Unidad 2: Estructuras de Datos

Tema III. Estructuras de datos Compuestas.

Pilas, Colas y Listas implementadas con punteros

Programación I (Plan 1999)

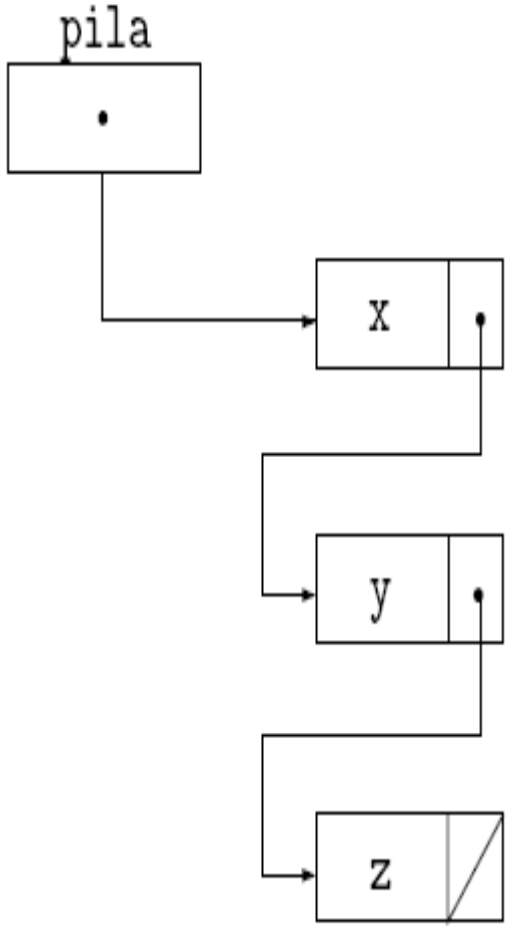
Algoritmos y Estructuras de Datos II (Plan 2009)

PILAS

Una pila es un tipo de lista en el que todas las inserciones y eliminaciones de elementos se realizan por el mismo extremo de la lista.

El nombre de pila procede de la similitud en el manejo de esta estructura de datos y la de una “pila de objetos".

Estas estructura también son llamadas listas LIFO, acrónimo que refleja la característica más importante de las pilas.

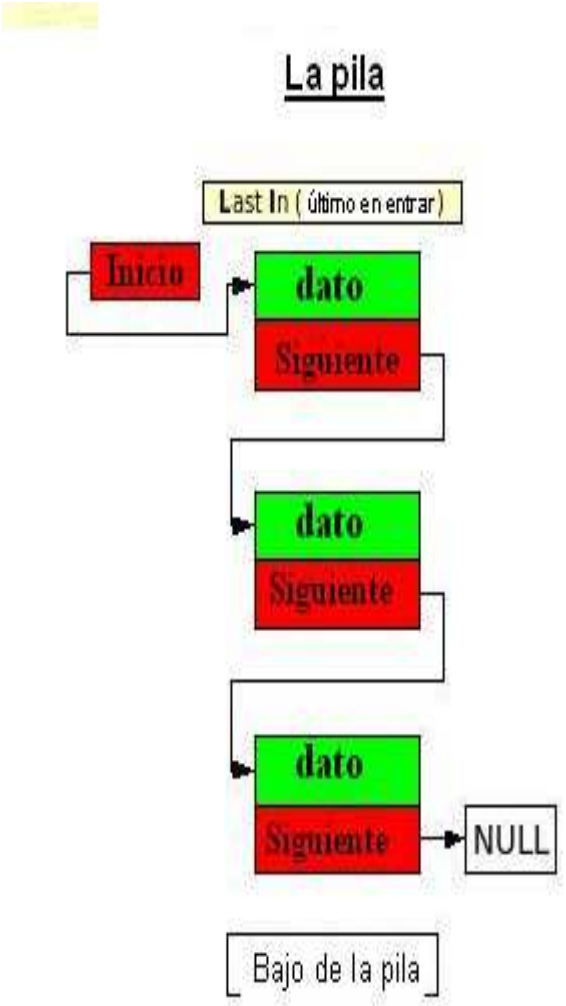


PILAS

Ya que la inserción es siempre hecha al inicio de la lista, el primer elemento de la lista será el ultimo elemento ingresado, por lo tanto estará en la cabeza de la pila.

No se ha utilizado un puntero ”**fin”**, como se hace en el caso de la lista enlazada simple, ya que el objetivo no es el de tratar una lista enlazada, sino una **pila**.

