typedef struct ElementoLista

{

char \*dato;

struct ElementoLista \*siguiente;

} Elemento;

typedef struct ListaIdentificar

{

Elemento \*inicio;

Elemento \*fin;

int tamaño;

} Lista;

/\* inicialización de la lista \*/

void inicializacion (Lista \* lista);

/\* INSERCION \*/

/\* inserción en una lista vacía \*/

int ins\_en\_lista\_vacia (Lista \* lista, char \*dato);

/\* inserción al inicio de la lista \*/

int ins\_inicio\_lista (Lista \* lista, char \*dato);

/\* inserción al final de la lista \*/

int ins\_fin\_lista (Lista \* lista, Elemento \* actual, char \*dato);

/\* inserción en otra parte \*/

int ins\_lista (Lista \* lista, char \*dato, int pos);

/\* SUPRESION \*/

int sup\_inicio (Lista \* lista);

int sup\_en\_lista (Lista \* lista, int pos);

int menu (Lista \*lista,int \*k);

void muestra (Lista \* lista);

void destruir ( Lista \* lista);

void inicializacion (Lista \* lista) {

lista ->inicio = NULL;

lista ->fin = NULL;

lista ->tamaño = 0;

}

/\* insercion en una lista vacia \*/

int ins\_en\_lista \_vacia (Lista \* lista, 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 = NULL;

lista ->inicio = nuevo\_elemento;

lista ->fin = nuevo\_elemento;

lista ->tamaño++;

return 0;

}

/\* inserción al inicio de la lista \*/

int ins\_inicio\_lista (Lista \* lista, 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 = lista->inicio;

lista ->inicio = nuevo\_elemento;

lista ->tamaño++;

return 0;

}

/\*insercion al final de la lista \*/

int ins\_fin\_lista (Lista \* lista, Elemento \* actual, 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);

actual->siguiente = nuevo\_elemento;

nuevo\_elemento->siguiente = NULL;

lista ->fin = nuevo\_elemento;

lista ->tamaño++;

return 0;

}

/\* insercion en la posicion solicitada \*/

int ins\_ lista (Lista \* lista, char \*dato, int pos) {

if (lista ->tamaño < 2)

return -1;

if (pos < 1 || pos >= lista ->tamaño)

return -1;

Elemento \*actual;

Elemento \*nuevo\_elemento;

int i;

if ((nuevo\_elemento = (Elemento \*) malloc (sizeof (Elemento))) == NULL)

return -1;

if ((nuevo\_elemento->dato = (char \*) malloc (50 \* sizeof (char))) == NULL)

return -1;

actual = lista ->inicio;

for (i = 1; i < pos; ++i)

actual = actual->siguiente;

if (actual->siguiente == NULL)

return -1;

strcpy (nuevo\_elemento->dato, dato);

nuevo\_elemento->siguiente = actual->siguiente;

actual->siguiente = nuevo\_elemento;

lista ->tamaño++;

return 0;

}

/\* supresión al inicio de la lista \*/

int sup\_inicio (Lista \* lista) {

if (lista ->tamaño == 0)

return -1;

Elemento \*sup\_elemento;

sup\_elemento = lista ->inicio;

lista ->inicio = lista ->inicio->siguiente;

if (lista ->tamaño == 1)

lista ->fin = NULL;

free (sup\_elemento->dato);

free (sup\_elemento);

lista ->tamaño--;

return 0;

}

/\* suprimir un elemento después de la posición solicitada \*/

int sup\_en\_lista (Lista \* lista, int pos) {

if (lista ->tamaño <= 1 || pos < 1 || pos >= lista ->tamaño)

return -1;

int i;

Elemento \*actual;

Elemento \*sup\_elemento;

actual = lista ->inicio;

for (i = 1; i < pos; ++i)

actual = actual->siguiente;

sup\_elemento = actual->siguiente;

actual->siguiente = actual->siguiente->siguiente;

if(actual->siguiente == NULL)

lista ->fin = actual;

free (sup\_elemento->dato);

free (sup\_elemento);

lista ->tamaño--;

return 0;

}

/\* visualización de la Lista \*/

void muestra (Lista \* lista)

{

Elemento \*actual;

actual = lista ->inicio;

while (actual != NULL){

printf ("%p - %s\n", actual, actual->dato);

actual = actual->siguiente;

}

}

/\* destruir la Lista \*/

void destruir (Lista\* lista) {

while (lista ->tamaño > 0)

sup\_inicio (lista);

}