1 #include <iostream>

2 #include <stdio.h>

3 #include <stdlib.h>

4 #include <string.h>

5 #include "cliente.h"

6

7

8 **using namespace std**;

9

10 Cliente crearCiente(){

11

12 Cliente e = **new** ClienteEstructura;

13

14 **return** e;

15

16 }

17

18 **char** \* getsNombre(Cliente a)

19 {

20 **return** a->nombre;

21 }

22

23 **int** getsIdUsuario(Cliente a)

24 {

25 **return** a->idUsuario;

26 }

27 **int** getsEdad(Cliente a)

28 {

29 **return** a->edad;

30 }

31

32 **char** \* getsDireccion(Cliente a)

33 {

34 **return** a->direccion;

35 }

36

37 **void** setsNombre(Cliente a, **char** nuevoNombre[30]){

38 strcpy(a->nombre,nuevoNombre);

39 }

40

41 **void** setsDireccion(Cliente a, **char** nuevoDireccion[50]){

42

43 strcpy(a->direccion,nuevoDireccion);

44 }

45

46 **void** setsIdusuario(Cliente a, **int** nuevoId){

47 a->idUsuario=nuevoId;

48 }

49

50 **void** setsEdad(Cliente a, **int** nuevoEdad){

51 a->edad=nuevoEdad;

52 }

1 #include <iostream>

2 #include<conio.h>

3 #include<stdio.h>

4 #include<stdlib.h>

5 #include <string.h>

6 #include<fstream>

7 #include "funcionesCargarArchivos.h"

8 #include "infoMensual.h"

9 #include "vino.h"

10 #include "cliente.h"

11 #include "lista.h"

12

13

14 **using namespace std**;

15

16 **void** leerYpasarAusuarios(Lista clien){

17 **ifstream** archivo;

18 **string** texto;

19 archivo.open("usuarios\_test.txt",ios::in);//abrimos el archivo en modo lectura

20

21

22 **if**(archivo.fail()){

23 **cout**<<"No se pudo abrir el archivo";

24 exit(1);

25 }

26

27 **while**(!archivo.eof()){//mientras no sea el final del archivo

28

29 getline(archivo,texto);

30 Cliente c;

31 c= pasarCharAStructConStringUsuario(texto);

32

33 agregarNodo(clien,c);

34 }

35

36 archivo.close();

37 }

38

39 **void** leerYpasarAseleccion(Lista selec){

40 **fstream** archivo;

41 **string** texto;

42 archivo.open("elecion.txt",ios::in);//abrimos el archivo en modo lectura

43

44

45 **if**(archivo.fail()){

46 **cout**<<"No se pudo abrir el archivo";

47 exit(1);

48 }

49

50 **while**(!archivo.eof()){//mientras no sea el final del archivo

51 getline(archivo,texto);

52 infoMesual c;

53 c= pasarCharAstructSeleccion(texto);

54 agregarNodo(selec,c);

55 }

56

57 archivo.close();

58

59 }

60

61 **void** leerYpasaraVinos(Lista vin){

62 **fstream** archivo;

63 **string** texto;

64 archivo.open("catalogos.txt",ios::in);//abrimos el archivo en modo lectura

65

66

67 **if**(archivo.fail()){

68 **cout**<<"No se pudo abrir el archivo";

69 exit(1);

70 }

71

72 **while**(!archivo.eof()){//mientras no sea el final del archivo

73 getline(archivo,texto);

74 Vino v;

75 v= pasarCharAStructConStringVino(texto);

76 agregarNodo(vin,v);

77

78 }

79

80 archivo.close();

81

82 }

83

84

85

86 Cliente pasarCharAStructConStringUsuario(**string** texto){

87 **int** pos=-1;

88 **int** pos2=-1;

89 **int** pos3=-1;

90

91

92 **char** auxNombre[30]= " ";

93 **char** auxEdad [50]=" ";

94 **char** idUsuario[30];

95 **char** auxDireccion[90]=" ";

