UNIVERSIDAD DE GUADALAJARA

CUCEI

DIVISIÓN DE TECNOLOGÍAS PARA LA INTEGRACIÓN CIBER-HUMANA

DEPARTAMENTO DE CIENCIAS COMPUTACIONALES

PRÁCTICA No. 6

TEMA: Pilas (utilizando una Lista Simplemente Ligada)

EQUIPO No. 4

INTEGRANTES:

Munguía Guizar Marlon Uriel

Padilla Perez Jorge Daray

Pérez Palacios Perla Michelle

Pulido Tobias Rafael Agustín

SEMINARIO DE SOLUCIÓN DE PROBLEMAS DE ESTRUCTURAS DE DATOS II | SECCIÓN D19 | 2022B

PROF. Mariscal Lugo Luis Felipe

MARCO TEÓRICO

11.3 PILA GENERICA CON LISTAS ENLAZADA

La realización dinámica de una pila utilizando una lista enlazada almacena cada elemento de la pila como un nodo de la lista. Como las operaciones en el TAD Pila se realizan por el mismo extremo, las acciones correspondientes con la lista se realizarán siempre por em mismo extremo de la lista.

Tiene la ventaja de que el tamaño se ajusta al numero de elementos de la pila. Sin embargo, es necesaria más memoria para guardar el campo de enlace entre nodos consecutivos.

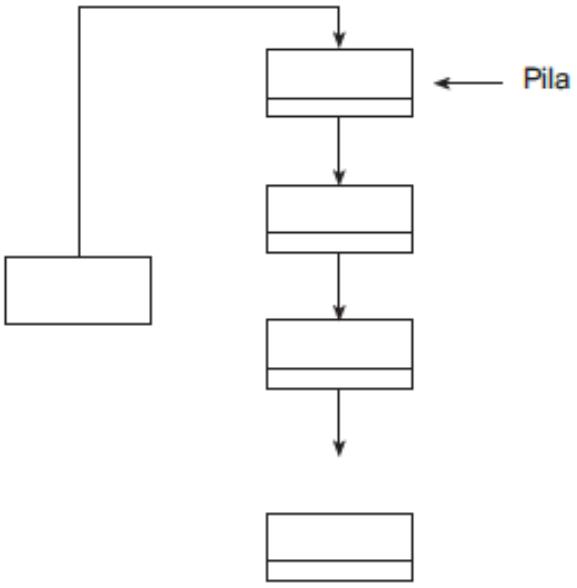


Figura 11.5. Representación de una pila con una lista enlazada.

11.3.1 Clase PilaGenerica y NodoPila

Los elementos de la pila son los nodos de la lista, con un atributo para guardar el elemento y otro de enlace. Las operaciones del tipo pila implementada con listas son las mismas, salvo la operación que controla si la pila está llena, ahora no tiene significado ya que el único límite es la memoria.

El tipo de dato de elemento se corresponde con el tipo de los elementos de la pila para que no dependa de un tipo concreto. La clase NodoPila representa un nodo de la lista enlazada, tiene dos atributos: elemento, guarda el elemento de la pila y siguiente, contiene la dirección del siguiente nodo de la lista.

11.3.2 Implementación de las operaciones del TAD Pila con listas enlazadas

El constructor de Pila inicializa a ésta como pila vacía (cima == NULL), realmente, a la condición de lista vacía. Las operaciones insertar, quitar y cimaPila acceden a la lista directamente con el puntero cima. Entonces, como no necesitan recorren los nodos de la lista, no dependen del número de nodos, la eficiencia de cada operación es constante, O(1).

Al crear una instancia de pila es cuando se informa del tipo concreto de sus elementos.

Bibliografía

Aguilar, L. J., Garcia, L. S., & Martinez, I. Z. (2007). *Estructura de Datos en C++.* Madrid: McGraw-Hill.

CÓDIGO FUENTE:

