

Practica

# Listas Ligadas Sencillas

---

Luis Eduardo Galindo Amaya (1274895)



Asignatura	Estructuras de Datos (341)
Docente	Itzel Barriba Cazares
Fecha	2022-09-11

# Listas Ligadas Sencillas

Luis Eduardo Galindo Amaya (1274895)

2022-09-11

## Información De La Actividad

**Nombre de la actividad** {{{title}}}

**Fecha** 2022-09-11

**Lugar** Edificio 6E, Salón 204.

**Carácter de la actividad** Individual.

**Participante(es)** Luis Eduardo Galindo Amaya (1274895).

Repositorio de Github

## Prodecimieto

Se analizaron los documentos enviados por la profesora y se usaron como base para crear una libreria nueva para manejar lista ssencillas. La lista original simplemente ordenaba los elementos de menor a menor, en la nueva lista los terminos se pueden agregar por pocision:

- 0. Al inicio
- -1. Al final
- y cualquier otra pocision definida por el usuario.

## Analisis del resultado

Los resultados son los esperados, se pudo siplicar los metodos y ahora son mas legibles por si es necesario hacer modificaciones en el futuro.

# Capturas de Pantalla

## Inserción de datos

```

<tos/EDO-2022/07-09-2022
+ ..
+ src/
+ lista-sencilla.c
+ makefile
+ a.out
+ practica.org
+ practica.pdf
+ practica.tex

Ingresar el caracter entre corchetes para
realizar la operacion.

a - agregar elementos a la lista
e - eliminar elementos de la lista
i - mostrar las instrucciones
s - salir

-----

La lista esta vacia.

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? a
valor a insertar: 12
posicion a insertar (0: inicio, -1: final): 0
-----

[0: 12]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? a
valor a insertar: 34
posicion a insertar (0: inicio, -1: final): -1
-----

[0: 12]->[1: 34]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? a
valor a insertar: 45
posicion a insertar (0: inicio, -1: final): 1
-----

[0: 12]->[1: 45]->[2: 34]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir?

[1/5] 07-09-2022 (0:1 F:4) U:*** *eshell* Bot (27,31) (Eshell yas # company WK ivy GCMH)
05:06 PM

```

## Eliminación de datos

```

<tos/EDO-2022/07-09-2022
+ ..
+ src/
+ lista-sencilla.c
+ makefile
+ a.out
+ practica.org
+ practica.pdf
+ practica.tex

valor a insertar: 45
posicion a insertar (0: inicio, -1: final): 1
-----

[0: 12]->[1: 45]->[2: 34]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? e
posicion a insertar (0: inicio, -1: final): 0
Valor 12 eliminado

-----

[0: 45]->[1: 34]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? e
posicion a insertar (0: inicio, -1: final): -1
-----

[0: 45]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? a
valor a insertar: 12
posicion a insertar (0: inicio, -1: final): 0
-----

[0: 12]->[1: 45]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir? e
posicion a insertar (0: inicio, -1: final): 1
Valor 45 eliminado

-----

[0: 12]->NULL

-----

[a]gregar [e]liminar [i]instrucciones [s]alir?

[3/5] 07-09-2022 (0:1 F:4) C-x 1 U:*** *eshell* Bot (74,48) (Eshell yas # company WK ivy GCMH)
05:51 PM

```

## Conclusiones

Las listas a pesar de tener un comportamiento bien definido pueden modificarse para cumplir con roles muy específicos para la aplicación que se desea crear.

## Código

```
1
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 // Tipos de datos
6
7 struct nodoLista {
8     int dato;
9     struct nodoLista *ptrSiguiente;
10 };
11
12 typedef struct nodoLista NodoLista;
13 typedef NodoLista *ptrNodoLista;
14
15 // Prototipos
16
17 int capturar_entero();
18 char capturar_caracter();
19 int menu_principal();
20 void insertar(ptrNodoLista *ptrS, int posicion, int valor);
21 void eliminar(ptrNodoLista *ptrS, int posicion);
22 int estaVacia(ptrNodoLista ptrS);
23 void imprimeLista(ptrNodoLista ptrActual);
24 void instrucciones(void);
25 void liberarMemoria(ptrNodoLista *ptrS);
26
27 // Main
28
29 int main() {
30     ptrNodoLista lista = NULL;
31     int op = -1;
32     int captura;
33     int posicion;
34
35     instrucciones();
36
37     while (op) {
38         printf("_____\\n");
```

```

39     printf("\n");
40     imprimeLista(lista);
41     printf("\n");
42     printf("_____+_____\\n");
43     printf("[a]gregar [e]liminar [i]nstrucciones | [s]alir? ");
44     op = menu_principal();
45
46     switch (op) {
47     case 1: /* Insertar */
48         printf("valor a insertar: ");
49         captura = capturar_entero();
50         printf("posicion a insertar (0: inicio , -1: final): ");
51         posicion = capturar_entero();
52         insertar(&lista , posicion , captura);
53         break;
54
55     case 2: /* Eliminar */
56         printf("posicion a insertar (0: inicio , -1: final): ");
57         posicion = capturar_entero();
58         eliminar(&lista , posicion);
59         break;
60
61     case 3:
62         instrucciones();
63         break;
64     }
65 }
66
67 liberarMemoria(&lista);
68
69 return 0;
70 }
71
72 // Funciones _____
73
74 /**
75  * Verificar si esta vacia la lista
76  *
77  * @param ptrS lista de punteros
78  * @return Regrasa 1 si la lista esta vacia de lo contrario 0.
79  */
80 int estaVacia(ptrNodoLista ptrS) { return ptrS == NULL; }
81
82 /**
83  * Liberar la memoria del programa
84  *
85  * @param ptrS
86  */