96

97 **int** longitud= texto.**size**();

98

99

100 fflush (stdin);

101 **for** (**int** i=0;i<longitud;i++)

102 {

103 **if**(texto[i]=='-')

104 {

105 pos=i;

106

107 i=longitud;//termina el for

108 }

109 }

110 **for** (**int** i=pos+1;i<longitud;i++)

111 {

112 **if**(texto[i]=='-')

113 {

114 pos2=i;

115

116 i=longitud;//termina el for

117 }

118 }

119 **for** (**int** i=pos2+1;i<longitud;i++)

120 {

121 **if**(texto[i]=='-')

122 {

123 pos3=i;

124

125 i=longitud;//termina el for

126 }

127 }

128

129

130

131 **for** (**int** i=0;i<pos;i++)

132 {

133 idUsuario[i]=texto[i];

134

135 }

136

137

138 **for** (**int** i=pos+1;i<pos2;i++)

139 {

140

141 auxNombre[i-pos-1]=texto[i];

142 }

143

144 **for** (**int** i=pos2+1;i<pos3;i++)

145 {

146

147 auxDireccion[i-pos2-1]=texto[i];

148

149 }

150 **for** (**int** i=pos3+1;i<longitud;i++)

151 {

152

153 auxEdad[i-pos3-1]=texto[i];

154 }

155 Cliente cli=crearCiente();

156

157 setsNombre(cli,auxNombre);

158 setsDireccion(cli,auxDireccion);

159 setsIdusuario(cli,atoi(idUsuario));

160 setsEdad(cli,atoi(auxEdad));

161

162 **return** cli;

163 }

164

165 Vino pasarCharAStructConStringVino(**string** texto){

166

167 **int** pos=-1;

168 **int** pos2=-1;

169 **int** pos3=-1;

170 **int** pos4=-1;

171 **int** pos5=-1;

172 **int** pos6=-1;

173

174 **char** idVino[50]=" ";

175 **char** etiqueta[50]=" ";

176 **char** bodega [50]=" ";

177 **char** segmento\_del\_vino[50]=" ";

178 **char** varietal[50]=" ";

179 **char** anioCosecha[50]=" ";

180 **char** terroir[50]=" ";

181

182

183 **int** longitud= texto.**size**();

184

185 fflush (stdin);

186 **for** (**int** i=0;i<longitud;i++)

187 {

188 **if**(texto[i]=='-')

189 {

190 pos=i;

191

192 i=longitud;//termina el for

193 }

194 }

195 **for** (**int** i=pos+1;i<longitud;i++)

196 {

197 **if**(texto[i]=='-')

198 {

199 pos2=i;

200

201 i=longitud;//termina el for

202 }

203 }

204 **for** (**int** i=pos2+1;i<longitud;i++)

205 {

206 **if**(texto[i]=='-')

207 {

208 pos3=i;

209

210 i=longitud;//termina el for

211 }

212 }

213 **for** (**int** i=pos3+1;i<longitud;i++)

214 {

215 **if**(texto[i]=='-')

216 {

217 pos4=i;

218

219 i=longitud;//termina el for

220 }

221 }

222 **for** (**int** i=pos4+1;i<longitud;i++)

223 {

224 **if**(texto[i]=='-')

225 {

226 pos5=i;

227

228 i=longitud;//termina el for

229 }

230 }

231 **for** (**int** i=pos5+1;i<longitud;i++)

232 {

233 **if**(texto[i]=='-')

234 {

235 pos6=i;

236

237 i=longitud;//termina el for

238 }

239 }

240

241

242

243 **for** (**int** i=0;i<pos;i++)

244 {

245 idVino[i]=texto[i];

246

247 }

248

249

250 **for** (**int** i=pos+1;i<pos2;i++)

251 {

252

253 etiqueta[i-pos-1]=texto[i];

254 }

255

256 **for** (**int** i=pos2+1;i<pos3;i++)

257 {

258

259 bodega[i-pos2-1]=texto[i];

