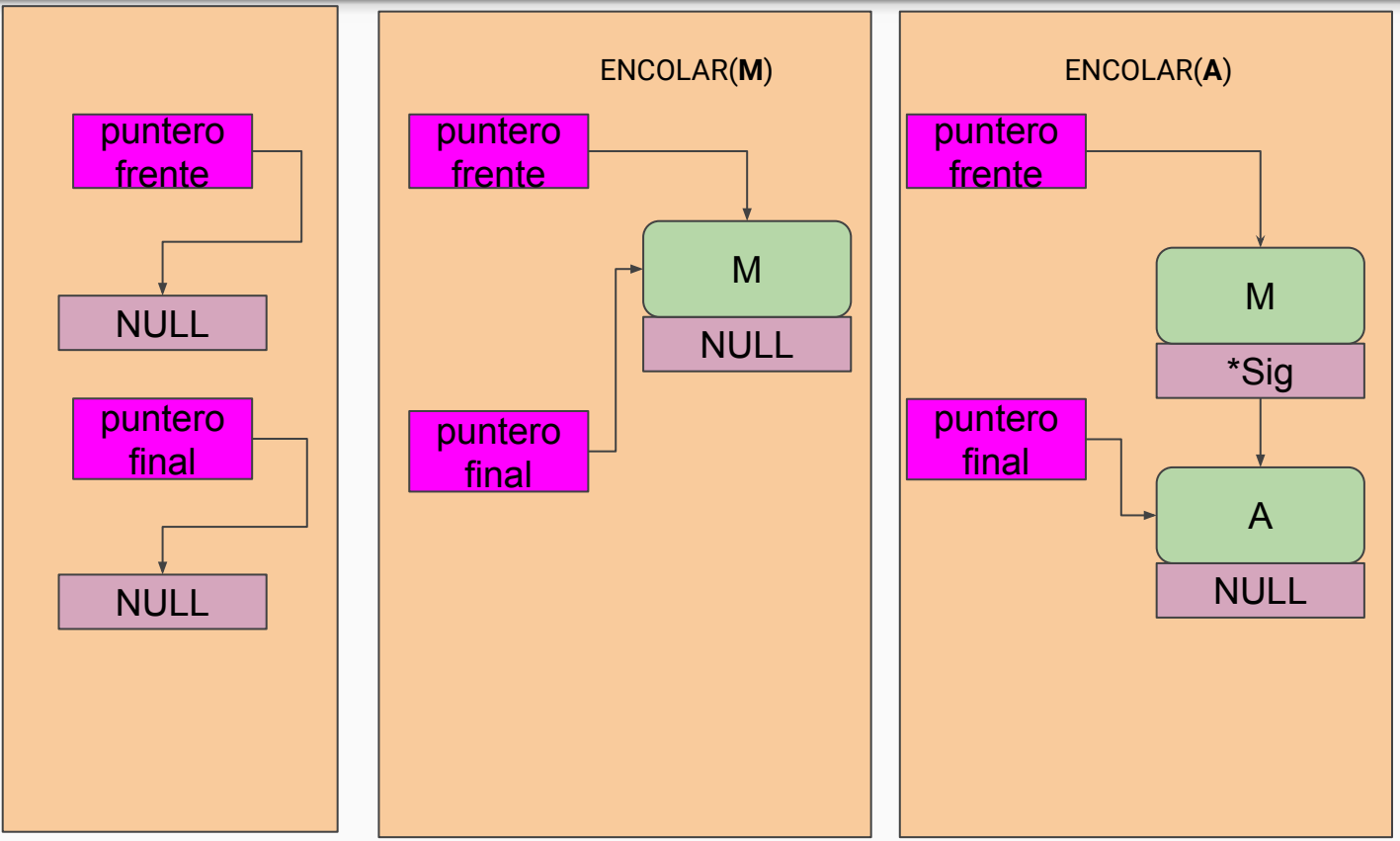
Ejemplo básico de funcionamiento **ENCOLAR**



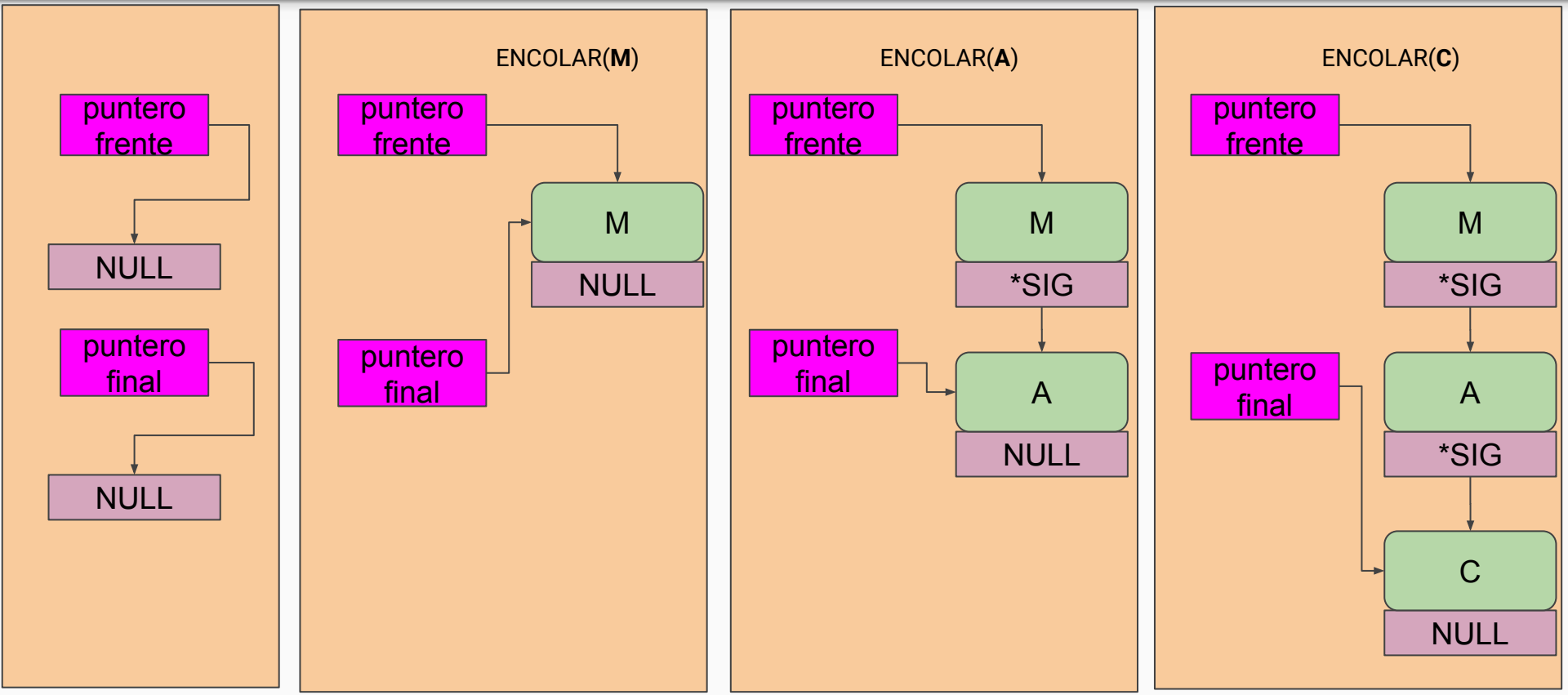
Ejemplo básico de funcionamiento **ENCOLAR**



