```
1   /*
2    * Proyecto: SolucionLaboratorio08_2023_1
3    * Archivo:  main.cpp
4    * Autor:    J. Miguel Guanira E.
5    *
6   < * Created on 18 de octubre de 2023, 08:51 PM
7    */
8
9   #include <iostream>
10  #include <iomanip>
11  using namespace std;
12  #include "Tesoreria.h"
13
14  int main(int argc, char** argv) {
15      class Tesoreria caja;  // {}
16
17      caja.cargaAlumnos("Alumnos.csv");
18      caja.actualizaBoleta("escalas.csv");
19      caja.imprimeBoleta("reporteDePagos.txt");
20
21      return 0;
22  }
23
24  /*
25   * Proyecto: SolucionLaboratorio089_2023_1
26   * Archivo:  Tesoreria.h
27   * Autor:    J. Miguel Guanira E.
28   *
29   * Creado el 29 de mayo de 2024, 10:40 AM
30   */
31
32
33  #ifndef TESORERIA_H
34  #define TESORERIA_H
35
36  #include "Arbol.h"
37
38  class Tesoreria {
39  private:
40      class Arbol aboleta;
41  public:
42      void cargaEscalas(const char *);
43      void cargaAlumnos(const char *);
44      void actualizaBoleta(const char *);
45      void imprimeBoleta(const char *);
46      void imprimeLinea(ofstream &, char, int);
47  };
48
49  #endif /* TESORERIA_H */
50
51  /*
52   * Proyecto: SolucionLaboratorio09_2023_1
53   * Archivo:  Tesoreria.cpp
54   * Autor:    J. Miguel Guanira E.
55   *
56   * Creado el 29 de mayo de 2024, 10:40 AM
57   */
58
59  #include <iostream>
60  #include <fstream>
61  #include <iomanip>
62  using namespace std;
63
64  #include "Tesoreria.h"
65
66  void Tesoreria::cargaAlumnos(const char*nombArch) {
```

```
67          aboleta.crear(nombArch);
68      }
69
70      void Tesoreria::actualizaBoleta(const char*nombArch) {
71          aboleta.cargaEscalas(nombArch);
72          aboleta.actualizaBoleta();
73      }
74
75      void Tesoreria::imprimeBoleta(const char*nombArch) {
76          aboleta.mostrarEnOrden(nombArch);
77      }
78
79      /*
80       * Proyecto: SolucionLaboratorio10_2023_1
81       * Archivo:  Nodo.h
82       * Autor:    J. Miguel Guanira E.
83       *
84       * Created on 4 de junio de 2024, 06:24 PM
85       */
86
87
88      #ifndef NODO_H
89      #define NODO_H
90
91      #include "Nodo.h"
92      #include "Boleta.h"
93      class Nodo {
94      private:
95          class Boleta dboleta;
96          class Nodo *izq;
97          class Nodo *der;
98      public:
99          Nodo();
100         friend class Arbol;
101     };
102
103     #endif /* NODO_H */
104
105     /*
106      * Proyecto: SolucionLaboratorio10_2023_1
107      * Archivo:  Nodo.cpp
108      * Autor:    J. Miguel Guanira E.
109      *
110      * Created on 4 de junio de 2024, 06:24 PM
111      */
112
113     #include <iostream>
114     #include <iomanip>
115     using namespace std;
116
117     #include "Nodo.h"
118
119     Nodo::Nodo() {
120         izq = nullptr;
121         der = nullptr;
122     }
123
124
125
126     /*
127      * Proyecto: SolucionLaboratorio10_2023_1
128      * Archivo:  Arbol.h
129      * Autor:    J. Miguel Guanira E.
