```cpp
/*
 * Proyecto: PunterosGenericos-Aplicacion
 * Archivo:  main.cpp
 * Autor:    J. Miguel Guanira E.//miguel.guanira.
 *
 * Created on 9 de abril de 2024, 09:45 AM
 */

#include <iostream>
#include <iomanip>
using namespace std;
#include "FuncionesAuxiliares.h"

int main(int argc, char** argv) {
    void *alumnos;

    cargarAlumnos(alumnos,"Alumnos.csv");
    cargarNotas(alumnos,"CursosNotas.csv");

    calcularPromedios(alumnos);
    probarCarga(alumnos);

    ordenar(alumnos);
    cout<<endl<<endl<<"Alumnos ordenados"<<endl;
    probarCarga(alumnos);

    // SOLO SI SE CONTINUA CON OTRo JUEGO DE DATOS
    eliminarDatos(alumnos);
    return 0;
}

/*
 * Proyecto: PunterosGenericos-Aplicacion
 * Archivo:  FuncionesAuxiliares.h
 * Autor:    J. Miguel Guanira E. //miguel.guanira.
 *
 * Created on 9 de abril de 2024, 09:47 AM
 */

#ifndef FUNCIONESAUXILIARES_H
#define FUNCIONESAUXILIARES_H

void cargarAlumnos(void *&alumnos,const char*nombArch);
void probarCarga(void *alumnos);
void *leerRegistro(ifstream &arch);
char *leeCadena(ifstream &arch, char='\n');
void incrementarEspacios(void **&alumnos,int &nd, int &cap);
void imprimirAlumno(void *al);
void cargarNotas(void *al,const char*nombArch);
int buscar(int cod,void **alumnos);
bool sonIguales(int cod,void*al);
void colocarCurso(ifstream &arch,void *al,int &nd,int &cap);
void incrementarEspacios(void *&al,int &nd, int &cap);
void *leeRCurso(ifstream &arch);
void agregarCurso(void*cur,void *alNota, int nd);
void imprimirCursos(void *cur);
void imprimirCurso(void *cur);
void calcularPromedios(void *alumnos);
void promedioDelAlumno(void *al);
void *promedio(void *al);
int obtenerNota(void *cur);
void imprimirPromedio(void *pro);
void ordenar(void *al);
void qsort(void **alumnos,int izq,int der );
void qsort2(void **notas,int izq,int der );
bool estanEnDesorden(void *aluI,void *alK);
```

```
67  bool estanEnDesorden2(void *notI,void *notK);
68  void cambiar(void*&al1,void*&al2);
69  void ordenarCursos(void *alu);
70  void eliminarDatos(void *alumnos);
71  void eliminarAlumno(void *al);
72  void eliminarCursos(void *no);
73  void eliminaNota(void *no);
74  #endif /* FUNCIONESAUXILIARES_H */
75
76  /*
77   * Proyecto: PunterosGenericos-Aplicacion
78   * Archivo:  FuncionesAuxiliares.cpp
79   * Autor:    J. Miguel Gunira E//miguel.guanira.
80   *
81   * Created on 9 de abril de 2024, 09:47 AM
82   */
83
84  #include <iostream>
85  #include <fstream>
86  #include <iomanip>
87  using namespace std;
88  #include <cstring>
89  #include <cmath>
90  #include "FuncionesAuxiliares.h"
91  #define INCREMENTO 5
92  enum Reg {COD,NOM,NOT,PRO};
93
94  void cargarAlumnos(void *&al,const char*nombArch){
95      ifstream arch(nombArch,ios::in);
96      if(not arch.is_open()){
97          cout<<"El archivo "<<nombArch<<"no se abrio"<<endl;
98          exit(1);
99      }
100     void **alumnos, *reg;
101     int nd=0, cap=0;
102     alumnos = nullptr;
103     while(true){
104         reg = leerRegistro(arch);
105         if(arch.eof())break;
106         if(nd==cap) incrementarEspacios(alumnos,nd,cap);
107         alumnos[nd-1] = reg;
108         nd++;
109     }
110     al = alumnos;
111 }
112
113 void *leerRegistro(ifstream &arch){
114     void**registro;
115     int *codigo, cod;
116     char*nombre;
117
118     arch>>cod;
119     if(arch.eof()) return nullptr;
120     codigo = new int;
121     *codigo = cod;
122     arch.get();
123     nombre = leeCadena(arch);
124     registro = new void*[4]{};
125     registro[COD]=codigo;
126     registro[NOM]=nombre;
127 //    registro[NOT]=nullptr;
128 //    registro[PRO]=nullptr;
129     return registro;
130 }
131
132 char *leeCadena(ifstream &arch, char car){
```

```
133         char cadena[60], *cad;
134         arch.getline(cadena,60,car);
135         cad = new char [strlen(cadena)+1];
136         strcpy(cad,cadena);
137         return cad;
138     }
139
140     void incrementarEspacios(void **&alumnos,int &nd, int &cap){
141         void **aux;
142
143         cap += INCREMENTO;
144         if (alumnos == nullptr){
