S5: Colas (queues)

Estructuras de Datos

Docente: Pamela Landero Sepúlveda
pamelalandero@Gmail.com

Colas

- Es una abstracción útil con muchos paralelismos en la vida real: colas de espera en bancos, supermercados, etc
  - Ello hace que su importancia sea crucial en simulaciones.
- Computacionalmente hay muchos casos en los que aparecen colas de espera, por ejemplo:
  - Procesos que esperan a ser ejecutados en sistemas multitarea
  - Procesos batch de cálculo intensivo
  - Acceso a recursos compartidos (impresora)

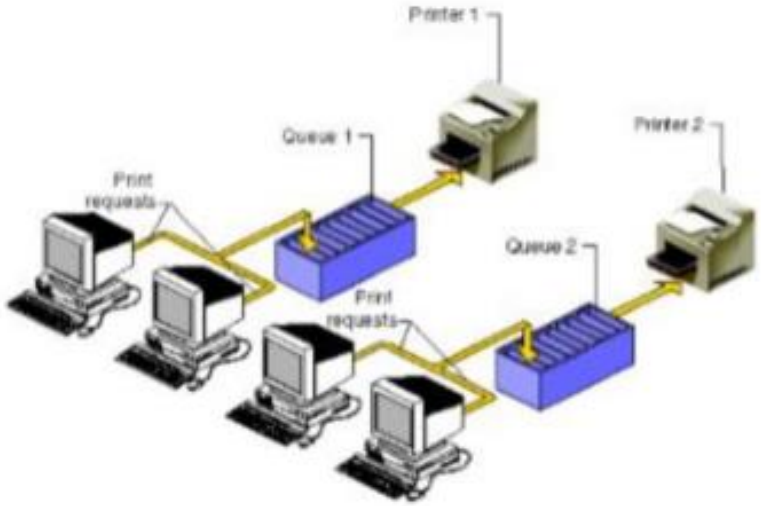
Colas

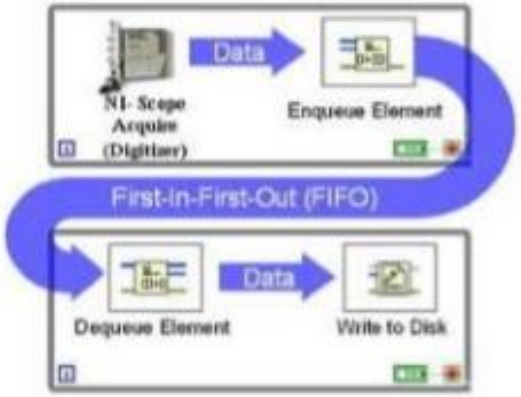
- Tipo de dato lineal con estructura **FIFO** (**F**irst **I**n, **F**irst **O**ut), indica que el primer elemento que se incorporó a la estructura, será el primero que salga, y el único que se puede consultar en un momento dado.

Colas

- Las colas son un subconjunto de las listas, en donde las eliminaciones se dan al comienzo de la lista y las inserciones al final.
- Los elementos se procesan en el orden como se reciben (similar a la cola de impresión en redes).
- Cuando se desea accesar un elemento que no está en un extremo, obligadamente se debe realizar una operación desplazamiento de los nodos anteriores (o posteriores) a él.

