```cpp
/*
 * Proyecto: PunterosGenericos_Aplicacion
 * Archivo:  main.cpp
 * Autor:    J. Miguel Guanira E.//miguel.guanira.
 *
 * Created on 17 de septiembre de 2024, 07:42 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);
    ordenarAlumnos(alumnos);
    probarLaLectura(alumnos,"ReporteDePrueba.txt");
    liberarEspacios(alumnos);

    return 0;
}

/*
 * Proyecto: PunterosGenericos_Aplicacion
 * Archivo:  FuncionesAuxiliares.h
 * Autor:    J. Miguel Guanira E. //miguel.guanira.
 *
 * Created on 17 de septiembre de 2024, 08:31 AM
 */

#ifndef FUNCIONESAUXILIARES_H
#define FUNCIONESAUXILIARES_H

void cargarAlumnos(void*&alumnos, const char *nombArch);
void probarLaLectura(void*alumnos, const char *nombArch);
void *leerRegistro(ifstream &arch);
char*leeCadena(ifstream &arch, char delimitador='\n');
void aumentamosEspacios(void **&alumnos,int &numDat,int &cap);
void imprimeAlumno(ofstream &arch,void *al);
int buscarAlumno(int cod, void **alumnos);
bool sonIguales(int cod, void *al);
void colocarCurso(ifstream& arch,void *al,int &nd,int &cap);
void *leerCurso(ifstream &arch);
void incrementarEspacios(void *&al,int &nd,int &cap);
void agregarCurso(void *cur,void *notas,int &nd);
void imprimirCursos(ofstream &arch,void *alNot);
void imprimeCurso(ofstream &arch,void *alNot);
void cargarNotas(void*al, const char *nombArch);
void calcularPromedios(void *alumnos);
void promedioDelAlumno(void *al);
void * promedio(void *al);
int obtenerNota(void *cur);
void ordenarAlumnos(void *alumnos);
void qSort(void **alumnos,int izq,int der);
void cambiar(void *&alumI, void *&alumK);
bool estanEnDesorden(void *alumI,void *alumK);
void liberarEspacios(void *alumnos);
void liberarAlumno(void *al);
void liberarNotas( void *alNotas);
void liberarCurso(void *alNot);

#endif /* FUNCIONESAUXILIARES_H */

```

```cpp
/*
 * Proyecto: PunterosGenericos_Aplicacion
 * Archivo:  FuncionesAuxiliares.cpp
 * Autor:    J. Miguel Gunira E//miguel.guanira.
 *
 * Created on 17 de septiembre de 2024, 08:31 AM
 */

#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
#include <cstring>
#include "FuncionesAuxiliares.h"
enum REG {COD,NOM,NOTA,PROM};
#define INCREMENTO 5

void cargarAlumnos(void*&al, const char *nombArch){
    ifstream arch(nombArch,ios::in);
    if(not arch.is_open()){
        cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
        exit(1);
    }
    void **alumnos, *reg;
    int numDat=0, cap=0;
    alumnos = nullptr;
    while(true){
        reg = leerRegistro(arch);
        if(arch.eof())break;
        if(numDat == cap) aumentamosEspacios(alumnos,numDat,cap);
        alumnos[numDat-1] = reg;
        numDat++;
    }
    al = alumnos;
}

void *leerRegistro(ifstream &arch){
    void **registro;
    int *codigo, cod;
    char *nombre;

    arch >> cod;
    if(arch.eof()) return nullptr;
    codigo = new int;
    *codigo = cod;
    arch.get();
    nombre = leeCadena(arch);
    registro = new void*[4]{};
    registro[COD] = codigo;
    registro[NOM] = nombre;
    return registro;
}

char*leeCadena(ifstream &arch, char delimitador){
    char buffer[60], *cad;
    arch.getline(buffer,60,delimitador);
    cad = new char[strlen(buffer)+1];
    strcpy(cad, buffer);
    return cad;
}

void aumentamosEspacios(void **&alumnos,int &numDat,int &cap){
    void **aux;
    cap += INCREMENTO;
    if(alumnos == nullptr){
        alumnos = new void*[cap]{};
```

```
133             numDat = 1;
134         }
135         else{
136             aux =new void*[cap]{};
137             for (int i = 0; i < numDat; i++) {