En resumen …el último elemento ingresado, será el primer elemento recuperado.



LA ELABORACIÓN DEL MODELO DE UN ELEMENTO DE LA PILA

1. Para definir un elemento de la **pila** será utilizado el tipo **struct**. El elemento de la pila contendrá un campo **dato** y un puntero **siguiente**.

2. El puntero **siguiente** tiene que ser de la misma clase que el elemento, de lo contrario no va a poder apuntar hacia el elemento. El puntero **siguiente** permitirá el acceso al próximo elemento.

```
typedef struct ElementoLista {
    char *dato;
    struct ElementoLista *siguiente;
}Elemento;
```

1

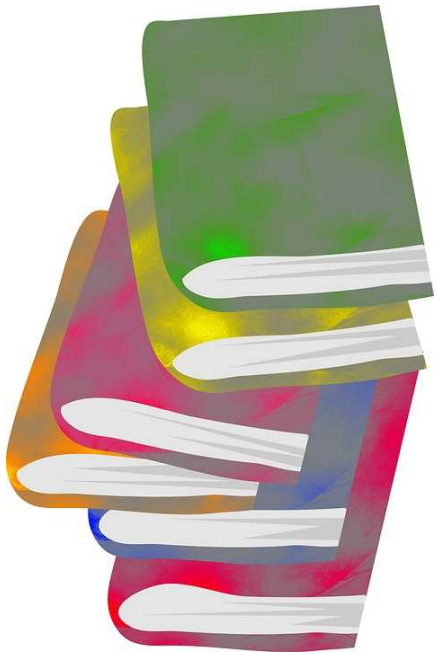
- Para permitir las operaciones sobre la pila, vamos a guardar ciertos elementos: el primer elemento, el numero de elementos