260

261 }

262 **for** (**int** i=pos3+1;i<pos4;i++)

263 {

264

265 segmento\_del\_vino[i-pos3-1]=texto[i];

266 }

267 **for** (**int** i=pos4+1;i<pos5;i++)

268 {

269

270 varietal[i-pos4-1]=texto[i];

271 }

272 **for** (**int** i=pos5+1;i<pos6;i++)

273 {

274

275 anioCosecha[i-pos5-1]=texto[i];

276 }

277 **for** (**int** i=pos6+1;i<longitud;i++)

278 {

279

280 terroir[i-pos6-1]=texto[i];

281 }

282 Vino vi=crearVino();

283

284 setsAnioCosecha(vi,atoi(anioCosecha));

285 setsIdVino(vi,atoi(idVino));

286 setsBodega(vi,bodega);

287 setsEtiqueta(vi,etiqueta);

288 setsSegmentoVino(vi,segmento\_del\_vino);

289 setsTerroir(vi,terroir);

290 setsVarietal(vi,varietal);

291

292

293 **return** vi;

294

295 };

296

297

298 infoMesual pasarCharAstructSeleccion(**string** texto){

299

300 **int** pos=-1;

301 **int** pos2=-1;

302 **int** pos3=-1;

303 **int** pos4=-1;

304 **int** pos5=-1;

305 **int** pos6=-1;

306 **int** pos7=-1;

307 **int** pos8=-1;

308

309 **char** idUsuario[50]=" ";

310 **char** mesSeleccion[50]=" ";

311 **char** anioSeleccion [50]=" ";

312 **char** idVino1[50]=" ";

313 **char** idVino2[50]=" ";

314 **char** idVino3[50]=" ";

315 **char** idVino4[50]=" ";

316 **char** idVino5[50]=" ";

317 **char** idVino6[50]=" ";

318

319 **int** longitud= texto.**size**();

320

321 fflush (stdin);

322 **for** (**int** i=0;i<longitud;i++)

323 {

324 **if**(texto[i]=='-')

325 {

326 pos=i;

327

328 i=longitud;//termina el for

329 }

330 }

331 **for** (**int** i=pos+1;i<longitud;i++)

332 {

333 **if**(texto[i]=='-')

334 {

335 pos2=i;

336

337 i=longitud;//termina el for

338 }

339 }

340 **for** (**int** i=pos2+1;i<longitud;i++)

341 {

342 **if**(texto[i]=='-')

343 {

344 pos3=i;

345

346 i=longitud;//termina el for

347 }

348 }

349 **for** (**int** i=pos3+1;i<longitud;i++)

350 {

351 **if**(texto[i]=='-')

352 {

353 pos4=i;

354

355 i=longitud;//termina el for

356 }

357 }

358 **for** (**int** i=pos4+1;i<longitud;i++)

359 {

360 **if**(texto[i]=='-')

361 {

362 pos5=i;

363

364 i=longitud;//termina el for

365 }

366 }

367 **for** (**int** i=pos5+1;i<longitud;i++)

368 {

369 **if**(texto[i]=='-')

370 {

371 pos6=i;

372

373 i=longitud;//termina el for

374 }

375 }

376 **for** (**int** i=pos6+1;i<longitud;i++)

377 {

378 **if**(texto[i]=='-')

379 {

380 pos7=i;

381

382 i=longitud;//termina el for

383 }

384 }

385 **for** (**int** i=pos7+1;i<longitud;i++)

386 {

387 **if**(texto[i]=='-')

388 {

389 pos8=i;

390

391 i=longitud;//termina el for

392 }

393 }

394

395

396

397 **for** (**int** i=0;i<pos;i++)

398 {

399 idUsuario[i]=texto[i];

400

401 }

402

403

404 **for** (**int** i=pos+1;i<pos2;i++)

405 {

406

407 mesSeleccion[i-pos-1]=texto[i];

408 }

409

410 **for** (**int** i=pos2+1;i<pos3;i++)

411 {

412

413 anioSeleccion[i-pos2-1]=texto[i];

