```
1   //
2   // Created by Silvia Vargas on 17/11/2025.
3   //
4
5   #include <iostream>
6   #include <fstream>
7   #include <iomanip>
8   using namespace std;
9
10  #include "Bibliotecas/Cliente.h"
11  #include "Bibliotecas/Venta.h"
12  #include "Bibliotecas/Libreria.h"
13  #include "Bibliotecas/Libro.h"
14
15  #include "Bibliotecas/NodoCliente.h"
16  #include "Bibliotecas/NodoLibro.h"
17
18  #include "Bibliotecas/funciones.h"
19
20  int main() {
21      struct NodoCliente *listaClientes;
22      cargarClientes("ArchivosDeDatos/clientes.csv",listaClientes);
23      mostrarClientes("ArchivosDeReportes/PruebaDeClientes.txt",
        listaClientes);
24      struct NodoLibro *listaLibros;
25      cargaLibros("ArchivosDeDatos/libros.csv",listaLibros);
26
        mostrarLibros("ArchivosDeReportes/PruebaDeLibros.txt",listaLibros)
        ;
27
        procesarVentas("ArchivosDeDatos/ventas.txt",listaClientes,listaLib
        ros);
28      mostrarClientes("ArchivosDeReportes/ReporteFinalClientes.txt",
        listaClientes);
29
        mostrarLibros("ArchivosDeReportes/ReporteFinalLibros.txt",listaLib
        ros);
30      return 0;
31  }
32
33  //
34  // Created by Silvia Vargas on 17/11/2025.
35  //
36  #ifndef CLIENTE_H
37  #define CLIENTE_H
38
39  struct Cliente {
40      int dni;
41      char *nombre;
42      double montoComprado;
43      double descuentoFuturo;
44  };
45
46  #endif //CLIENTE_H
47
48  //
49  // Created by Silvia Vargas on 17/11/2025.
50  //
51
52  #ifndef VENTA_H
53  #define VENTA_H
54
55  struct Venta {
```

```
    int dniComprador;
    char *nombreComprador;
    int fecha;
    int calificacion;
};
#endif //VENTA_H

//
// Created by Silvia Vargas on 17/11/2025.
//

#ifndef LIBRERIA_H
#define LIBRERIA_H

struct Libreria {
    int codigo;
    struct Venta *ventas;
    int cantidadVentas;
    double totalVentas;
};
#endif //LIBRERIA_H

//
// Created by Silvia Vargas on 17/11/2025.
//

#ifndef LIBRO_H
#define LIBRO_H

struct Libro {
    char *codigo;
    char *titulo;
    char *autor;
    double precio;
    struct Libreria *librerias;
    int cantidadLibrerias;
    double totalVentas;
    int unidadesVendidas;
    int cantidadBuenasCalificaciones;
    int sumaBuenasCalificaciones;
    int cantidadMalasCalificaciones;
    int sumaMalasCalificaciones;
};
#endif //LIBRO_H

//
// Created by Silvia Vargas on 17/11/2025.
//

#ifndef NODOCLIENTE_H
#define NODOCLIENTE_H

struct NodoCliente {
    struct Cliente datoCliente;
    struct NodoCliente* siguiente;
};

#endif //NODOCLIENTE_H

//
// Created by Silvia Vargas on 17/11/2025.
//

```

```
#ifndef NODOLIBRO_H
#define NODOLIBRO_H

struct NodoLibro {
    struct Libro datoLibro;
    struct NodoLibro *siguiente;
};

#endif //NODOLIBRO_H

//
// Created by Silvia Vargas on 17/11/2025.