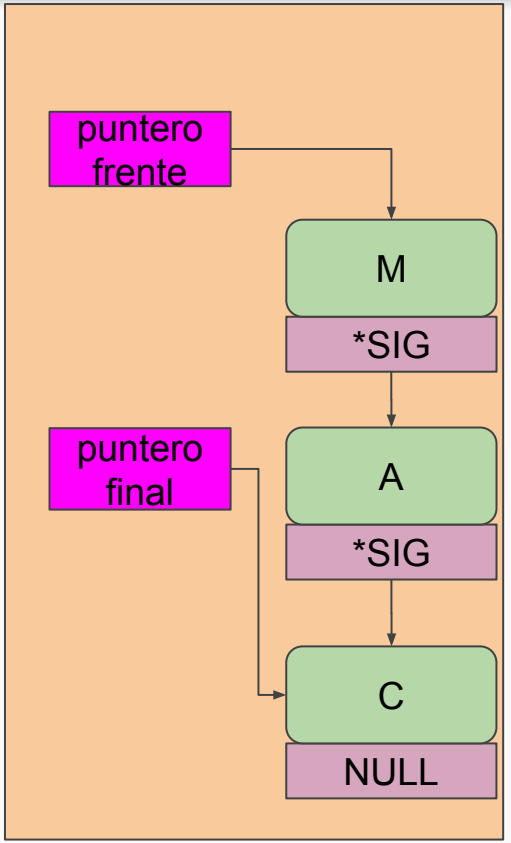
Ejemplo básico de funcionamiento **ENCOLAR**



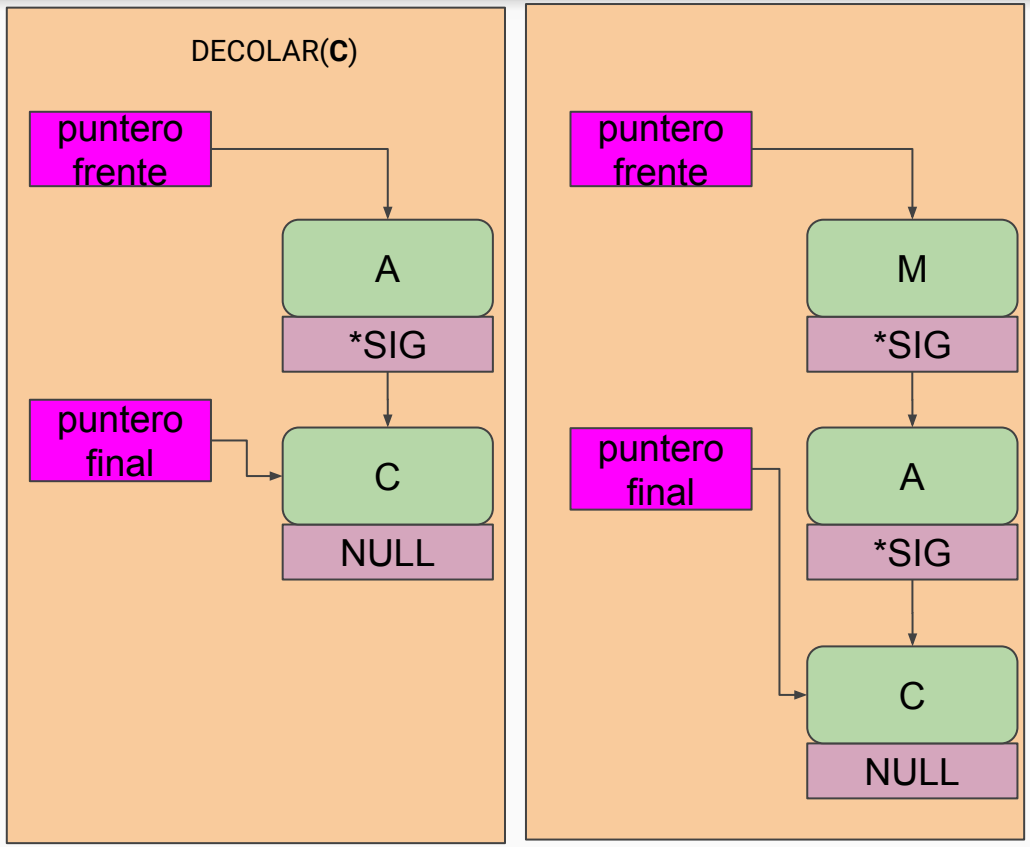
Ejemplo básico de funcionamiento **ENCOLAR**