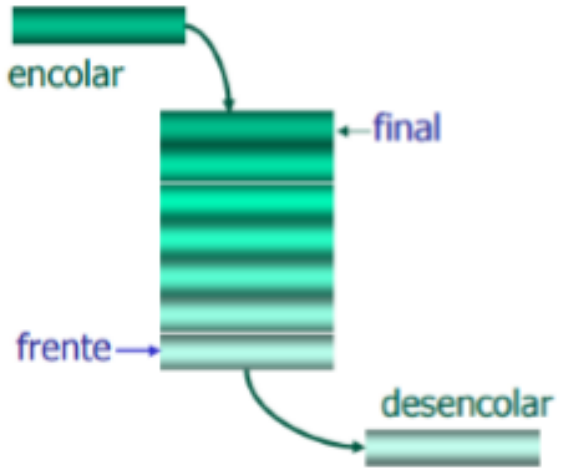
Colas - Ejemplos



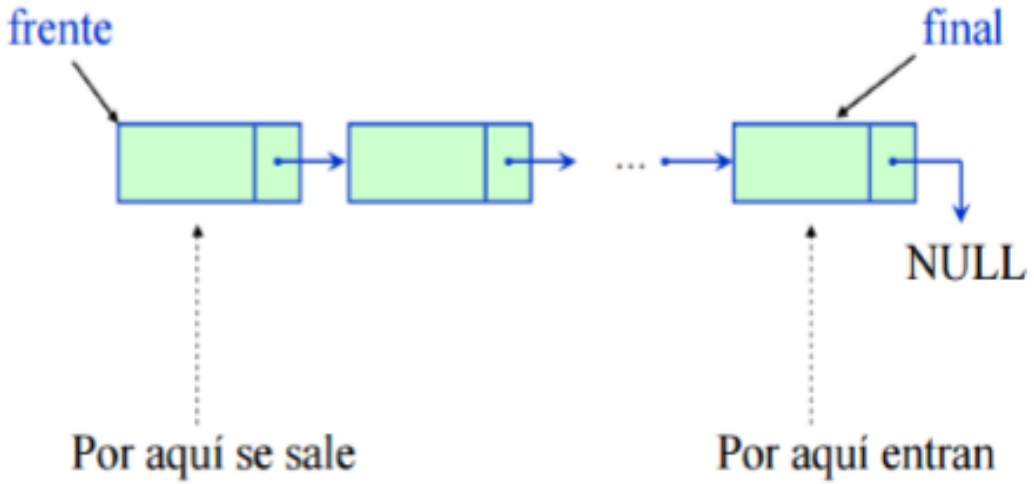


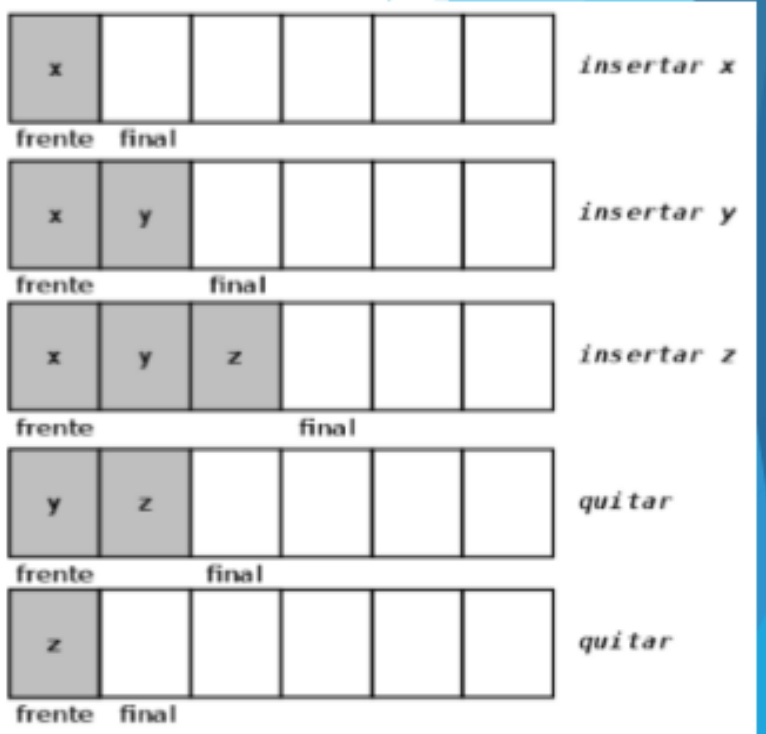
Colas - Operaciones

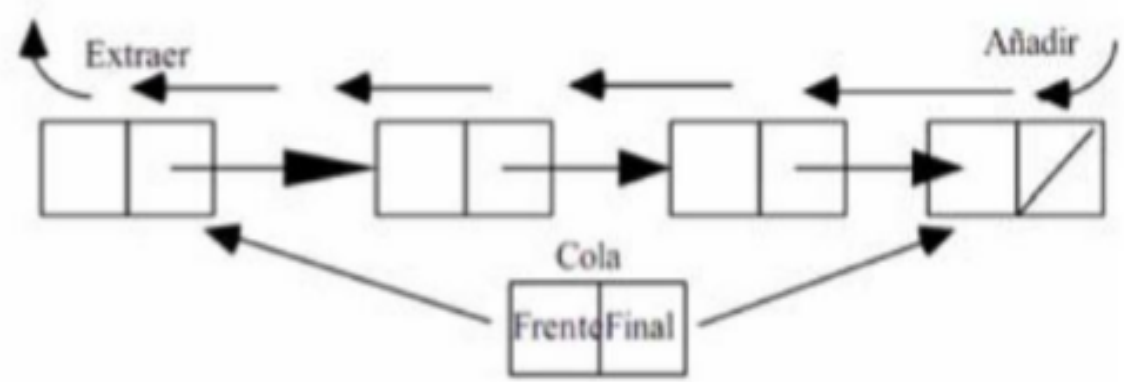
Crear	Se crea la cola vacía.
Tamaño	Regresa el numero de elementos de la cola.
Encola	Se añade un elemento a la cola.
Desencola	Se elimina un elemento de la cola.
Comienzo	Devuelve el elemento que está al comienzo de la cola.
Final	Devuelve el elemento que está al final de la cola.
Vacía	Devuelve cierto si la cola está vacía o falso en caso contrario.
Llena	Devuelve cierto si la cola está llena o falso en caso contrario.