//

#ifndef FUNCIONES_H
#define FUNCIONES_H

void cargarClientes(const char *,struct NodoCliente *&);
char *leeCadenaExactaDelim(ifstream &,char );
void insertarNodoCliente(struct NodoCliente *&,struct Cliente );
void mostrarClientes(const char *,struct NodoCliente *);
void cargaLibros(const char *,struct NodoLibro *&);
void insertarNodoLibro(struct NodoLibro *&,struct Libro ) ;
void mostrarLibros(const char *,struct NodoLibro *);
void procesarVentas(const char *,struct NodoCliente *,struct
NodoLibro *);
struct NodoLibro *buscarLibro(struct NodoLibro *,const char *) ;
struct NodoCliente *buscarCliente(struct NodoCliente *,int );
void actualizarLibreria(struct NodoLibro *,int ,double ,int ,int
,char *,int );
int buscarLibreria(struct Libreria *,int ,int );
void anadirVenta(struct Libreria &,struct Venta );
void actualizarValoresLibro(struct Libro &,int ,double ) ;
void actualizarDescuentoFuturo(struct NodoCliente *);

#endif //FUNCIONES_H

//
// Created by Silvia Vargas on 17/11/2025.
//
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string.h>
using namespace std;

#include "Cliente.h"
#include "Venta.h"
#include "Libreria.h"
#include "Libro.h"

#include "NodoCliente.h"
#include "NodoLibro.h"

#include "funciones.h"

#define MAX_LIBRERIAS 5
#define MAX_VENTAS 10

#define NO_ENCONTRADO -1

void cargarClientes(const char *nombArch,struct NodoCliente
*&listaClientes) {
```

```
179       ifstream arch(nombArch,ios::in);
180       if (not arch.is_open()) {
181           cout << "Error al abrir el archivo" << nombArch<<endl;
182           exit(1);
183       }
184       struct Cliente datoCliente{};
185       listaClientes=nullptr;
186       while (true) {
187           arch>>datoCliente.dni;
188           if (arch.eof()) break;
189           arch.get();
190           datoCliente.nombre=leeCadenaExactaDelim(arch,'\n');
191           insertarNodoCliente(listaClientes,datoCliente);
192       }
193   }
194
195   char *leeCadenaExactaDelim(ifstream &arch,char delim) {
196       char *ptr,cadena[100];
197       arch.getline(cadena,100,delim);
198       if (arch.eof()) return nullptr;
199       ptr=new char[strlen(cadena)+1];
200       strcpy(ptr,cadena);
201       return ptr;
202   }
203
204   void insertarNodoCliente(struct NodoCliente *&listaClientes,
205       struct Cliente datoCliente) {
206       struct NodoCliente *nuevoNodo;
207       nuevoNodo=new struct NodoCliente;
208       nuevoNodo->datoCliente=datoCliente;
209       nuevoNodo->siguiente=listaClientes;
210       listaClientes=nuevoNodo;
211   }
212
213   void mostrarClientes(const char *nombArch,
214       struct NodoCliente *listaClientes) {
215       ofstream arch(nombArch,ios::out);
216       if (not arch.is_open()) {
217           cout << "Error al abrir el archivo" << nombArch<<endl;
218           exit(1);
219       }
220       struct NodoCliente *ptrAux;
221       ptrAux=listaClientes;
222       arch<<setw(40)<<"LISTA DE CLIENTES"<<fixed<<setprecision(2)<<endl;
223       arch<<setw(6)<<"DNI"<<setw(20)<<"NOMBRE"<<setw(20)
224           <<"MONTO"<<setw(10)<<"DESCUENTO"<<endl;
225       while (ptrAux) {
226           struct Cliente auxCliente=ptrAux->datoCliente;
227           arch<<setfill('0')<<setw(8)<<auxCliente.dni
228               <<setfill(' ')<<setw(3)<<' '<<left<<setw(40)
229               <<auxCliente.nombre<<right<<setw(10)
230               <<auxCliente.montoComprado<<setw(10)
231               <<auxCliente.descuentoFuturo<<endl;
232           // arch<<setfill('0')<<setw(8)<<ptrAux->datoCliente.dni
233           //     <<setfill(' ')<<setw(3)<<' '<<left<<setw(40)
234           //     <<ptrAux->datoCliente.nombre<<right<<setw(10)
235           //     <<ptrAux->datoCliente.montoComprado<<setw(10)
236           //     <<ptrAux->datoCliente.descuentoFuturo<<endl;
