```
1   /*
2    * Proyecto: EX01_PREG02_PuntGenericos
3    * Archivo:  main.cpp
4    * Autor:    J. Miguel Guanira E.
5    *
6    * Created on 2 de octubre de 2021, 08:23 PM
7    */
8
9   #include "FuncionesEx01_20211_Preg02.h"
10  int main(int argc, char** argv) {
11      void *conductor, *falta, *consolidado;
12      cargadatos(conductor, falta);
13      comprobarCarga(conductor, falta);
14
15      registrafaltas(consolidado, conductor, falta);
16      imprime(consolidado);
17      return 0;
18  }
19
20  /*
21   * Proyecto: EX01_PREG02_PuntGenericos
22   * Archivo:  FuncionesEx01_20211_Preg02.h
23   * Autor:    J. Miguel Guanira E.
24   *
25   * Created on 2 de octubre de 2021, 08:26 PM
26   */
27
28  #ifndef FUNCIONESEX01_20211_PREG02_H
29  #define FUNCIONESEX01_20211_PREG02_H
30  #include <fstream>
31  using namespace std;
32  void cargadatos(void *&,void *&);
33  void comprobarCarga(void *,void *);
34  void mostrarConductor(void *);
35  void mostrarFalta(void*);
36  void registrafaltas(void *&,void *,void *);
37  void imprime(void *);
38  void cargarConductores(void *&);
39  void cargarFaltas(void *&);
40  void * leeConductor(ifstream &);
41  void * leeFalta(ifstream &);
42  int buscarConductor(int ,void **);
43  void leerDatos(ifstream &,int &,char*,int &,int &,double &,void **);
44  double obtenerMonto(int ,void **);
45  int buscarConsolidado(int, int ,int ,void **);
46  void aumentarMonto(double, void *);
47  void obtenerNombre(int ,void **,char*&);
48  void colocarConsolidado(int,char*,int,int,double,void**,int&);
49  void imprimeConsolidado(ofstream &, void **, int &);
50  void imprimeLinea(ofstream &,char ,int );
51
52  #endif /* FUNCIONESEX01_20211_PREG02_H */
53
54  /*
55   * Proyecto: EX01_PREG02_PuntGenericos
56   * Archivo:  FuncionesEx01_20211_Preg02.cpp
57   * Autor:    J. Miguel Guanira E.
58   *
59   * Created on 2 de octubre de 2021, 08:26 PM
60   */
61
62  #include <iostream>
63  #include <iomanip>
64  #include <fstream>
65  using namespace std;
66  #include <cstring>
```

```
67  #include <cmath>
68  #include "FuncionesEx01_20211_Preg02.h"
69  #define MAX_CAR_LIN 70
70  enum CONDUCTOR {LIC,NOMB};
71  enum INFRACCION {COD,DESC,MONT};
72  enum CONSOLIDADO {LICENCIA,NOMBRE,AA,MM,MONTO};
73
74  void cargadatos(void *&conductor,void *&falta){
75      cargarConductores(conductor);
76      cargarFaltas(falta);
77  }
78
79  void cargarConductores(void *&conductor){
80      void *aux[200], **cond;
81      int nd = 0;
82      ifstream arch("Conductores.csv",ios::in);
83      if(!arch.is_open()){
84          cout<<"ERROR: No se pudo abrir el archivo Conductores.csv"<<endl;
85          exit(1);
86      }
87      while(true){
88          aux[nd] = leeConductor(arch);
89          if(aux[nd] == nullptr) break;
90          nd++;
91      }
92      cond = new void*[nd+1];
93      for(int i=0; i<nd; i++)
94          cond[i] = aux[i];
95      cond[nd] =nullptr;
96      conductor = cond;
97  }
98
99  void * leeConductor(ifstream &arch){
100     int d, *dni;
101     char nomb[60], *nombre;
102     void **reg;
103     arch>>d;
104     if (arch.eof()) return nullptr;
105     dni = new int;
106     *dni = d;
107     arch.get(); // saco la coma
108     arch.getline(nomb,60);
109     nombre = new char [strlen(nomb)+1];
110     strcpy(nombre,nomb);
111     reg = new void*[2];
112     reg[CONDUCTOR::LIC] = dni;
113     reg[CONDUCTOR::NOMB] = nombre;
114     return reg;
115 }
116
117 void cargarFaltas(void *&falta){
118     void *aux[300], **falt;
119     int nd = 0;
120     ifstream arch("Infracciones.csv",ios::in);
121     if(!arch.is_open()){
122         cout<<"ERROR: No se pudo abrir el archivo Infracciones.csv"<<endl;
123         exit(1);
124     }
125     while(true){
126         aux[nd] = leeFalta(arch);
127         if(aux[nd] == nullptr) break;
128         nd++;
129     }
130     falt = new void*[nd+1];
131     for(int i=0; i<nd; i++)
132         falt[i] = aux[i];
```

```
133         falt[nd] = nullptr;
134         falta = falt;
135     }
136
137     void * leeFalta(ifstream &arch){
138         int cod, *codigo;
139         char desc[500], *descripcion, gravedad[20];
140         double *monto;