130      *
131      * Created on 4 de junio de 2024, 06:26 PM
132      */
```

```
133
134
135 #ifndef ARBOL_H
136 #define ARBOL_H
137 #include <fstream>
138
139 #include "Nodo.h"
140 #include "Escala.h"
141
142 class Arbol {
143 private:
144     class Nodo *raiz;
145     class Escala lescala[10];
146
147     void insertarR(class Nodo *&arbol, const class Boleta &boleta);
148     void mostrarEnOrdenR(ofstream &arch, class Nodo *arbol);
149     void actualizaBoletaR(class Nodo *arbol);
150 public:
151     Arbol();
152     virtual ~Arbol();
153     void cargaEscalas(const char *);
154     void crear(const char*);
155     void actualizaBoleta();
156     void mostrarEnOrden(const char*);
157     void imprimeLinea(ofstream&, char , int );
158 };
159
160 #endif /* ARBOL_H */
161
162 /*
163  * Proyecto: SolucionLaboratorio10_2023_1
164  * Archivo:  Arbol.cpp
165  * Autor:    J. Miguel Guanira E.
166  *
167  * Created on 4 de junio de 2024, 06:26 PM
168  */
169
170 #include <iostream>
171 #include <iomanip>
172 using namespace std;
173 #include "Arbol.h"
174 #include "Boleta.h"
175
176 Arbol::Arbol() {
177     raiz = nullptr;
178 }
179
180 Arbol::~Arbol() {
181 //    eliminaR(raiz);
182 }
183
184 void Arbol::cargaEscalas(const char*nombArch) {
185     ifstream arch(nombArch, ios::in);
186     if(not arch.is_open()){
187         cout<<"No se pudo abrir el archivo "<<nombArch<<endl;
188         exit(1);
189     }
190     int cod;
191     double precio;
192
193     while(true){
194         arch>>cod;
195         if(arch.eof())break;
196         arch.get();
197         arch>>precio;
198         lescala[cod-1].SetCodigo(cod);
```

```
199             lescala[cod-1].SetPrecio(precio);
200         }
201     }
202
203     void Arbol::crear(const char*nombArch) {
204         ifstream arch(nombArch, ios::in);
205         if(not arch.is_open()){
206             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
207             exit(1);
208         }
209         class Boleta boleta;
210         char tipo;
211         while(true){
212             arch>>tipo;
213             if(arch.eof())break;
214             arch.get();
215             boleta.asignaMemoria(tipo);
216             boleta.leeDatos(arch);
217             insertarR(raiz,boleta);
218         }
219         boleta.SetNull();
220     }
221
222     void Arbol::insertarR(class Nodo*& arbol, const class Boleta& dato) {
223         if(arbol == nullptr){
224             arbol= new class Nodo;
225             arbol->dboleta = dato;
226             return;
227         }
228         if(arbol->dboleta > dato)
229             insertarR(arbol->izq, dato);
230         else
231             insertarR(arbol->der,dato);
232     }
233
234     void Arbol::mostrarEnOrden(const char*nombArch) {
235         ofstream arch(nombArch, ios::out);
236         if(not arch.is_open()){
237             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
238             exit(1);
239         }
240         arch<<left<<setw(10)<<"Codigo"<<setw(40)<<"Nombre"<<setw(8)<<"Escala"
241             <<setw(10)<<"Creditos"<<setw(12)<<"Licencia"<<setw(10)<<"Total"<<endl;
242
243         imprimeLinea(arch,'=',90);
244         mostrarEnOrdenR(arch,raiz);
245         cout<<endl;
246     }
247
248     void Arbol::mostrarEnOrdenR(ofstream &arch, class Nodo *raiz) {
249         if(raiz){
250             mostrarEnOrdenR(arch, raiz->izq);
251             arch<<raiz->dboleta;
252             mostrarEnOrdenR(arch, raiz->der);
253         }
254     }
255
256     void Arbol::imprimeLinea(ofstream&arch, char car, int nd) {
257         for(int i=0; i<nd; i++)
258             arch.put(car);
259         arch<<endl;
260     }
261
262     void Arbol::actualizaBoleta() {
263         actualizaBoletaR(raiz);
264     }
```

```
265
266 void Arbol::actualizaBoletaR(class Nodo *arbol) {
267     if(arbol == nullptr) return;
268     int esc = arbol->dboleta.GetEscala();
269     double precio = lescala[esc-1].GetPrecio();
270
271     arbol->dboleta.actualizaBoleta(precio);
272     actualizaBoletaR(arbol->izq);
273     actualizaBoletaR(arbol->der);
274 }
275
276 /*
277  * Proyecto: SolucionLaboratorio10_2023_1
278  * Archivo:  Escala.h
279  * Autor:    J. Miguel Guanira E.