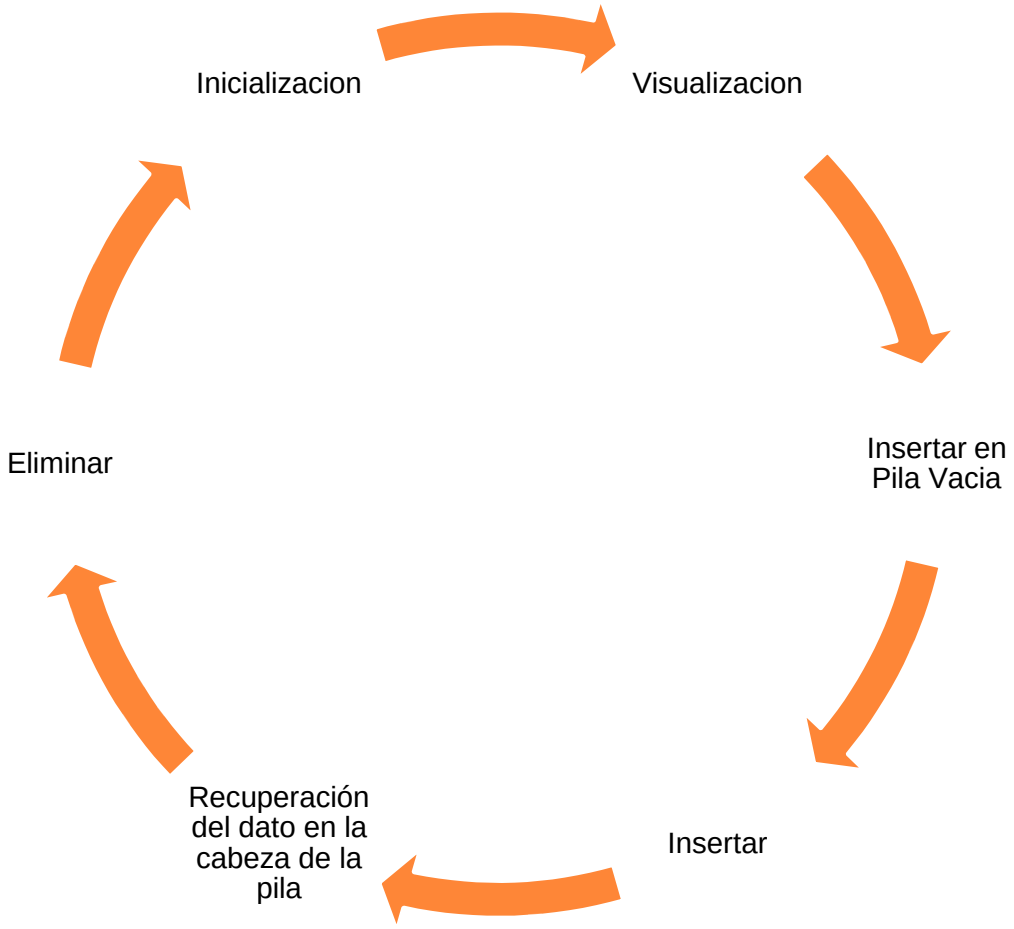
2

- El primer elemento, que se encuentra en la cabeza de la pila, nos permitirá realizar la operación de recuperación de los datos situados en la parte superior de la pila. Para ello, se utilizará otra estructura (no es obligatorio, pueden ser utilizadas variables).

```
typedef struct
ListaUbicación{
  Elemento *inicio;
  int tamaño;
}
```

Operaciones con Pilas





Magter. Oscar Adolfo Vallejos
FaCENA - UNNE

OPERACIONES SOBRE LAS PILAS

Inicialización

- Modelo de la función: void inicialización (Pila *tas);

Esta operación debe ser realizada antes de cualquier otra operación sobre la pila. Esta inicializa el puntero **inicio** con el valor **NULL** y el tamaño con el valor 0.

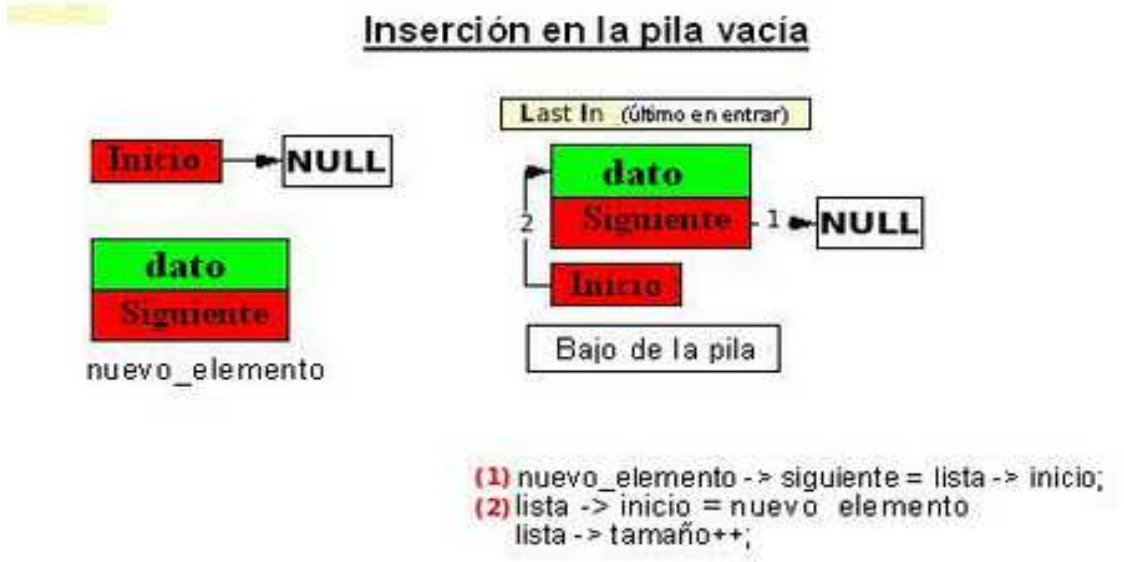
```
void inicialización (Pila * tas) {
        tas->inicio = NULL;
        tas->tamaño = 0;
 }
```

