```cpp
/*
 * Proyecto: ImplementacionDeUnQsortGenerico
 * Archivo:  main.cpp
 * Autor:    J. Miguel Guanira E.//miguel.guanira.
 *
 * Created on 23 de abril de 2024, 08:29 AM
 */

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

int main(int argc, char** argv) {
//    int a[50]={10, 15, 2, 82, 35, 44, 17, 91,33, 77,12, 21, 56,17, 4, 25}, nd=16;
//    ordenaGen(a,0,nd-1,intcmp);
//    for(int i=0; i<nd; i++)
//        cout<<setw(4)<<a[i];
//    cout<<endl

//        // Con cadenas
//    int nd;
//    char *nombres[100];
//    leerDatos(nombres,nd, "Personas.txt");
//    ordenaGen(nombres,0,nd-1,cadcmp);
//    for(int i=0; i<nd; i++)
//        cout<<nombres[i]<<endl;

    void *personal;
    int nd;
    leerDatos(personal,nd,"personal.csv");
//    ordenaGen(personal,0,nd-1,voidcmpSueldo);
//    ordenaGen(personal,0,nd-1,voidcmpCodigos);
    ordenaGen(personal,0,nd-1,voidcmpNombres);
    imprimirDatos(personal,nd);

    return 0;
}

/*
 * Proyecto: ImplementacionDeUnQsortGenerico
 * Archivo:  FuncionesDeOrdenacionGenerica.h
 * Autor:    J. Miguel Guanira E. //miguel.guanira.
 *
 * Created on 23 de abril de 2024, 08:30 AM
 */

#ifndef FUNCIONESDEORDENACIONGENERICA_H
#define FUNCIONESDEORDENACIONGENERICA_H

void ordenaGen(void *arr, int izq, int der, int (*cmp)(const void *, const void *));
void cambiarGen(void *&arrI, void*&arrK);

#endif /* FUNCIONESDEORDENACIONGENERICA_H */

/*
 * Proyecto: ImplementacionDeUnQsortGenerico
 * Archivo:  FuncionesDeOrdenacionGenerica.cpp
 * Autor:    J. Miguel Gunira E//miguel.guanira.
 *
 * Created on 23 de abril de 2024, 08:30 AM
 */
```

```
65
66  #include <iostream>
67  #include <iomanip>
68  using namespace std;
69  #include "FuncionesDeOrdenacionGenerica.h"
70
71  void ordenaGen(void *arr, int izq, int der,
72                          int (*cmp)(const void *, const void*)){
73      void **arreglo = (void**)arr;
74      int limite;
75
76      if(izq>=der) return;
77      cambiarGen(arreglo[izq], arreglo[(izq+der)/2]);
78      limite=izq;
79      for(int i=izq+1; i<=der; i++)
80          if(cmp(arreglo[izq],arreglo[i])>0)
81             cambiarGen(arreglo[++limite],arreglo[i]);
82      cambiarGen(arreglo[izq],arreglo[limite]);
83      ordenaGen(arreglo,izq, limite-1,cmp);
84      ordenaGen(arreglo,limite+1,der,cmp);
85  }
86
87  void cambiarGen(void *&arrI, void*&arrK){
88      void*aux;
89      aux = arrI;
90      arrI = arrK;
91      arrK = aux;
92  }
93
94  /*
95   * Proyecto: ImplementacionDeUnQsortGenerico
96   * Archivo:  FuncionesDeComparacion.h
97   * Autor:    J. Miguel Guanira E. //miguel.guanira.
98   *
99   * Created on 23 de abril de 2024, 08:51 AM
100  */
101
102 #ifndef FUNCIONESDECOMPARACION_H
103 #define FUNCIONESDECOMPARACION_H
104
105 int intcmp(const void*a, const void*b);
106 int cadcmp(const void*a, const void*b);
107 int voidcmpSueldo(const void*a, const void*b);
108 int voidcmpCodigos(const void*a, const void*b);
109 int voidcmpNombres(const void*a, const void*b);
110
111 #endif /* FUNCIONESDECOMPARACION_H */
112
113 /*
114  * Proyecto: ImplementacionDeUnQsortGenerico
115  * Archivo:  FuncionesDeComparacion.cpp
116  * Autor:    J. Miguel Gunira E//miguel.guanira.
