



UNIVERSIDAD DE GUADALAJARA

CUCEI

DIVISIÓN DE TECNOLOGÍAS PARA LA INTEGRACIÓN CIBER-HUMANA

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

PRÁCTICA No. 5

TEMA: Listas Simplemente Ligadas

EQUIPO No. 4

INTEGRANTES:

Munguía Guízar Marlon Uriel

Padilla Perez Jorge Daray

Pérez Palacios Perla Michelle

Pulido Tobías Rafael Agustín

SEMINARIO DE SOLUCIÓN DE PROBLEMAS DE ESTRUCTURAS DE DATOS

II | SECCIÓN D19 | 2022B

PROF. Mariscal Lugo Luis Felipe

MARCO TEÓRICO

FUNDAMENTOS TEÓRICOS DE LISTAS ENLAZADAS

Las estructuras lineales de elementos homogéneos (listas, tablas, vectores) necesitan ser implementados en memoria dinámica, ya que, si se implementan con arrays, estos necesitan fijar por adelantado el espacio que ocuparan en memoria, por ende, las hace ineficientes en este tipo de estructuras.

Gracias a esta asignación dinámica, con el uso de apuntadores y variables apuntadas en ejecución haga que coincida la memoria física utilizada en este.

Una lista enlazada es una colección o secuencia de elementos dispuestos uno detrás de otro, conectados de tal forma que el primer nodo (que contiene los campos con la información a guardar) conecta al siguiente (“enlace”) y este al siguiente y así hasta que el último nodo apunte a nulo.

Este “enlace” está representado por una ‘→’ lo que facilita la comprensión de la conexión entre los nodos indicando así la dirección de memoria del siguiente nodo desde el primer elemento (nodo 1) al último elemento (nodo n).

Las listas se pueden dividir en 4 categorías:

- Listas simplemente enlazadas. Cada nodo (elemento) contiene un único enlace que conecta ese nodo al nodo siguiente o nodo sucesor. La lista es eficiente en recorridos directos (“adelante”).
- Listas doblemente enlazadas. Cada nodo contiene dos enlaces, uno a su nodo predecesor y el otro a su nodo sucesor. La lista es eficiente tanto en recorrido directo (“adelante”) como en recorrido inverso (“atrás”).
- Lista circular simplemente enlazada. Una lista simplemente enlazada en la que el último elemento (cola) se enlaza al primer elemento (cabeza) de tal modo que la lista puede ser recorrida de modo circular (“en anillo”).
- Lista circular doblemente enlazada. Una lista doblemente enlazada en la que el último elemento se enlaza al primer elemento y viceversa. Esta lista se puede recorrer de modo circular (en anillo) tanto en dirección directa (“adelante”) como inversa (“atrás”).

La implementación de cada uno de los cuatro tipos de estructuras de listas se puede desarrollar utilizando punteros.

El primer nodo, frente, de una lista es el nodo apuntado por cabeza. La lista encadena nodos juntos desde el frente al final (cola) de la lista. El final se identifica como el nodo cuyo campo enlace tiene el valor NULL. La lista se recorre desde el primero al último nodo; en cualquier punto del recorrido la posición actual se referencia por el puntero actual. Una lista vacía, es decir, que no contiene nodos se representa con el puntero cabeza a nulo.

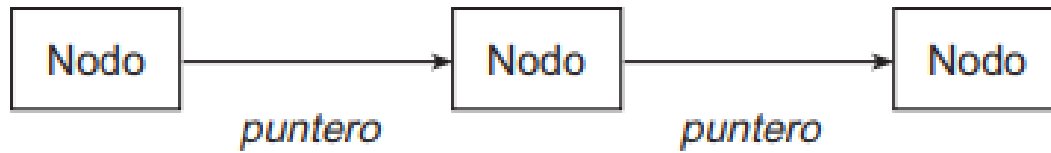
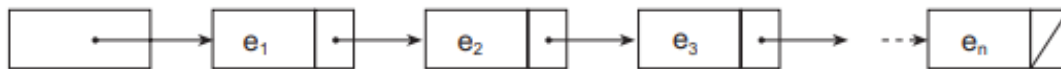


Figura 10.1. Lista enlazada (representación simple).



e_1, e_2, \dots, e_n son valores del tipo `TipoElemento`

Figura 10.2. Lista enlazada (representación gráfica típica).