141         void **reg;
142         arch>>cod;
143         if (arch.eof()) return nullptr;
144         codigo = new int;
145         *codigo = cod;
146         arch.get(); // saco la coma
147         arch.getline(desc,500,',');
148         descripcion = new char [strlen(desc)+1];
149         strcpy(descripcion,desc);
150         arch.getline(gravedad,20,',');
151         monto = new double;
152         arch>>*monto;
153         reg = new void*[3];
154         reg[INFRACCION::COD] = codigo;
155         reg[INFRACCION::DESC] = descripcion;
156         reg[INFRACCION::MONT] = monto;
157         return reg;
158     }
159
160     void comprobarCarga(void *cond,void *fal){
161         void **conductor = (void**)cond, **falta = (void**)fal;
162         cout<<"Conductores"<<endl;
163         for(int i = 0; conductor[i]; i++){
164             mostrarConductor(conductor[i]);
165         }
166         cout<<endl<<"Faltas"<<endl;
167         for(int i = 0; falta[i]; i++){
168             mostrarFalta( falta[i]);
169         }
170
171     }
172
173     void mostrarConductor(void *cond){
174         void **conductor = (void**)cond;
175         int *dni;
176         char*nombre;
177         dni = (int*)conductor[CONDUCTOR::LIC];
178         nombre = (char*)conductor[CONDUCTOR::NOMB];
179         cout<<left<<setw(10)<<*dni<<setw(45)<<nombre<<endl;
180     }
181
182     void mostrarFalta(void *fal){
183         void **falta = (void**)fal;
184         int *codigo;
185         char*descripcion;
186         double *monto;
187         cout.precision(2);
188         cout<<fixed;
189
190         codigo = (int*)falta[INFRACCION::COD];
191         descripcion = (char*)falta[INFRACCION::DESC];
192         monto = (double*)falta[INFRACCION::MONT];
193         cout<<left<<setw(10)<<*codigo<<setw(10)<<*monto
194                 <<setw(45)<<descripcion<<endl;
195     }
196
197     void registrafaltas(void *&cons,void *cond, void *fal){
198         void **conductor = (void**)cond, **falta = (void**)fal,
```

```
199                 **consolidado, *auxCons[700]{};
200         int nd=0, licencia, pos, mm, aa;
201         char placa[8], *nombreCond;
202         double monto;
203 
204         ifstream arch("RegistroDeFaltas.csv",ios::in);
205         if(!arch.is_open()){
206             cout<<"ERROR: No se pudo abrir el archivo RegistroDeFaltas.csv"<<endl;
207             exit(1);
208         }
209 
210         while(true){
211             leerDatos(arch,licencia,placa,mm,aa,monto,falta);
212             if(arch.eof())break;
213             pos = buscarConsolidado(licencia,mm,aa,auxCons);
214             if (pos != -1){ // Lo encontró
215                 aumentarMonto(monto, auxCons[pos]);
216             }
217             else{ //No lo encontró
218                 obtenerNombre(licencia,conductor,nombreCond);
219                 colocarConsolidado(licencia,nombreCond,mm,aa,monto,auxCons,nd);
220             }
221         }
222         consolidado = new void*[nd+1];
223         for(int i=0; auxCons[i]; i++) consolidado[i] = auxCons[i];
224         consolidado[nd] = nullptr;
225         cons = consolidado;
226     }
227 
228     void leerDatos(ifstream &arch, int &licencia, char*placa,
229                    int &mm,int &aa, double &monto, void **falta){
230         int dd, codFalta;
231         char car;
232         arch>>licencia;
233         if(arch.eof())return;
234         arch.get();
235         arch.getline(placa,8,',');
236         arch>>dd>>car>>mm>>car>>aa>>car>>codFalta;
237         monto = obtenerMonto(codFalta,falta);
238     }
239 
240     double obtenerMonto(int codFalta, void **falta){
241         void**reg;
242         int *codigo;
243         double *monto;
244         for (int i=0; falta[i]; i++){
245             reg = (void**)falta[i];
246             codigo = (int*)reg[INFRACCION::COD];
247             if(codFalta == *codigo) {
248                 monto = (double*)reg[INFRACCION::MONT];
249                 return *monto;
250             }
251         }
252         return -1.0;
253     }
254 
255     int buscarConsolidado(int licencia, int mm, int aa, void **auxCons){
256         void **reg;
257         int*lic, *mM, *aA;
258 