Colas - Representación





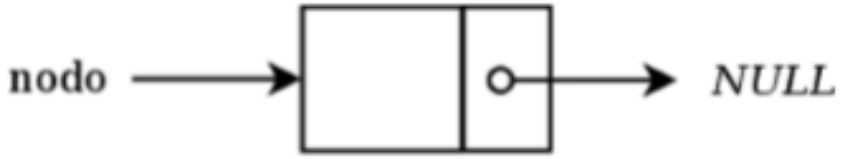


Crear cola

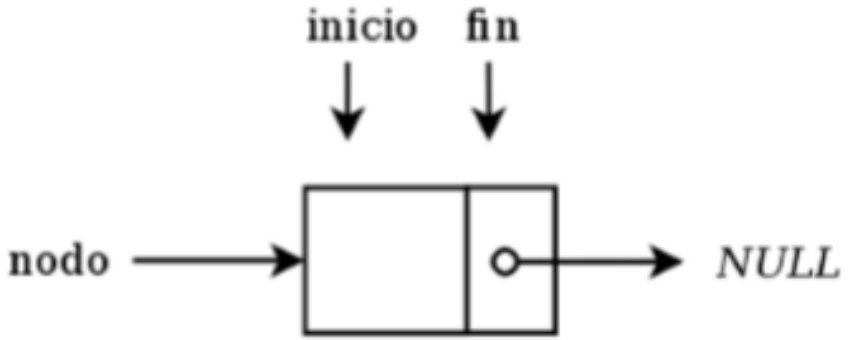
- Crear una Cola: hacer que $inicio$ y fin apunte a NULL. $inicio \longrightarrow$ NULL $fin \longrightarrow$ NULL

Insertar en cola vacía (encolar)

1. Crear un nodo y hacer que su siguiente apunte a NULL.



2. Hacer que $inicio$ y fin apunte a nodo.

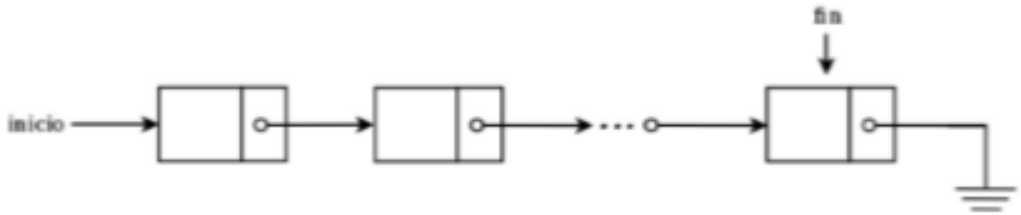


Insertar cola no vacía (encolar)

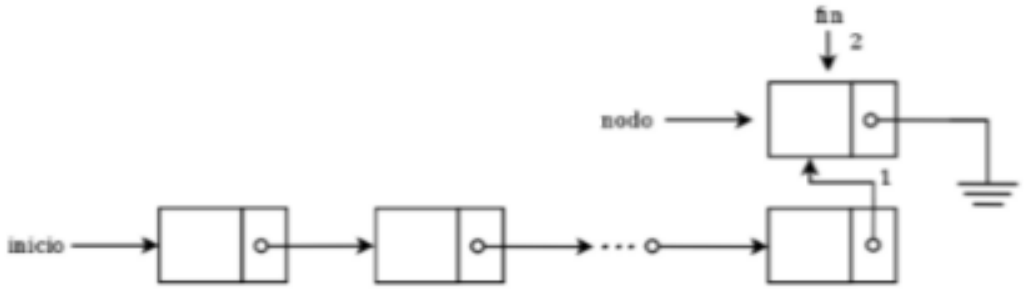
1. Crear un nodo y hacer que el siguiente del último nodo apunte al nuevo nodo.