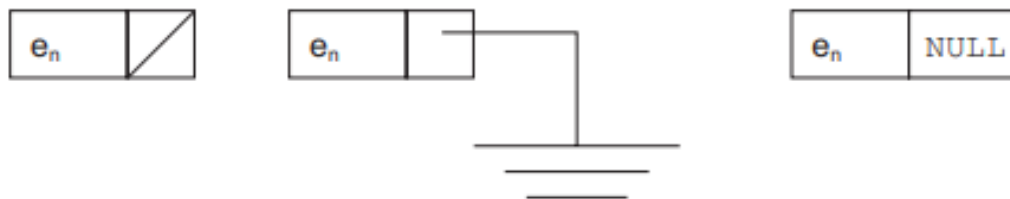


Figura 10.3. Diferentes representaciones gráficas del nodo último.

TIPO ABSTRACTO DE DATOS Lista

Una lista almacena información del mismo tipo, ordenada (cada elemento contiene la dirección del siguiente elemento), con un número indeterminado de elementos, dinámica (el número de nodos puede variar debido a un proceso insertando o eliminando nodos), Las inserciones se pueden realizar en cualquier punto de la lista, así como las eliminaciones. Cada elemento es un nodo de la lista

Para formalizar el Tipo Abstracto de Dato Lista a partir de la noción matemática, se define un conjunto de operaciones básicas con objetos de tipo Lista. Las operaciones:

Inicializa, Vacía, Insertar, Localizar, Eliminar, Anterior, Primero, y Anular

Estas operaciones son las básicas para manejar listas. En realidad, la decisión de qué operaciones son las básicas depende de las características de la aplicación que se va a realizar con los datos de la lista También dependerá del tipo de representación elegido para las listas.

Bibliografía.

Joyanes Aguilar, L., Sanchez Garcia, L. & Zahonero Martinez, I. (2007).
ESTRUCTURA DE DATOS EN C++. McGraw-Hill.

