FILA, COLA Y LISTAS

COLA → METODO FIFO.
IMPRIMA EL PRIMERO, LUEGO EL SEGUNDO Y ASI (HASTA Q' NO TERMINE EL PRIMERO, NO SEGUIRÁ CON EL OTRO)
FILA → METODO LIFO
LO ULTIMO QUE PUSE, ES LO PRIMERO QUE SACO.

PUNTERO ACA ES INDICE

LISTAS PRIMERO: L2

```
STRUCT CLIENTES {
    INT CODIGO;
    CHAR NOMBRE [20];
    CHAR LOCALIDAD [20];
    INT INDICE SIGUIENTE;
} CLI [50];
```

0: 07, RR, L1, [4] — 1: 02, LL, L2, [3] — 2: 01, AA, L3, [1]
3: 09, MM, L4, [0] — 4: 05, ZZ, L3, [9999];

0	07	RR	L1	4
1	02	LL	L2	3
2	01	AA	L3	2
3	09	MM	L4	0
4	05	ZZ	L3	9999

DOBLE INDICE → MULTIPLE LISTAS
NO SE USA.

→ CARGAR
SI ES 1RO ——— 0 → 9999
2DO Ó MAS COMPARO ≤ ———
≥ ———
SI ADEMAS = ULTIMO = 9999
MUESTRO

LISTAS ENLAZADAS: ES UNA SECUENCIA DE ELEMENTOS UNO DETRÁS DEL OTRO, EN LA QUE CADA ELEMENTO SE CONECTA AL SIGUIENTE ELEMENTO POR UN ENLACE O INDICE

NODO — INDICE

[DATO 1 | P] → [DATO 2 | P] → [DATO 3 | P] → [DATO 4 | /]

CADA NODO (ELEMENTO) CONTIENE UN ÚNICO ENLACE Q' CONECTA ESE NODO AL NODO SIGUIENTE. LA LISTA ES EFICIENTE EN RECORRIDOS DIRECTOS (ADELANTE).

SUS OPERACIONES:
- DECLARACION DE LOS TIPOS NODOS E INDICE A NODO.
- INICIALIZACIÓN O CREACIÓN
- INSERTAR ELEMENTOS EN UNA LISTA
- ELIMINAR " DE " "
- BUSCAR " " " " (COMPROBAR LA EXISTENCIA EN UNA LISTA)
- RECORRER UNA LISTA ENLAZADA (VISITAR CADA NODO DE LA LISTA)
- COMPROBAR SI LA LISTA ESTA VACIA.

7) Lista simplemente enlazada con PRIMER y ULTIMO nodo.

```
TYPEDEF STRUCT {
int dato;
struct Nodo *sgte; // estructura auto-referenciada q' apunta a la S/G pos de la lista
}Nodo;

Nodo *primero = NULL; // Primera pos del nodo vacio (esta vacia)
Nodo *ultimo = NULL; // ultima  "  "  "  "
// 5
void agregar (Nodo *nodo) {
  if (primero = NULL) { //La lista esta vacia si es asi
    primero = Nodo; // ya tiene un valor 5 (no esta vacia)
    ultimo = Nodo;
  }
// 5 -> 6
  else { // Caso contrario de q' no esta vacio (agregarle un nuevo valor de mi lista)
    ultimo -> sgte = Nodo; // ultimo apunta al S/G nodo
    ultimo = Nodo; // valor 6
  }
}
int main () { // Imprimir los datos.
  Nodo *primerNodo = malloc (sizeof (Nodo)); // Ahora reservo el todos los nodos
  primerNodo -> dato = 5; // Cargando el dato correspondiente (5)

  Nodo *segundoNodo = malloc (sizeof (Nodo));
  segundoNodo -> dato = 7;

agregar (primerNodo); // Funcion de primer Nodo (agregar)
agregar (segundoNodo);

Nodo *I = primero; // Apunta a I para q' imprima todos los datos de primero
while (I != NULL) { // mientras q' I sea recorriendo todos los pos de mi lista mientras sea distinto de
  printf ("%d\n", I -> dato); // Imprimir el dato
  I = I -> sgte; // va del 1er nodo, 2nodo y asi hasta el nodo q' tenia la lista.
}
  return 0;
}
```

7.9) Crear una lista simplemente enlazada de 3 números flotantes, pidiendo al usuario dichos números.

```
typedef struct {
    float numero;
    struct Nodo * sgte;
} Nodo;

Nodo * primer = NULL;
Nodo * ultimo = NULL;

void agregar(Nodo * nuevo){
    if (primer == NULL){
        primer = nuevo;
        ultimo = nuevo;
    } // 5 -> 7
    else {
        ultimo -> sgte = nuevo;
        ultimo = nuevo;
    }
}

int main(){
    float n1, n2, n3;

    printf("Ingrese 3 numeros flotantes: ");
    scanf("%f %f %f", &n1, &n2, &n3);

    Nodo * primerNodo = malloc(sizeof(Nodo));
    primerNodo -> numero = n1;
    Nodo * segundoNodo = malloc(sizeof(Nodo));
    segundoNodo -> numero = n2;
    Nodo * tercerNodo = malloc(sizeof(Nodo));
    tercerNodo -> numero = n3;

    agregar(primerNodo);
    agregar(segundoNodo);
    agregar(tercerNodo);

    // Recorriendo la lista
    Nodo *I = primerNodo;
    while(I != NULL){
        printf("%f", I -> numero);
        I = I -> sgte;
    }
}
```

Nodo -> sgte = NULL; // un N° a puntual siguiente nodo (inicialmente)