237           ptrAux=ptrAux->siguiente;
238       }
239   }
240
241   void cargaLibros(const char *nombArch,struct NodoLibro *&listaLibros)
```

```
    {
242     ifstream arch(nombArch,ios::in);
243     if (not arch.is_open()) {
244         cout << "Error al abrir el archivo" << nombArch<<endl;
245         exit(1);
246     }
247     struct Libro datoLibro{};
248     listaLibros=nullptr;
249     while (true) {
250         datoLibro.codigo=leeCadenaExactaDelim(arch,',');
251         if (arch.eof()) break;
252         datoLibro.titulo=leeCadenaExactaDelim(arch,',');
253         datoLibro.autor=leeCadenaExactaDelim(arch,',');
254         arch>>datoLibro.precio;
255         arch.get();
256         datoLibro.librerias=new struct Libreria[MAX_LIBRERIAS]{};
257         insertarNodoLibro(listaLibros,datoLibro);
258     }
259 }
260
261 void insertarNodoLibro(struct NodoLibro *&listaLibros,struct Libro
    datoLibro) {
262     struct NodoLibro *nuevoNodo;
263     nuevoNodo=new struct NodoLibro;
264     nuevoNodo->datoLibro=datoLibro;
265     nuevoNodo->siguiente=listaLibros;
266     listaLibros=nuevoNodo;
267 }
268
269 void mostrarLibros(const char *nombArch,struct NodoLibro
    *listaLibros) {
270     ofstream arch(nombArch,ios::out);
271     if (not arch.is_open()) {
272         cout << "Error al abrir el archivo" << nombArch<<endl;
273         exit(1);
274     }
275     struct NodoLibro *ptrAux;
276     ptrAux=listaLibros;
277     arch<<setw(40)<<"LISTA DE LIBROS"<<fixed<<setprecision(2)<<endl;
278
        arch<<"CODIGO"<<setw(20)<<"TITULO"<<setw(30)<<"AUTOR"<<setw(10)<<"
        PRECIO"
279         <<setw(10)<<"CANT.LIBR"<<setw(10)<<"TOT.VENT."<<setw(10)
280         <<"UNID.VEND."<<setw(10)<<"C.B.CAL"<<setw(10)<<"S.B.CAL"
281         <<setw(10)<<"C.M.CAL"<<setw(10)<<"S.M.CAL"<<endl;
282     while (ptrAux) {
283         struct Libro libroAux=ptrAux->datoLibro;
284         arch<<libroAux.codigo<<setw(3)<<' '<<left<<setw(50)
285
                <<libroAux.titulo<<setw(30)<<libroAux.autor<<right<<setw(6
                )
286             <<libroAux.precio<<setw(4)<<libroAux.cantidadLibrerias
287             <<setw(10)<<libroAux.totalVentas
288             <<setw(10)<<libroAux.unidadesVendidas
289             <<setw(4)<<libroAux.cantidadBuenasCalificaciones
290             <<setw(6)<<libroAux.sumaBuenasCalificaciones
291             <<setw(4)<<libroAux.cantidadMalasCalificaciones
292             <<setw(6)<<libroAux.sumaMalasCalificaciones<<endl;
293         ptrAux=ptrAux->siguiente;
294     }
295 }
296
297 void procesarVentas(const char *nombArch,struct NodoCliente
```

```
    *listaClientes,
298     struct NodoLibro *listaLibros) {
299     ifstream arch(nombArch,ios::in);
300     if (not arch.is_open()) {
301         cout << "Error al abrir el archivo" << nombArch<<endl;
302         exit(1);
303     }
304     int codigo,d,m,a,dni,cal,fecha;
305     char c,libro[8];
306     double precio;
307     struct NodoLibro *libroEncontrado;
308     struct NodoCliente *clienteEncontrado;
309     while (true) {
310         arch>>codigo;
311         if (arch.eof()) break;
312         arch>>d>>c>>m>>c>>a;
313         fecha=a*10000+m*100+d;
314         while (true) {
315             arch>>libro>>dni>>cal;
316             libroEncontrado=buscarLibro(listaLibros,libro);
317             clienteEncontrado=buscarCliente(listaClientes,dni);
318             if (libroEncontrado and clienteEncontrado) {
319                 precio=libroEncontrado->datoLibro.precio;
320                 clienteEncontrado->datoCliente.montoComprado+=precio;
321
                    actualizarLibreria(libroEncontrado,codigo,precio,fecha
                    ,dni,
322                     clienteEncontrado->datoCliente.nombre,cal);
323