Main:

```
/*Equipo 4:
   Munguia Guizar Marlon Uriel
   Padilla Perez Jorge Daray
   Perez Palacios Perla Michelle
   Pulido Tobias Rafael Agustin
   Seccion: D19
   Calendario: 2022B
   */
#include <iostream>
#include<iomanip>
#include<fstream>
#include<cstdlib>
#include "Libro.h"
#include "Pila.h"
#include "Nodo.h"

using std::fstream;

  using namespace std;
  void menu();
  void altas(fstream&);
  void bajas(fstream&, fstream&);
  void cambios(fstream&, fstream&);
  void consultas(fstream&,int);
  void imprimirLinea( ostream &salida, const Libro &registro );
  void recuperar(fstream&);
  Libro d;//un objeto o registro Libro
  Libro blanco;//un objeto o registro Libro
  Pila mi_pila;//un objeto o registro Pila

int main()
{
    menu();
    return 0;
}

int numeroDeSocio = 1;
void altas(fstream&archDeportSalida )
{
    system("cls");
    char nombre[15];
    char autor[15];
    char editorial[15];
    //Pone el puntero al inicio del archivo
    archDeportSalida.clear();
    archDeportSalida.seekg(0, archDeportSalida.beg);
        d.establecerposicion(numeroDeSocio);
            cout<<"Teclea nombre, autor y editorial\n?";
            cin>>setw(15)>>nombre;
            cin>>setw(15)>>autor;
            cin>>setw(15)>>editorial;
            //establecer los valores nombre, autor y editorial del
registro
```

```
 53              d.establecernombre(nombre);
 54              d.estableceraautor(autor);
 55              d.establecereditorial(editorial);
 56              mi_pila.push(d);
 57              //buscar la posicion en el archivo de registro
especificado por el usuario
 58              archDeportSalida.seekp((d.obtenerposicion()-
1)*sizeof(Libro)); //seekp(n, ios::beg);
 59              //escribir la informaciÃ³n especificada por el usuario
en el archivo
 60              archDeportSalida.write(reinterpret_cast<const char
*>(&d),sizeof(Libro));
 61              numeroDeSocio++;
 62    archDeportSalida.clear();
 63  }//main
 64
 65  void bajas(fstream&archDeportSalida, fstream&archdeportEntrada)
 66  {
 67      if (numeroDeSocio == 1)
 68      {
 69          cout << "Lista vacia"<<endl;
 70          system("pause");
 71          return;
 72      }
 73      //Pone el puntero al inicio
 74      archdeportEntrada.clear();
 75      archdeportEntrada.seekg(0, archdeportEntrada.beg);
 76      int elim;
 77          d.establecerposicion(numeroDeSocio-1);
 78          //Busca la posicion dada del registro
 79          archdeportEntrada.seekg((d.obtenerposicion()-
1)*sizeof(Libro),ios::beg); //seekp(n, ios::beg);
 80          // lee el siguiente registro del archivo
 81          archdeportEntrada.read( reinterpret_cast< char * >( &d ),
sizeof( Libro ) );
 82          if ( d.obtenerposicion() != 0)
 83              {
 84                  cout << left << setw( 15 ) << "Nombre" << setw( 15 )
 85                  << "Autor" << setw( 15 ) << "Editorial"<< endl;
 86                  imprimirLinea( cout, d);
 87                  cout << "Desea eliminar el socio ? '1' si, '0' no"
<< endl;
 88                  cin >> elim;
 89                  if (elim == 1)
 90                  {
 91                      //Pone el puntero al inicio
 92                      archDeportSalida.clear();
 93                      archDeportSalida.seekp(0, archDeportSalida.beg);
 94                      d.establecerposicion(numeroDeSocio);
 95                      //buscar la posicion en el archivo de registro
especificado por el usuario
 96                      archDeportSalida.seekp((d.obtenerposicion()-
1)*sizeof(Libro)); //seekp(n, ios::beg);
 97                      //escribir la informaciÃ³n especificada por el
usuario en el archivo
 98                      archDeportSalida.write(reinterpret_cast<const
char *>(&blanco),sizeof(Libro));
```

```
 99                     cout << "Borrado corectamente" << endl;
100                     mi_pila.pop();
101                     numeroDeSocio--;
102                     system("pause");
103                 }
104                 else
105                 {
106                     cout << "Regresando al menu..." << endl;
107                     system("pause");
108                 }
109             }
110         else
111             {
112                 cout << "EL NUMERO DE SOCIO NO EXISTE"<<endl;
113                 system("pause");
114             }
115     archdeportEntrada.clear();
116 }
117
118 void consultas(fstream&archdeportEntrada, int opc)
119 {
120     //Pone el puntero al inicio
121         archdeportEntrada.clear();
122         archdeportEntrada.seekg(0, archdeportEntrada.beg);
123         switch(opc)
124         {
125         case 4:
126             {
127                 system("cls");
128                 cout << "Consulta individual" << endl;
129                 int numero;
130                 cout << "Dame el numero de socio: "<<endl;
131                 cin >> numero;
132                 d.establecerposicion(numero);
133                 archdeportEntrada.seekg((d.obtenerposicion()-
1)*sizeof(Libro),ios::beg); //seekp(n, ios::beg);
134                 // lee el siguiente registro del archivo
135                     archdeportEntrada.read( reinterpret_cast< char *
>( &d ), sizeof( Libro ) );
136                 if ( d.obtenerposicion() != 0)
137                     {
138                         cout << left << setw( 15 ) << "Nombre" <<
setw( 15 )
139                         << "Autor" << setw( 15 ) << "Editorial"<<
endl;
140                         imprimirLinea( cout, d);
141                     }
142                 else
143                     {
144                         cout << "EL NUMERO DE SOCIO NO
EXISTE"<<endl;
145                     }
146                 system("pause");
147                 break;
148             }
149         case 3:
150             {
```

```
151                     system("cls");
152                     mi_pila.mostrar();
153                     system("pause");
154                 }
155                 break;
156             }
157         archdeportEntrada.clear();
158     } // fin de consultas
159 
160         // muestra un solo registro
161         void imprimirLinea( ostream &salida, const Libro &registro )
162         {
163         salida << left << setw( 15 ) << registro.obtenernombre()
164         << setw( 15 ) << registro.obtenerautor()
165         << setw( 15 ) << registro.obtenereditorial()<<endl;
166         } // fin de la funciÃ³n imprimirLinea
167 
168     void recuperar(fstream&archlibro)
169     {
170         system("cls");
171         // lee el primer registro del archivo
172         archlibro.read( reinterpret_cast< char * >( &d ), sizeof( Libro
) );
173         // lee todos los registros del archivo
174         while ( archlibro && !archlibro.eof() )
175         {
176             // muestra un registro
177             if ( d.obtenerposicion() != 0)
178             {
179                 mi_pila.push(d);
180                 numeroDeSocio++;
181             }
182             // lee el siguiente registro del archivo
183             archlibro.read( reinterpret_cast< char * >( &d ), sizeof(
Libro ) );
184         } // fin de while
185         system("pause");
186     }
187 
188     void menu()
189     {
190         fstream archDeportENSA("Libros.dat", ios::out | ios::binary |
ios::in );
191         if ( !archDeportENSA)
192         {
193             cerr << "No se pudo abrir el archivo ensa." << endl;
194             exit( 1 );
195         } // fin de if
196         recuperar(archDeportENSA);
197 
198         int opc;
199         do
200         {
201         system("cls");
202         cout << "MENU LIBRERIA CUCEI"<<endl;
203         cout << "1) Comprar libros(insertar)" << endl;
204         cout << "2) Vender libros(eliminar)" << endl;
```

```
205       cout << "3) Consultas generales" << endl;
206       cout << "4) Salir" << endl;
207       cout << "Ingresar opcion: "<<endl;
208       cin>> opc;
209       switch(opc)
210           {
211           case 1:
212               {
213                   altas(archDeportENSA);
214                   break;
215               }
216           case 2:
217               {
218                   bajas(archDeportENSA,archDeportENSA);
219                   break;
220               }
221           case 3:
222               {
223                   consultas(archDeportENSA,opc);
224                   break;
225               }
226           case 4:
227               {
228                   cout << "saliendo..."<<endl;
229                   archDeportENSA.close();
230                   exit(1);
231                   break;
232               }
233           }
234       }while(opc!=4);
235   }
236
```