INSERCIÓN DE UN ELEMENTO EN LA PILA

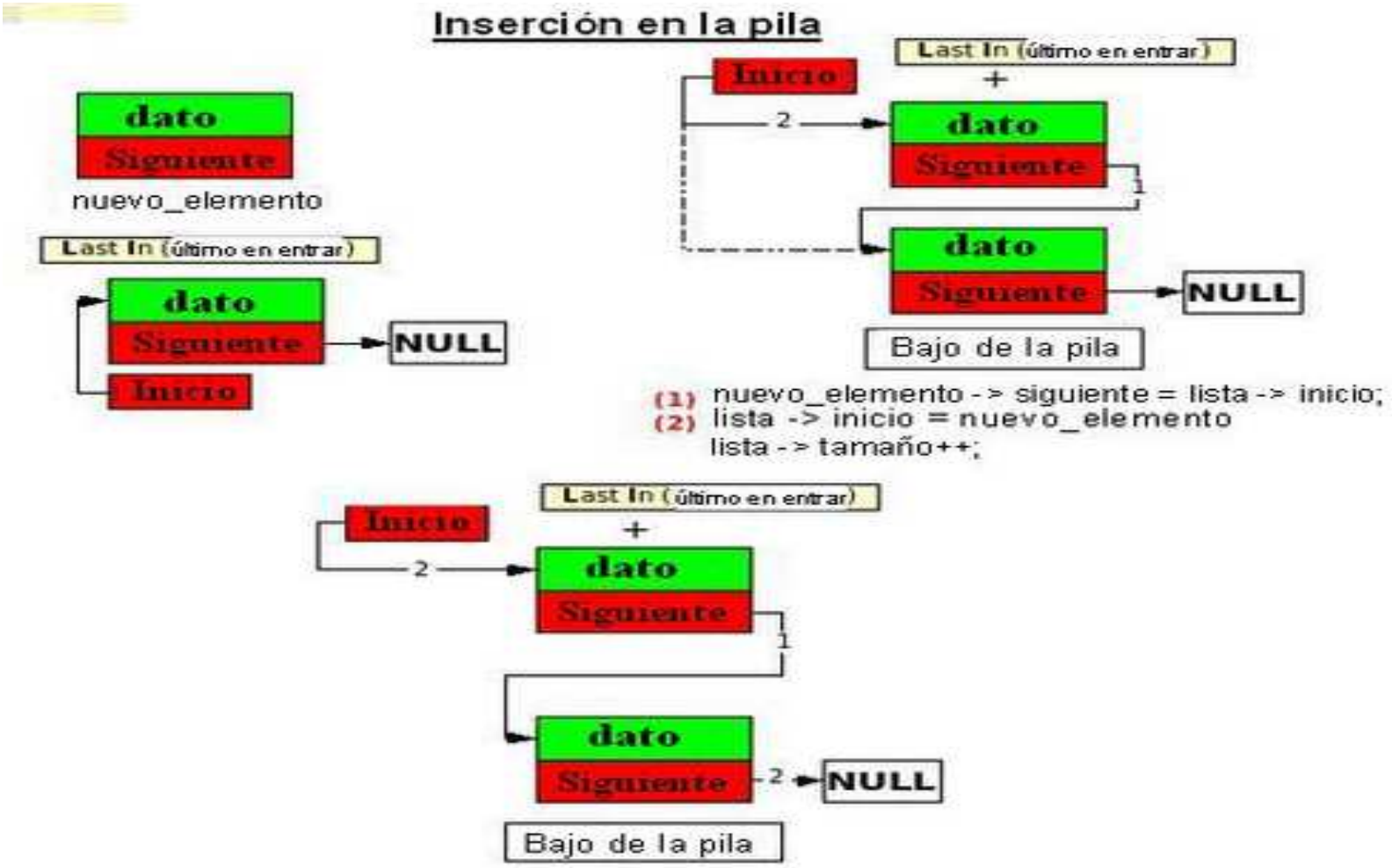
- El algoritmo de inserción y el registro de los elementos: declaración del elemento que va a insertarse, asignación de la memoria para el siguiente elemento, introducir el contenido del campo de los datos, actualizar el puntero **inicio** hacia el primer elemento (la cabeza de la pila). Actualizar el **tamaño** de la pila.

- Modelo de la función: int apilar (Pila *tas, char *dato);

- La primera imagen muestra el comienzo de la inserción, por lo tanto la lista de tamaño **1** después de la inserción. La característica de la pila no es muy apreciada con un solo elemento, ya que es el único a recuperar.



- Esta segunda imagen nos permite observar el comportamiento de la pila. La inserción siempre se hace en la parte superior de la pila (al inicio de la lista).



LA FUNCIÓN

```
/* apilar (añadir) un elemento en la pila */
int apilar (Pila * tas, char *dato){
  Elemento *nuevo_elemento;
  if ((nuevo_elemento = (Elemento *) malloc (sizeof
(Elemento))) == NULL)
    return -1;
  if ((nuevo_elemento->dato = (char *) malloc (50 * sizeof
(char)))
      == NULL)
    return -1;
  strcpy (nuevo_elemento->dato, dato);
  nuevo_elemento->siguiente = tas->inicio;
  tas->inicio = nuevo_elemento;
  tas->tamaño++;
}
```

ELIMINAR UN ELEMENTO DE LA PILA

- Simplemente hay que eliminar el elemento hacia el cual apunta el puntero **inicio**. Esta operación no permite recuperar el dato en la cabeza de la pila, solo eliminarlo.

