typedef struct dl\_Elemento Lista{

char \*dato;

struct dl\_ElementoLista \*anterior;

struct dl\_ElementoLista \*siguiente;

} dl\_Elemento;

typedef struct dl\_ListaIdentificar{

dl\_Elemento \*inicio;

dl\_Elemento \*fin;

int tamaño;

} dl\_Lista;

/\* inicialización de la lista \*/

void inicialización (dl\_Lista \* lista);

dl\_Elemento \*alloc (dl\_Elemento \* nuevo\_elemento);

/\* INSERCION \*/

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

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

int ins\_fin\_lista(dl\_Lista \* lista, char \*dato);

int ins\_después (dl\_Lista \* lista, char \*dato, int pos);

int ins\_antes (dl\_Lista \* lista, char \*dato, int pos);

/\*ELIMINACION\*/

int sup(dl\_Lista \*liste, int pos);

void muestra (dl\_Lista \* lista);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void muestra\_inv (dl\_Lista \* lista);

void destruir (dl\_Lista \* lista);

void inicialización (dl\_Lista \* lista)

{

lista->inicio = NULL;

lista->fin = NULL;

lista->tamaño = 0;}

int inserción\_en\_lista\_vacia (dl\_Lista \* lista, char \*dato){

dl\_Elemento \*nuevo\_elemento;

if ((nuevo\_elemento = alloc (nuevo\_elemento)) == NULL)

return -1;

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

nuevo\_elemento->anterior = NULL;

nuevo\_elemento->siguiente = NULL;

lista->inicio = nuevo\_elemento;

lista->fin = nuevo\_elemento;

lista->tamaño++;

return 0;

}

int ins\_inicio\_lista(dl\_Lista \* lista, char \*dato)

{

dl\_Elemento \*nuevo\_elemento;

if ((nuevo\_elemento = alloc (nuevo\_elemento)) == NULL)

return -1;

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

nuevo\_elemento->anterior = NULL;

nuevo\_elemento->siguiente = lista->inicio;

lista->inicio->anterior = nuevo\_elemento;

lista->inicio = nuevo\_elemento;

lista->tamaño++;

return 0;

}

int ins\_fin\_lista(dl\_Lista \* lista, char \*dato)

{

dl\_Elemento \*nuevo\_elemento;

if ((nuevo\_elemento = alloc (nuevo\_elemento)) == NULL)

return -1;

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

nuevo\_elemento->siguiente = NULL;

nuevo\_elemento->anterior = lista->fin;

lista->fin->siguiente = nuevo\_elemento;

lista->fin = nuevo\_elemento;

lista->tamaño++;

return 0;

}

int ins\_después (dl\_Lista \* lista, char \*dato, int pos)

{

int i;

dl\_Elemento \*nuevo\_elemento, \*actual;

if ((nuevo\_elemento = alloc (nuevo\_elemento)) == NULL)

return -1;

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

actual = lista->inicio;

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

actual = actual->siguiente;

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

nuevo\_elemento->anterior = actual;

if(actual->siguiente == NULL)

lista->fin = nuevo\_elemento;

else

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

actual->siguiente = nuevo\_elemento;

lista->tamaño++;

return 0;

}

int ins\_antes (dl\_Lista \* lista, char \*dato, int pos)

{

int i;

dl\_Elemento \*nuevo\_elemento, \*actual; if ((nuevo\_elemento = alloc (nuevo\_elemento)) == NULL)

return -1;

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

actual = lista->inicio;

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

actual = actual->siguiente;

nuevo\_elemento->siguiente = actual;

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

if(actual->anterior == NULL)

lista->inicio = nuevo\_elemento;

else

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

actual->anterior = nuevo\_elemento;

lista->tamaño++;

return 0;

}

int sup(dl\_Lista \*lista, int pos){

int i;

dl\_Elemento \*sup\_elemento,\*actual;

if(lista->tamaño == 0)

return -1;

if(pos == 1){

/\* eliminación de 1er elemento \*/

sup\_elemento = lista->inicio;

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

if(lista->inicio == NULL)

lista->fin = NULL;

else

lista->inicio->anterior == NULL; }

else if(pos == lista->tamaño){

/\* eliminacion del ultimo elemento \*/

sup\_elemento = lista->fin;

lista->fin->anterior->siguiente = NULL;

lista->fin = lista->fin->anterior; }

else {

/\* eliminacion en otra parte \*/

actual = lista->inicio;

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

actual = actual->siguiente;

sup\_elemento = actual;

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

actual->siguiente->anterior = actual->anterior; }

free(sup\_elemento->dato);

free(sup\_elemento);

lista->tamaño--;

return 0;

}

void destruir(dl\_Lista \*lista){

while(lista->tamaño > 0)

supp(lista,1);}

dl\_Elemento \*alloc (dl\_Elemento \* nuevo\_elemento){

if ((nuevo\_elemento = (dl\_Elemento \*) malloc (sizeof (dl\_Elemento))) == NULL)

return NULL;

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

return NULL;

return nuevo\_elemento;}