Libro.h:

```
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  #ifndef LIBRO_H
10 #define LIBRO_H
11 #include <iostream>
12
13 using namespace std;
14
15 class Libro
16 {
17     public:
18         Libro(int = 0, string = "", string = "", string = "");
19         void establecerposicion(int);
20         int obtenerposicion() const;
21         void establecernombre(string);
22         string obtenernombre() const;
23         void establecerautor(string);
24         string obtenerautor() const;
25         void establecereditorial(string);
26         string obtenereditorial() const;
27         void imprimir_registro();
28         ~Libro();
29     private:
30         int posicion;
31         char nombre[15];
32         char autor[15];
33         char editorial[15];
34 };
35
36 #endif // LIBRO_H
37
```

Libro.cpp:

```
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 "Libro.h"
10 #include <iostream>
11 #include <iomanip>
12 #include <string.h>
13
14 using namespace std;
15
16 Libro::Libro(int valorPosicion, string valorNombre, string
valorAutor, string valorEditorial)
17 {
18     establecerposicion(valorPosicion);
19     establecernombre(valorNombre);
20     estableceautor(valorAutor);
21     estableceditorial(valorEditorial);
22 }//ctor
23
24 void Libro::establecerposicion(int valorPosicion)
25 {
26     posicion = valorPosicion;
27 }
28 int Libro::obtenerposicion()const
29 {
30     return posicion;
31 }
32 //Establece el valor del nombre
33 void Libro::establecernombre(string cadenaNombre)
34 {
35     const char*valorNombre = cadenaNombre.data();
36     int longitud = cadenaNombre.size();
37     longitud = (longitud < 15 ? longitud : 14);
38     strncpy(nombre, valorNombre, longitud);
39     nombre[longitud] = '\0';
40 }//fin void establecerNombre
41
42 string Libro::obtenernombre()const
43 {
44     return nombre;
45 }
46
47 //Establece el valor del deporte
48 void Libro::estableceautor(string cadenaAutor)
49 {
50     const char*valorAutor = cadenaAutor.data();
51     int longitud = cadenaAutor.size();
52     longitud = (longitud < 15 ? longitud : 14);
53     strncpy(autor, valorAutor, longitud);
54     autor[longitud] = '\0';
```

```
55  }//fin void establecerDeporte
56  string Libro::obtenerautor()const
57  {
58      return autor;
59  }
60
61  //Establece el valor del deporte
62  void Libro::establecereditorial(string cadenaEditorial)
63  {
64      const char*valorEditorial = cadenaEditorial.data();
65      int longitud = cadenaEditorial.size();
66      longitud = (longitud < 15 ? longitud : 14);
67      strncpy(editorial, valorEditorial, longitud);
68      editorial[longitud] = '\0';
69  }//fin void establecerDeporte
70  string Libro::obtenereditorial()const
71  {
72      return editorial;
73  }
74
75  void Libro::imprimir_registro()
76  {
77      cout << left << setw( 16 ) << nombre
78      << setw( 16 ) << autor << setw( 16 ) << right << fixed
79      << showpoint << editorial << endl;
80  }
81
82  Libro::~Libro()
83  {
84
85  }//dtor
```