145             alumnos = new void*[cap]{}; // No olvidar las {}
146             nd = 1;
147         }
148         else{
149             aux = new void* [cap]{};
150             for (int i=0; i<nd; i++)
151                 aux[i] = alumnos[i];
152             delete alumnos;
153             alumnos = aux;
154         }
155     }
156
157
158     void probarCarga(void *al){
159         void **alumnos = (void **)al;
160         for(int i=0; alumnos[i]; i++)
161             imprimirAlumno(alumnos[i]);
162     }
163
164     void imprimirAlumno(void *al){
165         void **alumno = (void**)al;
166         int *codigo;
167         char *nombre;
168
169         codigo = (int*)(alumno[COD]);
170         nombre = (char*)(alumno[NOM]);
171         cout<<left<<setw(10)<<*codigo<<setw(50)<<nombre<<endl;
172         if(alumno[NOT])
173             imprimirCursos(alumno[NOT]);
174         imprimirPromedio(alumno[PRO]);
175     }
176
177     void imprimirCursos(void *cur){
178         void **cursos = (void**)cur;
179         for(int i=0; cursos[i]; i++)
180             imprimirCurso(cursos[i]);
181     }
182
183     void imprimirCurso(void *cur){
184         void **curso= (void**)cur;
185         char*codigo= (char*)(curso[0]);
186         int*nota = (int*)(curso[1]);
187         cout<<right<<setw(10)<<codigo<<setw(5)<<*nota<<endl;
188     }
189
190     void cargarNotas(void *al,const char*nombArch){
191         ifstream arch(nombArch,ios::in);
192         if(not arch.is_open()){
193             cout<<"El archivo "<<nombArch<<"no se abrio"<<endl;
194             exit(1);
195         }
196         void **alumnos=(void**)al;
197         int nd[50]{}, cap[50]{}, cod, pos;
198
```

```
199         while(true){
200             arch>>cod;
201             if(arch.eof())break;
202             arch.get();
203             pos = buscar(cod,alumnos);
204             if(pos!= -1)
205                 colocarCurso(arch,alumnos[pos],nd[pos],cap[pos]);
206             else
207                 while(arch.get()!='\n');
208         }
209     }
210
211     void imprimirPromedio(void *pro){
212         double * promedio = (double*)pro;
213         cout.precision(2);
214         cout<<fixed;
215         cout<<"Promedio = "<<setw(10)<<*promedio<<endl;
216     }
217
218     int buscar(int cod,void **alumnos){
219         for(int i=0; alumnos[i]; i++)
220             if(sonIguales(cod,alumnos[i])) return i;
221         return -1;
222     }
223
224     bool sonIguales(int cod,void*al){
225         void **alumno = (void**)al;
226         int *codigo = (int*)alumno[COD];
227         return cod==*codigo;
228     }
229
230     void colocarCurso(ifstream &arch,void *al,int &nd,int &cap){
231         void **alumno=(void**)al;
232         void *cur;
233         cur = leeRCurso(arch);
234         if(nd == cap)
235             incrementarEspacios(alumno[NOT],nd,cap);
236         agregarCurso(cur,alumno[NOT],nd);
237         nd++;
238     }
239
240     void incrementarEspacios(void *&al,int &nd, int &cap){
241         void **aux, **alumnos = (void**)al;
242
243         cap += INCREMENTO;
244         if (alumnos == nullptr){
245             alumnos = new void*[cap]{}; // No olvidar las {}
246             nd = 1;
247         }
248         else{
249             aux = new void* [cap]{};
250             for (int i=0; i<nd; i++)
251                 aux[i] = alumnos[i];
252             delete alumnos;
253             alumnos = aux;
254         }
255         al = alumnos;
256     }
257
258     void *leeRCurso(ifstream &arch){
259         char*codigo;
260         int*nota = new int;
261         void**registro = new void*[2];
262         codigo = leeCadena(arch,',');
263         arch>>*nota;
264         registro[0]=codigo;
```

```
265         registro[1]=nota;
266         return registro;
267     }
268
269     void agregarCurso(void*cur,void *alNota, int nd){
270         void **alumnoNota = (void**)alNota;
271         alumnoNota[nd-1] = cur;
272     }
273
274     void calcularPromedios(void *al){
275         void **alumnos = (void**)al;
276
