

2-11-2021



Lista dinamica

Practica: 5

Materia: Estructura de datos

Sección: D01.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge
Daray

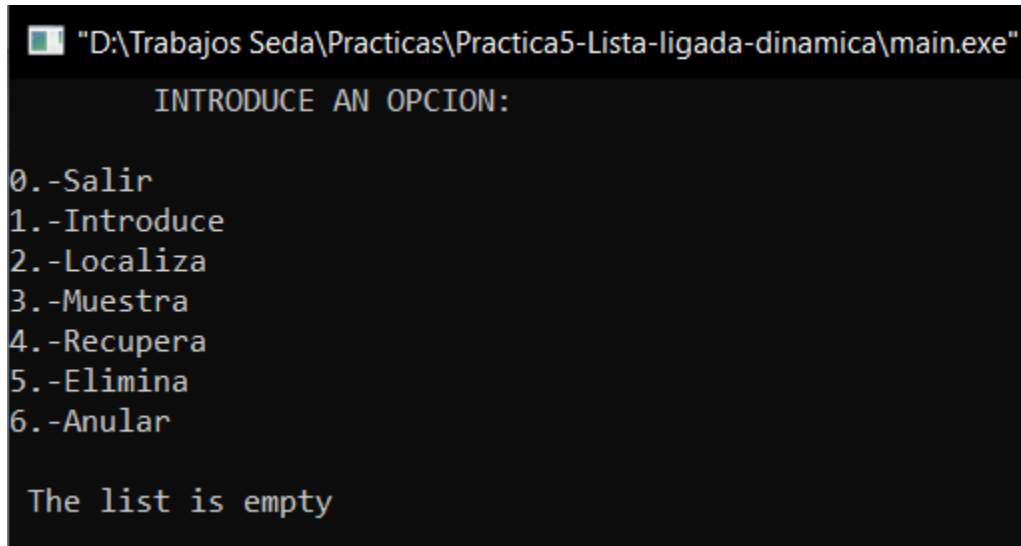
Nombre profesor: Julio Esteban
Valdes Lopez

Introducción

Mi práctica consiste en la implementación de una lista dinámica en la cual se implementan lo típico de un TDA Lista lo que viene siendo insertar en orden como si fuera una lista consultar elemento remove etc. En esta practica se logro hacer todo lo pedido para esta, además de poder realizar la función de localizar posición por elemento la cual me costó más, pero si se logró.

También aunque no se implementaron las funciones anterior y siguiente creo que es una función fácil de comprender y de hacer por lo que me parece bien que se hayan saltado esas 2 funciones que en si no existe una utilidad buena para ellos.

Pantallazos



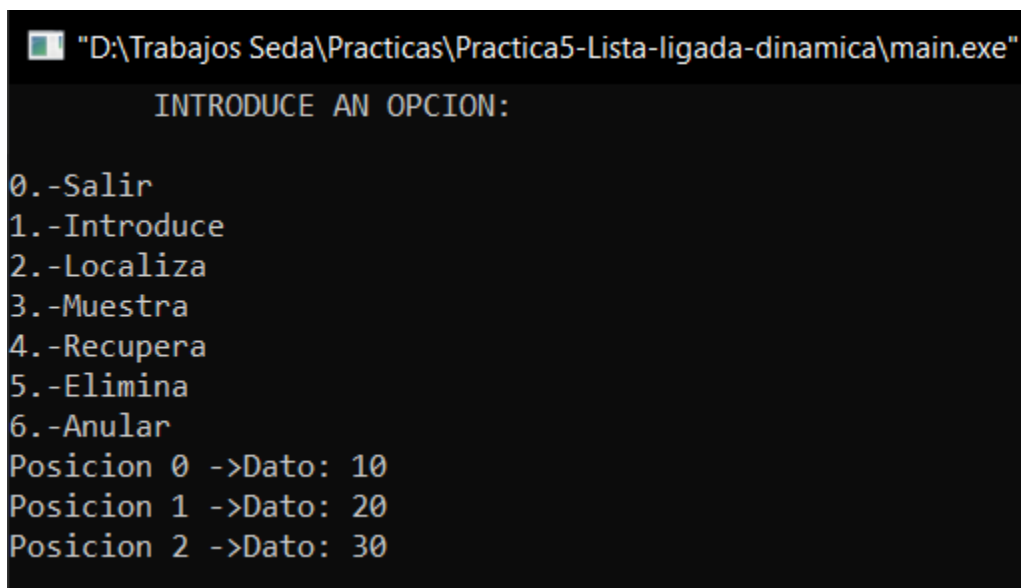
```
"D:\Trabajos Seda\Practicas\Practica5-Lista-ligada-dinamica\main.exe"

INTRODUCE AN OPCION:

0.-Salir
1.-Introduce
2.-Localiza
3.-Muestra
4.-Recupera
5.-Elimina
6.-Anular

The list is empty
```