Código:

```
1  /*Equipo 4:
2     Munguia Guizar Marlon Uriel
3     Padilla Perez Jorge Daray
4     Perez Palacios Perla Michelle
5     Pulido Tobias Rafael Agustin
6     Seccion: D19
7     Calendario: 2022B
8  */
9  #include <iostream>
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <string.h>
13 #include <cstring>
14 #define TAMLISTA 20
15
16 using namespace std;
17
18 void menu();
19
20 struct Alumno
21 {
22     char nombre[30];
23     unsigned int edad;
24     char sexo[1];
25     unsigned int bandera;
26     //struct Alumno *next;
27 }datos[TAMLISTA];
28
29 struct Lista
30 {
31     Alumno datos[TAMLISTA];
32     void inicializa();
33     bool vacia();
34     bool llena();
35     void insertar(int pos);
36     void elimina(int pos);
37     int ultimo;
38     int primero;
39     int guarda_m,guarda_h;
40     void imprimir();
41     void anular();
42     void burbuja_mejorada();
43     bool lista_ordenada ();
44     Lista()//Constructor
45     {
46         inicializa();
47     }
48 };
49
50 //Inicializa la lista con sus variables
51 void Lista::inicializa()
52 {
53     ultimo = -1;
54     primero = 0;
55     guarda_h = 1,guarda_m = 0;
```

```

56  }
57
58  //Verifica si la lista esta vacia
59  bool Lista::vacía()
60  {
61      return ultimo == -1;
62  }
63
64  //Verifica si la lista esta llena
65  bool Lista::llena()
66  {
67      return ultimo == TAMLISTA - 1;
68  }
69
70  //Elimina toda la lista
71  void Lista::anular()
72  {
73      ultimo = -1;
74  }
75
76  //Inserta alumno por posicion
77  void Lista::insertar(int pos){
78      if (llena() || pos < 0 || pos > ultimo + 1)
79      {
80          cout<<"Ingresa un elemento consecutivo valido"<<endl;
81          return;
82      }
83      char cadena_h[TAMLISTA]{'M'};
84      char cadena_m[TAMLISTA]{'F'};
85      //Recorre la posicion de la lista si ya existe para no
sobrecribir
86      for(int i = ultimo+1 ; i > pos ; i-- )
87      {
88          datos[i] = datos[i - 1];
89      }
90      cout << "Ingresa el nombre: " <<endl;
91      cin >> datos[pos].nombre;
92      cout << "Ingresa la edad: " <<endl;
93      cin >> datos[pos].edad;
94      cout << "Ingresa el sexo 'f' para femenino o 'm' para masculino:
"<<endl;
95      cin >> datos[pos].sexo;
96      //0x5F Quita el 32 a nivel bits para hacer mayusculas
97      if (*datos[pos].sexo = ( *datos[pos].sexo >= 'a' &&
*datos[pos].sexo <= 'z')? *datos[pos].sexo & 0x5F :
*datos[pos].sexo)//cadena de entrada rango, entonces hace un if corto
98      ;
99      //Compara si sexo es masculino o femenino
100     if ( strcmp(datos[pos].sexo, cadena_m) ==0 )
101     {
102         datos[pos].bandera = guarda_m;
103         guarda_m = guarda_m + 2;
104     }
105     else if ( strcmp(datos[pos].sexo, cadena_h) ==0 )
106     {
107         datos[pos].bandera = guarda_h;
108         guarda_h = guarda_h + 2;

```

```

109     }
110     else
111     {
112         cout<<"Ingresa un elemento valido"<<endl;
113         return;
114     }
115     ultimo++;
116 }
117
118
119 void Lista::elimina(int pos)
120 {
121     if (vacía() || pos < 0 || pos > ultimo ) {
122         cout<<"La lista esta vacía"<<endl;
123         return;
124     }
125
126     for (int i = pos ; i <= ultimo ; i++){
127         datos[i] = datos[i + 1];
128     }
129     ultimo--;
130 }
131
132 void Lista::imprimir()
133 {
134     if (vacía()){
135         cout<<"La lista esta vacía"<<endl;
136         return;
137     }
138     for(int i = primero ; i <= ultimo ; i++)
139     {
140         cout << "Posicion numero: "<<i+1<<endl;
141         cout << "nombre: "<<datos[i].nombre<<endl;
142         cout << "edad: "<<datos[i].edad<<endl;
143         cout << "sexo: "<<datos[i].sexo<<endl;
144     }
145 }
146
147 bool Lista::lista_ordenada(){
148     int j = primero, i = ultimo;
149     while (j < i){
150         if (datos[j].bandera > datos[j+1].bandera){
151             return 0;
152         }
153         j++;
154     }
155     i--;
156     return 1
157 ;
158 }
159
160 void Lista::burbuja_mejorada()
161 {
162     if (vacía()){
163         cout<<"La lista esta vacía"<<endl;
164         return;
165     }

