LISTAS

```
typedef struct tipo_nodo{
        int valor;
        struct tipo_nodo *sig;
}nodo_lista;

/*Funciones prototipo*/
void crear_lista(nodo_lista **lista);
void insertar_ordenado(nodo_lista **lista, int dato);
void eliminar_nodo(nodo_lista **lista, int dato);
nodo_lista *buscar_valor(nodo_lista *lista, int dato);
void imprimir_lista(nodo_lista *lista);
int lista_vacia(nodo_lista *lista);

/*Definición de funciones*/
void crear_lista(nodo_lista **lista){*lista=NULL;}

void insertar_ordenado(nodo_lista **lista, int dato){
        nodo_lista *actual, *anterior;
        nodo_lista *nuevo=(nodo_lista *)malloc(sizeof(nodo_lista));
        nuevo->valor=dato;
        actual=*lista;
        anterior=NULL;
        while(actual!=NULL && actual->valor<dato){
                anterior=actual;
                actual=actual->sig;
        }
        if(anterior!=NULL){/*Inserto en el cuerpo*/
                anterior->sig=nuevo;
                nuevo->sig=actual;
        }else{                                  /*Inserto al inicio*/
                nuevo->sig=*lista;
                *lista=nuevo;
        }
}
```

```
void eliminar_nodo(nodo_lista **lista, int dato){
	nodo_lista *actual, *anterior;
	actual=*lista;
	anterior=NULL;
	while(actual!=NULL && actual->valor!=dato){
		anterior=actual;
		actual=actual->sig;
	}
	if(actual!=NULL){   /*dato encontrado*/
		if(anterior!=NULL){/*borrar del cuerpo*/
			anterior->sig=actual->sig;
		}else{                                  /*borrar del inicio*/
			*lista=actual->sig;
		}
		free(actual);
	}
}

nodo_lista *buscar_valor(nodo_lista *lista, int dato){
	nodo_lista *aux;
	aux=lista;
	while(aux!=NULL && aux->valor!=dato)
		aux=aux->sig;
	return aux;
}

void imprimir_lista(nodo_lista *lista){
	nodo_lista *aux;
	aux=lista;
	printf("\n********** LISTA ORDENADA **********\n");
	while(aux!=NULL){
		printf("\t\t%d\n", aux->valor);
		aux=aux->sig;
	}
}

int lista_vacia(nodo_lista *lista){
	if(lista==NULL)
		return 1;
	else
		return 0;
}
```

COLAS

```
//DEFINICION DE ESTRUCTURAS
typedef struct tipo_nodo{
        int valor;
        struct tipo_nodo *sig;
}nodo;

typedef struct tipo_cola{
        nodo *primero, *ultimo;
}t_cola;

//CREA LA COLA
void crear_cola(t_cola *cola){
        cola->primero=NULL;
        cola->ultimo=NULL;
}

//ENCOLAR UN ELEMENTO
void encolar(t_cola *cola, int dato){
        nodo *nuevo=(nodo *)malloc(sizeof(nodo));
        nuevo->valor=dato;
        nuevo->sig=NULL;
        if(cola->primero==NULL){
                cola->primero=nuevo;
                cola->ultimo=nuevo;
        }else{
                cola->ultimo->sig=nuevo;
                cola->ultimo=nuevo;
        }
}

//DESENCOLAR UN ELEMENTO
/*Antes de invocar a descencolar debe
verificarse que la cola no este vacia*/
int desencolar(t_cola *cola){
        int valor;
        nodo *aux;
        aux=cola->primero;
        cola->primero=aux->sig;
        valor=aux->valor;
        free(aux);
        return valor;
}
//VERIFICA SI LA COLA ESTA VACIA
int cola_vacia(t_cola cola){
        if(cola.primero==NULL)
                return 1;
        else
                return 0;
}
```

```
// LISTA ELEMENTOS DE LA COLA
void muestra(t_cola cola){
	nodo *actual;
	actual=cola.primero;
	printf("\n********** COLA **********\n");
	while(actual!=NULL){
		printf("\t\t%d\n", actual->valor);
		actual=actual->sig;
	}
	printf("\n********** ---------- **********\n");
}
```

PILAS

```
//ESTRUCTURA PILA
typedef struct tipo_nodo{
	int valor;
	struct tipo_nodo *sig;
}nodo_pila;

//CREA LA PILA
void crear_pila(nodo_pila **pila){*pila=NULL;}

//APILA UN ELEMENTO EN LA PILA
void apilar(nodo_pila **pila, int dato){
	nodo_pila *nuevo;
	nuevo=(nodo_pila *)malloc(sizeof(nodo_pila));
	nuevo->valor=dato;
	nuevo->sig=*pila;
	*pila=nuevo;
}

//DESAPILA UN ELEMENTO DE LA PILA
int desapilar(nodo_pila **pila){
	int valor;
	nodo_pila *aux;
	aux=*pila;
	*pila=aux->sig;
	valor=aux->valor;
	free(aux);
	return valor;
}

//INDICA SI LA PILA ESTA VACÍA
int pila_vacia(nodo_pila *pila){
	if(pila==NULL)
		return 1;
	else
		return 0;
}

// LISTA ELEMENTOS DE LA PILA
void muestra(nodo_pila *pila){
	nodo_pila *actual;
	actual=pila;
	printf("\n********** PILA **********\n");
	while(actual!=NULL){
		printf("\t\t%d\n", actual->valor);
		actual=actual->sig;
	}
	printf("\n********** ---------- **********\n");
}
```