  Modelo de la función: int desapilar (Pila *tas);

  La función da como resultado -1 en caso de error, si no devuelve 0.

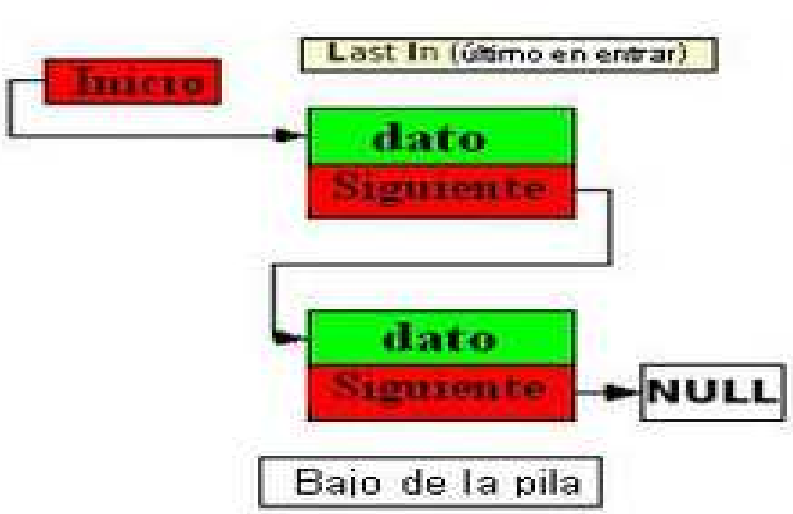
  Las etapas:

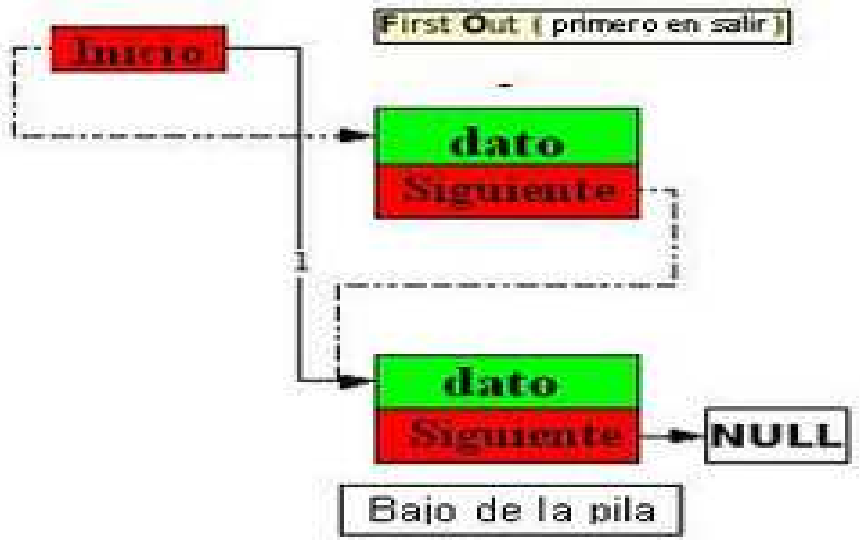
  El puntero **sup_elemento** contendrá la dirección del primer elemento.

  El puntero **inicio** apuntará hacia el segundo elemento (después de la eliminación del primer elemento, el segundo elemento estará en la cabeza de la pila).

  El **tamaño** de la pila disminuirá un elemento.

Eliminar un elemento de la pila






```
int desapilar (Pila * tas){
  Elemento *sup_elemento;
  if (tas->tamaño == 0)
    return -1;
  sup_elemento = tas->inicio;
  tas->inicio = tas->inicio->siguiente;
  free (sup_elemento->dato);
  free (sup_elemento);
  tas->tamaño--;
  return 0;
}
```

```
(1) sup_elemento = lista -> inicio;
    lista -> inicio = lista -> Inicio -> siguiente;
    lista -> tamaño++;
```

VISUALIZACIÓN DE LA PILA

- Para mostrar la pila entera, es necesario posicionarse al inicio de la pila (el puntero **inicio** lo permitirá). Luego, utilizando el puntero **siguiente** de cada elemento, la pila es recorrida del primero hacia el ultimo elemento. La condición para detenerse es determinada por el **tamaño** de la pila.

```
/* visualización de la pila */
void muestra (Pila * tas){
  Elemento *actual;
  int i;
  actual = tas->inicio;

  for(i=0;i<tas->tamaño;++i){
    printf("\t\t%s\n", actual->dato);
    actual = actual->siguiente;
  }
}
```