2. Hacer que fin apunte al nuevo nodo.

   - Antes



   - Después

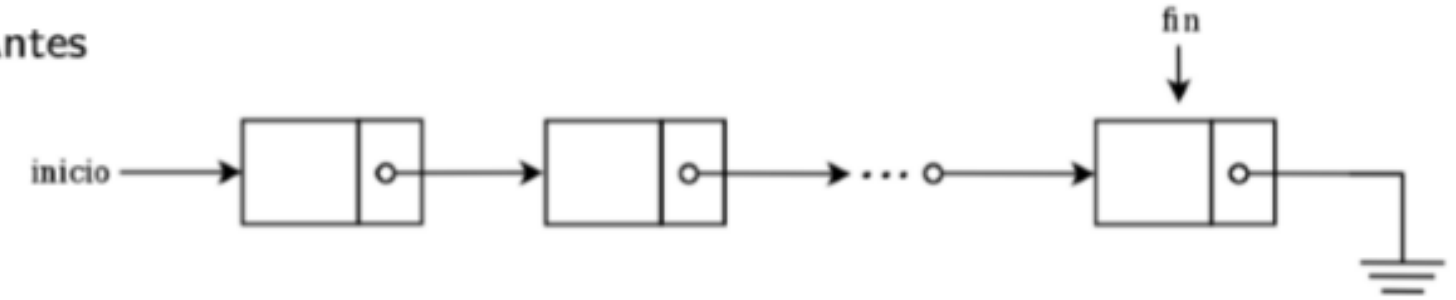


Eliminar un elemento (desencolar)

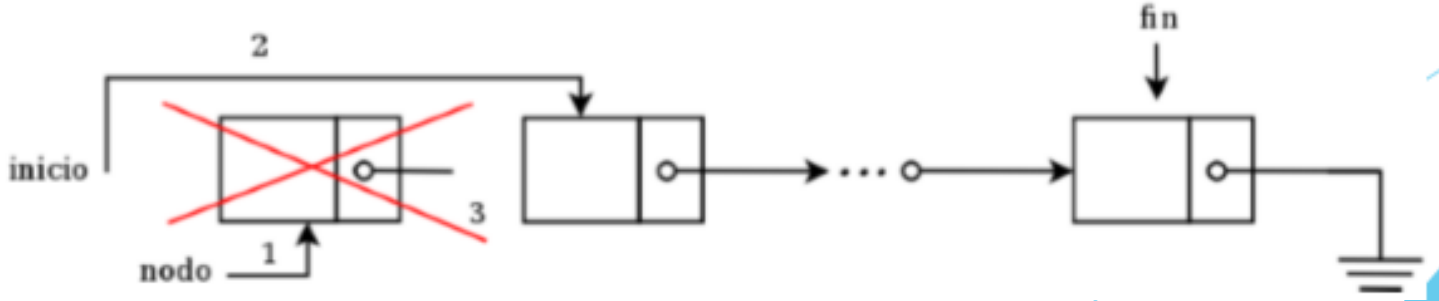
Suponiendo que se parte de una cola con uno o más nodos, considere un apuntador auxiliar nodo:

1. Hacer que aux apunte al primer elemento de la cola, es decir a *inicio*.
2. Asignar a *inicio* la dirección del segundo nodo de la cola; es decir, el de su nodo siguiente.
3. Liberar la memoria asignada a nodo, que es el que se desea eliminar.