Nodo.h:

```
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  #ifndef NODO_H
10 #define NODO_H
11 #include "Libro.h"
12 #include <iostream>
13
14 using namespace std;
15
16 class Nodo
17 {
18     public:
19         Nodo();
20         Libro Datos; //tipo objeto Datos de Libro
21         Nodo *next;//Apuntador a siguiente
22         ~Nodo();
23     private:
24 };
25
26 #endif // NODO_H
```

Nodo.cpp:

```
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 "Nodo.h"
10 #include "Libro.h"
11 #include <iostream>
12
13 using namespace std;
14
15 Nodo::Nodo()
16 {
17     Datos.establecernombre("");
18     Datos.establecerautor("");
19     Datos.establecereditorial("");
20     next = nullptr;
21 }//ctor
22
23 Nodo::~Nodo()
24 {
25
26 }//dtor
```

Pila.h:

```
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  #ifndef PILA_H
10 #define PILA_H
11 #include "Nodo.h"
12 #include "Libro.h"
13 #include <iostream>
14 #include <fstream>
15
16 using namespace std;
17
18 class Pila
19 {
20     public:
21         Pila();
22         void push(Libro);
23         void pop();
24         void mostrar();
25         ~Pila();
26     private:
27         Nodo *top;
28 };
29
30 #endif // PILA_H
```

Pila.cpp:

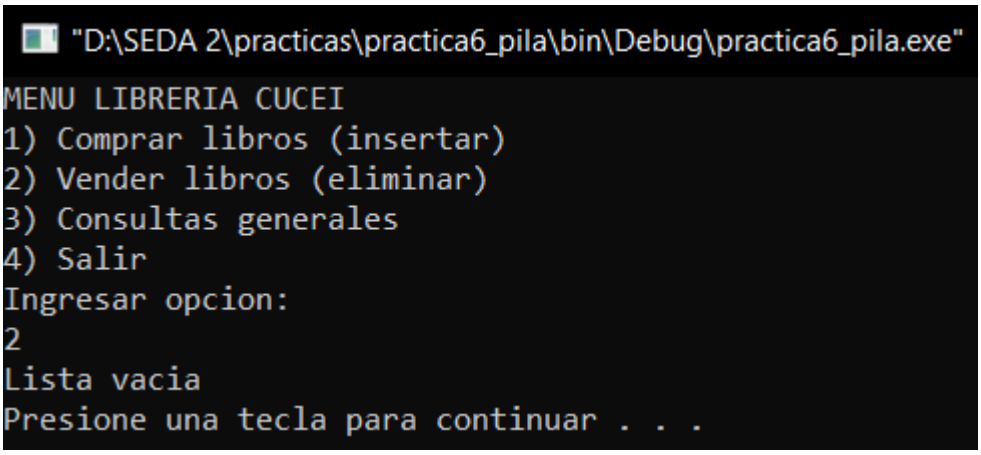
```
1  /*Equipo 4:
2      Munguia Guizar Marlon Uriel
3      Padilla Perez Jorge Daray
4      Perez Palacios Perla Michelle
5      Pulido Tobias Rafael Agustin
6      Seccion: D19
7      Calendario: 2022B
8      */
9  #include "Pila.h"
10 #include "Nodo.h"
11 #include "Libro.h"
12 #include <iostream>
13 #include <iomanip>
14
15 using namespace std;
16
17 void Pila::push(Libro book)
18 {
19     Nodo *temp = new Nodo();
20     temp->Datos = book;
21     if (top == nullptr)
22     {
23         top = temp;
24     }
25     else
26     {
27         temp ->next = top;
28         top = temp;
29     }
30     top = temp;
31 }
32
33 void Pila::pop()
34 {
35     Nodo *temp;
36     temp = top;
37     if(top != nullptr)
38     {
39         temp = top;
40         top = temp->next;
41         delete(temp);
42     }
43     if(top == nullptr)
44     {
45         cout << "Se elimino todo correctamente" <<endl;
46     }
47 }
48
49 void Pila::mostrar()
50 {
51     Nodo *temp;
52     temp = top;
53     cout << left << setw( 15 ) << "nombre"
54     << setw( 15 ) << "autor" << setw( 15 ) << right << fixed
55     << showpoint << "editorial" << endl;
```

```
56      while(temp)
57      {
58          temp ->Datos.imprimir_registro();
59          temp = temp ->next;
60      }
61  }
62
63  Pila::Pila()
64  {
65      top = nullptr;
66  }//ctor
67
68  Pila::~Pila()
69  {
70
71  }//dtor
```