RECUPERACIÓN DEL DATO EN LA CABEZA DE LA PILA

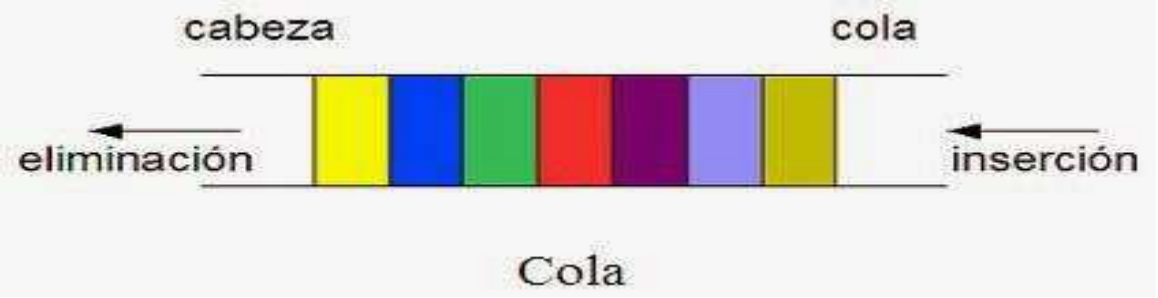
- Para recuperar el dato en la cabeza de la pila sin eliminarlo, he utilizado una macro. La macro lee los datos en la parte superior de la pila utilizando el puntero **inicio**.

```
#define pila_dato(tas) tas->inicio->dato
```

COLAS

- Una cola es una lista en la que todas las inserciones se realizan por un extremo y todas las eliminaciones se realizan por el otro extremo de la lista.

- El ejemplo más importante de esta estructura de datos, del cual recibe el nombre, es el de una cola de personas ante una ventanilla. En esta situación, el primero en llegar es el primero en ser servido; por esto, las colas también se llaman listas FIFO.

