```
1   /*
2    * Proyecto: SolucionLab10-2022-1
3    * Archivo:  main.cpp
4    * Autor:    J. Miguel Guanira E.
5    *
6    * Created on 17 de noviembre de 2022, 07:23 PM
7    */
8
9   #include "Promociones.h"
10
11  int main(int argc, char** argv) {
12      Promociones pro;
13
14      pro.leepedidos();
15      pro.actualizapedidos();
16      pro.imprimepedidos();
17
18  return 0;
19  }
20
21  /*
22   * Proyecto: SolucionLab10-2022-1
23   * Archivo:  Pedido.h
24   * Autor:    J. Miguel Guanira E.
25   *
26   * Created on 17 de noviembre de 2022, 07:25 PM
27   */
28
29
30  #ifndef PEDIDO_H
31  #define PEDIDO_H
32  #include <fstream>
33  using namespace std;
34
35  class Pedido {
36  private:
37      int codigo;
38      char *nombre;
39      int cantidad;
40      int dni;
41      int fecha;
42      double total;
43  public:
44      Pedido();
45      Pedido(const Pedido& orig);
46      virtual ~Pedido();
47      void SetTotal(double total);
48      double GetTotal() const;
49      void SetFecha(int fecha);
50      int GetFecha() const;
51      void SetDni(int dni);
52      int GetDni() const;
53      void SetCantidad(int cantidad);
54      int GetCantidad() const;
55      void SetNombre(const char* nombre);
56      void GetNombre(char*) const;
57      void SetCodigo(int codigo);
58      int GetCodigo() const;
59      void operator =(const Pedido& orig);
```

```
60          virtual void lee(ifstream &);
61          virtual void imprime(ofstream &);
62          void fecha_dd_mm_aa(int&,int&,int&);
63          virtual void descuentaFlete()=0;
64      };
65
66      #endif /* PEDIDO_H */
67
68      /*
69       * Proyecto: SolucionLab10-2022-1
70       * Archivo:  Pedido.cpp
71       * Autor:    J. Miguel Guanira E.
72       *
73       * Created on 17 de noviembre de 2022, 07:25 PM
74       */
75
76      #include <iostream>
77      #include <fstream>
78      #include <iomanip>
79      #include <cstring>
80      using namespace std;
81
82      #include "Pedido.h"
83
84      Pedido::Pedido() {
85          nombre = nullptr;
86      }
87
88      Pedido::Pedido(const Pedido& orig) {
89          nombre = nullptr;
90          *this = orig;
91      }
92
93      Pedido::~Pedido() {
94          if(nombre != nullptr) delete nombre;
95      }
96
97      void Pedido::operator=(const Pedido& orig) {
98          char nomb[60];
99          codigo = orig.dni;
100         orig.GetNombre(nomb);
101         SetNombre(nomb);
102         cantidad = orig.cantidad;
103         dni = orig.dni;
104         fecha = orig.fecha;
105         total = orig.total;
106     }
107
108     void Pedido::SetTotal(double total) {
109         this->total = total;
110     }
111
112     double Pedido::GetTotal() const {
113         return total;
114     }
115
116     void Pedido::SetFecha(int fecha) {
117         this->fecha = fecha;
118     }
```

```
119
120 int Pedido::GetFecha() const {
121     return fecha;
122 }
123
124 void Pedido::SetDni(int dni) {
125     this->dni = dni;
126 }
127
128 int Pedido::GetDni() const {
129     return dni;
130 }
131
132 void Pedido::SetCantidad(int cantidad) {
133     this->cantidad = cantidad;
134 }
135
136 int Pedido::GetCantidad() const {
137     return cantidad;
138 }
139
140 void Pedido::SetNombre(const char* nomb) {
