Estructura de datos

Pilas

Colas

Descripción del TDA Cola

- Una cola es un caso particular de lista en el cual los elementos se insertan en un extremo (el posterior o final) y se suprimen en el otro (el anterior o frente).
- Las colas se conocen también como listas *FIFO* (*first-in first-out*) o listas ``primero en entrar, primero en salir''.
- Algunas de las operaciones vistas para listas pierden sentido en el TDA Cola y se definen nuevas operaciones

- Podemos definir las siguientes operaciones:
  - CREA. Crea una cola vacía.
  - VACIA. Devuelve un valor cierto si la cola está vacía, y falso en caso contrario.
  - PRIMERO. Devuelve el primer elemento de una cola.
  - INSERTA. Añade un elemento por el extremo final de una cola.
  - SUPRIME. Suprime el primer elemento de una cola.

// Declaración de la estructura

```
typedef struct nodoCola {
    int dato;
    struct nodoCola *ptrSiguiente;
} NodoCola;
```

// Apuntadores a los extremos

```
NodoCola *ptrCabeza = NULL;
NodoCola *ptrTalon = NULL;
```

// Insertar elementos

```
void encolar( NodoCola *ptrCabeza, NodoCola *ptrTalon, const int info )
{
   NodoCola *ptrNuevo;

   ptrNuevo = (NodoCola *)malloc( sizeof( NodoCola ) );

   if ( ptrNuevo == NULL )
      printf( "No se agrego el entero %d. Memoria insuficiente\n", info );
   else {
      ptrNuevo->dato = info;
      ptrNuevo->ptrSiguiente = NULL;


      if ( estaVacia( *ptrCabeza ) )
         *ptrCabeza = ptrNuevo;
      else
         (*ptrTalon)->ptrSiguiente = ptrNuevo;

      *ptrTalon = ptrNuevo;
   }
};
```

Eliminar elementos

```
int desencolar( NodoCola **ptrCabeza, NodoCola **ptrTalon )
{
   NodoCola *ptrTemp;

   int valorElim;


   valorElim = (*ptrCabeza)->dato;
   ptrTemp = *ptrCabeza;
   *ptrCabeza = (*ptrCabeza)->ptrSiguiente;


   if ( *ptrCabeza == NULL ) *ptrTalon = NULL;

   free( ptrTemp );

   return valorElim;
```

// Mostrar cola y Vacia

```
void imprimirCola( NodoCola *ptrActual )
{
   printf( "La cola es:\n\n" );


   while ( ptrActual != NULL) {
      printf( "%d -> ", ptrActual->dato );
      ptrActual = ptrActual->ptrSiguiente;
   }
   printf( "NULL\n" );
}
```

```
int estaVacia( NodoCola *ptrCima )
{
   return ptrCima == NULL;
}
```

Descripción del TDA Pila

- Una pila es un caso especial de lista en la cual todas las inserciones y supresiones tienen lugar en un extremo determinado llamado *tope*.

- A las pilas se les llama también listas *LIFO (last-in first-out)* o listas "primero en entrar, primero en salir".

- Al igual que ocurría con el TDA Cola, en el TDA Pila tampoco se definen operaciones de posicionamiento en la pila. Esto es debido a que todas las operaciones de acceso se realizan en la misma posición, el tope de la pila.

Un TDA de la familia pila incluye a menudo las cinco operaciones siguientes:

- CREA. Crea una pila vacía.
- VACIA. Devuelve un valor cierto si la pila está vacía, y falso en caso contrario.
- TOPE. Devuelve el elemento situado el tope de la pila, sin extraerlo.
- PUSH. Añade un elemento a la pila, quedando éste situado en el tope.
- POP. Suprime el elemento situado en el tope de la pila.

// Declaración de la estructura

```
typedef struct nodoPila {
    char dato;
    struct nodoPila *ptrSiguiente;
} NodoPila;
```

// Apuntadores a los extremos

```
NodoPila *ptrPila = NULL;
```

// Insertar elementos

```
void apilar( NodoPila **ptrCima, const char info )
{
   NodoPila *ptrNuevo;

   ptrNuevo = (NodoPila *)malloc( sizeof( NodoPila ) );

   if ( ptrNuevo == NULL )
      printf( "No se agrego el caracter \'%c\'. Memoria insuficiente\n", info );
   else {
      ptrNuevo->dato = info;
      ptrNuevo->ptrSiguiente = *ptrCima;
      *ptrCima = ptrNuevo;
   }
}
```

Eliminar elementos

```
desapilar( NodoPila **ptrCima )
{
   NodoPila *ptrTemp;

   char valorElim;

   ptrTemp = *ptrCima;
   valorElim = (*ptrCima)->dato;
   *ptrCima = (*ptrCima)->ptrSiguiente;
   free( ptrTemp );

   return valorElim;
}
```

// Mostrar cola y Vacia

```
void imprimirPila( NodoPila *ptrActual )
{
   printf( "La pila es:\n\n" );


   while ( ptrActual != NULL) {
      printf( "%c -> ", ptrActual->dato );
      ptrActual = ptrActual->ptrSiguiente;
   }
   printf( "NULL\n" );
}



int estaVacia( NodoPila *ptrCima )
{
   return ptrCima == NULL;
```