```
/*
 * Proyecto: ListaConPlantillas
 * Archivo:  PlantillaListaLigada.h
 * Autor:    J. Miguel Guanira E.
 *
 * Created on 12 de noviembre de 2024, 08:40 AM
 */


#ifndef PLANTILLALISTALIGADA_H
#define PLANTILLALISTALIGADA_H

#include <iostream>
#include <iomanip>
#include <fstream>
using namespace std;

template <typename T>class PlantillaListaLigada;

template<typename T>
class NodoPlantilla{
private:
    T dato;
    class NodoPlantilla<T> *siguente;
public:
    NodoPlantilla<T>();
    friend class PlantillaListaLigada<T>;
};

template <typename T>
NodoPlantilla<T>::NodoPlantilla(){
    siguente = nullptr;
}
template <typename T>
class PlantillaListaLigada {
private:
    class NodoPlantilla<T> *lista;
    void imprimirR(ofstream &arch, class NodoPlantilla<T> *lista);
public:
    PlantillaListaLigada();
    virtual ~PlantillaListaLigada();
    void crear(const char* nombArch);
    void insrtear(const T&dato);
    void imprimir(const char*nombArch);
    void imprimirRecursivo(const char*nombArch);
    void elimina();
};

template <typename T>
PlantillaListaLigada<T>::PlantillaListaLigada(){
    lista = nullptr;
}

template <typename T>
PlantillaListaLigada<T>::~PlantillaListaLigada(){
    elimina();
}

template <typename T>
void PlantillaListaLigada<T>::crear(const char* nombArch){
    ifstream arch(nombArch,ios::in);
    if(not arch.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
        exit(1);
    }
    T dato;
```

```
67          while(true){
68              arch>>dato;
69              if(arch.eof())break;
70              insrtear(dato);
71          }
72      }
73
74      template <typename T>
75      void PlantillaListaLigada<T>::insrtear(const T &dato){
76          class NodoPlantilla<T> *p=lista, *ant=nullptr, *nuevo;
77          nuevo = new class NodoPlantilla<T>;
78          nuevo->dato = dato;
79          while(p){
80              if(p->dato > dato) break;
81              ant = p;
82              p = p->siguente;
83          }
84          nuevo->siguente = p;
85          if(ant != nullptr) ant->siguente = nuevo;
86          else lista = nuevo;
87      }
88
89      template <typename T>
90      void PlantillaListaLigada<T>::imprimir(const char* nombArch){
91          ofstream arch(nombArch,ios::out);
92          if(not arch.is_open()){
93              cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
94              exit(1);
95          }
96          class NodoPlantilla<T> *p=lista;
97          while(p){
98              arch<<p->dato<<endl;
99              p = p->siguente;
100         }
101         arch<<endl;
102     }
103
104     template <typename T>
105     void PlantillaListaLigada<T>::imprimirRecursivo(const char* nombArch){
106         ofstream arch(nombArch,ios::out);
107         if(not arch.is_open()){
108             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
109             exit(1);
110         }
111         imprimirR(arch,lista);
112         arch<<endl;
113     }
114
115     template <typename T>
116     void PlantillaListaLigada<T>::imprimirR(ofstream& arch,
117                                             NodoPlantilla<T>* lista){
118         if(lista == nullptr) return;
119         imprimirR(arch,lista->siguente);
120         arch<<lista->dato<<endl;
121     }
122
123     template <typename T>
124     void PlantillaListaLigada<T>::elimina(){
125         class NodoPlantilla<T>*sale;
126         while(lista){
127             sale = lista;
128             lista = lista->siguente;
129             delete sale;
130         }
131     }
132
```

```
133  #endif /* PLANTILLALISTALIGADA_H */
134
135  /*
136   * Archivo: Persona.h
137   * Autor:   J. Miguel Guanira E. (mguanir)
138   *
139   * Creado el 18 de junio de 2019, 09:34 AM
140   */
141
142  #ifndef PERSONA_H
143  #define PERSONA_H
144
145  class Persona {
146  private:
147      int dni;
148      char *nombre;
149      double sueldo;
150  public:
151      Persona();
152      virtual ~Persona();
153      void SetSueldo(double sueldo);
154      double GetSueldo() const;
155      void SetNombre(char* nombre);
156      void GetNombre(char*) const;
157      void SetDni(int dni);
158      int GetDni() const;
159      void operator =(const class Persona&);
160      bool operator >(const class Persona&);
161  };
162
163  void operator >>(istream&,class Persona&);
164  ostream &operator<<(ostream&, const class Persona&);
165
166  #endif /* PERSONA_H */
167
168  /*
169   * Archivo: Persona.cpp
170   * Autor:   J. Miguel Guanira E. (mguanir)
171   *
172   * Creado el 18 de junio de 2019, 09:34 AM
173   */
174  #include <iostream>
175  #include <iomanip>
176  using namespace std;
177  #include <cstdlib>
178  #include <cstring>
179  #include "Persona.h"
180
181  Persona::Persona() {
182      nombre = nullptr;
183  }
184
185  Persona::~Persona() {
186      if (nombre)delete nombre;
187  }
188
189  void Persona::SetSueldo(double sueldo) {
190      this->sueldo = sueldo;
191  }
192
193  double Persona::GetSueldo() const {
194      return sueldo;
195  }
196
197  void Persona::SetNombre(char* nomb) {
198      if (nombre!=nullptr)delete nombre;
```

```
199         nombre = new char [strlen(nomb)+1];
200         strcpy(nombre,nomb);
201     }
202
203     void Persona::GetNombre(char*nomb) const {
204         if(nombre ==nullptr) nomb[0]=0;
205         else strcpy(nomb, nombre);
206     }
207
208     void Persona::SetDni(int dni) {
209         this->dni = dni;
210     }
211
212     int Persona::GetDni() const {
213         return dni;
214     }
215
216     void Persona::operator=(const Persona&per) {
217         char nomb[100];
218         dni = per.dni;
219         per.GetNombre(nomb);
220         SetNombre(nomb);
221         sueldo = per.sueldo;
222     }
223
224     bool Persona::operator>(const Persona&per) {
225         return dni>per.GetDni();
226     }
227
228     void operator >>(istream&in,class Persona&per){
229         char nomb[100];
230         int dni;
231         double sueldo;
232         in>>dni;
233         per.SetDni(dni);
234         in.get();
235         in.getline(nomb,100,',');
236         per.SetNombre(nomb);
237         in>>sueldo;
238         per.SetSueldo(sueldo);
239     }
240
241     ostream &operator<<(ostream&out, const class Persona&per){
242         char nomb[100];
243         out.precision(2);
244         out<<fixed;
245         out<<setw(10)<<per.GetDni()<<"  ";
246         per.GetNombre(nomb);
247         out<<left<<setw(50)<<nomb
248            <<right<<setw(12)<<per.GetSueldo();
249         return out;
250     }
251
252     /*
253      * Proyecto: ListaConPlantillas
254      * Archivo:  main.cpp
255      * Autor:    J. Miguel Guanira E.
256      *
257      * Created on 11 de noviembre de 2024, 09:24 PM
258      */
259
260     #include <iostream>
261     #include <iomanip>
262     #include <string>
263     using namespace std;
264     #include "PlantillaListaLigada.h"
```

```
265  #include "Persona.h"
266
267  int main(int argc, char** argv) {
268  //    class PlantillaListaLigada<int> listaInt;
269  //    listaInt.crear("Datos.txt");
270  //    listaInt.imprimir("ReporteEnteros.txt");
271  //    listaInt.imprimirRecursivo("ReporteEnterosRec.txt");
272
273  //    class PlantillaListaLigada<string> listaStr;
274  //    listaStr.crear("Nombres.txt");
275  //    listaStr.imprimir("ReporteNombres.txt");
276  //    listaStr.imprimirRecursivo("ReporteNombresRec.txt");
277
278      class PlantillaListaLigada<class Persona> listaPer;
279      listaPer.crear("personal.csv");
280      listaPer.imprimir("ReportePersonas.txt");
281      listaPer.imprimirRecursivo("ReportePersonasRec.txt");
282      return 0;
283  }
284
285
```