141    if(nombre!=nullptr) delete nombre;
142    nombre = new char[strlen(nomb)+1];
143    strcpy(nombre,nomb);
144 }
145
146 void Pedido::GetNombre(char*nomb) const {
147     if(nombre==nullptr) nomb[0]=0;
148     strcpy(nomb,nombre);
149 }
150
151 void Pedido::SetCodigo(int codigo) {
152     this->codigo = codigo;
153 }
154
155 int Pedido::GetCodigo() const {
156     return codigo;
157 }
158
159 void Pedido::lee(ifstream&arch) {
160     char cad[100],c;
161     int dd,mm,aa;
162
163     arch.getline(cad,100,',');
164     SetNombre(cad);
165     arch>>cantidad>>c>>total>>c>>dni>>c>>dd>>c>>mm>>c>>aa;
166     fecha = 10000*aa+100*mm+dd;
167 }
168
169 void Pedido::imprime(ofstream&arch) {
170     int dd,mm,aa;
171     arch.precision(2);
172     arch<<fixed;
173     arch<<endl;
174     fecha_dd_mm_aa(dd,mm,aa);
175     arch<<right<<setfill('0')<<setw(2)<<dd<<'/'<<setw(2)<<mm<<'/'<<setw(4)<<aa
176         <<setfill(' ')<<endl;
177     arch<<left<<setw(10)<<codigo<<nombre<<endl;
```

```
178         arch<<left<<setw(15)<<"DNI:"<<right<<setw(10)<<dni<<endl;
179         arch<<left<<setw(15)<<"Monto Total:"<<right<<setw(10)<<total<<endl;
180     }
181
182     void Pedido::fecha_dd_mm_aa(int &dd, int &mm, int &aa) {
183         int f;
184         aa = fecha/10000;
185         f = fecha%10000;
186         mm = f/100;
187         dd = f%100;
188     }
189
190     /*
191      * Proyecto: SolucionLab10-2022-1
192      * Archivo:  PedidoEspecial.h
193      * Autor:    J. Miguel Guanira E.
194      *
195      * Created on 17 de noviembre de 2022, 07:40 PM
196      */
197
198
199     #ifndef PEDIDOESPECIAL_H
200     #define PEDIDOESPECIAL_H
201     #include "Pedido.h"
202
203     class PedidoEspecial : public Pedido{
204     private:
205         double descuento;
206     public:
207         void SetDescuento(double descuento);
208         double GetDescuento() const;
209         void lee(ifstream &);
210         void imprime(ofstream &);
211         void descuentaFlete();
212     };
213
214     #endif /* PEDIDOESPECIAL_H */
215
216     /*
217      * Proyecto: SolucionLab10-2022-1
218      * Archivo:  PedidoEspecial.cpp
219      * Autor:    J. Miguel Guanira E.
220      *
221      * Created on 17 de noviembre de 2022, 07:40 PM
222      */
223
224     #include <iostream>
225     #include <fstream>
226     #include <iomanip>
227     using namespace std;
228
229     #include "PedidoEspecial.h"
230
231     void PedidoEspecial::SetDescuento(double descuento) {
232         this->descuento = descuento;
233     }
234
235     double PedidoEspecial::GetDescuento() const {
236         return descuento;
```

```
237 }
238
239 void PedidoEspecial::lee(ifstream &arch) {
240     char c;
241     double monto;
242     arch>>descuento>>c;
243     Pedido::lee(arch);
244     monto=GetTotal();
245     monto *= (1-descuento/100);
246     SetTotal(monto);
247 }
248
249 void PedidoEspecial::imprime(ofstream&arch) {
250     Pedido::imprime(arch);
251     arch<<left<<setw(15)<<"Descuento:"<<right<<setw(10)<<descuento<<'%'<<endl;
252 }
253
254 void PedidoEspecial::descuentaFlete() {}
255 //No tiene código porque no hay flete que descontar
256
257 /*
258  * Proyecto: SolucionLab10-2022-1
259  * Archivo:  PedidoUsual.h
260  * Autor:    J. Miguel Guanira E.