```

```
87 void liberarMemoria(ptrNodoLista *ptrS) {
88     ptrNodoLista ptrAnterior;
89     ptrNodoLista ptrActual;
90
91     /* determinar si esta vacia */
92     if (estaVacia(*ptrS)) {
93         printf("La lista esta Vacía , nada que liberar.\n");
94         return;
95     }
96
97     ptrActual = *ptrS;
98     ptrAnterior = NULL;
99
100    for (int i = 0; ptrActual != NULL; i++) {
101        ptrAnterior = ptrActual;
102        ptrActual = ptrActual->ptrSiguiente;
103        free(ptrAnterior);
104    }
105
106    printf("MEMORIA LIBERADA CON EXITO!\n");
107 }
108
109 /**
110  * Muestra las intrucciones del progrma
111  */
112 void instrucciones() {
113     printf("\nIngresa el caracter entre corchetes para\n");
114     printf("realizar la operacion.\n\n");
115     printf("a – agregar elementos a la lista\n");
116     printf("e – eliminar elementos de la lista\n");
117     printf("i – mostrar las instrucciones \n");
118     printf("s – salir\n\n");
119 }
120
121 /**
122  * Imprime los valore de la lista
123  *
124  * @param ptrActual puntero a la lista
125  */
126 void imprimeLista(ptrNodoLista ptrActual) {
127
128     if (estaVacia(ptrActual)) {
129         printf("La lista esta vacia.\n");
130         return;
131     }
132
133     for (int i = 0; ptrActual != NULL; i++) {
134         printf("[%d: %d]->", i, ptrActual->dato);
```

```
135     ptrActual = ptrActual->ptrSiguiente;
136 }
137
138     printf("NULL\n");
139 }
140
141 /**
142  * Elimina un elemento de la lista
143  * NOTE: posicion < 0, Inserta en el ultimo elemento de la lista
144  * NOTE: 0, Inserta en el Primer elemento de la lista
145  *
146  * @param posicion del elemento a eliminar
147  */
148 void eliminar(ptrNodoLista *ptrS, int posicion) {
149     ptrNodoLista ptrAnterior;
150     ptrNodoLista ptrActual;
151
152     /* determinar si esta vacia */
153     if (estaVacia(*ptrS)) {
154         printf("La lista esta Vacia\n");
155         return;
156     }
157
158     ptrActual = *ptrS;
159     ptrAnterior = NULL;
160
161     /* eliminar el primer elemento o el unico elemento en la
162      lista si solo tiene un termino */
163     if (posicion == 0 || ptrActual->ptrSiguiente == NULL) {
164         *ptrS = ptrActual->ptrSiguiente;
165         printf("Valor %d eliminado\n", ptrActual->dato);
166         free(ptrActual);
167         return;
168     }
169
170     /* eliminar elemento al final */
171     if (posicion == -1) {
172         while (ptrActual->ptrSiguiente != NULL) {
173             ptrAnterior = ptrActual;
174             ptrActual = ptrActual->ptrSiguiente;
175         }
176
177         /* asigna el como nodo siguiente NULL por que se elimina
178          el ultimo */
179         ptrAnterior->ptrSiguiente = NULL;
180         free(ptrActual);
181         return;
182     }
```

```
183
184 /* eliminar elemento por posicion */
185 for (int i = posicion; i > 0; —i) {
186     /* iterar los punteros */
187     ptrAnterior = ptrActual;
188     ptrActual = ptrActual->ptrSiguiente;
189     /* si el siguiente putero no existe */
190     if (ptrActual == NULL) {
191         printf("Fuera de rango\n");
192         return;
193     }
194 }
195
196 ptrAnterior->ptrSiguiente = ptrActual->ptrSiguiente;
197 printf("Valor %d eliminado\n", ptrActual->dato);
198 free(ptrActual);
199 }
200
201 /**
202  * Inserta un elemento en la lista ,
203  * NOTE: posicion < 0, Inserta en el ultimo elemento de la lista
204  * NOTE: 0, Inserta en el Primer elemento de la lista
205  *
206  * @param *ptrS puntero a la lista
207  * @param posicion en la lista
208  * @param valor a insertar
209  */
210 void insertar(ptrNodoLista *ptrS, int posicion, int valor) {
211     ptrNodoLista ptrNuevo;
212     ptrNodoLista ptrAnterior;
213     ptrNodoLista ptrActual;
214