Ejemplo básico de funcionamiento `DECOLAR`

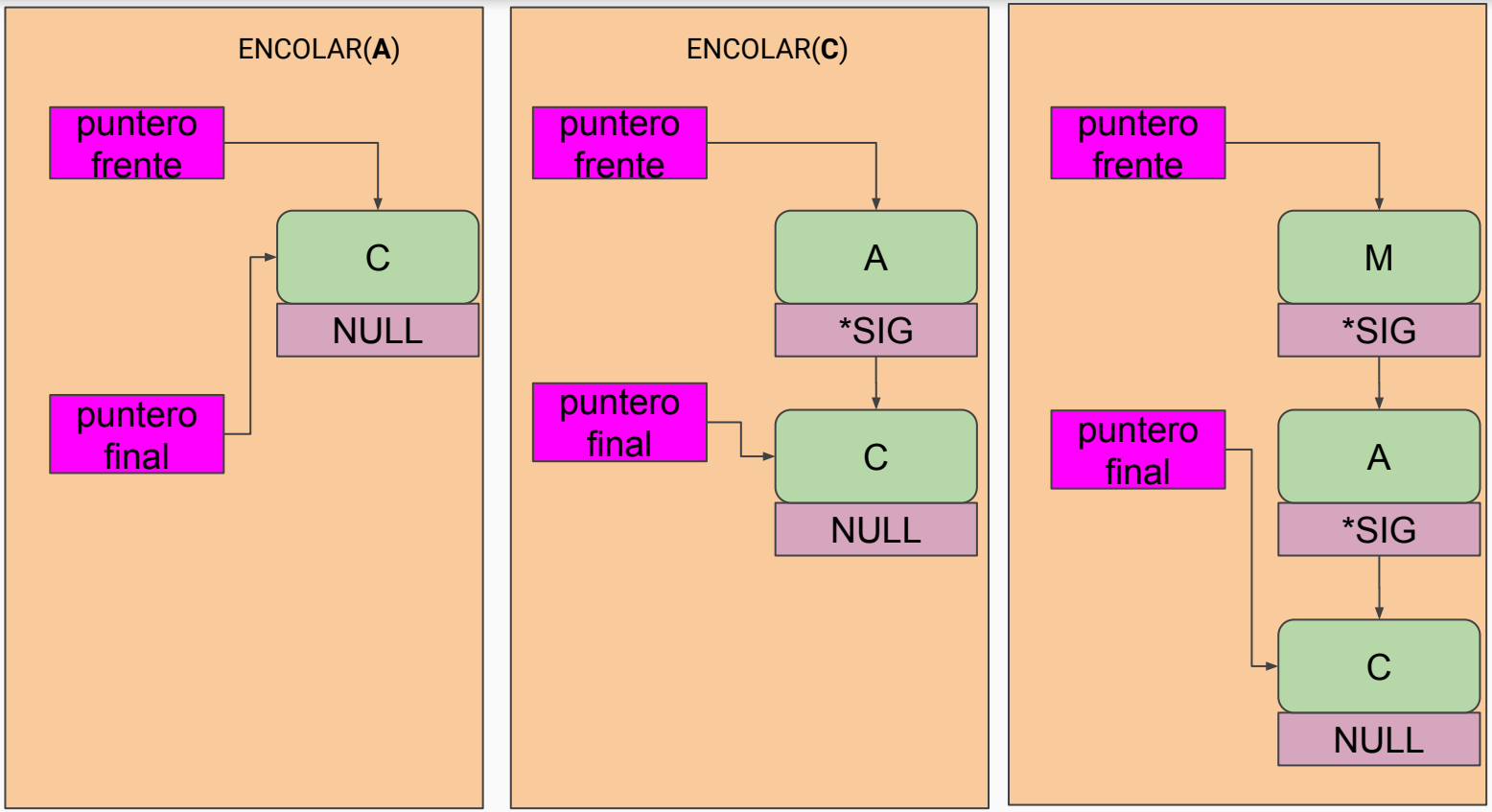


Ejemplo básico de funcionamiento **DECOLAR**



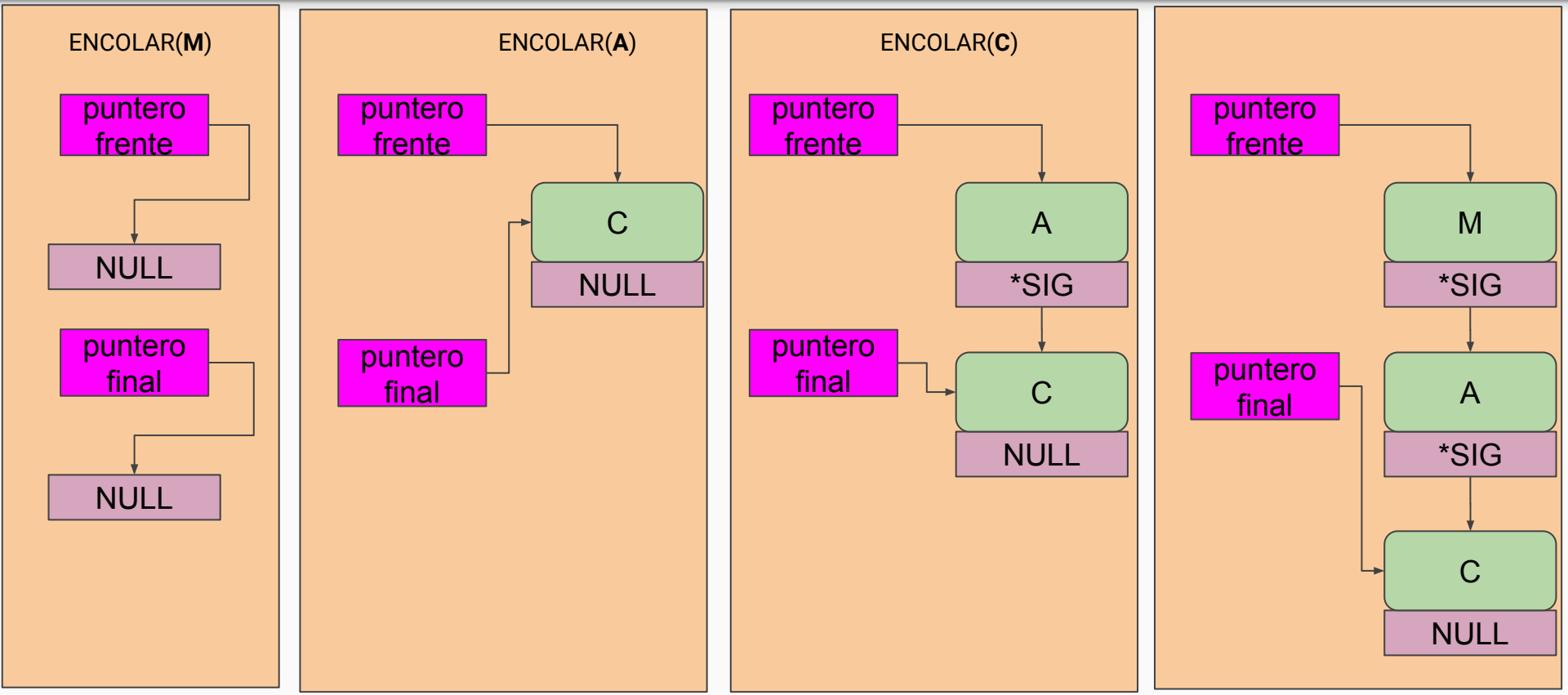
Salida MAC

Ejemplo básico de funcionamiento **DECOLAR**



Salida MAC

Ejemplo básico de funcionamiento **DECOLAR**



Salida MAC

Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

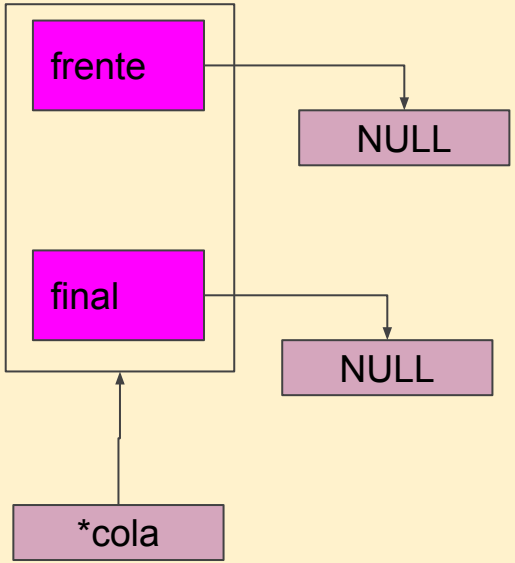
```
/*archivo coladinamica.h*/

struct nodo{
    TipoDato elemento;
    struct nodo * siguiente;
}
typedef struct nodo Nodo;

struct cola{
    Nodo * frente;
    Nodo * final;
}
typedef struct cola Cola;

/*Operaciones sobre la Pila*/
void crearCola(Cola * cola);
void insertarCola(Cola * cola, TipoDato elemento);
TipoDato quitarCola(Cola * cola);
void borrarCola(Cola * cola);

int frenteCola (Cola cola);
int colaVacia(Cola pila);
```

Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

```
/*archivo coladinamica.c*/
```

```
/*Crea una Cola nueva*/
void crearCola(Cola * cola){
     cola -> frente = cola -> final = NULL;
}
```

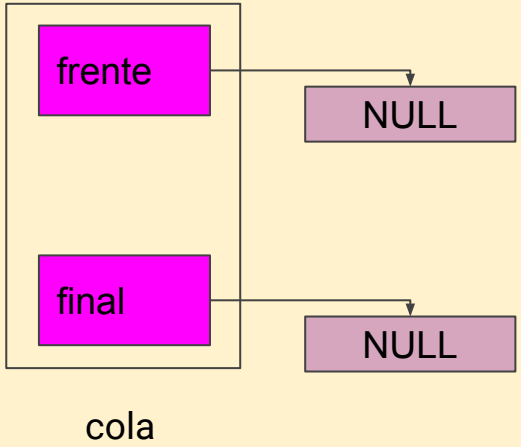


Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

```
/*archivo coladinamica.c*/
```

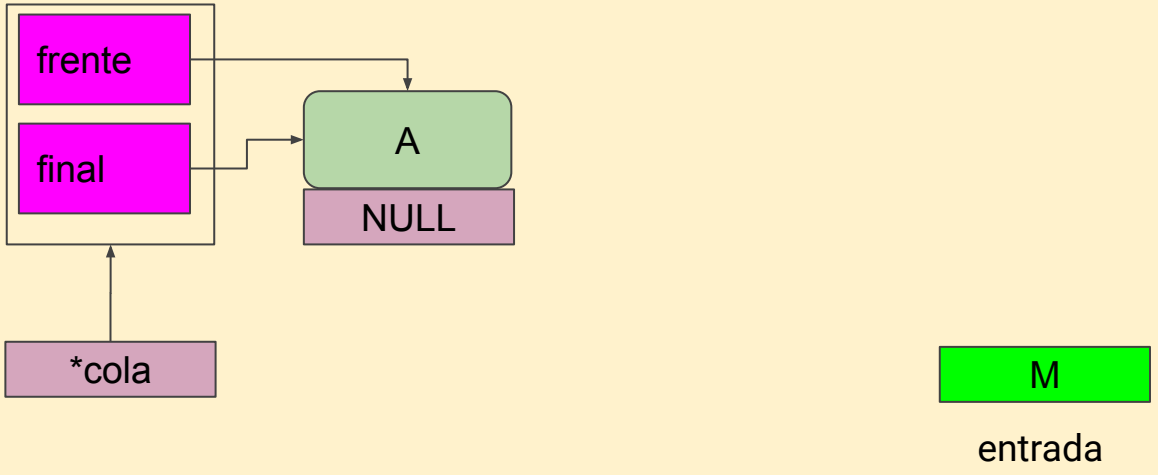
```
/*Verifica si la cola está vacía*/
int colaVacía(Cola cola){
    return (cola.frente == NULL);
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

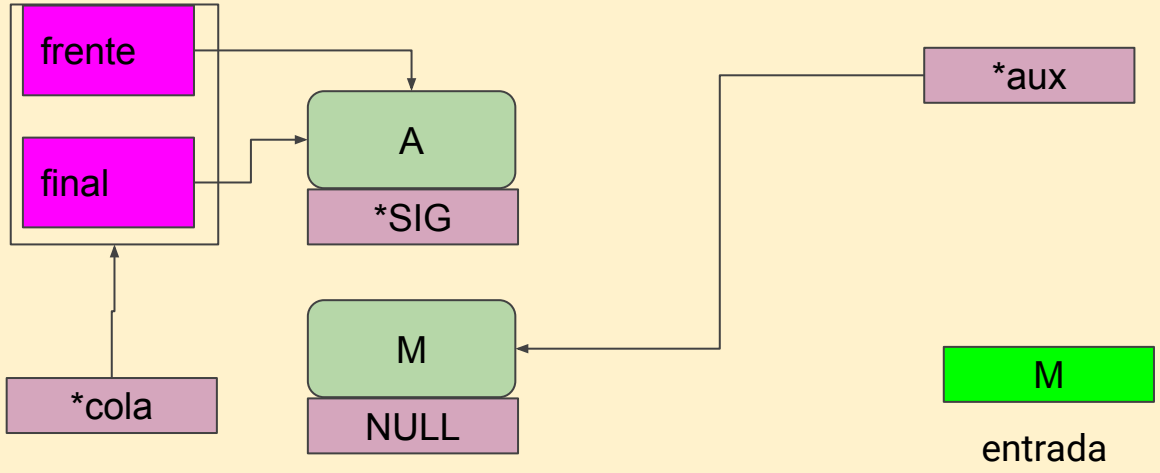
```
/*archivo coladinamica.c*/
/*Insertar en la Cola*/
void insertarCola(Cola * cola, TipoDato entrada){
     Nodo * aux;
     aux = crearNodo (entrada);
     if (colaVacia(*cola)){
          cola -> frente = aux;
     }else{
          cola -> final -> siguiente = aux;
     }
     cola -> final = aux;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