280  *
281  * Creado el 04 de junio de 2024, 10:40 AM
282  */
283
284
285 #ifndef ESCALA_H
286 #define ESCALA_H
287
288 class Escala {
289 private:
290     int codigo;
291     double precio;
292 public:
293     Escala();
294     void SetPrecio(double precio);
295     double GetPrecio() const;
296     void SetCodigo(int codigo);
297     int GetCodigo() const;
298 };
299
300 #endif /* ESCALA_H */
301
302 /*
303  * Proyecto: SolucionLaboratorio10_2023_1
304  * Archivo:  Escala.cpp
305  * Autor:    J. Miguel Guanira E.
306  *
307  * Creado el 04 de junio de 2024, 10:40 AM
308  */
309
310 #include <iostream>
311 #include <iomanip>
312 using namespace std;
313
314 #include "Escala.h"
315
316 Escala::Escala() {
317     codigo = 0;
318 }
319
320 void Escala::SetPrecio(double precio) {
321     this->precio = precio;
322 }
323
324 double Escala::GetPrecio() const {
325     return precio;
326 }
327
328 void Escala::SetCodigo(int codigo) {
329     this->codigo = codigo;
330 }
```

```
331
332 int Escala::GetCodigo() const {
333     return codigo;
334 }
335
336 /*
337  * Proyecto: SolucionLaboratorio10_2023_1
338  * Archivo:  Boleta.h
339  * Autor:    J. Miguel Guanira E.
340  *
341  * Creado el 04 de junio de 2024, 10:40 AM
342  */
343
344
345 #ifndef BOLETA_H
346 #define BOLETA_H
347 #include <fstream>
348
349 #include "Alumno.h"
350
351 class Boleta {
352 private:
353     class Alumno *pboleta;
354 public:
355     Boleta();
356     virtual ~Boleta();
357     void asignaMemoria(char tipo);
358
359     void leeDatos(ifstream &arch) const;
360     void actualizaBoleta(double);
361     void imprimeBoleta(ofstream&) const;
362     bool operator >(const class Boleta &dato)const;
363     int GetCodigo()const;
364     int GetEscala()const;
365     void SetNull();
366 };
367
368 void operator >>(ifstream &arch, class Boleta &boleta);
369 void operator <<(ofstream &arch, const class Boleta &boleta);
370
371 #endif /* BOLETA_H */
372
373 /*
374  * Proyecto: SolucionLaboratorio10_2023_1
375  * Archivo:  Boleta.cpp
376  * Autor:    J. Miguel Guanira E.
