```cpp
/*
 * Proyecto: Solucion_Laboratorio2_2024-1
 * Archivo:  main.cpp
 * Autor:    J. miguel.guanira.
 *
 * Created on 9 de setiembre de 2024, 08:15
 */
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
#include "MetodoExactoDeMemoria.h"

int main(int argc, char** argv) {
   char ***libros,***pedidosLibros;
   int **stock,  **pedidosClientes;
   bool **pedidosAtendidos;

   lecturaDeLibros ("Libros.csv",libros, stock);
   pruebaDeLecturaDeLibros ("ReporteDeLibrosInicial.txt" ,libros, stock);
   atencionDePedidos ("Pedidos.txt", libros, stock, pedidosClientes,
                      pedidosLibros, pedidosAtendidos);
   pruebaDeLecturaDeLibros ("ReporteDeLibrosFinal.txt", libros, stock);
   reporteDeEntregaDePedidos ("ReporteDeEntregaDePedisos.txt", pedidosClientes,
                              pedidosLibros, pedidosAtendidos);
    return 0;
}

/*
 * Proyecto: Solucion_Laboratorio2_2024-1
 * Archivo:  MetodoExactoDeMemoria.cpp
 * Autor:    J. miguel.guanira.
 *
 * Created on 9 de setiembre de 2024, 08:24
 */

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
#include <cstring>
#include <mutex>
#include "MetodoExactoDeMemoria.h"
#define MAX_CAR_LIN 70

void lecturaDeLibros (const char*nombArch,char ***&libros, int **&stock){
    ifstream arch(nombArch,ios::in);
    if(not arch.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
        exit(1);
    }
    char**buffLibros[300]{}, **regLibro;
    int *buffStock[300]{}, numLibros=0;

    while(true){
        regLibro = leeCodTitAut(arch);
        if(arch.eof())break;
        buffLibros[numLibros] = regLibro;
        buffStock[numLibros] = leeStock(arch);
        numLibros++;
    }
    libros = new char**[numLibros+1]{};
    stock = new int*[numLibros+1]{};
    for (int i = 0; i < numLibros; i++) {
        libros[i] = buffLibros[i];
        stock[i] = buffStock[i];
```

```
67      }
68  }
69
70  char **leeCodTitAut(ifstream &arch){
71      char *codigo, **regLibro;
72      codigo = leeCadena(arch,',');
73      if(arch.eof())return nullptr;
74      regLibro = new char*[3];
75      regLibro[0] = codigo;
76      regLibro[1] = leeCadena(arch,',');
77      regLibro[2] = leeCadena(arch,',');
78      return regLibro;
79  }
80  char *leeCadena(ifstream &arch,char delimitador){
81      char buffCadena[100], *cadena;
82      arch.getline(buffCadena,100,delimitador);
83      if(arch.eof())return nullptr;
84      cadena = new char[strlen(buffCadena)+1];
85      strcpy(cadena,buffCadena);
86      return cadena;
87  }
88
89  int *leeStock(ifstream &arch){
90      int *regStock;
91      double precio;
92      char c;
93      regStock = new int[2]{};
94      arch>>regStock[0]>>c>>precio;
95      arch.get(); // saco el cambio de linea
96      return regStock;
97  }
98
99  char *leeCadena(ifstream &arch,char delimitador){
100     char buffCadena[100], *cadena;
101     arch.getline(buffCadena,100,delimitador);
102     if(arch.eof())return nullptr;
103     cadena = new char[strlen(buffCadena)+1];
104     strcpy(cadena,buffCadena);
105     return cadena;
106 }
107
108 void pruebaDeLecturaDeLibros (const char*nombArch,char ***libros,int **stock){
109     ofstream arch(nombArch,ios::out);
110     if(not arch.is_open()){
111         cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
112         exit(1);
113     }
114     arch<<left<<setw(10)<<"CODIGO"<<setw(60)<<"TITULO"<<setw(35)<<"AUTOR"
115         <<right<<"STOCK"<<setw(15)<<"NO ATENDIDOS"<<endl;
116     for (int i = 0; libros[i]; i++) {
117         imprimeLibro(arch, libros[i],stock[i]);
118     }
119 }
120
121 void imprimeLibro(ofstream &arch, char**libro,int *stock){
122     arch<<left<<setw(10)<<libro[0]<<setw(60)<<libro[1]<<setw(35)<<libro[2]
123         <<right<<setw(4)<<stock[0]<<setw(9)<<stock[1]<<endl;
124
125 }
126
127 void atencionDePedidos (const char*nombArch, char ***libros, int **stock,
128                         int **&pedidosClientes, char ***&pedidosLibros,
129                         bool **&pedidosAtendidos){
130     ifstream arch(nombArch,ios::in);
