```
1   /*
2    * Proyecto: ArbolConClases
3    * Archivo:  Nodo.h
4    * Autor:    J.  Miguel Guanira E. //miguel.guanira.
5    *
6    * Created on 6 de noviembre de 2024, 08:31 AM
7    */
8
9
10  #ifndef NODO_H
11  #define NODO_H
12  #include "Arbol.h"
13  #include "Persona.h"
14
15  class Nodo {
16  private:
17      class Persona dato;
18      class Nodo *izquiera;
19      class Nodo *derecha;
20  public:
21      Nodo();
22      friend class Arbol;
23  };
24
25  #endif /* NODO_H */
26
27  /*
28   * Proyecto: ArbolConClases
29   * Archivo:  Nodo.cpp
30   * Autor:    J. Miguel Guanira E. //miguel.guanira.
31   *
32   * Created on 6 de noviembre de 2024, 08:31 AM
33   */
34
35  #include <iostream>
36  #include <iomanip>
37  using namespace std;
38
39  #include "Nodo.h"
40
41  Nodo::Nodo() {
42      izquiera = derecha =nullptr;
43  }
44
45  /*
46   * Proyecto: ArbolConClases
47   * Archivo:  Arbol.h
48   * Autor:    J.  Miguel Guanira E. //miguel.guanira.
49   *
50   * Created on 6 de noviembre de 2024, 08:33 AM
51   */
52
53
54  #ifndef ARBOL_H
55  #define ARBOL_H
56  #include "Nodo.h"
57  #include "Persona.h"
58
59  class Arbol {
60  private:
61      class Nodo *arbol;
62      void insertarR(class Nodo *&arbol, const class Persona &dato);
63      void mostrarEnOrdenR(class Nodo *arbol);
64      class Persona buscarR(class Nodo *arbol,int dato);
65      void eliminaR(class Nodo *arbol);
66  public:
```

```
67          Arbol();
68          virtual ~Arbol();
69          void crear(const char *nombArch);
70          void insertar(const class Persona &dato);
71          void mostrarEnOrden();
72          class Persona buscar(int dato);
73          void elimina();
74      };
75
76      #endif /* ARBOL_H */
77
78      /*
79       * Proyecto: ArbolConClases
80       * Archivo:  Arbol.cpp
81       * Autor:    J. Miguel Guanira E. //miguel.guanira.
82       *
83       * Created on 6 de noviembre de 2024, 08:33 AM
84       */
85
86      #include <iostream>
87      #include <fstream>
88      #include <iomanip>
89      using namespace std;
90
91      #include "Arbol.h"
92
93      Arbol::Arbol() {
94          arbol = nullptr;
95      }
96
97      Arbol::~Arbol() {
98          elimina();
99      }
100
101     void Arbol::crear(const char* nombArch) {
102         ifstream arch (nombArch,ios::in);
103         if(not arch.is_open()){
104             cout<<"ERROR: no se pudo abrie el archivo "<<nombArch<<endl;
105             exit(1);
106         }
107         class Persona per;
108         while(true){
109             arch>>per;
110             if(arch.eof())break;
111             insertarR(arbol,per);
112         }
113     }
114
115     void Arbol::insertar(const class Persona &dato) {
116         insertarR(arbol,dato);
117     }
118
119     void Arbol::insertarR(class Nodo* &arbol, const class Persona &dato) {
120         if(arbol==nullptr){
121             arbol = new class Nodo;
122             arbol->dato = dato;
123             return;
124         }
125         if(arbol->dato > dato) insertarR(arbol->izquiera,dato);
126         else insertarR(arbol->derecha,dato);
127     }
128
129     void Arbol::mostrarEnOrden() {
130         mostrarEnOrdenR(arbol);
131         cout<<endl;
132     }
```

```
133
134 void Arbol::mostrarEnOrdenR(class Nodo* arbol) {
135     if(arbol){
136         mostrarEnOrdenR(arbol->izquiera);
137         cout<<arbol->dato;
138         mostrarEnOrdenR(arbol->derecha);
139     }
140 }
141
142 class Persona Arbol::buscar(int dato) {
143     return buscarR(arbol,dato);
144 }
145
146 class Persona Arbol::buscarR(class Nodo* arbol, int dato) {
147     class Persona per;
148     per.SetDni(-1);
149     per.SetNombre(" ");
150     per.SetSueldo(0.0);
151     if(arbol==nullptr) return per;
152     if(arbol->dato == dato) return arbol->dato;
153     if(arbol->dato > dato) return buscarR(arbol->izquiera,dato);
154     else return buscarR(arbol->derecha,dato);
155 }
156
157 void Arbol::elimina() {
158     eliminaR(arbol);
159 }
160
161 void Arbol::eliminaR(class Nodo* arbol) {
162     if(arbol){
163         eliminaR(arbol->izquiera);
164         eliminaR(arbol->derecha);
165         delete arbol;
166     }
167 }
168
169 /*
170  * Proyecto: ListaLigadaConClasePersona
171  * Archivo:  Persona.h
172  * Autor:    J.  Miguel Guanira E. //miguel.guanira.