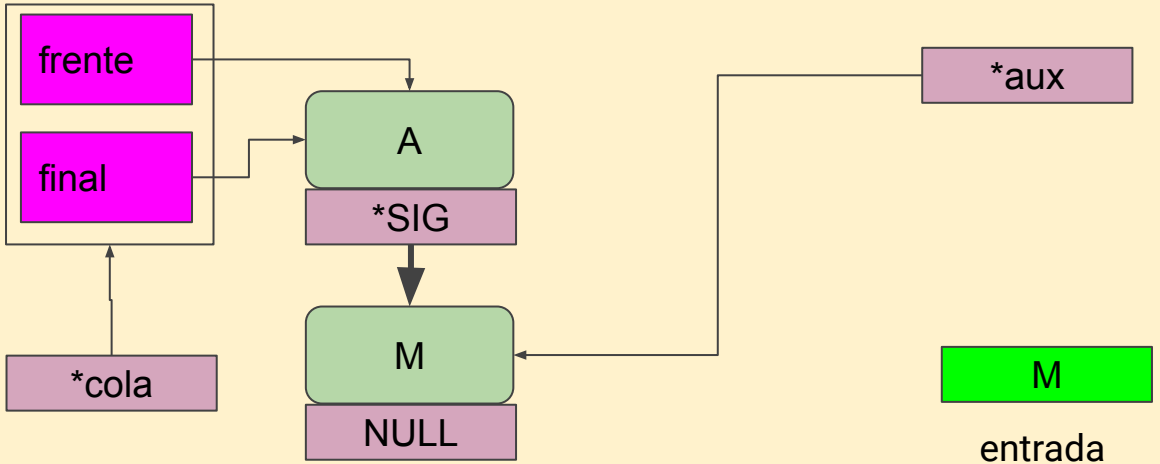
```
/*archivo coladinamica.c*/
/*Insertar en la Cola*/
void insertarCola(Cola * cola, TipoDato entrada){
     Nodo * aux;
     aux = crearNodo (entrada);
     if (colaVacia(*cola)){
          cola -> frente = aux;
     }else{
          cola -> final -> siguiente = aux;
     }
     cola -> final = aux;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

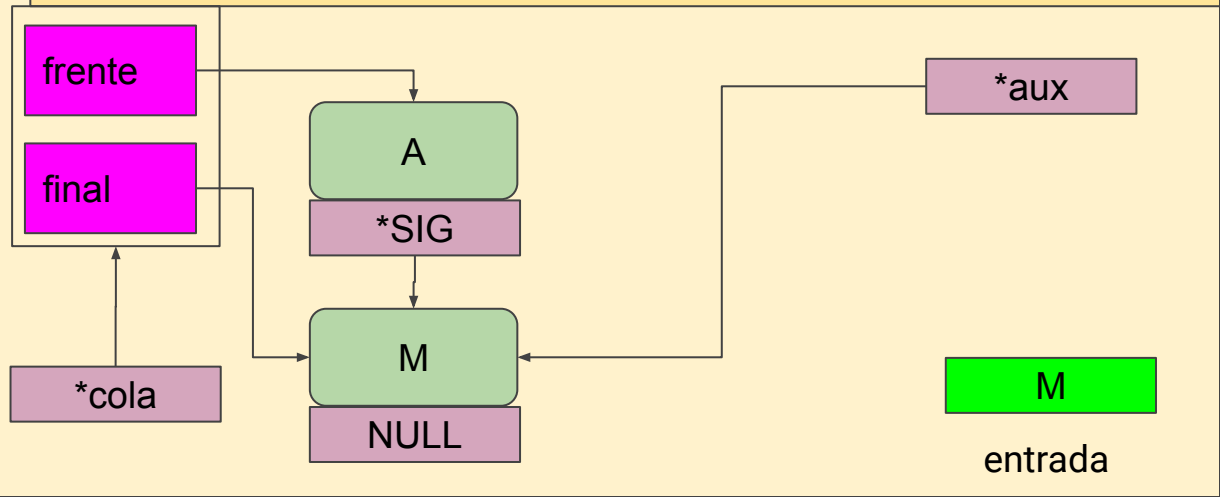
```
/*archivo coladinamica.c*/
/*Insertar en la Cola*/
void insertarCola(Cola * cola, TipoDato entrada){
     Nodo * aux;
     aux = crearNodo (entrada);
     if (colaVacia(*cola)){
          cola -> frente = aux;
     }else{
          cola -> final -> siguiente = aux;
     }
     cola -> final = aux;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