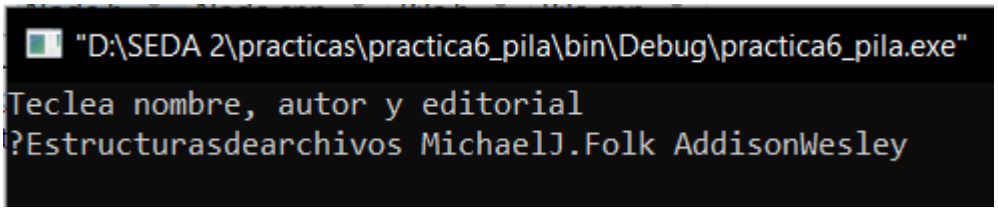
IMAGEN DEL ARCHIVO FÍSICO

IMÁGENES DE LA CORRIDA DEL PROGRAMA

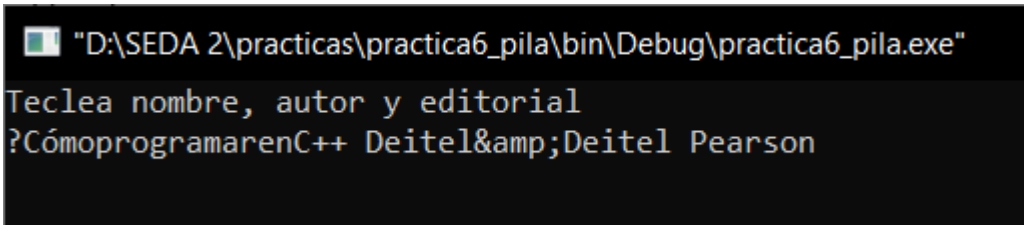
PASO 1: Elegir la opción 2 (eliminar, mostrar mensaje de validación, la pila está vacía)



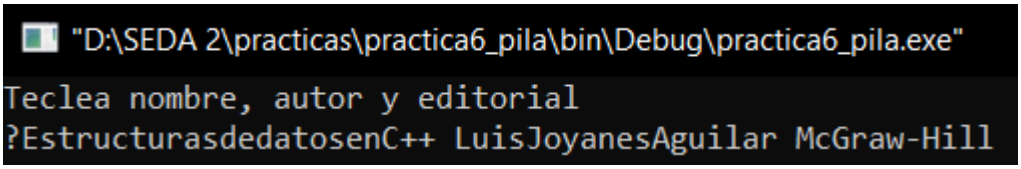
PASO 2: Elegir la opción 1 (insertar)



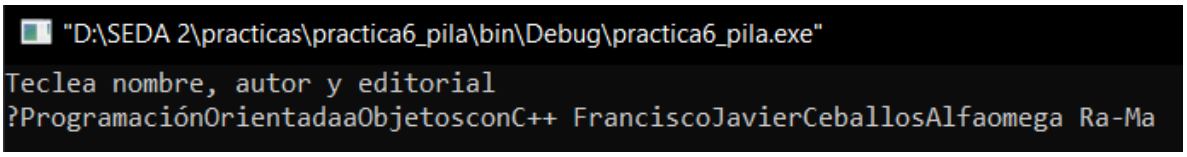
PASO 3: Elegir la opción 1 (insertar)



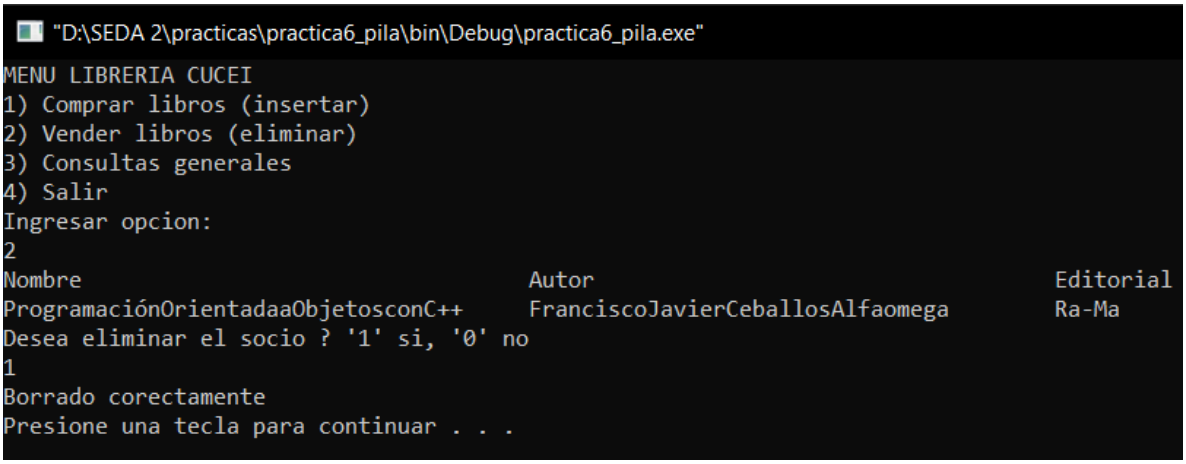
PASO 4: Elegir la opción 1 (insertar)