414

415 }

416 **for** (**int** i=pos3+1;i<pos4;i++)

417 {

418

419 idVino1[i-pos3-1]=texto[i];

420 }

421 **for** (**int** i=pos4+1;i<pos5;i++)

422 {

423

424 idVino2[i-pos4-1]=texto[i];

425 }

426

427 **for** (**int** i=pos5+1;i<pos6;i++)

428 {

429

430 idVino3[i-pos5-1]=texto[i];

431 }

432 **for** (**int** i=pos6+1;i<pos7;i++)

433 {

434

435 idVino4[i-pos6-1]=texto[i];

436 }

437 **for** (**int** i=pos7+1;i<pos8;i++)

438 {

439

440 idVino5[i-pos7-1]=texto[i];

441 }

442 **for** (**int** i=pos8+1;i<longitud;i++)

443 {

444

445 idVino6[i-pos8-1]=texto[i];

446 }

447 infoMesual selec=crearInfoMensual();

448

449 setsIdusuario(selec,atoi(idUsuario));

450 setsMesSeleccion(selec,atoi(mesSeleccion));

451 setsAnioSeleccion(selec,atoi(anioSeleccion));

452 setsIdVino1(selec,atoi(idVino1));

453 setsIdVino2(selec,atoi(idVino2));

454 setsIdVino3(selec,atoi(idVino3));

455 setsIdVino4(selec,atoi(idVino4));

456 setsIdVino5(selec,atoi(idVino5));

457 setsIdVino6(selec,atoi(idVino6));

458

459

460 **return** selec;

461

462 };

1 #include <iostream>

2 #include <stdlib.h>

3 #include <string.h>

4 #include "infoMensual.h"

5

6 **using namespace std**;

7

8 infoMesual crearInfoMensual(){

9

10 infoMesual e = **new** infoMesualEstructura;

11

12 **return** e;

13 }

14

15 **int** getsIdUsuarioInfo(infoMesual a){

16 **return** a->idUsuario;

17 }

18

19 **int** getsAnioSeleccion(infoMesual a){

20 **return** a->anioSeleccion;

21 }

22

23 **int** getsVino1(infoMesual a){

24 **return** a->idVino1;

25 }

26

27 **int** getsVino2(infoMesual a){

28 **return** a->idVino2;

29 }

30 **int** getsVino3(infoMesual a){

31 **return** a->idVino3;

32 }

33 **int** getsVino4(infoMesual a){

34 **return** a->idVino4;

35 }

36 **int** getsVino5(infoMesual a){

37 **return** a->idVino5;

38 }

39 **int** getsVino6(infoMesual a){

40 **return** a->idVino6;

41 }

42

43 **void** setsIdusuario(infoMesual a, **int** nuevoId){

44 a->idUsuario=nuevoId;

45 }

46 **void** setsMesSeleccion(infoMesual a, **int** nuevoMesSeleccion){

47 a->mesSeleccion=nuevoMesSeleccion;

48 }

49 **void** setsAnioSeleccion(infoMesual a, **int** nuevoAnioSeleccion){

50 a->anioSeleccion=nuevoAnioSeleccion;

51 }

52 **void** setsIdVino1(infoMesual a, **int** nuevoIdVino1){

53 a->idVino1=nuevoIdVino1;

54 }

55 **void** setsIdVino2(infoMesual a, **int** nuevoIdVino2){

56 a->idVino2=nuevoIdVino2;

57 }

58 **void** setsIdVino3(infoMesual a, **int** nuevoIdVino3){

59 a->idVino3=nuevoIdVino3;

60 }

61 **void** setsIdVino4(infoMesual a, **int** nuevoIdVino4){

62 a->idVino4=nuevoIdVino4;

63 }

64 **void** setsIdVino5(infoMesual a, **int** nuevoIdVino5){

65 a->idVino5=nuevoIdVino5;

66 }

67 **void** setsIdVino6(infoMesual a, **int** nuevoIdVino6){

68 a->idVino6=nuevoIdVino6;