```

```


166     if(lista_ordenada())
167     {
168         cout<<"La lista esta intercalada, no hace falta
intercalar"<<endl;
169         return;
170     }
171     int i = ultimo, j;
172     Alumno mi_alumno;
173     bool cambio;
174     do{
175         cambio = false;
176         j = 0;
177         while (j < i){
178             if (datos[j].bandera > datos[j+1].bandera){
179                 mi_alumno = datos[j];
180                 datos[j] = datos[j+1];
181                 datos[j+1] = mi_alumno;
182                 cambio = true;
183             }
184             j++;
185         }
186         i--;
187     }while(cambio);
188     cout << "Alumnos intercalados correctamente"<<endl;
189 }
190
191
192 int main()
193 {
194     menu();
195     return 0;
196 }
197
198 struct Lista mi_lista;
199 void menu(){
200     FILE *archivo;//Creacion de archivo logico
201     archivo = fopen("alumnos.dat","w+");//Creacion de archivo fisico
a traves del archivo logico en modo escritura y lectura
202     int i=0;
203     int opc=0;
204     while(opc != 4)
205     {
206         system("cls");
207         int pos;
208         char cambio[5];
209         cout<<" MenÃº - Sistema Control Escolar"<<endl;
210         cout<<"1) Insertar alumnos"<<endl;
211         cout<<"2) Mostrar todos los alumnos"<<endl;
212         cout<<"3) Transformar la lista original
(Intercalar)"<<endl;
213         cout<<"4) Salir"<<endl;
214         cout<<"Seleccione opcion:"<<endl;
215         cin>>opc;
216         switch (opc)
217         {
218             case 1:
219                 {

```



```
220             system("cls");
221             fflush(stdin);
222             cout<<"En que posicion desea insertar el
alumno?, Inicio en 1."<<endl;
223             gets(cambio);
224             pos = atoi(cambio); //atoi convierte de char* a
int, si no hay un entero lo pasa a 0 por default.
225             pos = pos-1;
226             mi_lista.insertar(pos);
227             i++;
228             system("pause");
229         }
230         break;
231         case 2:
232         {
233             system("cls");
234             mi_lista.imprimir();
235             system("pause");
236         }break;
237         case 3:
238         {
239             system("cls");
240             mi_lista.burbuja_mejorada();
241             system("pause");
242         }break;
243         case 4:break;
244
245         default:
246             cout<<"La opcion: "<<opc<<"No existe"<<endl;
247     }
248 }
249
250     if (archivo){
251         fwrite(reinterpret_cast <const char*>
(&mi_lista), sizeof(struct Alumno), i, archivo);
252         fclose(archivo);
253     }
254 }
```


imágenes del archivo físico:

 alumnos.dat: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
Alison		F		Emmanuel

 alumnos.dat: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
	M	Sabina	F	Hazhiel

 alumnos.dat: Bloc de notas

Archivo	Edición	Formato	Ver	Ayuda
Hazhiel		M		

Capturas:

- Insertar dos alumnas y luego dos alumnos (nombre = "Alison", edad = 21, sexo = 'F', nombre = "Sabina", edad = 20, sexo = 'F', nombre = "Emmanuel", edad = 19, sexo = 'M', nombre = "Hazhiel", edad = 18, sexo = 'M') opción (1):

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
En que posicion desea insertar el alumno?, Inicio en 1.
1
Ingresa el nombre:
Alison
Ingresa la edad:
21
Ingresa el sexo 'f' para femenino o 'm' para masculino:
f
Presione una tecla para continuar . . .
```

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
En que posicion desea insertar el alumno?, Inicio en 1.
2
Ingresa el nombre:
Sabina
Ingresa la edad:
20
Ingresa el sexo 'f' para femenino o 'm' para masculino:
F
Presione una tecla para continuar . . .
```

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
En que posicion desea insertar el alumno?, Inicio en 1.
3
Ingresa el nombre:
Emmanuel
Ingresa la edad:
19
Ingresa el sexo 'f' para femenino o 'm' para masculino:
m
Presione una tecla para continuar . . .
```

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
En que posicion desea insertar el alumno?, Inicio en 1.
4
Ingresa el nombre:
Hazhiel
Ingresa la edad:
18
Ingresa el sexo 'f' para femenino o 'm' para masculino:
M
Presione una tecla para continuar . . .
```

- Mostrar la lista opción (2):

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
Posicion numero: 1
nombre: Alison
edad: 21
sexo: F
Posicion numero: 2
nombre: Sabina
edad: 20
sexo: F
Posicion numero: 3
nombre: Emmanuel
edad: 19
sexo: M
Posicion numero: 4
nombre: Hazhiel
edad: 18
sexo: M
Presione una tecla para continuar . . .
```

- Elegir la transformación (intercalación)

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
Alumnos intercalados correctamente
Presione una tecla para continuar . . .
```