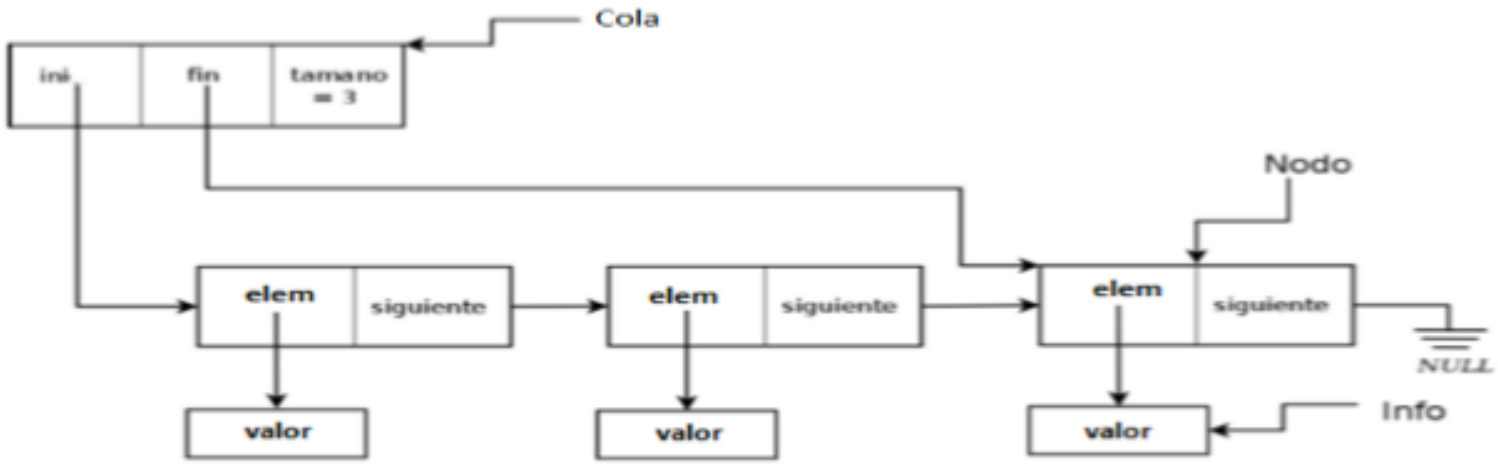
- Antes



- Después



Colas - Implementación




```c
int main(int argc, char *argv[]){
    Cola *C;
    C = crearCola();
    encolar(C,15); //push
    encolar(C,4);
    encolar(C,5);
    encolar(C,10);
    recorrerCola(C);
    printf("\n Valor desencolado: %d", desencolar(C)->valor);
    recorrerCola(C);
    printf("\n Tamaño de la cola: %d", C->tam);
    destruirCola(C);
}
```

```c
typedef struct info{
    int valor;
}Info;

typedef struct nodo{
    Info *elem;
    struct nodo *sgte;
}Nodo;

typedef struct cola{
    Nodo *ini;
    Nodo *fin;
    int tam;
}Cola;
```

Colas - Implementación

```c
int main(int argc, char *argv[]){
    Cola *C;
    C = crearCola();
    encolar(C,15); //push
    encolar(C,4);
    encolar(C,5);
    encolar(C,10);
    recorrerCola(C);
    printf("\n Valor desencolado: %d", desencolar(C)->valor);
    recorrerCola(C);
    printf("\n Tamaño de la cola: %d", C->tam);
    destruirCola(C);
}
```

```c
//Crea la información del nodo
Info *crearInfo(int pval){
    Info *newInfo;
    if (newInfo= (Info *) malloc(sizeof(Info))){
        newInfo->valor = pval;
    }else{
        printf("ERROR: MEMORIA INFO NO ASIGNADA");
    }
    return newInfo;
}
```

```c
//Crear Cola
Cola *crearCola(){
    Cola *c;
    if (c = (Cola *) malloc(sizeof(Cola)))
    {
        c->ini=c->fin=NULL;
        c->tam=0;
    }else{
        printf("ERROR: MEMORIA COLA NO ASIGNADA");
    }
    return c;
}
```

```c
//Encolar
void encolar(Cola *c, int pval){
    //Crear Info
      Info *I;
      I = crearInfo(pval);
    //crear Nodo
    Nodo  *newNodo;
    if (newNodo = (Nodo *) malloc(sizeof(Nodo))) {
        if (esVacia(c)){
            c->ini = newNodo;
        }else{
            c->fin->sgte = newNodo;
        }
        c->fin = newNodo;
        newNodo->sgte = NULL;
        newNodo->elem = I;
        c->tam++;
    }else{
        printf("ERROR: MEMORIA NODO NO ASIGNADA");
        free(newNodo);
    }
}
```

Colas - Implementación

```
//Desencolar
Info *desencolar(Cola *c){
    Nodo *aux;
    Info *inf;
    if (esVacia(c))
        printf("La Cola está vacía no se puede desencolar\n");
    else
    {
      aux = c->ini; //aux apunta al inicio de la cola
      inf = aux->elem; //Se guarda la información a eliminar
      c->ini = aux->sgte; //El inicio de c se asigna al nodo sgte (segundo nodo)
      c->tam--;
      free(aux);
    }
    return inf; //Retorna la información desencolada
}
```

```
//Función que revisa si la Cola está vacía
bool esVacia(Cola *c){
    if(c->ini == NULL) return true;
    else return false;
}
```

TAREA:
¿Qué funciones falta implementar?. Impleméntelas junto con el main

¿ cuál es la eficiencia de las operaciones de una cola?

Ejercicios Propuestos:

- **Escriba una función que reciba como parámetro una cola de números enteros y nos devuelva el mayor y el menor de la cola.**
- **Escriba una función que reciba como parámetro una cola de números enteros y devuelva la misma cola pero invertida (no usar estructura auxiliar)**
- Escribir una función que reciba como argumento dos colas, las mezcle y retorne una única cola como resultado. Los datos en la cola de resultado pueden quedar en cualquier orden.