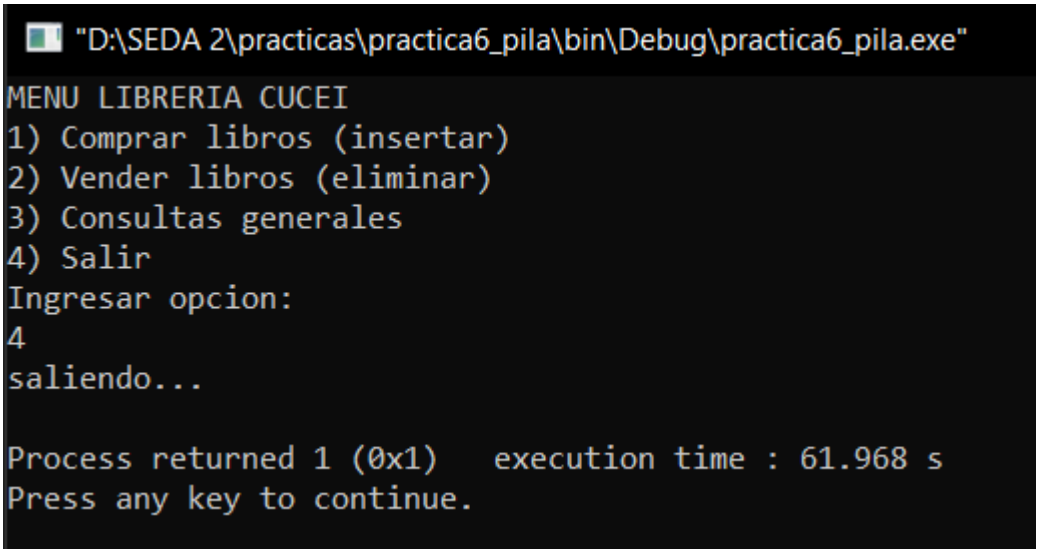
PASO 5: Elegir la opción 1 (insertar)



PASO 6: Elegir la opción 2 (eliminar)



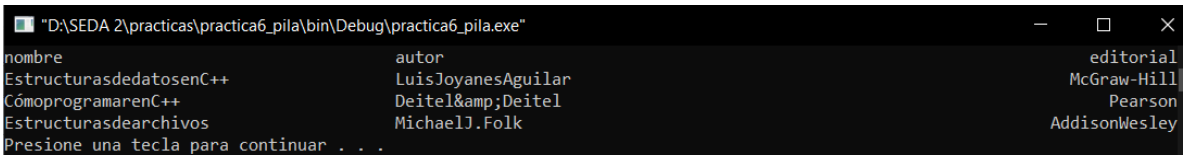
PASO 7: Elegir la opción 3 (salir, respaldar toda la información que tiene la pila en el archivo físico)



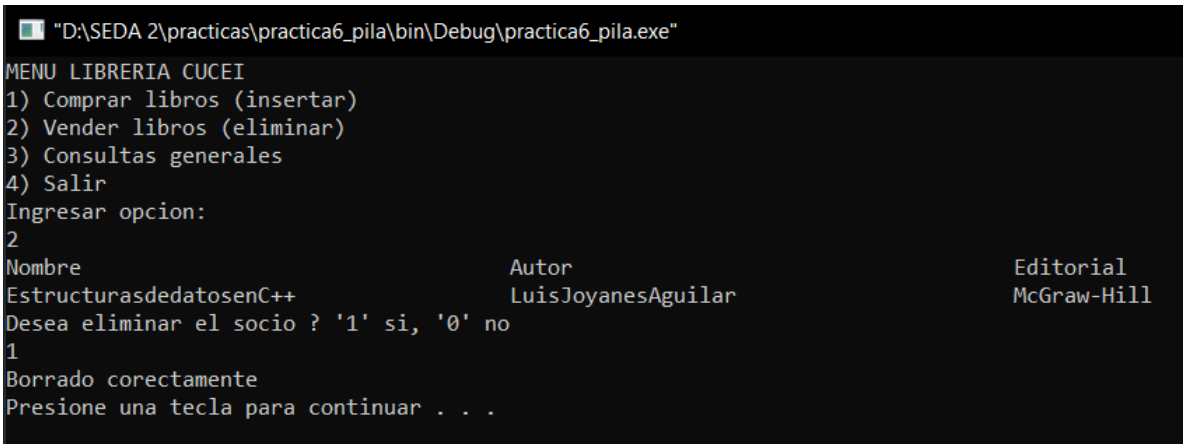
PASO 8: Abrir el archivo físico para verificar que se haya guardado la información de los libros. REALIZAR CAPTURA.