131     if(not arch.is_open()){
132         cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
```

```
133             exit(1);
134         }
135         int *buffPedCli[200]{}, numeroDePedido, dni, cantLib[200]{},maxPed=0,
136                 cantCli=0;
137         char **buffPedLib[200]{},c;
138         bool *buffPeddAte[200]{};
139 
140         inicilizarBuffers(buffPedCli,buffPedLib,buffPeddAte);
141         while(true){
142             arch>>numeroDePedido;
143             if(arch.eof())break;
144             arch>>c>>dni;
145             asignarPedidoAlCliente(numeroDePedido,dni,buffPedCli,cantCli);
146             asignarLibros(arch,buffPedLib[numeroDePedido],
147                     buffPeddAte[numeroDePedido],cantLib[numeroDePedido],libros,
148                     stock);
149             if(numeroDePedido>maxPed) maxPed = numeroDePedido;
150         }
151         pedidosClientes = new int*[cantCli+1]{};
152         for(int i=0; i<200; i++) {
153            if(i<cantCli) pedidosClientes[i] = liberaEspacios(buffPedCli[i]);
154            else delete buffPedCli[i];
155         }
156         memoriaExactaPedidos(pedidosLibros,pedidosAtendidos,buffPedLib,buffPeddAte,
157                              cantLib,maxPed);
158     }
159 
160     void inicilizarBuffers(int**buffPedCli,char ***buffPedLib,bool **buffPeddAte){
161         for (int i = 0; i < 200; i++) {
162             buffPedCli[i] = new int[20]{};
163             buffPedLib[i] = new char*[10]{};
164             buffPeddAte[i] = new bool[10]{};
165         }
166     }
167 
168     void asignarPedidoAlCliente(int numeroDePedido,int dni,int **buffPedCli,
169                                 int &cantCli){
170         int posCli, *cliente;
171         posCli = buscarCliente(dni,buffPedCli,cantCli);
172 
173         if(posCli == -1){
174            cliente = buffPedCli[cantCli];
175            cliente[0] = dni;
176            cantCli++;
177         }
178         else cliente = buffPedCli[posCli];
179 
180         cliente[2+cliente[1]] = numeroDePedido;
181         cliente[1]++;
182     }
183 
184     int buscarCliente(dni,buffPedCli,cantCli);
185 
186     void reporteDeEntregaDePedidos (const char*nombArch, int **pedidosClientes,
187                                     char ***pedidosLibros, bool **pedidosAtendidos){
188         ofstream arch(nombArch,ios::out);
189         if(not arch.is_open()){
190             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
191             exit(1);
192         }
193 
194         arch<<right<<setw(55)<<"REPORTE DE ATENCION DE PEDIDOS"<<endl;
195         imprimeLinea(arch,'=',MAX_CAR_LIN);
196         for(int i=0; pedidosClientes[i]; i++)
197             imprimeCliente(arch,pedidosClientes[i],pedidosLibros,pedidosAtendidos);
198     }
```

```
199
200 void imprimeCliente(ofstream &arch, int *cliente,
201                       char ***pedidosLibros, bool **pedidosAtendidos){
202     int pedido;
203     arch<<right<<"Cliente: "<<setw(10)<<cliente[0]<<endl;
204     imprimeLinea(arch,'=',MAX_CAR_LIN);
205     arch<<setw(14)<<"Pedido No."<<setw(18)<<"Codigo del libro"
206         <<setw(12)<<"Observacion"<<endl;
207     imprimeLinea(arch,'-',MAX_CAR_LIN);
208     for (int i = 0; i <cliente[1]; i++){
209         pedido = cliente[i+2];
210         imprimeLinea(arch,'-',MAX_CAR_LIN);
211         arch<<"     "<<setfill('0')<<setw(6)<<pedido<<setfill(' ');
212         imprimeLibros(arch,pedidosLibros[pedido],pedidosAtendidos[pedido]);
213     }
214 }
215
216 void imprimeLibros(ofstream &arch,char**pedidosLibros,bool*pedidosAtendidos){
217     for (int i = 0; pedidosLibros[i]; i++) {
218         if(i==0) arch<<setw(18);
219         else arch<<setw(29);
220         arch<<pedidosLibros[i];
221         if(pedidosAtendidos[i])
222             arch<<setw(14)<<"ATENDIDO"<<endl;
223         else arch<<setw(17)<<"NO ATENDIDO"<<endl;
224     }
225
226 }
227
228 void imprimeLinea(ofstream &arch,char c,int n){
229     for (int i = 0; i < n; i++)arch.put(c);
230     arch<<endl;
231 }
```