Aquí se muestra el menú



```
"D:\Trabajos Seda\Practicas\Practica5-Lista-ligada-dinamica\main.exe"

INTRODUCE AN OPCION:

0.-Salir
1.-Introduce
2.-Localiza
3.-Muestra
4.-Recupera
5.-Elimina
6.-Anular
Posicion 0 ->Dato: 10
Posicion 1 ->Dato: 20
Posicion 2 ->Dato: 30
```

Aquí se aprecia que se introducen bien los datos y funciona bien la función mostrar

```
"D:\Trabajos Seda\Practicas\Practica5-Lista-ligada-dinamica\main.exe"

INTRODUCE AN OPCION:

0.-Salir
1.-Introduce
2.-Localiza
3.-Muestra
4.-Recupera
5.-Elimina
6.-Anular
Posicion 0 ->Dato: 10
Posicion 1 ->Dato: 20
Posicion 2 ->Dato: 30
2

    Seek a node in the list

    Introduce the posicion of the node that you want to seek: 1
Posicion 1Dato: 20

Presione una tecla para continuar . . .
```

Aquí se localiza por posición el dato

```
"D:\Trabajos Seda\Practicas\Practica5-Lista-ligada-dinamica\main.exe"

INTRODUCE AN OPCION:

0.-Salir
1.-Introduce
2.-Localiza
3.-Muestra
4.-Recupera
5.-Elimina
6.-Anular
Posicion 0 ->Dato: 10
Posicion 1 ->Dato: 20
Posicion 2 ->Dato: 30
4

    Modify the list

    Introduce the dato of the node that you want to recuperate: 20
Posicion 1Dato: 20
```

Aquí se recupera la posición de forma lineal

```
"D:\Trabajos Seda\Practicas\Practica5-Lista-ligada-dinamica\main.exe"
INTRODUCE AN OPCION:
0.-Salir
1.-Introduce
2.-Localiza
3.-Muestra
4.-Recupera
5.-Elimina
6.-Anular
Posicion 0 ->Dato: 10
Posicion 2 ->Dato: 30
```

Aquí se elimina la posición 1 de la lista

Conclusión

Respecto a la realización del código concluyo que se logró bien la realización de este programa ya que a lo que se puede apreciar funciona de manera correcta como debería hacerlo una lista además de implementar bien sus funciones recalco mis conocimientos, y estoy abierto a posibles errores que pueda tener el programa al momento de que el profe la evalúe.

También se concluye que aunque no parezca que tiene una utilidad buena al momento de referirse a trabajo si lo piensas un rato le hayas muchas utilidades importantes tanto que pensándolo muchas aplicaciones y juegos realizan una lista implementada en estos mismos por lo cual espero poder aprender mas sobre las listas y los tipos de datos abstractos

Código fuente

```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include <string.h>
4
5  struct node{
6      int data;
7      struct node *next;
8      int dato;
9  };
10
11  struct Lista{
12  void inicializa();
13  int inserta();
14  void Localiza();
15  void Imprime();
16  void Recupera();
17  void eliminateNode();
18  void anular();
19  node *first;
20  node *last;
21  Lista(){
22  inicializa();
23  }
24  };
25
26  void Lista::inicializa(){
27      first = nullptr;
28      last = nullptr;
```

```
29  };
30
31  Lista mi_lista;
32  int main()
33  {
34      int opc;
35      do{
36          printf("\tINTRODUCE AN OPCION:\n");
37          printf("\n0.-Salir\n1.-Introduce\n2.-Localiza\n3.-Muestra\n4.-
38  Recupera\n5.-Elimina\n6.-Anular\n");
39          mi_lista.Imprime();
40          scanf("%d", &opc);
41
42          switch(opc){
43              case 0: printf("\tAdios \n");
44                      break;
45              case 1: printf("\tInsert a node in the list\n\n");
46                      mi_lista.inserta();
47                      break;
48              case 2: printf("\tSeek a node in the list\n\n");
49                      mi_lista.Localiza();
50                      system("pause");
51                      break;
52              case 3: printf("\tShow the list\n\n");
53                      mi_lista.Imprime();
54                      system("pause");
55                      break;
56              case 4: printf("\tRecuperate the list\n\n");
57                      mi_lista.Recupera();
58                      system("pause");
```



```

59             break;
60         case 5: printf("\tEliminate node\n\n");
61             mi_lista.eliminateNode();
62             system("pause");
63             break;
64         case 6: printf("\tAnulate List\n\n");
65             mi_lista.anular();
66             system("pause");
67             break;
68         default: printf("\tChoose a correct value");
69             break;
70     }
71     system("cls");
72     }while(opc!=0);
73
74
75     return 0;
76 }
77
78 int Lista::inserta()
79 {
80     node *new_n = (node *) malloc(sizeof(node));
81     if(!new_n){// new_==NULL
82         printf("Memory allocation error, new node could not be created");
83         return -1;
84     }
85
86     printf("INTRODUCE THE VALUE OF THE NEW NODE: ");
87     scanf("%d", &new_n -> data);
88     //se guardara en el nuevo nodo en su propiedad dato