138                 aux[i] = alumnos[i];
139             }
140             delete alumnos;
141             alumnos = aux;
142         }
143     }
144
145     void cargarNotas(void*al, const char *nombArch){
146         ifstream arch(nombArch,ios::in);
147         if(not arch.is_open()){
148             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
149             exit(1);
150         }
151         void **alumnos = (void**)al;
152         int nd[50]{}, cap[50]{}, cod, pos;
153         while(true){
154             arch>>cod;
155             if(arch.eof())break;
156             arch.get();
157             pos = buscarAlumno(cod, alumnos);
158             if(pos!=-1)
159                 colocarCurso(arch,alumnos[pos],nd[pos],cap[pos]);
160             else while(arch.get()!='\n');
161         }
162     }
163
164     int buscarAlumno(int cod, void **alumnos){
165         for (int i = 0; alumnos[i]; i++)
166             if(sonIguales(cod,alumnos[i])) return i;
167         return -1;
168
169     }
170
171     bool sonIguales(int cod, void *al){
172         void **alumno = (void**)al;
173         int *codigo = (int*)alumno[COD];
174         return cod == *codigo;
175     }
176
177     void colocarCurso(ifstream& arch,void *al,int &nd,int &cap){
178         void ** alumno = (void**)al;
179         void *cur;
180         cur = leerCurso(arch);
181         if(nd == cap) incrementarEspacios(alumno[NOTA],nd,cap);
182         agregarCurso(cur,alumno[NOTA],nd);
183     }
184
185     void *leerCurso(ifstream &arch){
186         char*codigo;
187         int *nota = new int;
188         void **registro = new void *[2];
189         codigo = leeCadena(arch,',');
190         arch>>*nota;
191         registro[0] = codigo;
192         registro[1] = nota;
193         return registro;
194     }
195
196     void incrementarEspacios(void *&al,int &nd,int &cap){
197         void **aux, **alumno = (void**)al;
198         cap += INCREMENTO;
```

```
199         if(alumno == nullptr){
200             alumno = new void*[cap]{};
201             nd = 1;
202         }
203         else{
204             aux = new void*[cap]{};
205             for (int i = 0; i < nd; i++)
206                 aux[i] = alumno[i];
207             delete alumno;
208             alumno = aux;
209         }
210         al = alumno;
211     }
212
213     void agregarCurso(void *cur,void *alNot,int &nd){
214         void **notas = (void**)alNot;
215         notas[nd-1] = cur;
216         nd++;
217     }
218
219     void calcularPromedios(void *al){
220         void **alumnos = (void **)al;
221         for (int i = 0; alumnos[i]; i++)
222             promedioDelAlumno(alumnos[i]);
223     }
224
225     void promedioDelAlumno(void *al){
226         void **alumno = (void **)al;
227         if(alumno[NOTA])
228             alumno[PROM] = promedio(alumno[NOTA]);
229     }
230
231     void * promedio(void *alNot){
232         void **notas = (void **)alNot;
233         int suma=0, numDat=0;
234         double *prom;
235         for (int i = 0; notas[i]; i++) {
236             suma += obtenerNota(notas[i]);
237             numDat++;
238         }
239         prom = new double;
240         *prom = (double)suma/numDat;
241         return prom;
242     }
243
244     int obtenerNota(void *cur){
245         void **curso = (void**)cur;
246         int *nota = (int*)curso[1];
247         return *nota;
248     }
249
250     void probarLaLectura(void*al, const char *nombArch){
251         ofstream arch(nombArch,ios::out);
252         if(not arch.is_open()){