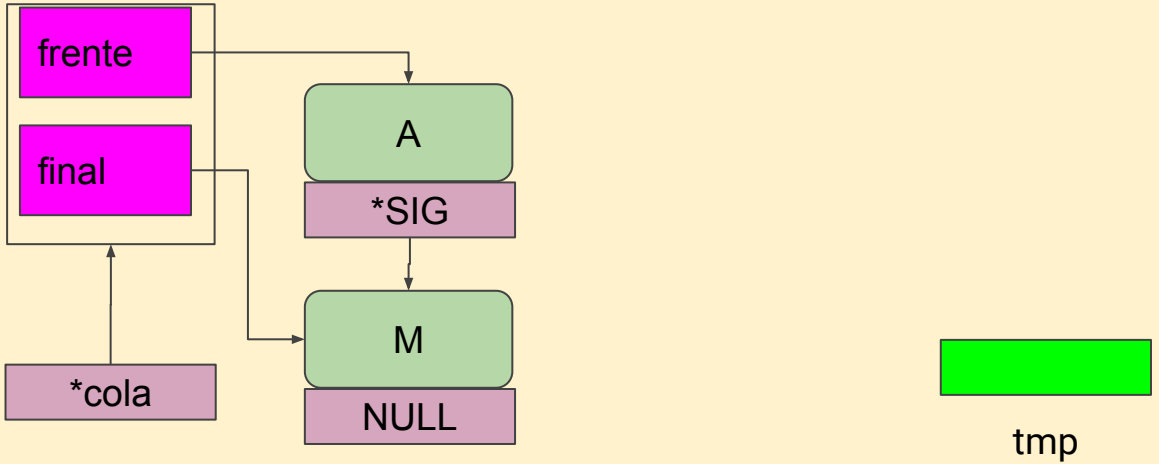
```
/*archivo coladinamica.c*/
/*Insertar en la Cola*/
void insertarCola(Cola * cola, TipoDato entrada){
     Nodo * aux;
     aux = crearNodo (entrada);
     if (colaVacia(*cola)){
          cola -> frente = aux;
     }else{
          cola -> final -> siguiente = aux;
     }
     cola -> final = aux;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

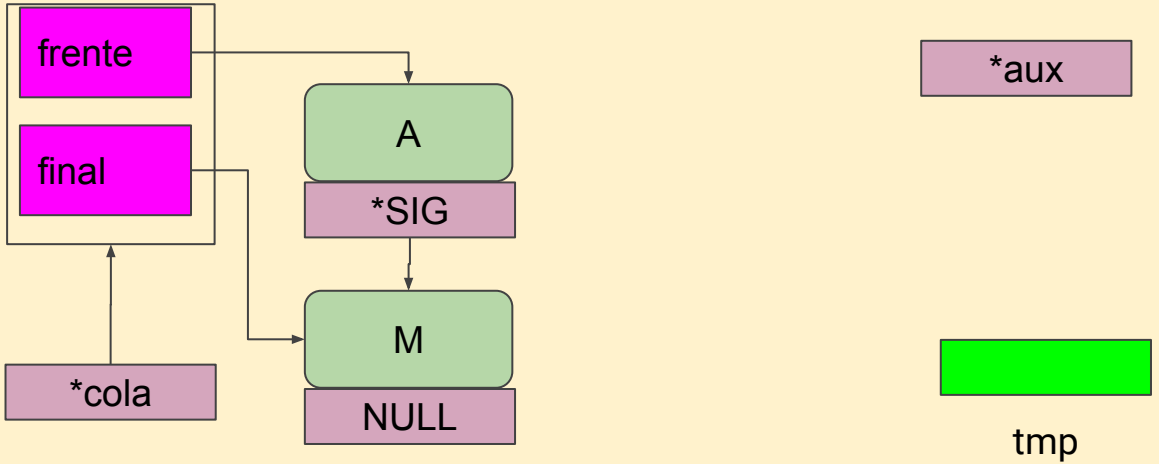
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

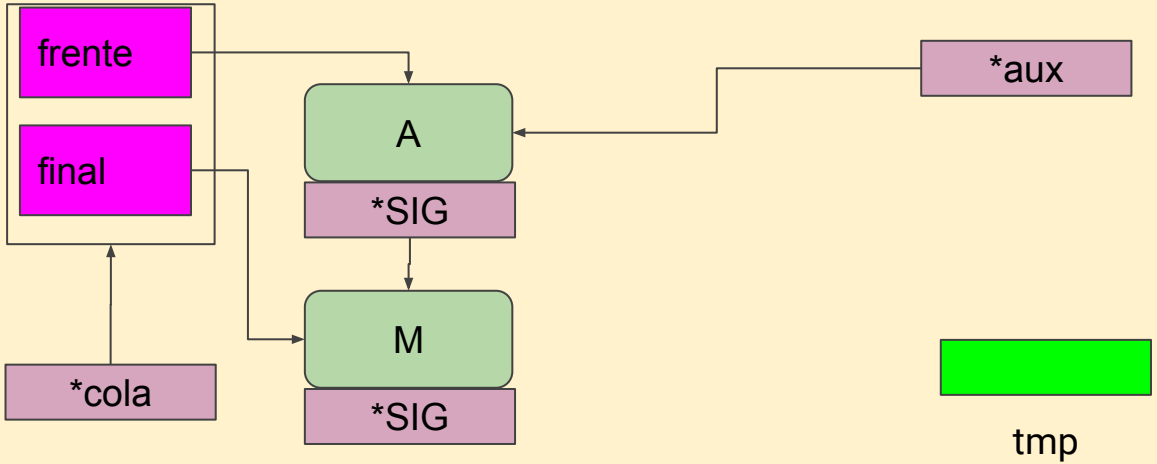
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

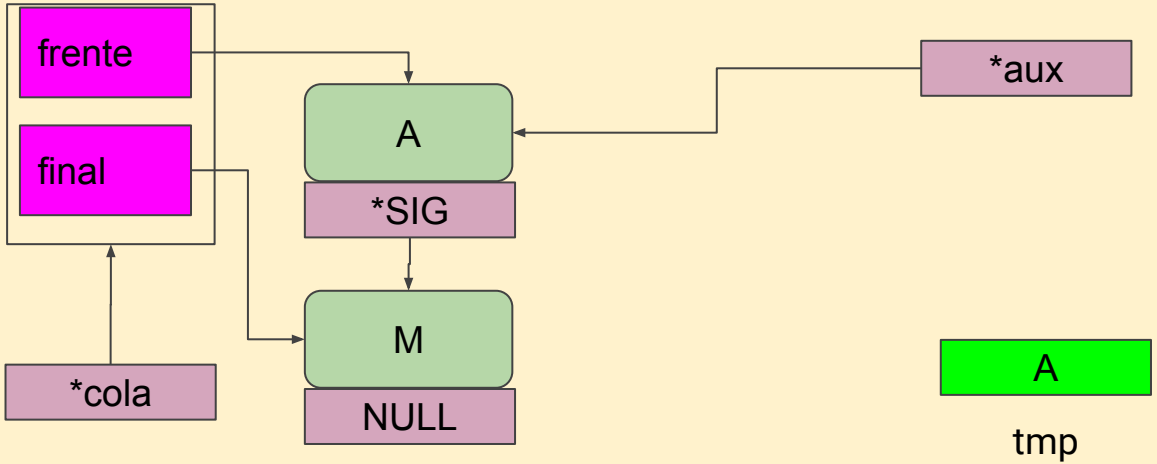
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

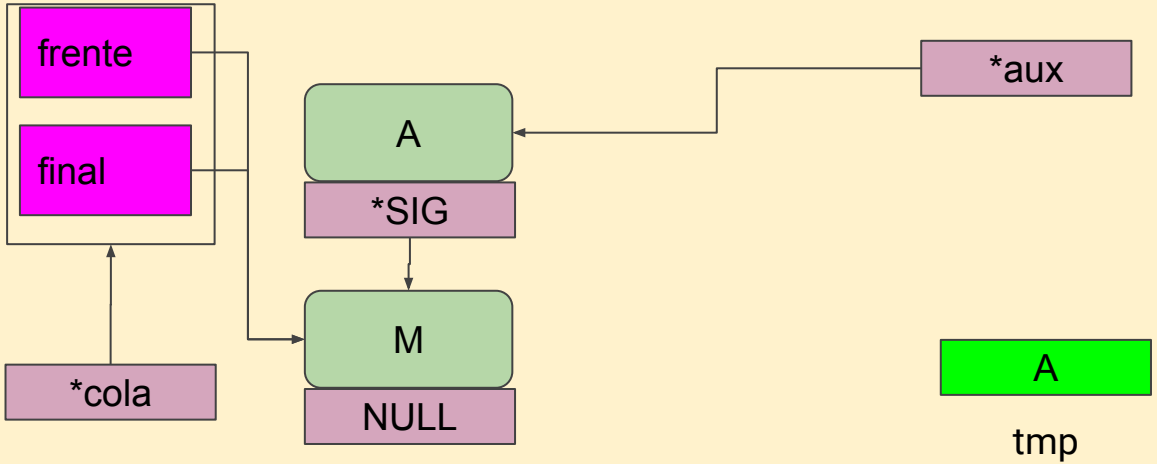
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

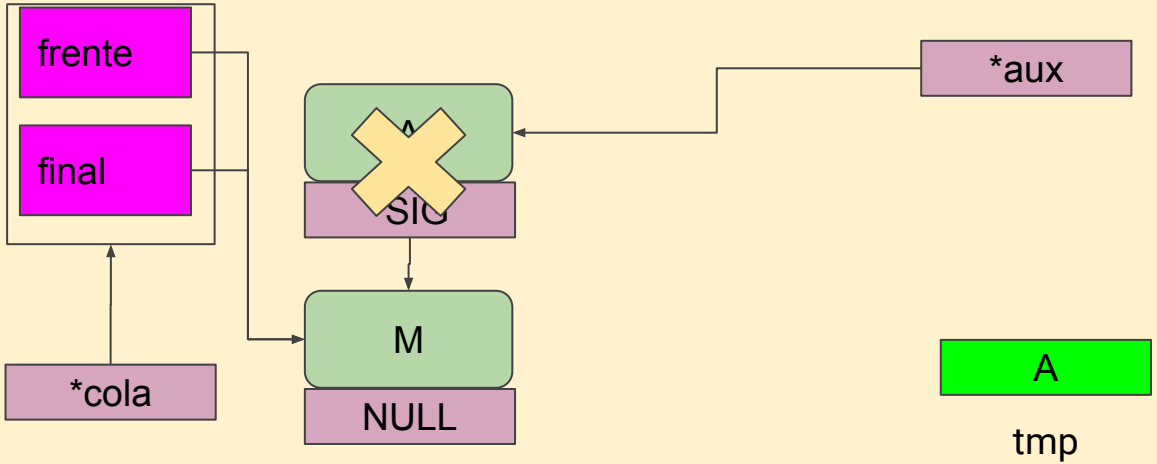
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente ;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

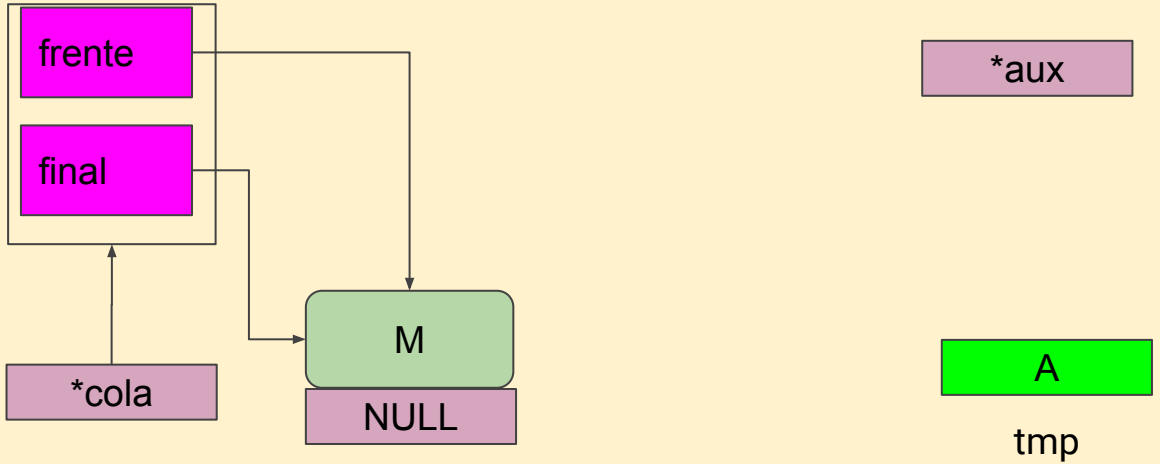
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

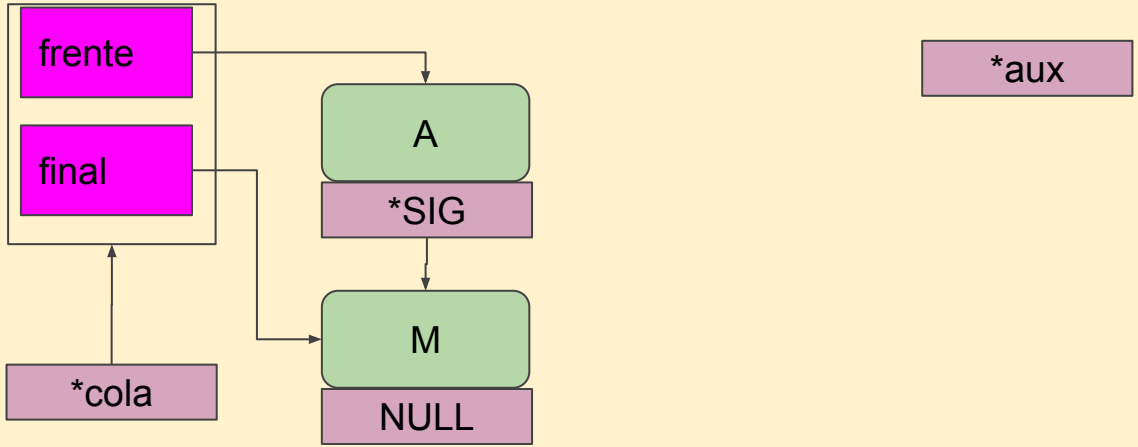
```
/*archivo coladinamica.c*/
/*Quitar de la Cola*/
TipoDato quitarCola(Cola * cola){
      TipoDato tmp;
      if (!colaVacia(*cola)){
            Nodo * aux;
            aux = cola->frente;
            tmp = cola->frente->elemento;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }else{
            puts("Cola Vacía\n");
            exit(1);
      }
      return tmp;
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