```

```
89
90     if(first == NULL){
91         first = new_n;
92         first ->next = NULL;
93         last = new_n;
94     }
95     else{
96         last -> next = new_n;
97         new_n -> next = NULL;
98         last = new_n;
99     }
100
101     printf("Dato: ");
102     scanf("%d",&new_n->dato);
103     printf("\nTHE NODE HAVE BEEN INTRODUCE CORRECTLY\n\n");
104     system("pause");
105     return 0;
106 }
107
108 void Lista::Imprime(){
109     node *temp = (node *) malloc(sizeof(node));
110     temp = first;
111     if (first != NULL){
112         while (temp != NULL){
113             printf("Posicion %d ->", temp->data);
114             printf("Dato: %i\n", temp->dato);
115             temp = temp ->next;
116         }
117     }
118     else{
```

```

119         printf("\n The list is empty\n");
120     }
121 }
122
123 void Lista::eliminateNode()
124 {
125     node *actual = (node *) malloc(sizeof(node));
126     actual = first;
127
128     node* before = (node *) malloc(sizeof(node));
129     before = NULL;
130
131     int soughtnode = 0, found = 0 ;
132
133     printf(" Introduce the posicion of the node that you want to eliminate:
134 ");
135     scanf("%d", &soughtnode);
136
137     if(first != NULL){
138         while(actual != NULL && found != 1){
139
140             if(actual -> data == soughtnode){
141
142                 if(actual == first){
143                     first = first ->next;
144                 }
145                 else{
146                     before -> next = actual -> next;
147                 }
148                 printf("\nThe link of the node have been eliminated");

```

[illegible]

```

179             printf("Dato: %i\n\n", temp->dato);
180             found = 1;
181         }
182         temp = temp ->next;
183     }
184     if (found == 0){
185         printf("El nodo no fue encontrado");
186     }
187 }
188 else{
189     printf("\n The list is empty\n");
190 }
191 }
192
193 void Lista::Recupera(){
194     node *temp = (node *) malloc(sizeof(node));
195     temp = first;
196     int soughtnode = 0, found = 0 ;
197
198     printf(" Introduce the dato of the node that you want to recuperate: ");
199     scanf("%d", &soughtnode);
200
201     if (first != NULL){
202         while (temp != NULL && found != 1){
203             if (temp -> dato == soughtnode){
204                 printf("Posicion %d", temp->data);
205                 printf("Dato: %i\n\n", temp->dato);
206                 found = 1;
207             }
208             temp = temp ->next;

```

```
209         }
210         if (found == 0){
211             printf("El nodo no fue encontrado");
212         }
213     }
214     else{
215         printf("\n The list is empty\n");
216     }
217 }
218
219 void Lista::anular()
220 {
221     node *actual = (node *) malloc(sizeof(node));
222     actual = first;
223
224     node* before = (node *) malloc(sizeof(node));
225     before = NULL;
226
227     if(first != NULL){
228         while(actual != NULL){
229
230             if(actual == first){
231                 first = first ->next;
232             }
233             else{
234                 before -> next = actual -> next;
235             }
236             before = actual;
237             actual = actual ->next;
238         }
```

```
239         free(before);
240         printf("\n\nNODE ELIMINATED SUCCESSFUL");
241     }
242     else{
243         printf("\nTHE LIST IS EMPTY\n\n");
244     }
245     printf("\n");
246 }
```