259         for (int i=0; auxCons[i]; i++){
260             reg = (void**)auxCons[i];
261             lic = (int*)reg[CONSOLIDADO::LICENCIA];
262             mM = (int*)reg[CONSOLIDADO::MM];
263             aA = (int*)reg[CONSOLIDADO::AA];
264             if (licencia == *lic && mm == *mM && aa == *aA) return i;
```

```
265         }
266         return -1;
267     }
268
269     void aumentarMonto(double monto, void *cons){
270         void **reg = (void**)cons;
271         double *mont;
272         mont = (double*)reg[CONSOLIDADO::MONTO];
273         (*mont) += monto;
274     }
275
276     void obtenerNombre(int licencia,void **conductor,char*&nombreCond){
277         void **reg;
278         int *lic;
279         char *nombre;
280         for(int i=0; conductor[i]; i++){
281             reg = (void**)conductor[i];
282             lic = (int*)reg[CONDUCTOR::LIC];
283             if(*lic == licencia){
284                 nombre = (char*)reg[CONDUCTOR::NOMB];
285                 nombreCond = new  char[strlen(nombre)+1];
286                 strcpy(nombreCond,nombre);
287             }
288         }
289     }
290
291     void colocarConsolidado(int licencia,char *nombreCond,int mm,int aa,
292                             double monto,void**auxCons,int &nd){
293         void **reg;
294         int i = nd, *lic, *m, *a;
295         char *nombre;
296         double *mont;
297         while(true){
298             i--;
299             if(i<0) break;
300             reg = (void**)auxCons[i];
301             lic = (int*)reg[CONSOLIDADO::LICENCIA];
302             m = (int*)reg[CONSOLIDADO::MM];
303             a = (int*)reg[CONSOLIDADO::AA];
304             if(*lic<licencia or *lic==licencia and *m<mm or
305                     *lic==licencia and *m==mm and *a<aa) break;
306             auxCons[i+1] = auxCons[i];
307         }
308         lic = new int;  *lic = licencia;
309         nombre = new char[strlen(nombreCond)+1];
310         strcpy(nombre,nombreCond);
311         m = new int;  *m = mm;
312         a = new int;  *a = aa;
313         mont = new double;  *mont = monto;
314
315         reg = new void*[5];
316         reg[CONSOLIDADO::LICENCIA] = lic;
317         reg[CONSOLIDADO::NOMBRE] = nombre;
318         reg[CONSOLIDADO::MM] = m;
319         reg[CONSOLIDADO::AA] = a;
320         reg[CONSOLIDADO::MONTO] = mont;
321         auxCons[i+1] = reg;
322         nd++;
323     }
324
325     void imprime(void *cons){
326         void **consolidado =(void**)cons;
327         ofstream arch("RepCosolidadoMultas.txt",ios::out);
328         if(!arch.is_open()){
329             cout<<"ERROR: No se pudo abrir el archivo RepCosolidadoMultas.csv"
330                 <<endl;
```

```
331             exit(1);
332         }
333         arch.precision(2);
334         arch<<fixed;
335         arch<<setw(50)<<"REPORTE CONSOLIDADO DE MULTAS"<<endl;
336         for(int i=0; consolidado[i]; i++)
337             imprimeConsolidado(arch, consolidado,i);
338     }
339
340     void imprimeConsolidado(ofstream &arch, void **cons, int &i){
341         void **reg =(void**)cons[i];
342         int *lic = (int*)reg[CONSOLIDADO::LICENCIA], *m,*a, licencia;
343         char*nomb = (char *)reg[CONSOLIDADO::NOMBRE];
344         double *monto;
345         licencia = *lic;
346
347         arch<<left<<"Licencia: "<<setw(15)<<*lic<<"Nombre: "<<nomb<<endl;
348         imprimeLinea(arch,'=',MAX_CAR_LIN);
349         arch<<setw(20)<<"Año"<<setw(20)<<"Mes"<<setw(20)<<"Monto"<<endl;
350         imprimeLinea(arch,'=',MAX_CAR_LIN);
351         while (true){
352             m = (int*)reg[CONSOLIDADO::MM];
353             a = (int*)reg[CONSOLIDADO::AA];
354             monto = (double*)reg[CONSOLIDADO::MONTO];
355             arch<<left<<setw(20)<<*a<<setw(2)<<*m<<right<<setw(20)<<*monto<<endl;
356             if(cons[i+1]==nullptr)break;
357             reg =(void**)cons[i+1];
358             lic = (int*)reg[CONSOLIDADO::LICENCIA];
359             if (licencia != *lic) break;
360             i++;
361         }
362     }
363
364     void imprimeLinea(ofstream &arch,char car,int n){
365         for(int i=0; i<n; i++) arch<<car;
366         arch<<endl;
367     }
```