69 }

1 #include <iostream>

2 #include<fstream>

3 #include "Lista.h"

4 #include "funcionesCargarArchivos.h"

5

6 **using namespace std**;

7

8 Lista crearLista(){

9 Lista lista=**new** listaStruct();

10 lista->tamanio=0;

11 **return** lista;

12 }

13

14 **void** agregarNodo(Lista lista,**void**\* dato){

15 Nodo newNodo=**new** nodo();

16 newNodo->dato=dato;

17 **if**(lista->tamanio>0){

18 //Busca el ultimo nodo actual

19 **int** i=1;

20 Nodo aux=lista->inicio;

21 **while**(aux->siguiente!=NULL){

22 i++;

23 aux=aux->siguiente;

24 }

25 //Se le asigna el nro del ultimo nodo + 1 y se lo asigna como siguente al

ultimo

26 newNodo->nro=(i+1);

27 newNodo->siguiente=NULL;

28 aux->siguiente=newNodo;

29 }**else**{

30 //lista vacia, crea el nodo y le asigna 1 al numero

31 newNodo->nro=1;

32 newNodo->siguiente=NULL;

33 lista->inicio=newNodo;

34 }

35 //aumenta el tamaño de la lista

36 lista->tamanio=(lista->tamanio+1);

37 }

38

39

40 **void** eliminarNodo(Lista lista,**int** nro){

41 //verifica que la lista no este vacia

42 **if**(lista->tamanio>0){

43 //busca el nodo a eliminar

44 Nodo ant=NULL;

45 Nodo aux=lista->inicio;

46 **int** encontrado=0;

47 **while**(aux!=NULL && encontrado==0){

48 **if**(aux->nro==(nro-1)){

49 ant=aux;

50 }

51 **if**(aux->nro==nro){

52 encontrado=1;

53 }**else**{

54 aux=aux->siguiente;

55 }

56 }

57 **if**(encontrado==1){

58 //si lo encuentra corrige los punteros del anterior nodo

59 **if**(ant!=NULL){

60 ant->siguiente=aux->siguiente;

61 }**else**{

62 lista->inicio=aux->siguiente;

63 }

64 //corrige los numeros de todos los nodos siguentes

65 Nodo sig=aux->siguiente;

66 **while**(sig!=NULL){

67 sig->nro=(sig->nro-1);

68 sig=sig->siguiente;

69 }

70 //elimina el nodo

71 **delete** aux->dato;

72 **delete** aux;

73 //ajusta el tamaño de la lista

74 lista->tamanio=(lista->tamanio-1);

75 }**else**{

76 **cout** << "No se encontro el elemento "<<nro<<" en la lista"<< **endl**;

77 }

78 }**else**{

79 **cout** << "La lista esta vacia"<< **endl**;

80 }

81 }

82

83

84 **void** eliminarLista(Lista lista){

85 Nodo aux=lista->inicio;

86 **while**(aux!=NULL){

87 **delete** aux->dato;

88 **delete** aux;

89 aux=aux->siguiente;

90 }

91 **delete** lista;

92 }

93

94 Nodo nodoSeleccionado(Lista lista,**int** nro){

95 Nodo aux=lista->inicio;

96 **if**(aux->nro==nro){

97 **return** aux;

98 }

99 **while**(aux!=NULL){

100 **if**(aux->nro==nro){

101 **return** aux;

102 }**else**{

103 aux=aux->siguiente;

104 }

105 }

106 **cout** << "No se encontro el elemento "<<nro<<" en la lista"<< **endl**;

107 **return** NULL;

108 }

109

110 **bool** listaVacia(Lista lista){

111 **if**(lista==NULL || lista->tamanio==0 || lista->inicio==NULL){

112 **return true**;

113 }**else**{

114 **return false**;

115 }

116 }

117

118 Nodo ultimoNodo(Lista lista){

119 Nodo aux=lista->inicio;

120 **while**(aux->siguiente!=NULL){

121 aux=aux->siguiente;