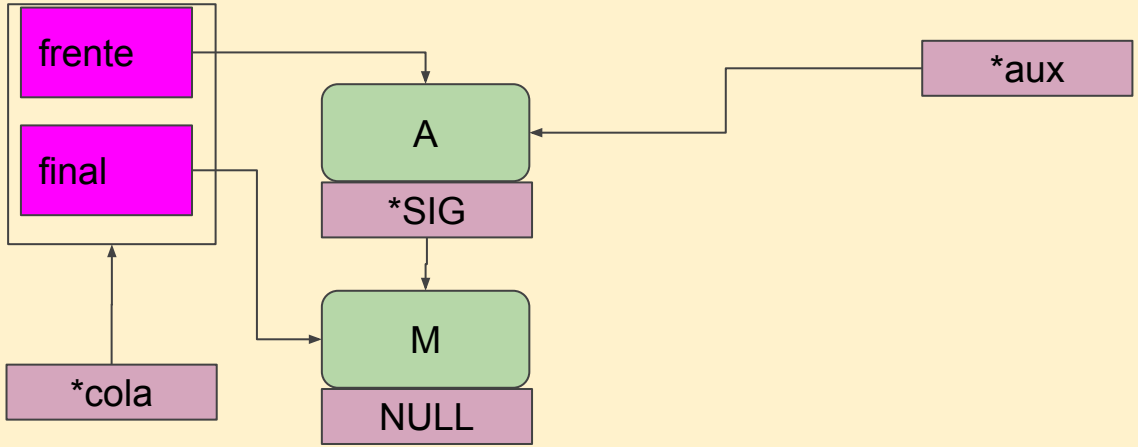
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

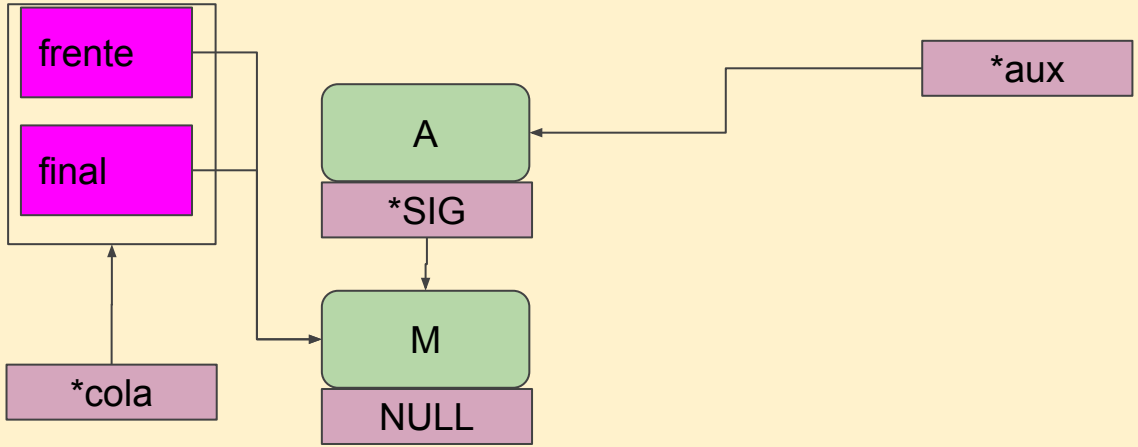
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