Volver a correr el programa para recuperar la información almacenada en el archivo, cargar dicha información a la Pila y continuar con el escenario.

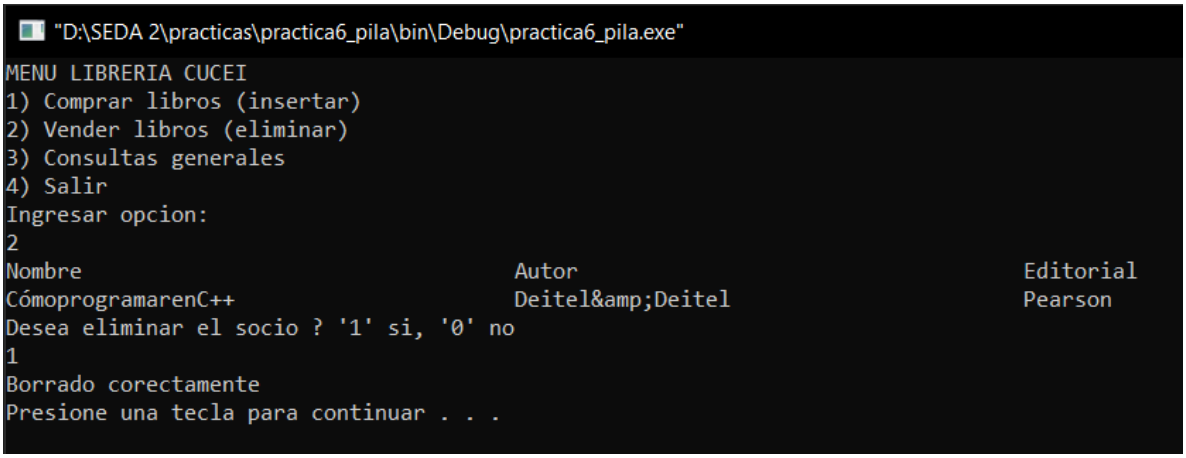
```
void recuperar(fstream&archlibro)
{
    system("cls");
    // lee el primer registro del archivo
    archlibro.read( reinterpret_cast< char * >( &d ), sizeof( Libro ) );
    // lee todos los registros del archivo
    while ( archlibro && !archlibro.eof() )
    {
        // muestra un registro
        if ( d.obtenerposicion() != 0)
        {
            mi_pila.push(d);
            numeroDeSocio++;
        }
        // lee el siguiente registro del archivo
        archlibro.read( reinterpret_cast< char * >( &d ), sizeof( Libro ) );
    } // fin de while
    system("pause");
}
```



PASO 9: Elegir la opción 2 (eliminar)

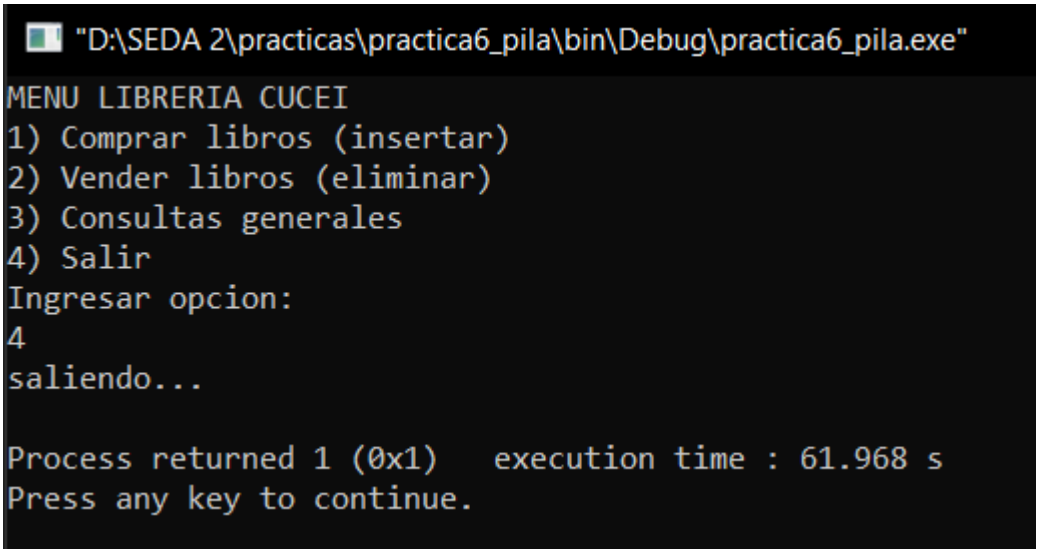


PASO 10: Elegir la opción 2 (eliminar)