377  *
378  * Creado el 04 de junio de 2024, 10:40 AM
379  */
380
381 #include <iostream>
382 #include <iomanip>
383 using namespace std;
384 #include "Presencial.h"
385 #include "Semipresencial.h"
386 #include "Virtual.h"
387
388 #include "Boleta.h"
389
390 Boleta::Boleta() {
391     pboleta = nullptr;
392 }
393
394 Boleta::~Boleta() {
395     if(pboleta!=nullptr) delete pboleta;
396 }
```

```
397
398 void Boleta::asignaMemoria(char tipo) {
399     switch(tipo){
400         case 'P':
401             pboleta = new class Presencial;
402             break;
403         case 'S':
404             pboleta = new class Semipresencial;
405             break;
406         case 'V':
407             pboleta = new class Virtual;
408             break;
409     }
410 }
411
412 void Boleta::leeDatos(ifstream& arch) const {
413     pboleta->lee(arch);  // Aquí aplicamos Polimorfismo
414 }
415
416 void Boleta::actualizaBoleta(double precioEscala) {
417     pboleta->actualizaTotal(precioEscala);
418 }
419
420 void Boleta::imprimeBoleta(ofstream&arch) const {
421     pboleta->imprime(arch);
422 }
423
424 bool Boleta::operator >(const class Boleta &dato)const { // <===== OJO
425     return pboleta->GetCodigo()>dato.GetCodigo();
426 }
427
428 int Boleta::GetCodigo() const{
429     return pboleta->GetCodigo();
430 }
431
432 int Boleta::GetEscala() const {
433     return pboleta->GetEscala();
434 }
435
436
437 void Boleta::SetNull() {
438     pboleta = nullptr;
439 }
440
441 void operator >>(ifstream &arch, class Boleta &boleta){
442     boleta.leeDatos(arch);
443 }
444
445 void operator <<(ofstream &arch, const class Boleta &boleta) {
446     boleta.imprimeBoleta(arch);
447 }
448
449 /*
450  * Proyecto: SolucionLaboratorio10_2023_1
451  * Archivo:  Alumno.h
452  * Autor:    J. Miguel Guanira E.
453  *
454  * Creado el 04 de junio de 2024, 10:40 AM
455  */
456
457 #ifndef ALUMNO_H
458 #define ALUMNO_H
459 #include <fstream>
460
461 class Alumno {
462 private:
```

```
463         int codigo;
464         char *nombre;
465         int escala ;
466         double creditos;
467         double total;
468     public:
469         Alumno();
470         virtual ~Alumno();
471         void SetTotal(double total);
472         double GetTotal() const;
473         void SetEscala(int escala);
474         int GetEscala() const;
475         void SetNombre(const char* nombre);
476         void GetNombre(char*) const;
477         void SetCodigo(int codigo);
478         int GetCodigo() const;
479         void SetCreditos(double creditos);
480         double GetCreditos() const;
481 
482         virtual void lee(ifstream &);
483         virtual void actualizaTotal(double);
484         virtual void imprime(ofstream&);
485     };
486 
487     #endif /* ALUMNO_H */
488 
489     /*
490      * Proyecto: SolucionLaboratorio10_2023_1
491      * Archivo:  Alumno.cpp
492      * Autor:    J. Miguel Guanira E.
493      *
494      * Creado el 04 de junio de 2024, 10:40 AM
495      */
496 
497     #include <iostream>
498     #include <fstream>
499     #include <iomanip>
500     using namespace std;
501     #include <cstring>
502     #include "Alumno.h"
503 
504     Alumno::Alumno() {
505         nombre = nullptr;
506         total = 0;
507     }
508 
509     Alumno::~Alumno() {
510         if(nombre!=nullptr) delete nombre;
511     }
512 
513     void Alumno::SetTotal(double total) {
514         this->total = total;
515     }
516 
517     double Alumno::GetTotal() const {
518         return total;
519     }
520 
521     void Alumno::SetEscala(int escala) {
522         this->escala = escala;
523     }
524 
525     int Alumno::GetEscala() const {
526         return escala;
527     }
528 
```

```
529  void Alumno::GetNombre(char*cad) const {
530      if(nombre==nullptr) cad[0]=0;
531      else strcpy(cad,nombre);
532  }
533
534  void Alumno::SetNombre(const char* cad) {
535      if(nombre!=nullptr) delete nombre;
536      nombre = new char[strlen(cad)+1];
537      strcpy(nombre,cad);
538  }
539
540  void Alumno::SetCodigo(int codigo) {
541      this->codigo = codigo;
542  }
543
544  int Alumno::GetCodigo() const {
545      return codigo;
546  }
547
548  void Alumno::SetCreditos(double creditos) {
549      this->creditos = creditos;
550  }
551
552  double Alumno::GetCreditos() const {
553      return creditos;
554  }
555
556  void Alumno::lee(ifstream&arch) {
557      char nomb[60],c;
558      arch>>codigo;
559      arch.get();
560      arch.getline(nomb,60,',');
561      SetNombre(nomb);
562      arch>>escala>>c>>creditos;
563      arch.get();
564  }
565
566  void Alumno::actualizaTotal(double pago) {
567      total = pago;  // también   SetTotal(pago);
568  }
569
570  void Alumno::imprime(ofstream&arch) {
571      arch.precision(2);
572      arch<<fixed;
573      arch<<left<<setw(10)<<codigo<<setw(40)<<nombre<<right<<setw(3)<<escala
574          <<setw(12)<<creditos;
575
576  }
577
578  /*
579   * Proyecto: SolucionLaboratorio10_2023_1
580   * Archivo:  Presencial.h
581   * Autor:    J. Miguel Guanira E.