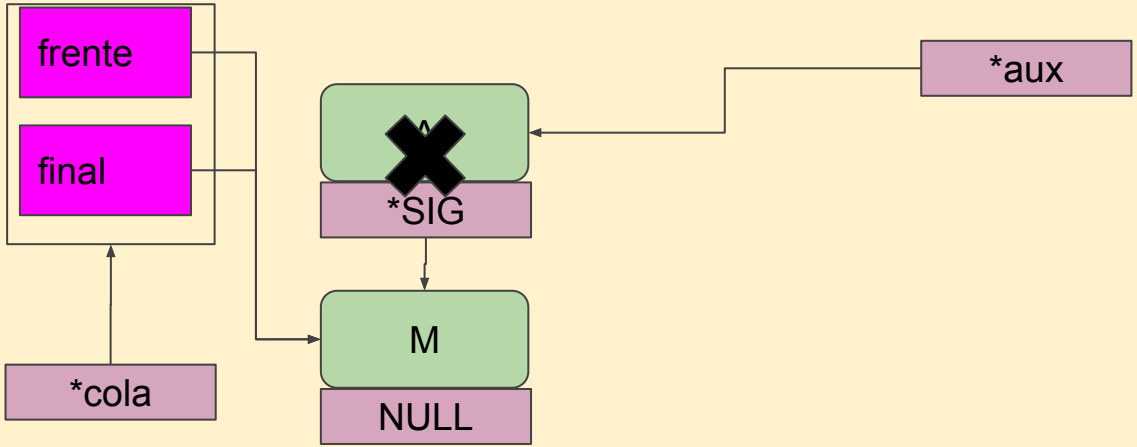
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

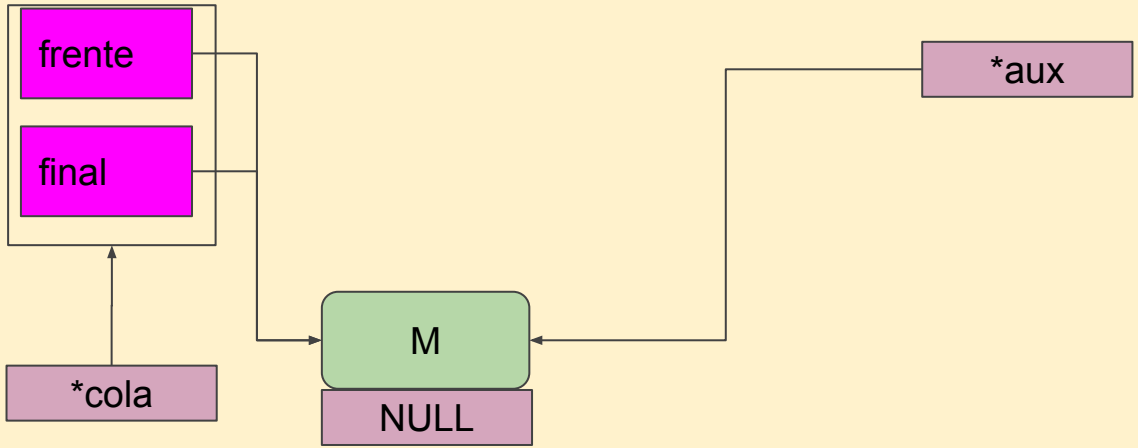
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

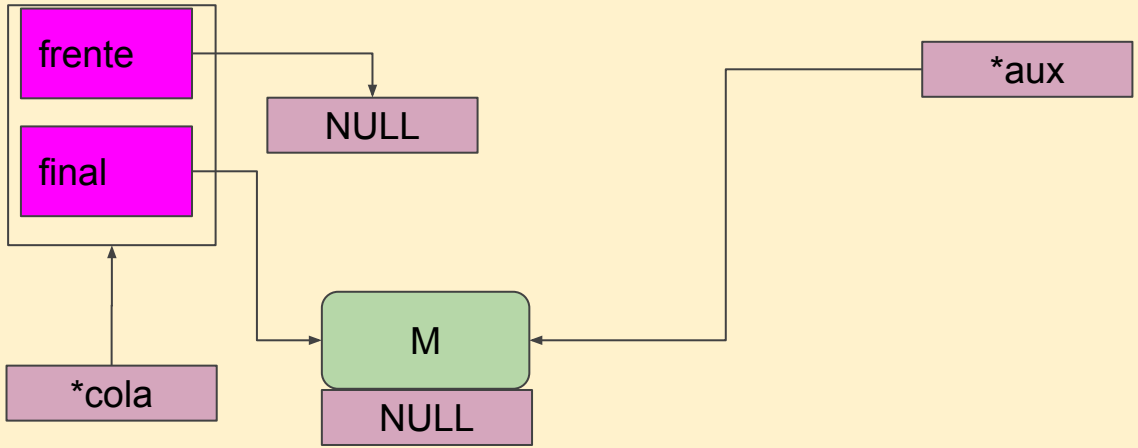
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

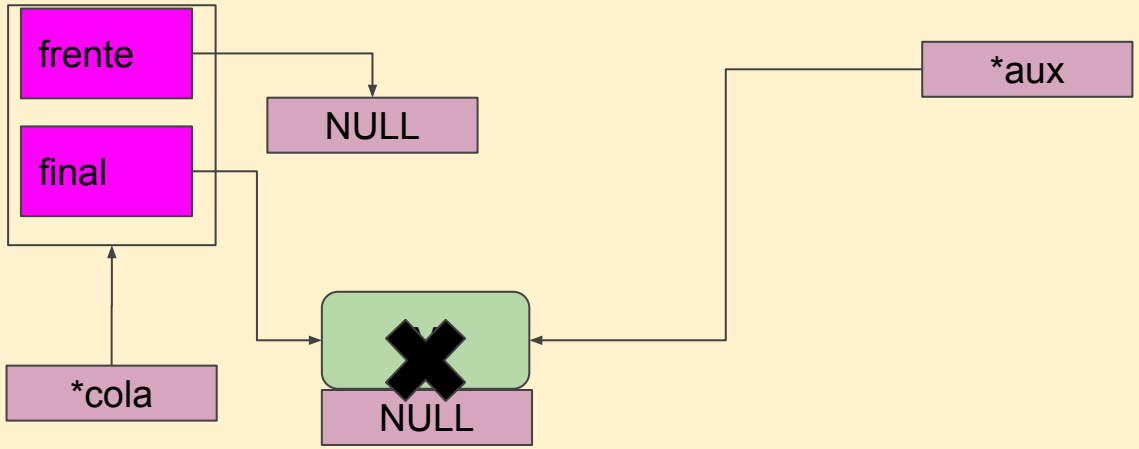
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

