```cpp
/*
 * Proyecto: STL-list
 * Archivo:  main.cpp
 * Autor:    J. Miguel Guanira E.
 *
 * Created on 13 de noviembre de 2024, 09:26 AM
 */

#include <iostream>
#include <fstream>
#include <iomanip>
#include <list>
#include <iterator>
using namespace std;
#include "Persona.h"

int main(int argc, char** argv) {
    list<double> lDbl{12.3, 54.33, 17.57, 61.99, 129.51, 57.57};
    list<double> lDbl2{88.44, 77.22, 11.99};

    cout.precision(2);
    cout<<fixed;

//    cout<<"Numero de elementos: "<<lDbl.size()<<endl;
//    for(double x: lDbl)
//        cout<<setw(10)<<x;
//    cout<<endl<<endl;
//
//    lDbl.push_front(77.11);
//    lDbl.push_back(1.3);
//    for(double x: lDbl)
//        cout<<setw(10)<<x;
//    cout<<endl<<endl;
//
//    list<double>::iterator it = lDbl.begin();
//    //it += 3;  esto no se puede hacer
//    it++;
//    it++;
//    it++;
//    cout<<setw(10)<<*it<<endl;
//    lDbl.insert(it,66.66);
//    for(double x: lDbl)
//        cout<<setw(10)<<x;
//    cout<<endl<<endl;

//    for(list<double>::iterator it= lDbl.begin(); it!=lDbl.end();it++)
//        cout<<setw(10)<<*it;
//    cout<<endl<<endl;
//
//    list<double>::iterator it2 = lDbl.end();
//    while(true){
//        it2--;
//        cout<<setw(10)<<*it2;
//        if(it2 == lDbl.begin())break;
//    }
//    cout<<endl<<endl;
//
//    lDbl.sort();
//    for(list<double>::iterator it= lDbl.begin(); it!=lDbl.end();it++)
//        cout<<setw(10)<<*it;
//    cout<<endl<<endl;
//    lDbl2.sort();
//
//    lDbl.merge(lDbl2);
//    for(list<double>::iterator it= lDbl.begin(); it!=lDbl.end();it++)
//        cout<<setw(10)<<*it;
```

```
67  //    cout<<endl<<endl;
68  //
69      list<class Persona> listPer;
70      ifstream arch("Personal.csv",ios::in);
71      if(not arch.is_open()){
72          cout<<"ERROR: No se pudo abrir el archivo "<<"Personal.csv"<<endl;
73          exit(1);
74      }
75      ofstream archRep("RepPersonal.txt",ios::out);
76      if(not archRep.is_open()){
77          cout<<"ERROR: No se pudo abrir el archivo "<<"RepPersonal.txt"<<endl;
78          exit(1);
79      }
80      class Persona persona;
81
82      while(true){
83          arch>>persona;
84          if(arch.eof())break;
85          listPer.push_back(persona);
86      }
87
88      listPer.sort();
89      for(Persona p: listPer)
90          archRep<<p;
91
92      return 0;
93  }
94
95  *
96   * Proyecto:  PlantillaDeClases
97   * Archivo: Persona.h
98   * Autor:   J. Miguel GuaniraErazo (Juan Miguel)
99   *
100  * Creado el 30 de junio de 2020, 06:23 PM
101  */
102
103 #ifndef PERSONA_H
104 #define PERSONA_H
105 #include <iostream>
106 class Persona {
107 private:
108     int dni;
109     char*nombre;
110     double sueldo;
111 public:
112     Persona();
113     Persona(const Persona& orig);
114     virtual ~Persona();
115     void SetSueldo(double sueldo);
116     double GetSueldo() const;
117     void SetNombre(const char* nombre);
118     void GetNombre(char* ) const;
119     void SetDni(int dni);
120     int GetDni() const;
121     void operator =(const class Persona&);
122     bool operator <(const class Persona&); //OBLIGATORIO PARA ORDENAR
123 };
124 void operator >> (istream &, class Persona&);
125 ostream& operator << (ostream &,const class Persona&);
126 #endif /* PERSONA_H */
127
128 /*
129  * Proyecto:  PlantillaDeClases
130  * Archivo: Persona.cpp
131  * Autor:   J. Miguel GuaniraErazo (Juan Miguel)
132  *
```

```
133     * Creado el 30 de junio de 2020, 06:23 PM
134     */
135    #include <iostream>
136    #include <iomanip>
137    using namespace std;
138    #include <cstring>
139    #include "Persona.h"
140
141    Persona::Persona() {
142        nombre = nullptr;
143    }
144
145    Persona::Persona(const Persona& orig) {
146        nombre = nullptr;
147        *this = orig;
148    }
149
150    Persona::~Persona() {
151        if (nombre != nullptr) delete nombre;
152    }
153
154    void Persona::SetSueldo(double sueldo) {
155        this->sueldo = sueldo;
156    }
157
158    double Persona::GetSueldo() const {
159        return sueldo;
160    }
161
162    void Persona::SetNombre(const char* nomb) {
163        if (nombre != nullptr) delete nombre;
164        nombre = new char[strlen(nomb)+1];
165        strcpy(nombre,nomb);
166    }
167
168    void Persona::GetNombre(char*nomb) const {
169        if(nombre == nullptr) nomb[0]=0;
170        else strcpy(nomb,nombre);
171    }
172
173    void Persona::SetDni(int dni) {
174        this->dni = dni;
175    }
176
177    int Persona::GetDni() const {
178        return dni;
179    }
180
181    void Persona::operator =(const class Persona&per) {
182        char nomb[60];
183        dni = per.dni;
184        per.GetNombre(nomb);
185        SetNombre(nomb);
186        sueldo = per.sueldo;
187    }
188
189    bool Persona::operator<(const class Persona&orig) {
190        return strcmp(nombre,orig.nombre)>0;
191    }
192
193    void operator >> (istream &in, class Persona&per){
194        int dni;
195        char nomb[60];
196        double sueldo;
197        in>>dni;
198        in.get(); //coma
```

```
199         in.getline(nomb,60,',');
200         in>>sueldo;
201         per.SetDni(dni);
202         per.SetNombre(nomb);
203         per.SetSueldo(sueldo);
204     }
205
206     ostream& operator << (ostream &out,const class Persona&per){
207         out.precision(2);
208         out<<fixed;
209         char nomb[60];
210         per.GetNombre(nomb);
211         out<<left<<setw(10)<<per.GetDni()<<setw(40)<<nomb
212                 <<right<<setw(10)<<per.GetSueldo()<<endl;
213         return out;
214     }
```