- Mostrar la lista

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
Posicion numero: 1
nombre: Alison
edad: 21
sexo: F
Posicion numero: 2
nombre: Emmanuel
edad: 19
sexo: M
Posicion numero: 3
nombre: Sabina
edad: 20
sexo: F
Posicion numero: 4
nombre: Hazhiel
edad: 18
sexo: M
Presione una tecla para continuar . . .
```

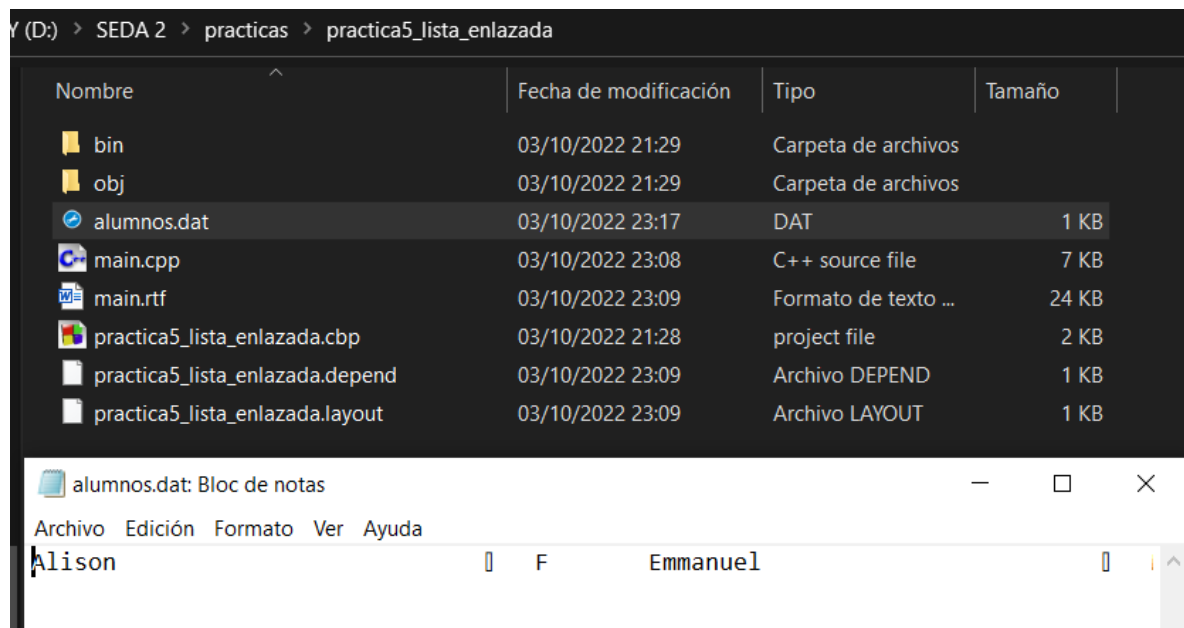
- Elegir salir del programa (se respalda el contenido de la lista)

```
"D:\SEDA 2\practicas\practica5_lista_enlazada\bin\Debug\practica5_lista_enlazada.exe"
Men. - Sistema Control Escolar
1) Insertar alumnos
2) Mostrar todos los alumnos
3) Transformar la lista original (Intercalar)
4) Salir
Seleccione opcion:
4