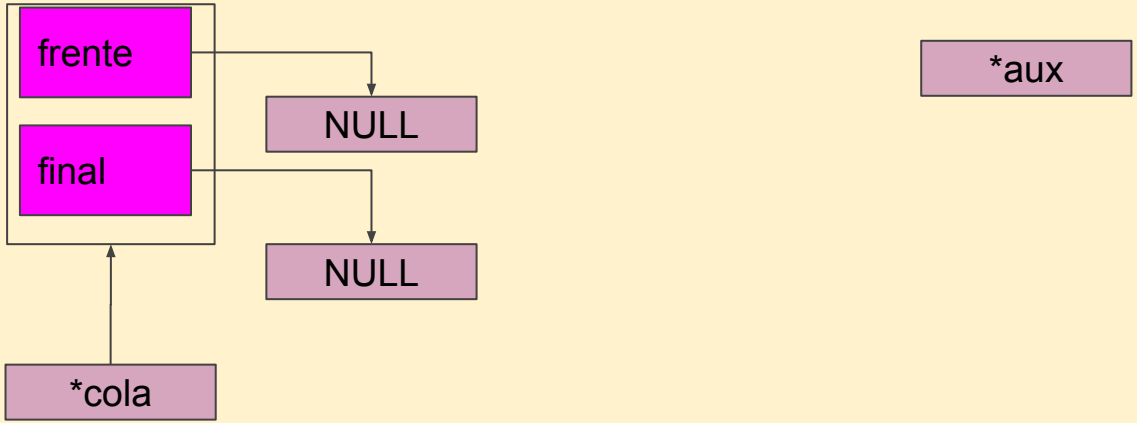
```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Funciones

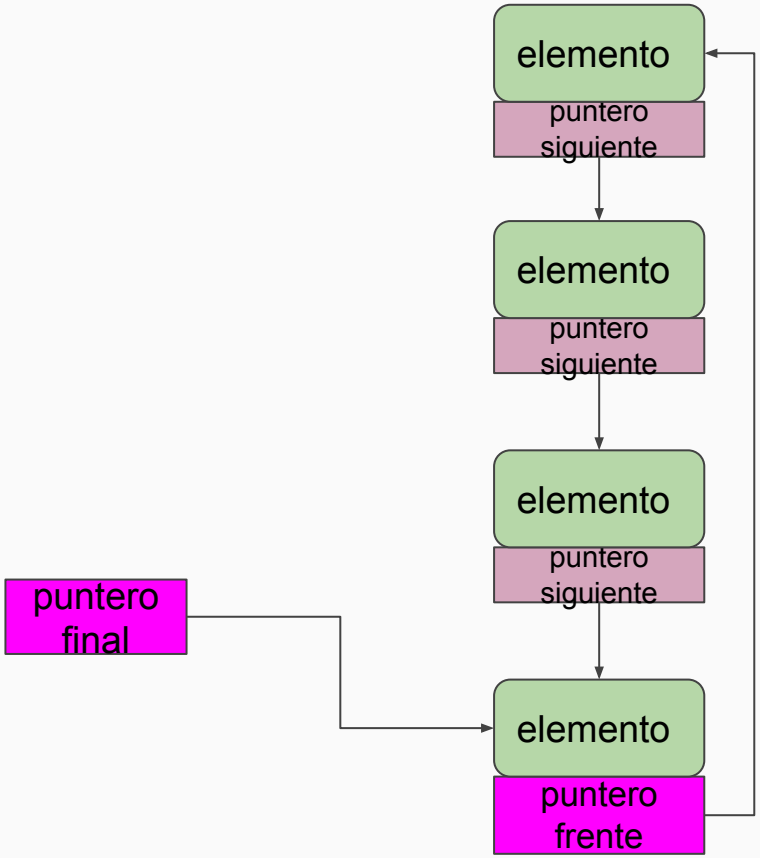
1. crearCola
2. insertarCola
3. quitarCola
4. borrarCola
5. frenteCola
6. colaVacia

```
/*archivo coladinamica.c*/
/*Elimina toda la Cola*/
void borrarCola(Cola * cola){

      for( ; cola -> frente != NULL; ){
            Nodo * aux;
            aux = cola->frente;
            cola->frente = cola->frente->siguiente;
            free(aux);
      }
}
```



Otras variantes de Colas

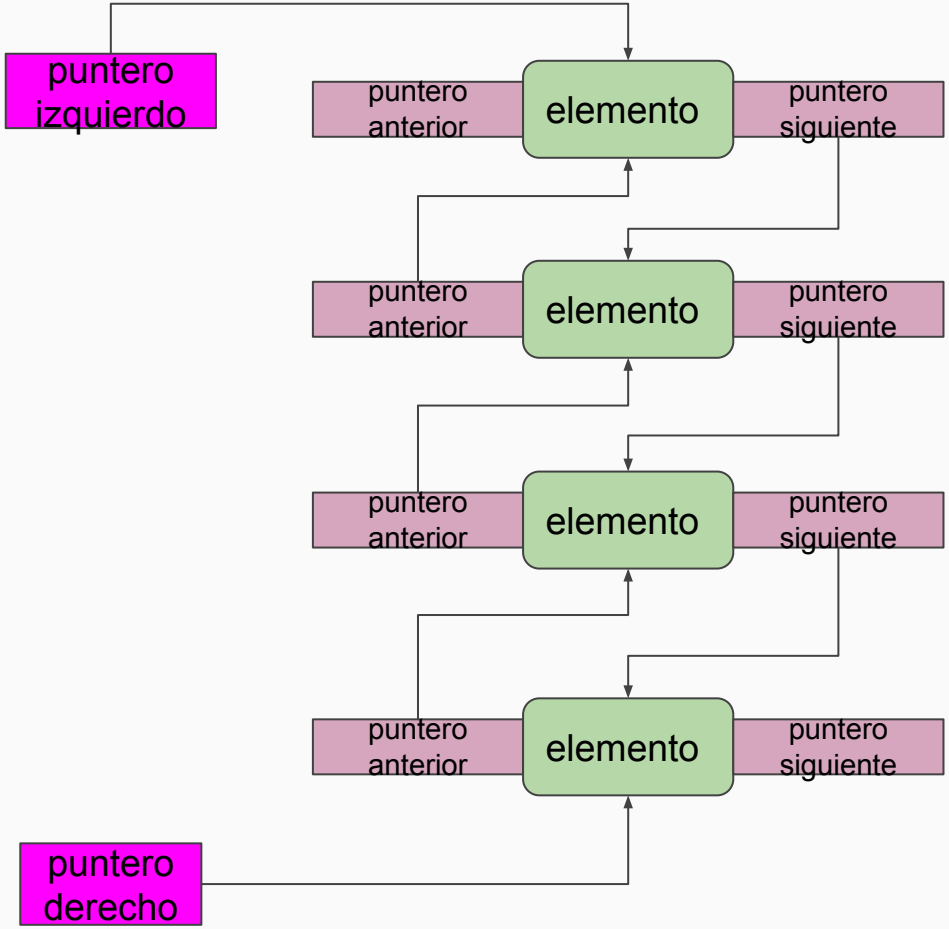
Colas con listas enlazadas circulares simples



Otras variantes de Colas

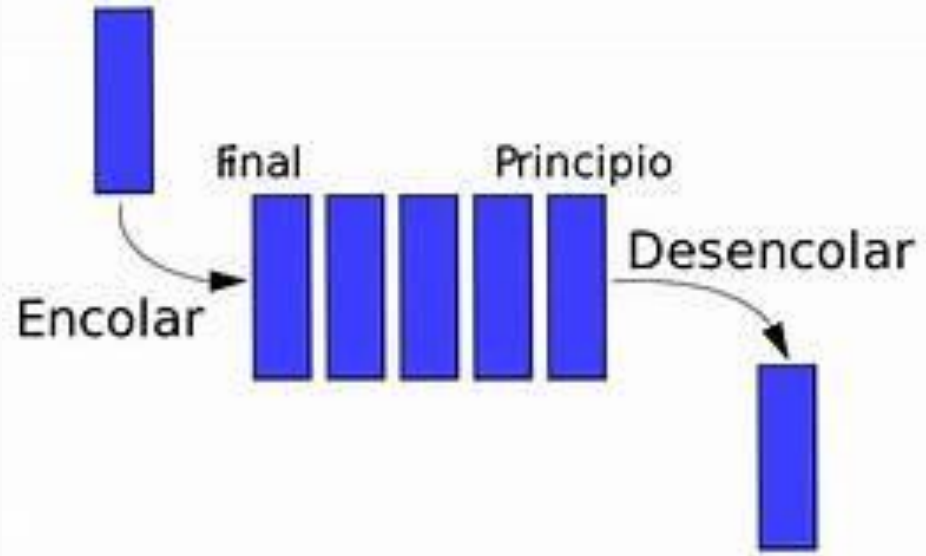
BICOLAS

Se puede encolar y decolar por ambos lados



Una Cola es una estructura de dados FIFO.

¡Gracias!



https://tse2.mm.bing.net/th?id=OIP.72n0q41vjLDc1AXUY1y8BAHaEd&pid=Api&P=0&w=262&h=158