122 }

123 **return** aux;

124 }

125

126 Nodo anteriorNodo(Lista lista,Nodo nodo){

127 Nodo anterior=lista->inicio;

128 **while**(anterior->siguiente!=nodo && anterior->siguiente!=NULL){

129 anterior=anterior->siguiente;

130 }

131 **if**(anterior->siguiente==nodo){

132 **return** anterior;

133 }**else**{

134 **cout** << "No se encontro el nodo anterior al nodo "<<nodo->nro<<" en la

lista"<< **endl**;

135 **return** NULL;

136 }

137 }

138

139 **void** mostrarLista(Lista lista){

140 Nodo nodo=lista->inicio;

141 **cout** << "Nodo nro:"<<nodo->nro<<" Dato:"<<(nodo->dato)<< **endl**;

142 **while**(nodo!=NULL){

143 **cout** << "Nodo nro:"<<nodo->nro<<" Dato:"<<(nodo->dato)<< **endl**;

144 nodo=nodo->siguiente;

145 }

146 }

147

148 **int** ultimoAnio(){

149 Lista elecciones = crearLista();

150 leerYpasarAseleccion(elecciones);

151 **int** anio = 0;

152 **for**(**int** i = 1; i <= elecciones->tamanio; i++){

153 Nodo nodoInforMensual=nodoSeleccionado(elecciones,i);

154 infoMesual info = (infoMesual) nodoInforMensual->dato;

155 **if**(anio<getsAnioSeleccion(info)){

156 anio = getsAnioSeleccion(info);

157 }

158 }

159

160 **return** anio;

161 }

1 #include <iostream>

2 #include "menu.h"

3

4

5 **using namespace std**;

6

7 **int** main(){

8 IniciarMenu();

9 **return** 0;

10 }

1 #include <iostream>

2 #include "menu.h"

3 #include "lista.h"

4 #include "funcionesCargarArchivos.h"

5 #include "RankingVinos.h"

6 #include "RankingBodegas.h"

7 #include "RankingVarietales.h"

8

9 **void** IniciarMenu(){

10 Lista miListaCliente=crearLista();

11 Lista miListaVinos=crearLista();

12 Lista miListaSeleccion=crearLista();

13

14 leerYpasarAusuarios(miListaCliente);

15 leerYpasarAseleccion(miListaSeleccion);

16 leerYpasaraVinos(miListaVinos);

17 **int** opcion;

18 **bool** repetir=**true**;

19 **do**

20 {

21 system("cls");

22 **cout**<<"\n-----------------------------Menu

principal-----------------------------\n"<<**endl**;

23 **cout**<<"1-Ranking de vinos"<<**endl**;

24 **cout**<<"2-Ranking de bodegas"<<**endl**;

25 **cout**<<"3-Ranking de varietales"<<**endl**;

26 **cout**<<"4- Salir"<<**endl**;

27 **cin**>>opcion;

28 **switch**(opcion)

29 {

30 **case** 1:

31 rankingPorVinos(miListaSeleccion,miListaVinos);

32 system("pause");

33 **break**;

34 **case** 2:

35 lstBodegas(miListaVinos,miListaSeleccion);

36 system("pause");

37 **break**;

38 **case** 3:

39 rankingDeVarietales(miListaSeleccion,miListaCliente,miListaVinos);

40 system("pause");

41 **break**;

42 **case** 4:

43 **cout**<<"Adios!!"<<**endl**;

44 repetir=**false**;

45 **break**;

46 }

47 }

48 **while**(repetir);

49 }

1 #include <iostream>

2 #include<conio.h>

3 #include<stdio.h>

4 #include<stdlib.h>

5 #include <string.h>

6 #include<fstream>

7 #include "lista.h"

8 #include "vino.h"

9 #include "infoMensual.h"

10 #include "funcionesCargarArchivos.h"

11 #include "rankingBodegas.h"

12 #include "RankingVarietales.h"

13 #include <iomanip>

14 **using namespace std**;