173  *
174  * Created on 5 de noviembre de 2024, 09:34 AM
175  */
176
177
178 #ifndef PERSONA_H
179 #define PERSONA_H
180
181 class Persona {
182 private:
183     int dni;
184     char *nombre;
185     double sueldo;
186 public:
187     Persona();
188     Persona(const Persona& orig);
189     virtual ~Persona();
190     void SetSueldo(double sueldo);
191     double GetSueldo() const;
192     void SetNombre(const char* nombre);
193     void  GetNombre(char*) const;
194     void SetDni(int dni);
195     int GetDni() const;
196     void operator = (const Persona& orig);
197     bool operator >(const Persona& orig);
198     bool operator ==(const Persona& orig);
```

```
199
200         bool operator >(int dato);
201         bool operator ==(int dato);
202 
203     };
204 
205     #endif /* PERSONA_H */
206 
207     void operator>>(ifstream &, class Persona &per);
208     void operator<<(ofstream &, class Persona &per);
209     void operator<<(ostream &, class Persona &per);
210 
211     /*
212      * Proyecto: ListaLigadaConClasePersona
213      * Archivo:  Persona.cpp
214      * Autor:    J. Miguel Guanira E. //miguel.guanira.
215      *
216      * Created on 5 de noviembre de 2024, 09:34 AM
217      */
218 
219     #include <iostream>
220     #include <fstream>
221     #include <iomanip>
222     using namespace std;
223     #include <cstring>
224     #include "Persona.h"
225 
226     Persona::Persona() {
227         nombre = nullptr;
228     }
229 
230     Persona::Persona(const Persona& orig) {
231         nombre = nullptr;
232         *this = orig;
233     }
234 
235     Persona::~Persona() {
236         if (nombre!=nullptr) delete nombre;
237     }
238 
239     void Persona::SetSueldo(double sueldo) {
240         this->sueldo = sueldo;
241     }
242 
243     double Persona::GetSueldo() const {
244         return sueldo;
245     }
246 
247     void Persona::SetNombre(const char* nomb) {
248         if(nombre) delete nombre;
249         nombre = new char [strlen(nomb)+1];
250         strcpy(nombre,nomb);
251     }
252 
253     void Persona::GetNombre(char*nomb) const {
254         if(nombre == nullptr) nomb[0] = 0;
255         else strcpy(nomb,nombre);
256     }
257 
258     void Persona::SetDni(int dni) {
259         this->dni = dni;
260     }
261 
262     int Persona::GetDni() const {
263         return dni;
264     }
```

```
265
266 void Persona::operator = (const Persona& orig){
267     char aux[60];
268     dni = orig.dni;
269     sueldo = orig.sueldo;
270     orig.GetNombre(aux);
271     SetNombre(aux);
272 }
273
274 bool Persona::operator>(const class Persona& orig) {
275     return dni > orig.dni;
276 }
277
278 bool Persona::operator==(const class Persona& orig) {
279     return dni == orig.dni;
280 }
281
282 bool Persona::operator>(int dato) {
283     return dni > dato;
284 }
285
286 bool Persona::operator==(int dato) {
287     return dni == dato;
288 }
289
290
291 void operator>>(ifstream &arch, class Persona &per){
292     int dni;
293     char nomb[60];
294     double sueldo;
295     arch>>dni;
296     if(arch.eof())return;
297     arch.get();
298     arch.getline(nomb,60,',');
299     arch>>sueldo;
300     per.SetDni(dni);
301     per.SetNombre(nomb);
302     per.SetSueldo(sueldo);
303 }
304
305 void operator<<(ofstream &arch, class Persona &per){
306     arch.precision(2);
307     arch<<fixed;
308     char cad[60];
309     per.GetNombre(cad);
310     arch<<right<<setw(10)<<per.GetDni()<<"  "<<left<<setw(45)<<cad
311         <<right<<setw(10)<<per.GetSueldo()<<endl;
312 }
313 void operator<<(ostream &out, class Persona &per){
314     out.precision(2);
315     out<<fixed;
316     char cad[60];
317     per.GetNombre(cad);
318     out<<right<<setw(10)<<per.GetDni()<<"  "<<left<<setw(45)<<cad
319         <<right<<setw(10)<<per.GetSueldo()<<endl;
320 }
321
```