261  *
262  * Created on 17 de noviembre de 2022, 07:48 PM
263  */
264
265
266 #ifndef PEDIDOUSUAL_H
267 #define PEDIDOUSUAL_H
268 #include "Pedido.h"
269 class PedidoUsual : public Pedido{
270 private:
271     double descuento;
272     double flete;
273 public:
274     void SetFlete(double flete);
275     double GetFlete() const;
276     void SetDescuento(double descuento);
277     double GetDescuento() const;
278     void lee(ifstream &);
279     void imprime(ofstream &);
280     void descuentaFlete();
281 };
282
283 #endif /* PEDIDOUSUAL_H */
284
285 /*
286  * Proyecto: SolucionLab10-2022-1
287  * Archivo:  PedidoUsual.cpp
288  * Autor:    J. Miguel Guanira E.
289  *
290  * Created on 17 de noviembre de 2022, 07:48 PM
291  */
292
293 #include <iostream>
294 #include <fstream>
295 #include <iomanip>
```

```
296  using namespace std;
297
298  #include "PedidoUsual.h"
299
300  void PedidoUsual::SetFlete(double flete) {
301      this->flete = flete;
302  }
303
304  double PedidoUsual::GetFlete() const {
305      return flete;
306  }
307
308  void PedidoUsual::SetDescuento(double descuento) {
309      this->descuento = descuento;
310  }
311
312  double PedidoUsual::GetDescuento() const {
313      return descuento;
314  }
315
316  void PedidoUsual::lee(ifstream&arch) {
317      char c;
318      double monto;
319      arch>>descuento>>c>>flete>>c;
320      Pedido::lee(arch);
321      monto=GetTotal();
322      monto *= (1-descuento/100);
323      monto *= (1-flete/100);
324      SetTotal(monto);
325  }
326
327  void PedidoUsual::imprime(ofstream&arch) {
328      Pedido::imprime(arch);
329      arch<<left<<setw(15)<<"Descuento:"<<right<<setw(10)<<descuento<<'%'<<endl;
330      arch<<left<<setw(15)<<"Flete:"<<right<<setw(10)<<flete<<'%'<<endl;
331  }
332
333  void PedidoUsual::descuentaFlete() {
334      double monto;
335      monto=GetTotal();
336      monto /= (1-flete/100);
337      flete = 0.0;
338      SetTotal(monto);
339  }
340
341  /*
342   * Proyecto: SolucionLab10-2022-1
343   * Archivo:  PedidoEventual.h
344   * Autor:    J. Miguel Guanira E.
345   *
346   * Created on 17 de noviembre de 2022, 07:52 PM
347   */
348
349
350  #ifndef PEDIDOEVENTUAL_H
351  #define PEDIDOEVENTUAL_H
352  #include "Pedido.h"
353
354  class PedidoEventual : public Pedido{
```

```
355   private:
356       double flete;
357   public:
358       void SetFlete(double flete);
359       double GetFlete() const;
360       void lee(ifstream &);
361       void imprime(ofstream &);
362       void descuentaFlete();
363   };
364
365   #endif /* PEDIDOEVENTUAL_H */
366
367   /*
368    * Proyecto: SolucionLab10-2022-1
369    * Archivo:  PedidoEventual.cpp
370    * Autor:    J. Miguel Guanira E.