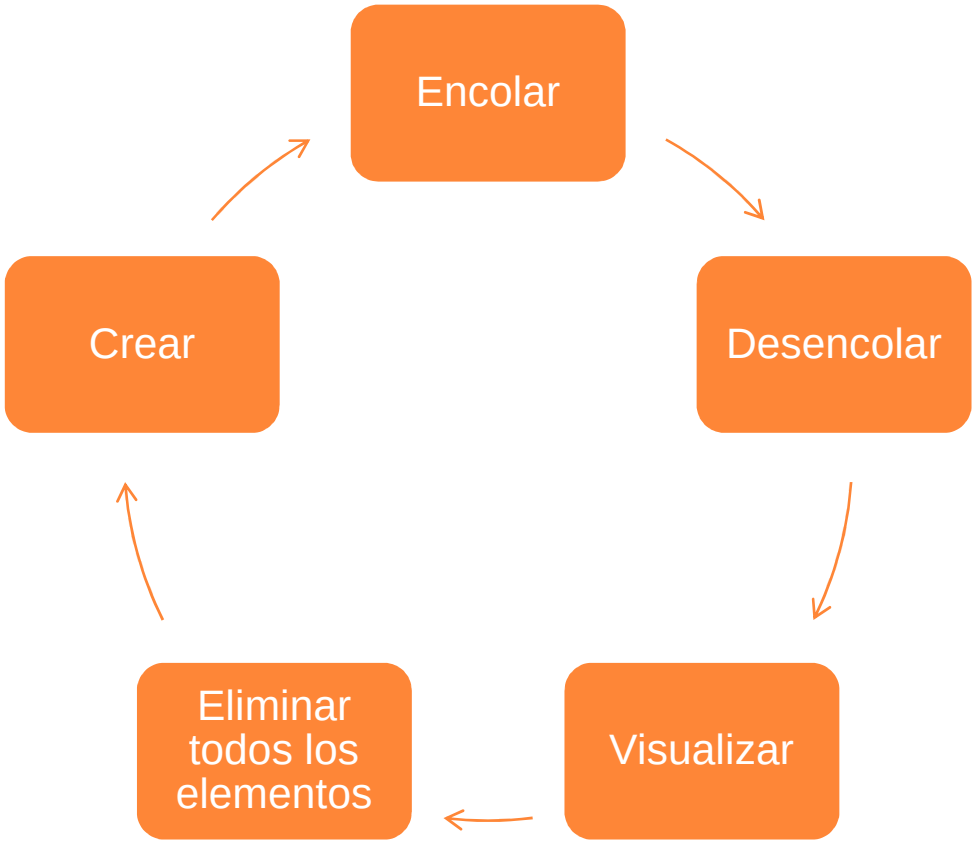


COLAS

- Una cola es un **tipo especial de lista enalazada en la que sólo se pueden insertar nodos en uno de los** extremos de la lista y sólo se pueden **eliminar nodos en el otro. Además, como sucede con las pilas, las** escrituras de datos siempre son inserciones de nodos, y las lecturas siempre eliminan el nodo leído.
- Este tipo de lista es conocido como lista **FIFO (First In First Out), el primero en entrar es el primero en** salir.
- El nodo típico para construir colas es el mismo que para la construcción de listas y pilas:

```
/*                Estructura de los nodos de la cola
--------------------------------------------------------------------------*/
struct nodo
{
    int nro;
    struct nodo *sgte;
};


/*                      Estructura de la cola
--------------------------------------------------------------------------*/
struct cola
{
    nodo *delante;
    nodo *atras  ;
};
```

OPERACIONES BÁSICAS CON COLAS



Magter. Oscar Adolfo Vallejos
FaCENA - UNNE

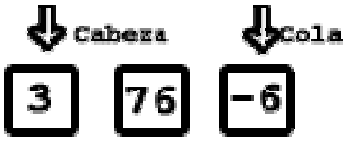
CREAR COLA

```
cola CREAR () {
   cola C;
   C = (tcola *) malloc(sizeof(tcola));
   if (C == NULL)
         error("Memoria insuficiente.");
   C->ant = C->post = (celda *)malloc(sizeof(celda));
   if (C->ant == NULL) error("Memoria insuficiente.");
   C->ant->siguiente = NULL;
return C;
}
```

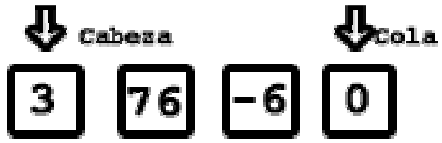
Magter. Oscar Adolfo Vallejos
FaCENA - UNNE

Insertar (encolar)

La inserción en las colas se realiza por la cola de las mismas, es decir, se inserta al final de la estructura.
Para llevar a cabo esta operación únicamente hay que reestructurar un par de punteros, el último nodo debe pasar a apuntar al nuevo nodo (que pasará a ser el último) y el nuevo nodo pasa a ser la nueva cola de la cola.



Si a esta cola le añadimos el elemento 0, la cola resultante sería:



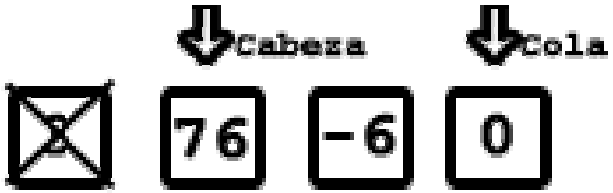
ENCOLAR

```
/*                        Encolar elemento
-------------------------------------------------------------------------*/
void encolar( struct cola &q, int valor )
{
     struct nodo *aux = new(struct nodo);

     aux->nro = valor;
     aux->sgte = NULL;

     if( q.delante == NULL)
         q.delante = aux;   // encola el primero elemento
     else
         (q.atras)->sgte = aux;

     q.atras = aux;         // puntero que siempre apunta al ultimo elemento

}
```

BORRAR (DESENCOLAR)

- El borrado es una operación muy simple en las colas. Esta operación supone extraer la cabeza de la cola, ya que es el elemento que más tiempo lleva en la estructura. Para llevar a cabo esta operación únicamente hay que extraer el elemento situado en la cabeza de la cola y avanzar el puntero *cabeza* una posición, para que de esta forma la nueva cabeza sea el segundo elemento que más tiempo lleva en la cola.

- Si realizamos la operación eliminar sobre la colas de 4 elementos del último gráfico el resultado sería el siguiente:



DESENCOLAR

Se quita de la cola el primer elemento. Para ello, una variable auxiliar apunta al inicio de la cola. Luego el siguiente elemento de la cola se mueve al inicio. Y finalmente se borra (delete) la variable auxiliar.

```
/*                          Desencolar elemento

------------------------------------------------------------------------*/
int desencolar( struct cola &q )
{
     int num ;
     struct nodo *aux ;


     aux = q.delante;      // aux apunta al inicio de la cola
     num = aux->nro;
     q.delante = (q.delante)->sgte;
     delete(aux);          // libera memoria a donde apuntaba aux


     return num;
}
```