253             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
254             exit(1);
255         }
256         void **alumnos = (void **)al;
257         for (int i = 0; alumnos[i] ; i++)
258             imprimeAlumno(arch,alumnos[i]);
259     }
260
261     void imprimeAlumno(ofstream &arch,void *al){
262         void **alumno = (void **)al;
263         int *codigo = (int *)alumno[COD];
264         char*nombre = (char*)alumno[NOM];
```

```
265         double *promedio;
266         arch.precision(2);
267         arch<<fixed;
268         arch<<right<<setw(10)<<*codigo<<"  "<<left<<nombre<<endl;
269         if(alumno[NOTA]){
270             imprimirCursos(arch,alumno[NOTA]);
271         }
272         if(alumno[PROM]){
273             promedio = (double*)alumno[PROM];
274             arch<<"  PROMEDIO = "<<setw(10)<<*promedio<<endl<<endl;
275         }
276     }
277
278     void imprimirCursos(ofstream &arch,void *alNot){
279         void **notas = (void**)alNot;
280         for (int i = 0; notas[i]; i++)
281             imprimeCurso(arch,notas[i]);
282     }
283
284     void imprimeCurso(ofstream &arch,void *alNot){
285         void **nota = (void **)alNot;
286         char*codCur = (char*)nota[0];
287         int*notaCur = (int*)nota[1];
288         arch<<right<<setw(15)<<codCur<<setw(5)<<*notaCur<<endl;
289     }
290
291     void ordenarAlumnos(void *al){
292         void **alumnos = (void**)al;
293         int numDat = 0;
294         for (numDat = 0; alumnos[numDat]; numDat++);
295         qSort(alumnos,0,numDat-1);
296     }
297
298     void qSort(void **alumnos,int izq,int der){
299         int limite;
300         if(izq>=der)return;
301         cambiar(alumnos[izq], alumnos[(izq+der)/2]);
302         limite = izq;
303         for (int i = izq+1; i <=der; i++)
304             if(estanEnDesorden(alumnos[i],alumnos[izq]))
305                 cambiar(alumnos[++limite], alumnos[i]);
306          cambiar(alumnos[izq], alumnos[limite]);
307          qSort(alumnos,izq,limite-1);
308          qSort(alumnos,limite+1,der);
309     }
310
311     void cambiar(void *&alumI, void *&alumK){
312         void *aux;
313         aux = alumI;
314         alumI = alumK;
315         alumK = aux;
316     }
317
318     bool estanEnDesorden(void *alI,void *alK){
319         void **alumI= (void**)alI, **alumK = (void**)alK;
320         char*nombreI = (char*)alumI[NOM], *nombreK = (char*)alumK[NOM];
321         return strcmp(nombreI,nombreK)<0;
322     }
323
324     void liberarEspacios(void *al){
325         void **alumnos = (void **)al;
326         for (int i = 0; alumnos[i]; i++)
327             liberarAlumno(alumnos[i]);
328         delete alumnos;
329     }
330
```

```
331  void liberarAlumno(void *al){
332      void **alumno = (void **)al;
333      int* codigo = (int *)alumno[COD];
334      char* nombre = (char*)alumno[NOM];
335      double *promedio = (double*)alumno[PROM];
336
337      delete codigo;
338      delete nombre;
339      if(promedio) delete promedio;
340      if(alumno[NOTA])liberarNotas(alumno[NOTA]);
341      delete alumno;
342  }
343
344  void liberarNotas( void *alNotas){
345      void **notas = (void**)alNotas;
346      for (int i = 0; notas[i]; i++)
347          liberarCurso(notas[i]);
348      delete notas;
349  }
350
351  void liberarCurso(void *alNot){
352      void **curso = (void **)alNot;
353      char* codigo = (char*)curso[0];
354      int* nota = (int*)curso[1];
355      delete codigo;
356      delete nota;
357      delete curso;
358  }
```