371    *
372    * Created on 17 de noviembre de 2022, 07:52 PM
373    */
374
375   #include <iostream>
376   #include <fstream>
377   #include <iomanip>
378   using namespace std;
379
380   #include "PedidoEventual.h"
381
382   void PedidoEventual::SetFlete(double flete) {
383       this->flete = flete;
384   }
385
386   double PedidoEventual::GetFlete() const {
387       return flete;
388   }
389
390   void PedidoEventual::lee(ifstream&arch) {
391       char c;
392       double monto;
393       arch>>flete>>c;
394       Pedido::lee(arch);
395       monto=GetTotal();
396       monto *= (1-flete/100);
397       SetTotal(monto);
398   }
399
400   void PedidoEventual::imprime(ofstream&arch) {
401       Pedido::imprime(arch);
402       arch<<left<<setw(15)<<"Flete:"<<right<<setw(10)<<flete<<'%'<<endl;
403
404   }
405
406   void PedidoEventual::descuentaFlete() {
407       double monto;
408       monto=GetTotal();
409       monto /= (1-flete/100);
410       SetTotal(monto);
411       flete = 0;
412   }
413
```

```
414  /*
415   * Proyecto: SolucionLab10-2022-1
416   * Archivo:  Nodo.h
417   * Autor:    J. Miguel Guanira E.
418   *
419   * Created on 17 de noviembre de 2022, 07:56 PM
420   */
421
422
423  #ifndef NODO_H
424  #define NODO_H
425  #include "Pedido.h"
426  #include "Lista.h"
427
428  class Nodo {
429  private:
430      class Pedido *ped;
431      class Nodo *sig;
432      class Nodo *ant;
433  public:
434      Nodo(){
435          sig = ant = nullptr;
436      }
437      friend class Lista;
438  };
439
440  #endif /* NODO_H */
441
442  /*
443   * Proyecto: SolucionLab10-2022-1
444   * Archivo:  Lista.h
445   * Autor:    J. Miguel Guanira E.
446   *
447   * Created on 17 de noviembre de 2022, 07:58 PM
448   */
449
450
451  #ifndef LISTA_H
452  #define LISTA_H
453  #include "Nodo.h"
454
455  class Lista {
456  private:
457      class Nodo *lini;
458      class Nodo *lfin;
459      void insertar(class Nodo*); // para preservar el encapsulamiento
460  public:
461      Lista();
462      virtual ~Lista();
463      void crearLista(ifstream &);
464      void imprimeLista(ofstream &);
465      void actualiza(ifstream &);
466      void actualizaCliente(int,int);
467  };
468
469  #endif /* LISTA_H */
470
471  /*
472   * Proyecto: SolucionLab10-2022-1
```

```
473     * Archivo:  Lista.cpp
474     * Autor:    J. Miguel Guanira E.
475     *
476     * Created on 17 de noviembre de 2022, 07:58 PM
477     */
478
479   #include <iostream>
480   #include <fstream>
481   #include <iomanip>
482   using namespace std;
483
484   #include "Lista.h"
485   #include "PedidoEspecial.h"
486   #include "PedidoEventual.h"
487   #include "PedidoUsual.h"
488
489   Lista::Lista() {
490       lini = nullptr;
491       lfin = nullptr;
492   }
493
494   Lista::~Lista() {
495       class Nodo *sale;
496       while(lini){
497           sale = lini;
498           lini = lini->sig;
499           delete sale;
500       }
501   }
502
503   void Lista::crearLista(ifstream &arch) {
504       int cod;
505       char c;
506       class Nodo *nuevo;
507
508       while(true){
509           arch>>cod;
510           if(arch.eof())break;
511           arch>>c;
512           nuevo = new class Nodo;
513           if(cod<400000) nuevo->ped = new class PedidoEspecial;
514           else if(cod<600000) nuevo->ped = new class PedidoUsual;
515           else nuevo->ped = new class PedidoEventual;
516           nuevo->ped->SetCodigo(cod);
517           nuevo->ped->lee(arch);  //Usamos el polimorfismo
518           insertar(nuevo);
519       }
520   }
521