VISUALIZAR (MOSTRAR LOS ELEMENTOS)

Utilizando un “while” se recorre toda la cola, desde la cabeza hasta el ultimo elemento de la misma

```
/*                        Mostrar Cola
--------------------------------------------------------------------------*/
void muestraCola( struct cola q )
{
     struct nodo *aux;

     aux = q.delante;

     while( aux != NULL )
     {
            cout<<"   "<< aux->nro ;
            aux = aux->sgte;
     }
}
```

ELIMINAR TODOS LOS ELEMENTOS DE LA COLA

Utilizando un “while” se recorre toda la cola, desde la cabeza hasta el ultimo elemento de la misma y uno a uno se los borra (delete)

```
/*              Eliminar todos los elementos de la Cola
--------------------------------------------------------------------------*/
void vaciaCola( struct cola &q)
{
     struct nodo *aux;

     while( q.delante != NULL)
     {
            aux = q.delante;
            q.delante = aux->sgte;
            delete(aux);
     }
     q.delante = NULL;
     q.atras   = NULL;

}
```

OPERACIONES BÁSICAS CON COLAS. RESUMIENDO

- **Añadir elemento en una cola vacía:**

Partiremos de que ya tenemos el **nodo a insertar y, por supuesto un puntero que apunte a él, además** los **punteros que definen la cola, primero y ultimo que valdrán NULL:**

El proceso es muy simple, bastará con que:

nodo->siguiente apunte a NULL.

Y que los punteros primero y ultimo apunten a nodo.

OPERACIONES BÁSICAS CON COLAS

Magter. Oscar Adolfo Vallejos
FaCENA - UNNE

- **Añadir elemento en una cola no vacía:**

De nuevo partiremos de un nodo a insertar, con un puntero que apunte a él, y de una cola, en este caso, al no estar vacía, los punteros primero y ultimo no serán nulos:

El proceso sigue siendo muy sencillo:

Hacemos que nodo->siguiente apunte a NULL.

Después que ultimo->siguiente apunte a nodo.

Y actualizamos ultimo, haciendo que apunte a nodo.

OPERACIONES BÁSICAS CON COLAS

Magter. Oscar Adolfo Vallejos
FaCENA - UNNE

- **Añadir elemento en una cola, caso general:**

Para generalizar el caso anterior, sólo necesitamos añadir una operación:

Hacemos que nodo->siguiente apunte a NULL.

Si ultimo no es NULL,

hacemos que ultimo->siguiente apunte a nodo.

Y actualizamos ultimo, haciendo que apunte a nodo.

Si primero es NULL, significa que la **cola estaba vacía, así**

que haremos que primero apunte también a nodo.

Eliminar un elemento en una cola:

Ahora también existen dos casos:

- que la cola tenga un solo elemento
- que tenga más de uno.

Usaremos un puntero a un nodo auxiliar:

- Hacemos que nodo apunte al primer elemento de la cola
  - primero.
- Asignamos a primero la dirección del segundo nodo de la pila:
  - primero->siguiente.
- Guardamos el contenido del nodo para devolverlo como retorno.
- Liberamos la memoria asignada al primer nodo, el que queremos eliminar.

Eliminar un elemento en una cola con un solo elemento:

- También necesitamos un puntero a un nodo auxiliar
- Hacemos que nodo apunte al primer elemento de la pila, es decir
  - primero.
- Asignamos NULL a primero, que es la dirección del segundo nodo teórico de la cola:
  - primero->siguiente.
- Guardamos el contenido del nodo para devolverlo como retorno,
- Liberamos la memoria asignada al primer nodo, el que queremos eliminar.
- Hacemos que ultimo apunte a NULL, ya que la lectura ha dejado la cola vacía.

Bibliografía

- Mark Allen Weiss - Estructuras de Datos y Algoritmos - Florida International University - Año: 1995 - Editorial: Addison-Wesley Iberoamericana .

- Joyanes Aguilar, Luis - Programación en Pascal - 4ª Edición - Año: 2006 - Editorial: McGraw-Hill/Interamericana de España, S.A.U.

- Cristóbal Pareja Flores, Manuel Ojeda Aciego, Ángel Andeyro Quesada, Carlos Rossi Jiménez - Algoritmos y Programación en Pascal.

- Joyanes Aguilar, Luis - Fundamentos de la Programación. Algoritmos, Estructuras de Datos y Objetos - 3ª Edición - Editorial: McGraw-Hill