582   *
583   * Creado el 04 de junio de 2024, 10:40 AM
584   */
585
586
587  #ifndef PRESENCIAL_H
588  #define PRESENCIAL_H
589  #include <fstream>
590  using namespace std;
591  #include "Alumno.h"
592
593  class Presencial : public Alumno{
594  private:
```

```
595         double recargo;
596         double total;
597     public:
598         Presencial();
599         void SetTotal(double total);
600         double GetTotal() const;
601         void SetRecargo(double recargo);
602         double GetRecargo() const;
603         void lee(ifstream &);
604         void actualizaTotal(double);
605         void imprime(ofstream&);
606     };
607
608     #endif /* PRESENCIAL_H */
609
610     /*
611      * Proyecto: SolucionLaboratorio10_2023_1
612      * Archivo:  Presencial.cpp
613      * Autor:    J. Miguel Guanira E.
614      *
615      * Creado el 04 de junio de 2024, 10:40 AM
616      */
617
618     #include <iostream>
619     #include <iomanip>
620     using namespace std;
621
622     #include "Presencial.h"
623
624     Presencial::Presencial() {
625         total = 0.0;
626     }
627
628     void Presencial::SetTotal(double total) {
629         this->total = total;
630     }
631
632     double Presencial::GetTotal() const {
633         return total;
634     }
635
636     void Presencial::SetRecargo(double recargo) {
637         this->recargo = recargo;
638     }
639
640     double Presencial::GetRecargo() const {
641         return recargo;
642     }
643
644     void Presencial::lee(ifstream&arch) {
645         Alumno::lee(arch);
646         arch>>recargo;
647         arch.get(); // Saco el cambio de línea porque la línea empieza en
648                     // un caracter
649     }
650
651     void Presencial::actualizaTotal(double precioEscala) {
652         total = precioEscala*GetCreditos()*(1+GetRecargo()/100.0);
653         Alumno::SetTotal(total);
654     }
655
656     void Presencial::imprime(ofstream&arch) {
657         Alumno::imprime(arch);
658         arch<<setw(22)<<total<<endl;
659     }
660
```

```
661  /*
662   * Proyecto: SolucionLaboratorio10_2023_1
663   * Archivo:  Semipresencial.h
664   * Autor:    J. Miguel Guanira E.
665   *
666   * Creado el 04 de junio de 2024, 10:40 AM
667   */
668
669
670  #ifndef SEMIPRESENCIAL_H
671  #define SEMIPRESENCIAL_H
672  #include <fstream>
673  using namespace std;
674  #include "Alumno.h"
675
676  class Semipresencial: public Alumno {
677  private:
678      double descuento;
679      double total;
680  public:
681      Semipresencial();
682      void SetTotal(double total);
683      double GetTotal() const;
684      void SetDescuento(double descuento);
685      double GetDescuento() const;
686      void lee(ifstream&);
687      void actualizaTotal(double);
688      void imprime(ofstream&);
689  };
690
691  #endif /* SEMIPRESENCIAL_H */
692
693  /*
694   * Proyecto: SolucionLaboratorio10_2023_1
695   * Archivo:  Semipresencial.cpp
696   * Autor:    J. Miguel Guanira E.