522   void Lista::insertar(class Nodo*nuevo) {
523       class Nodo *p = lini, *ant = nullptr;
524       if(lini == nullptr) {
525           lini = nuevo;
526           lfin = nuevo;
527       }
528       else {
529           while(p){
530               if(p->ped->GetDni() > nuevo->ped->GetDni() or
531                  p->ped->GetDni() == nuevo->ped->GetDni() and
```

```
532                 p->ped->GetFecha() > nuevo->ped->GetFecha() )break;
533             ant = p;
534             p = p->sig;
535         }
536         nuevo->sig = p;
537         nuevo->ant = ant;
538         if(ant!=nullptr) ant->sig = nuevo;
539         else lini = nuevo;
540         if (p!=nullptr) p->ant =nuevo;
541         else lfin = nuevo;
542     }
543 }
544 
545 void Lista::imprimeLista(ofstream&arch) {
546     class Nodo *p = lini;
547     while(p){
548         p->ped->imprime(arch);  //aplicamos el polimorfismo
549         p = p->sig;
550     }
551 }
552 
553 void Lista::actualiza(ifstream&arch) {
554     int dni,fecha,dd,mm,aa;
555     char c;
556     while(true){
557         arch>>dni;
558         if(arch.eof())break;
559         arch>>c>>dd>>c>>mm>>c>>aa;
560         fecha = 10000*aa+100*mm+dd;
561         actualizaCliente(dni,fecha);
562     }
563 }
564 
565 void Lista::actualizaCliente(int dni, int fecha) {
566     class Nodo *p = lfin;
567 
568     while(p){
569         if(p->ped->GetDni() == dni and p->ped->GetFecha()<fecha)break;
570         p = p->ant;
571     }
572     while(p and p->ped->GetDni()==dni){
573         p->ped->descuentaFlete();
574         p = p->ant;
575     }
576 }
577 
578 /*
579  * Proyecto: SolucionLab10-2022-1
580  * Archivo:  Promociones.h
581  * Autor:    J. Miguel Guanira E.
582  *
583  * Created on 17 de noviembre de 2022, 08:04 PM
584  */
585 
586 
587 #ifndef PROMOCIONES_H
588 #define PROMOCIONES_H
589 #include "Lista.h"
590 
```

```
591  class Promociones {
592  private:
593      class Lista lpedidos;
594  public:
595      void leepedidos();
596      void actualizapedidos();
597      void imprimepedidos();
598      void imprimeLinea(ofstream&,char,int);
599  };
600
601  #endif /* PROMOCIONES_H */
602
603  /*
604   * Proyecto: SolucionLab10-2022-1
605   * Archivo:  Promociones.cpp
606   * Autor:    J. Miguel Guanira E.
607   *
608   * Created on 17 de noviembre de 2022, 08:04 PM
609   */
610
611  #include <iostream>
612  #include <fstream>
613  #include <iomanip>
614  using namespace std;
615
616  #include "Promociones.h"
617
618  void Promociones::leepedidos() {
619      ifstream arch("Pedidos5.csv",ios::in);
620      if(not arch.is_open()){
621          cout<<"Error: no se pudo abrir el archivo Pedidos5.csv"<<endl;
622          exit(1);
623      }
624      lpedidos.crearLista(arch);
625  }
626
627  void Promociones::actualizapedidos() {
628      ifstream arch("promocion.csv",ios::in);
629      if(not arch.is_open()){
630          cout<<"Error: no se pudo abrir el archivo promocion.csv"<<endl;
631          exit(1);
632      }
633      lpedidos.actualiza(arch);
634  }
635
636  void Promociones::imprimepedidos() {
637      ofstream arch("ReportePedidos.txt",ios::out);
638      if(not arch.is_open()){
639          cout<<"Error: no se pudo abrir el archivo ReportePedidos.txt"<<endl;
640          exit(1);
641      }
642      arch<<right<<setw(36)<<"REPORTE DE PROMOCIONES"<<endl;
643      imprimeLinea(arch,'=',50);
644      lpedidos.imprimeLista(arch);
645  }
646
647  void Promociones::imprimeLinea(ofstream &arch, char c, int n) {
648      for(int i=0; i<n; i++) arch.put(c);
649      arch<<endl;
```

```
650     }
651
```