Process returned 0 (0x0)   execution time : 210.196 s
Press any key to continue.
```

Equipo 4 | Practica 5: Listas Simplemente Enlazadas.

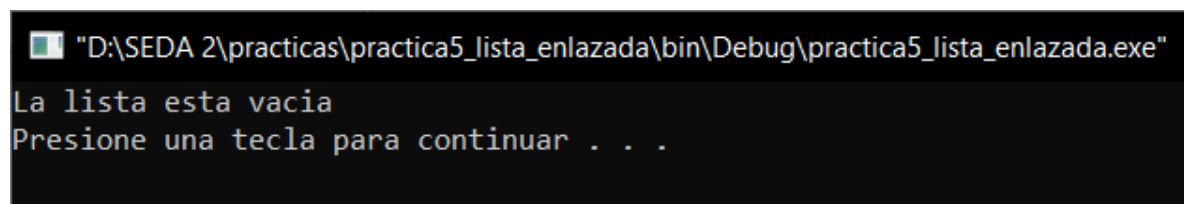
- Abrir el archivo físico para la toma de la captura de pantalla



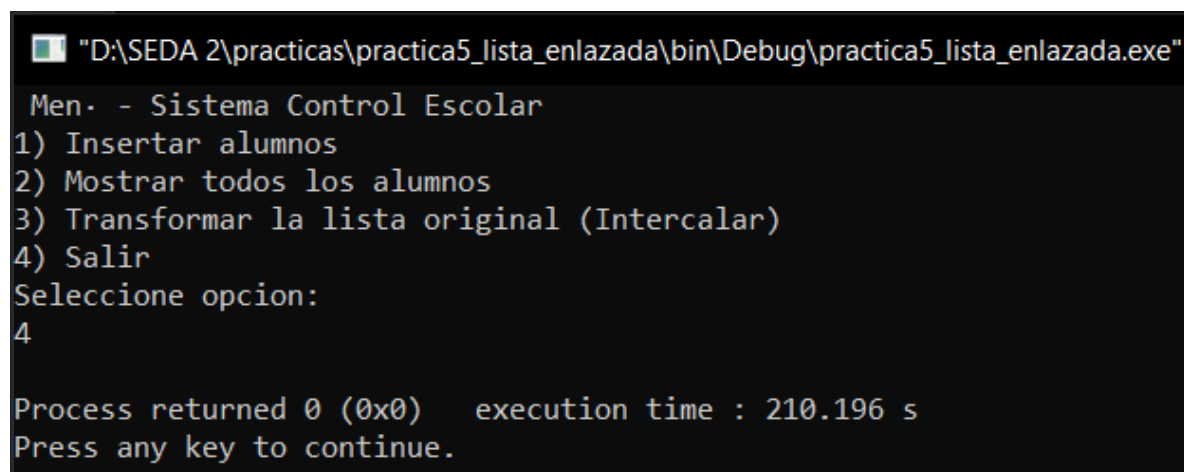
- Volver a ejecutar el programa (se recupera el contenido del archivo físico y se carga a la lista)

```
FILE *archivo;//Creacion de archivo logico
archivo = fopen("alumnos.dat","w+");//Creacion de archivo fisico a traves del archivo logico en modo escritura y lectura
```

- Mostrar la lista



- Salir



Conclusiones:

- **Munguía Guízar Marlon Uriel.**
 - Esta quinta práctica fue bastante entretenida, ya que retoma el tema de listas que ya llegué a ver en semestres anteriores, pero con su enfoque distintivo, ya que en esta ocasión se utiliza el flujo de datos a archivos físicos, todo esto le dio un punto interesante bastante interesante a el trabajo
- **Padilla Perez Jorge Daray.**
 - En conclusión, la practica 5 me hizo recordar el semestre pasado, con la diferencia de la implementación del almacenamiento en archivos, y con una función nueva que seria el intercalar a los alumnos, en lo cual es necesario pensar de una forma diferente para resolverlo a como era el semestre anterior.
- **Pérez Palacios Perla Michelle.**
 - En resumen, pudimos completar esta práctica gracias a los ejercicios anteriores que hemos estado trabajando sobre archivos, pero ahora insertando la información por medio de una lista. Decidimos utilizar estructuras para esta práctica ya que estábamos más familiarizados con el tipo de programación.
- **Pulido Tobías Rafael Agustín.**
 - En esta práctica la elaboración de la lista retoma lo anteriormente visto en semestres anteriores, con la diferencia de que aquí se almacenan los datos en archivos lo cual lo hace más eficiente y eficaz.