697   *
698   * Creado el 04 de junio de 2024, 10:40 AM
699   */
700
701  #include <iostream>
702  #include <iomanip>
703  using namespace std;
704
705  #include "Semipresencial.h"
706
707  Semipresencial::Semipresencial() {
708      total = 0.0;
709  }
710
711  void Semipresencial::SetTotal(double total) {
712      this->total = total;
713  }
714
715  double Semipresencial::GetTotal() const {
716      return total;
717  }
718
719  void Semipresencial::SetDescuento(double descuento) {
720      this->descuento = descuento;
721  }
722
723  double Semipresencial::GetDescuento() const {
724      return descuento;
725  }
726
```

```
727  void Semipresencial::lee(ifstream&arch) {
728      Alumno::lee(arch);
729      arch>>descuento;
730      arch.get(); // Saco el cambio de línea porque la línea empieza en
731                  // un caracter
732  }
733
734  void Semipresencial::actualizaTotal(double precioEscala) {
735      total = precioEscala*GetCreditos()*(1-descuento/100.0);
736      Alumno::SetTotal(total);
737  }
738
739  void Semipresencial::imprime(ofstream&arch) {
740      Alumno::imprime(arch);
741      arch<<setw(22)<<total<<endl;
742  }
743
744  /*
745   * Proyecto: SolucionLaboratorio10_2023_1
746   * Archivo:  Virtual.h
747   * Autor:    J. Miguel Guanira E.
748   *
749   * Creado el 04 de junio de 2024, 10:40 AM
750   */
751
752
753  #ifndef VIRTUAL_H
754  #define VIRTUAL_H
755  #include <fstream>
756  using namespace std;
757  #include "Alumno.h"
758
759  class Virtual: public Alumno {
760  private:
761      char *licencia;
762      double total;
763  public:
764      Virtual();
765      virtual ~Virtual();
766      void SetTotal(double total);
767      double GetTotal() const;
768      void SetLicencia(const char*);
769      void GetLicencia(char*) const;
770      void lee(ifstream&);
771      void actualizaTotal(double);
772      void imprime(ofstream&);
773  };
774
775  #endif /* VIRTUAL_H */
776
777  /*
778   * Proyecto: SolucionLaboratorio10_2023_1
779   * Archivo:  Virtual.cpp
780   * Autor:    J. Miguel Guanira E.
781   *
782   * Creado el 04 de junio de 2024, 10:40 AM
783   */
784
785  #include <iostream>
786  #include <iomanip>
787  using namespace std;
788  #include <cstring>
789  #include "Virtual.h"
790
791  Virtual::Virtual() {
792      licencia = nullptr;
```

```
793         total = 0.0;
794     }
795
796     Virtual::~Virtual() {
797         if(licencia!=nullptr) delete licencia;
798     }
799
800     void Virtual::SetTotal(double total) {
801         this->total = total;
802     }
803
804     double Virtual::GetTotal() const {
805         return total;
806     }
807
808     void Virtual::SetLicencia(const char* cad) {
809         if(licencia!=nullptr) delete licencia;
810         licencia = new char[strlen(cad)+1];
811         strcpy(licencia,cad);
812     }
813
814     void Virtual::GetLicencia(char*cad) const {
815         if(licencia==nullptr) cad[0]=0;
816         else strcpy(cad,licencia);
817     }
818
819     void Virtual::lee(ifstream&arch) {
820         char lic[10];
821         Alumno::lee(arch);
822         arch>>lic;
823         arch.get();
824         SetLicencia(lic);
825     }
826
827     void Virtual::actualizaTotal(double precioEscala) {
828         total = precioEscala*GetCreditos() + 100.0;
829         Alumno::SetTotal(total);
830     }
831
832     void Virtual::imprime(ofstream&arch) {
833         Alumno::imprime(arch);
834         arch<<setw(12)<<licencia<<setw(10)<<total<<endl;
835     }
836
```