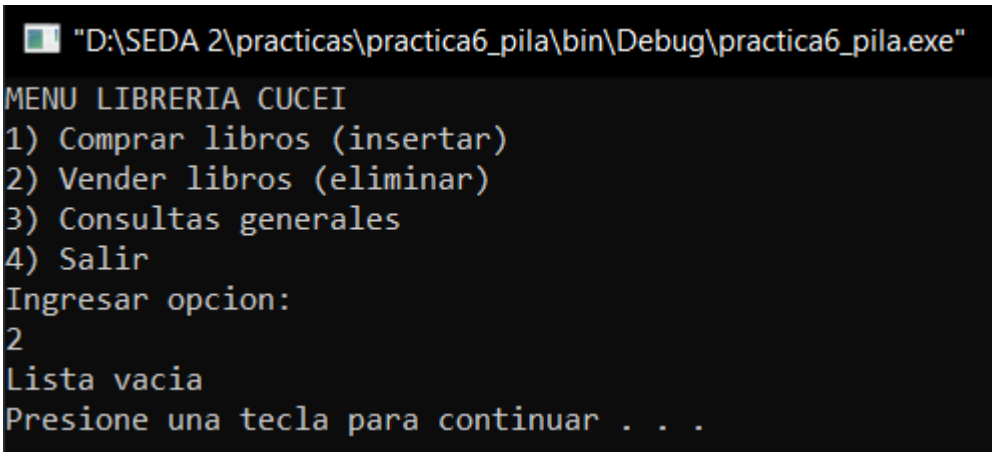


PASO 11: Elegir la opción 2 (eliminar)

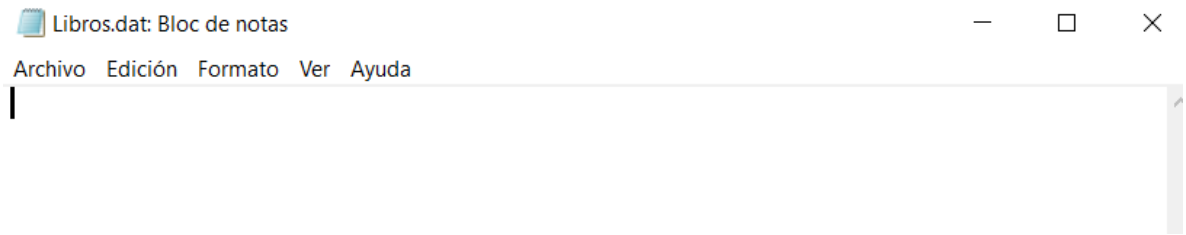
PASO 12: Elegir la opción 3 (salir, respaldar toda la información que tiene la pila en el archivo físico)



¿LA PILA TIENE INFORMACIÓN O SE ENCUENTRA VACÍA?

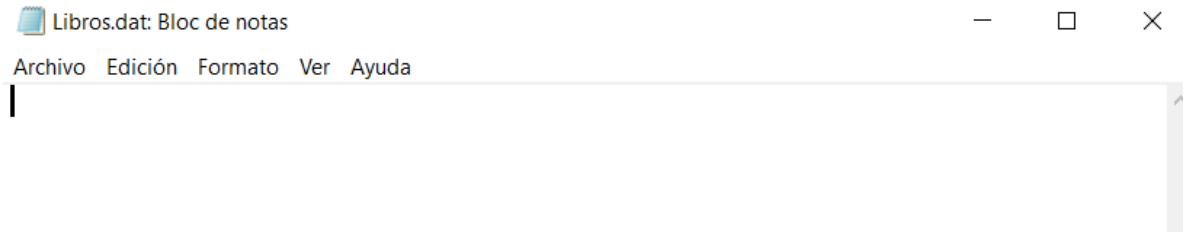


PASO 13: Abrir el archivo físico para verificar que se haya guardado la información de los libros. REALIZAR CAPTURA



¿EL ARCHIVO FÍSICO SE ENCONTRARÁ VACÍO O CON INFORMACIÓN?

Vacio



CONCLUSIONES

Munguía Guizar Marlon Uriel: Dentro de lo que conllevó la realización de está práctica me pareció bastante interesante la implementación de la pila junto con la lista enlazada, llegando a que se nos complicara en ciertas partes su implementación, pero al final se logró gracias a la ayuda de mis compañeros de equipo.

Padilla Perez Jorge Daray: Para la practica estuvo pesado realizar la opción de recuperar los libros y meterlos en la pila, ya que daba muchos errores al inicio pero se logro resolver al final, espero que para las próximas clases podamos mejorar.

Pérez Palacios Perla Michelle: En esta práctica pudimos aprender y reforzar conocimientos ya que utilizamos una pila genérica pero ahora con una lista enlazada, fue un reto el manejo de archivos en una pila.

Pulido Tobias Rafael Agustín: El uso de listas simplemente ligadas para generar pilas genéricas en esta ocasión me pareció un un tanto complicado pero en general me parece que las pilas genéricas son muy útiles en muchas situaciones mas aún cuándo se utilizan archivos para gestionar los datos almacenados dentro de la pila. Por otro lado en esta práctica el uso de las listas ligadas me quedó mas claro**.**