117  *
118  * Created on 23 de abril de 2024, 08:51 AM
119  */
120
121 #include <iostream>
122 #include <iomanip>
123 using namespace std;
124 #include <cstring>
125 int intcmp(const void*a, const void*b){
126     int ai = (int)(a), bi=(int)b;
127     return bi-ai;
128 }
129
130 int cadcmp(const void*a, const void*b){
```

```
131         char *ai=(char*)a, *bi=(char*)b;
132         return strcmp(ai,bi);
133     }
134 
135     int voidcmpSueldo(const void*a, const void*b){
136         void**regA=(void**)a, **regB=(void**)b;
137         double *sueldoA=(double*)regA[2], *sueldoB=(double*)regB[2];
138         return *sueldoA-*sueldoB;
139     }
140 
141     int voidcmpCodigos(const void*a, const void*b){
142         void**regA=(void**)a, **regB=(void**)b;
143         int *codigoA=(int *)regA[0],*codigoB=(int *)regB[0];
144         return *codigoA-*codigoB;
145     }
146 
147 
148      int voidcmpNombres(const void*a, const void*b){
149          void**regA=(void**)a, **regB=(void**)b;
150          char *nombreA = (char*)regA[1],*nombreB = (char*)regB[1];
151          return strcmp(nombreA,nombreB);
152      }
153 
154     /*
155      * Proyecto: ImplementacionDeUnQsortGenerico
156      * Archivo:  FuncionesAuxiliares.h
157      * Autor:    J. Miguel Guanira E. //miguel.guanira.
158      *
159      * Created on 23 de abril de 2024, 09:14 AM
160      */
161 
162     #ifndef FUNCIONESAUXILIARES_H
163     #define FUNCIONESAUXILIARES_H
164 
165     void leerDatos(char**nombres,int &nd, const char*nombArch);
166     void leerDatos(void *&persona,int &numDat,const char*nombArch);
167     void *leerReg(ifstream&arch);
168     void imprimirDatos(void *persona ,int numDat);
169     void imprimeRegistro(void*reg);
170 
171     #endif /* FUNCIONESAUXILIARES_H */
172 
173     /*
174      * Proyecto: ImplementacionDeUnQsortGenerico
175      * Archivo:  FuncionesAuxiliares.cpp
176      * Autor:    J. Miguel Gunira E//miguel.guanira.
177      *
178      * Created on 23 de abril de 2024, 09:14 AM
179      */
180 
181     #include <iostream>
182     #include <fstream>
183     #include <iomanip>
184     using namespace std;
185     #include "FuncionesAuxiliares.h"
186     #include <cstring>
187 
188     void leerDatos(char**nombres,int &nd, const char*nombArch){
189         ifstream arch(nombArch,ios::in);
190         if(not arch.is_open()){
191             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
192             exit(1);
193         }
194         char nomb[60];
195         nd=0;
196 
```

```
197         while(true){
198             arch>>nomb;
199             if(arch.eof())break;
200             nombres[nd] = new char[strlen(nomb)+1];
201             strcpy(nombres[nd],nomb);
202             nd++;
203         }
204     }
205
206     enum Registro {CODIGO, NOMBRE, SUELDO};
207
208     void leerDatos(void *&persona,int &numDat,const char*nombArch){
209         ifstream arch(nombArch,ios::in);
210         if(not arch.is_open()){
211             cout<<"ERROR: No se pudo abrir el archivo "<<nombArch<<endl;
212             exit(1);
213         }
214         void *buff[500], **per, *p;
215         numDat = 0;
216         while(1){
217             p = leerReg(arch);
218             if(p == nullptr) break;
219             buff[numDat] = p;
220             numDat++;
221         }
222         per = new void*[numDat];
223         for(int i=0; i<numDat; i++)
224             per[i] = buff[i];
225         persona = per;
226     }
227
228     void *leerReg(ifstream&arch){
229         void **r;
230         int *codigo, cod;
231         char*nombre, buff[100];
232         double*sueldo;
233
234         arch>>cod;
235         if(arch.eof()) return nullptr;
236         codigo = new int;
237         *codigo = cod;
238         arch.get(); // Sacamos la coma
239         arch.getline(buff,100,',');
240         nombre = new char[strlen(buff)+1];
241         strcpy(nombre,buff);
242         sueldo = new double;
243         arch>>*sueldo;
244
245         r = new void*[3];
246         r[CODIGO] = codigo;
247         r[NOMBRE] = nombre;
248         r[SUELDO] = sueldo;
249
250         return r;
251     }
252
253     void imprimirDatos(void *persona ,int numDat){
254         void **per = (void **)persona;
255
256         for(int i=0; i<numDat; i++)
257             imprimeRegistro(per[i]);
258     }
259
260     void imprimeRegistro(void*reg){
261         void **r = (void**)reg;
262         int *codigo = (int *)r[CODIGO];
```

```
263         char *nombre = (char*)r[NOMBRE];
264         double *sueldo = (double*)r[SUELDO];
265 
266         cout.precision(2);
267         cout<<fixed;
268         cout<<left<<setw(10)<<*codigo<<setw(45)<<nombre
269             <<right<<setw(10)<<*sueldo<<endl;
270     }
271 
```