215     ptrNuevo = malloc(sizeof(NodoLista));
216
217     /* Verificar que hay memoria para contirnuar */
218     if (ptrNuevo == NULL) {
219         printf("%d no se inserto. Memoria no disponible.\n", valor);
220         exit(EXIT_FAILURE);
221     }
222
223     /* Si la memoria se puede reservar entonces podemos
224        usar el nuevo nodo */
225     ptrNuevo->dato = valor;
226     /* todavia no sabemos si hay mas nodos antes o despues así */
227     ptrNuevo->ptrSiguiente = NULL;
228
229     /* insertar al inicio o si la lista esta vacia insertamos
230        termino en el primer nodo y termina */
```



```
231     if (posicion == 0 || estaVacia(*ptrS)) {
232         ptrNuevo->ptrSiguiente = *ptrS;
233         *ptrS = ptrNuevo;
234         return;
235     }
236
237     /* Variables de iteracion */
238     ptrActual = *ptrS;
239     ptrAnterior = NULL;
240
241     /* Si la posicion es -1 entonces insertamos al final */
242     if (posicion < 0) {
243         /* hay dos formas de resolver este problema :*/
244
245         /* La lenta , iterando un numero muy grande con la
246            funcion de insertar en posicion arbitraria , es lenta por
247            que la funcion tiene que hacer varias verificaciones.
248            NOTE: 0x7FFF es el maximo valor entero con signo en C */
249
250         posicion = 0x7FFF;
251
252         /* La rapida , que itera desde el 0 hasta el ultimo termino
253            en general es mas rapida porque no tienes que reasignar
254            cada variable en cada iteracion */
255
256         /* while (ptrActual->ptrSiguiente != NULL) */
257         /*     ptrActual = ptrActual->ptrSiguiente; */
258         /* ptrActual->ptrSiguiente = ptrNuevo; */
259         /* return; */
260
261         /* las dos hacen exactamente lo mismo */
262     }
263
264     /* Si el valor es mayor a 0 entonces sustituyes el valor en la
265        posicion especificada */
266     for (int i = posicion; i > 0; —i) {
267         /* iterar los punteros */
268         ptrAnterior = ptrActual;
269         ptrActual = ptrActual->ptrSiguiente;
270         /* si el siguiente putero no existe */
271         if (ptrActual == NULL)
272             break;
273     }
274     ptrAnterior->ptrSiguiente = ptrNuevo;
275     ptrNuevo->ptrSiguiente = ptrActual;
276 }
277
278 /**
```

```
279 * Muestra el menu principal , captura el caracter de opcion y
280 * retorna el indice de la operacion a realizar
281 *
282 * @return operacion
283 */
284 int menu_principal() {
285     char foo = 0;
286     foo = capturar_caracter();
287
288     // convertr a minuscula
289     foo = foo | 32;
290
291     if (foo == 's')
292         return 0;
293     if (foo == 'a')
294         return 1;
295     if (foo == 'e')
296         return 2;
297     if (foo == 'i')
298         return 3;
299
300     printf("\nERROR: No existe esa opcion\n");
301     return -1;
302 }
303
304 /**
305  * Retorna el caracter capturado o 10 si esta vacio .
306  * el numero 10 corresponde a 'Line Feed / newline '
307  *
308  * Extraido de: https://stackoverflow.com/a/40949122
309  *
310  * @return caracter
311  */
312 char capturar_caracter() {
313     char line[256];
314     char ch;
315
316     if (fgets(line , sizeof line , stdin) == NULL) {
317         printf("Input error.\n");
318         exit(EXIT_FAILURE);
319     }
320
321     ch = line[0];
322     return ch;
323 }
324
325 /**
326  * Captura un numero entero y vacia el stdin
```

```
327  *
328  * @return valor capturado
329  */
330  int capturar_entero() {
331      int foo = 0;
332      scanf("%d", &foo);
333
334      /* vaciar inputbuffer de LF (10)
335         https://stackoverflow.com/a/7898516 */
336      int c;
337      while ((c = getchar()) != '\n' && c != EOF);
338
339      return foo;
340  }
```