277         for(int i=0; alumnos[i]; i++)
278             promedioDelAlumno(alumnos[i]);
279     }
280
281     void promedioDelAlumno(void *al){
282         void **alumno = (void**)al;
283         if(alumno[NOT])
284             alumno[PRO] = promedio(alumno[NOT]);
285     }
286
287     void *promedio(void *al){ //Promedio de un alumno
288         void **cursos = (void**)al;
289         int suma=0, numDat=0;
290         double *prom = new double;
291         for(int i=0; cursos[i]; i++){
292             suma += obtenerNota(cursos[i]);
293             numDat++;
294         }
295         *prom = (double)suma/numDat;
296         return prom;
297     }
298
299     int obtenerNota(void *cur){
300         void **curso= (void**)cur;
301         int *nota = (int*)curso[1]; // return *(int*)curso[1];
302         return *nota;               //
303     }
304
305     void ordenar(void *al){
306         void **alumnos = (void**)al;
307         int numDat=0;
308         for(; alumnos[numDat];numDat++)
309             ordenarCursos(alumnos[numDat]);
310         qsort(alumnos,0,numDat-1);
311
312     }
313
314     void qsort(void **alumnos,int izq,int der ){
315         int limite;
316         if(izq>=der)return;
317         cambiar(alumnos[izq], alumnos[(izq+der)/2]);
318         limite = izq;
319         for(int i=izq+1; i<=der; i++)
320             if(estanEnDesorden(alumnos[i],alumnos[izq]))
321                 cambiar(alumnos[++limite],alumnos[i]);
322         cambiar(alumnos[izq],alumnos[limite]);
323         qsort(alumnos,izq,limite-1);
324         qsort(alumnos,limite+1,der);
325     }
326
327     void cambiar(void*&al1,void*&al2){
328         void *aux;
329         aux=al1;
330         al1=al2;
```

```
331         al2=aux;
332     }
333
334     bool estanEnDesorden(void *alI,void *alK){
335         void **alumnoI = (void**)alI, **alumnoK=(void**)alK;
336         char*nombreI = (char*)alumnoI[NOM], *nombreK=(char*)alumnoK[NOM];
337         return strcmp(nombreI,nombreK)<0;
338     }
339
340     void ordenarCursos(void *alu){
341         void**alumno = (void**)alu;
342         void **notas = (void**)alumno[NOT];
343         int nd=0;
344         for(; notas[nd];nd++);
345         qsort2(notas,0,nd-1);
346     }
347
348     void qsort2(void **notas,int izq,int der ){
349         int limite;
350         if(izq>=der)return;
351         cambiar(notas[izq], notas[(izq+der)/2]);
352         limite = izq;
353         for(int i=izq+1; i<=der; i++)
354             if(estanEnDesorden2(notas[i],notas[izq]))
355                 cambiar(notas[++limite],notas[i]);
356         cambiar(notas[izq],notas[limite]);
357         qsort2(notas,izq,limite-1);
358         qsort2(notas,limite+1,der);
359     }
360
361     bool estanEnDesorden2(void *notI,void *notK){
362         void **notaI = (void**)notI, **notaK=(void**)notK;
363         char*nombreI = (char*)notaI[0], *nombreK=(char*)notaK[0];
364         return strcmp(nombreI,nombreK)<0;
365     }
366
367     void eliminarDatos(void *al){
368         void **alumnos = (void**)al;
369         for(int i=0; alumnos[i];i++)
370             eliminarAlumno(alumnos[i]);
371         delete alumnos;
372     }
373
374     void eliminarAlumno(void *al){
375         void **alumno = (void**)al;
376         int *codigo = (int*)alumno[COD];
377         char *nombre = (char*)alumno[NOM];
378         double *promedio = (double*)alumno[PRO];
379         delete codigo;
380         delete nombre;
381         if(promedio!=nullptr) {
382             delete promedio;
383             eliminarCursos(alumno[NOT]);
384         }
385         delete alumno;
386     }
387
388     void eliminarCursos(void *no){
389         void **notas = (void**)no;
390         for(int i=0; notas[i];i++)
391             eliminaNota(notas[i]);
392     }
393
394     void eliminaNota(void *cur){
395         void **curso = (void**)cur;
396         char* codigo=(char*)curso[0];
```

```
397         int* nota = (int*)curso[1];
398         delete curso;
399         delete codigo;
400         delete nota;
401 
402     }
```