                    actualizarValoresLibro(libroEncontrado->datoLibro,cal,
                    precio);
324             }
325             if (arch.get()=='\n') break;
326         }
327     }
328     actualizarDescuentoFuturo(listaClientes);
329 }
330
331 struct NodoLibro *buscarLibro(struct NodoLibro *listaLibros,const
    char *libro) {
332     struct NodoLibro *ptrAux=listaLibros;
333     while (ptrAux) {
334         if (strcmp(ptrAux->datoLibro.codigo,libro)==0)
335             return ptrAux;
336         ptrAux=ptrAux->siguiente;
337     }
338     return nullptr;
339 }
340
341 struct NodoCliente *buscarCliente(struct NodoCliente
    *listaClientes,int dni) {
342     struct NodoCliente *ptrAux=listaClientes;
343     while (ptrAux) {
344         if (ptrAux->datoCliente.dni==dni)
345             return ptrAux;
346         ptrAux=ptrAux->siguiente;
347     }
348     return nullptr;
349 }
350
351 void actualizarLibreria(struct NodoLibro *libroEncontrado,int codigo,
352     double precio,int fecha,int dni,char *nombre,int cal) {
353     int idLibreria,cantLibrerias;
```

```
354         struct Venta nuevaVenta;
355         cantLibrerias=libroEncontrado->datoLibro.cantidadLibrerias;
356         idLibreria=buscarLibreria(libroEncontrado->datoLibro.librerias,
357             cantLibrerias,codigo);
358         if (idLibreria==NO_ENCONTRADO) {
359 
                libroEncontrado->datoLibro.librerias[cantLibrerias].codigo=cod
                igo;
360 
                libroEncontrado->datoLibro.librerias[cantLibrerias].ventas=new
                 struct Venta[MAX_VENTAS];
361             idLibreria=libroEncontrado->datoLibro.cantidadLibrerias;
362             libroEncontrado->datoLibro.cantidadLibrerias++;
363         }
364         nuevaVenta.dniComprador=dni;
365         nuevaVenta.fecha=fecha;
366         nuevaVenta.calificacion=cal;
367         nuevaVenta.nombreComprador=new char[strlen(nombre)+1];
368         strcpy(nuevaVenta.nombreComprador,nombre);
369 
            anadirVenta(libroEncontrado->datoLibro.librerias[idLibreria],nueva
            Venta);
370 
            libroEncontrado->datoLibro.librerias[idLibreria].totalVentas+=prec
            io;
371     }
372 
373     int buscarLibreria(struct Libreria *librerias,int cantLibrerias,int
        codigo){
374         for (int i=0;i<cantLibrerias;i++)
375             if (librerias[i].codigo==codigo) return i;
376         return NO_ENCONTRADO;
377     }
378 
379 
380     void anadirVenta(struct Libreria &libreria,struct Venta nuevaVenta) {
381         int cantVentas=libreria.cantidadVentas;
382         libreria.ventas[cantVentas]=nuevaVenta;
383         libreria.cantidadVentas++;
384     }
385 
386     void actualizarValoresLibro(struct Libro &libro,int cal,double
        precio) {
387         libro.totalVentas+=precio;
388         libro.unidadesVendidas++;
389         if (cal>50) {
390             libro.cantidadBuenasCalificaciones++;
391             libro.sumaBuenasCalificaciones+=cal;
392         }
393         else {
394             libro.cantidadMalasCalificaciones++;
395             libro.sumaMalasCalificaciones+=cal;
396         }
397 
398     }
399 
400     void actualizarDescuentoFuturo(struct NodoCliente *listaClientes) {
401         struct NodoCliente *ptrAux=listaClientes;
402         double monto;
403         while (ptrAux) {
404             monto=ptrAux->datoCliente.montoComprado;
405             if (monto>5000)
406                 ptrAux->datoCliente.descuentoFuturo=0.3;
```

```
407             else
408                 if (monto>3000)
409                     ptrAux->datoCliente.descuentoFuturo=0.2;
410                 else
411                     if (monto>0)
412                         ptrAux->datoCliente.descuentoFuturo=0.1;
413             ptrAux=ptrAux->siguiente;
414         }
415     }
416
```