15

16 **void** lstBodegas(Lista vinos, Lista infoSeleccionados)

17 {

18 Lista lista = crearLista();

19 //Guarda los datos de los regiustros catalogo.txt primero en nodos(infoVinos)

20 //Y luego en struct Vino

21 **int** ultimo = ultimoAnio();

22 **for**(**int** j = 1; j <= infoSeleccionados->tamanio; j++)

23 {

24 Nodo infoMen = nodoSeleccionado(infoSeleccionados,j);

25 infoMesual info = (infoMesual) infoMen->dato;

26 **if**(getsAnioSeleccion(info) == ultimo){

27 Vino vino1 = traerVino(vinos, getsVino1(info));

28 Vino vino2 =traerVino(vinos, getsVino2(info));

29 Vino vino3 =traerVino(vinos, getsVino3(info));

30 Vino vino4 =traerVino(vinos, getsVino4(info));

31 Vino vino5 =traerVino(vinos, getsVino5(info));

32 Vino vino6 =traerVino(vinos, getsVino6(info));

33

34 agregarBodega(lista,vino1);

35 agregarBodega(lista,vino2);

36 agregarBodega(lista,vino3);

37 agregarBodega(lista,vino4);

38 agregarBodega(lista,vino5);

39 agregarBodega(lista,vino6);

40 }

41 }

42

43 Lista ordenada=crearLista();

44

45 ordenarListaAuxBodega(lista,ordenada);

46

47 mostrarListaBodega(ordenada);

48

49 }

50

51 **void** mostrarListaBodega(Lista ordenada){

52 **cout** <<**endl**<<"Ranking de bodegas:"<<**endl**;

53 **cout** << **setw**( 60 ) << setfill( '-' ) << '\n' << setfill( ' ' );

54 **cout** << "| "<< left << **setw**( 14 ) << "Posicion"<< "|" << right << **setw**( 1) << "

Bodega" << right << **setw**( 13 )<< "|"<< "Cantidad Selecionado" << " |"<<**endl**;

55 **cout** << **setw**( 60 ) << setfill( '-' ) << '\n' << setfill( ' ' ) << '\n';

56 **for**(**int** b=1; b< ordenada->tamanio+1 ; b++)

57 {

58 Nodo nodoRanking=nodoSeleccionado(ordenada,b);

59 Bodega bod=(Bodega)nodoRanking->dato;

60

61 **int** resultado=strlen(bod->nombreBodega);

62

63 **cout** <<"Nro en lista:"<<nodoRanking->nro<<" -Bodega "<<bod->nombreBodega<<

right<< **setw**(33-resultado)<<" -Cant seleccionada: "<<bod->cant<<**endl**;

64 }

65

66 **cout**<<"\n"<<**endl**;

67 }

68

69 Bodega buscarBodega(Lista lista,**char** nombre[50])

70 {

71 Nodo aux=lista->inicio;

72 **int** encontrado=0;

73 **while**(aux!=NULL && encontrado==0)

74 {

75 Bodega bodega=(Bodega) aux->dato;

76 **if**(strcmp(bodega->nombreBodega,nombre)==0)

77 {

78 encontrado=1;

79 }

80 **else**

81 {

82 aux=aux->siguiente;

83 }

84 }

85 **if**(encontrado==1)

86 {

87 **return** (Bodega)aux->dato;

88 }

89 **else**

90 {

91 **return** NULL;

92 }

93 }

94

95 **void** agregarBodega(Lista bodega, Vino vino){

96

97 Bodega bod = buscarBodega(bodega,getsNombreBodega(vino));

98 **if**(bod!=NULL)

99 {

100 bod->cant++;

101 }

102 **else**

103 {

104 bod = **new** BodegaStruct;

105 bod->cant=1;

106 strcpy(bod->nombreBodega, getsNombreBodega(vino));

107 agregarNodo(bodega,bod);

108 }

109

110

111 }

112

113 **void** ordenarListaAuxBodega(Lista lista, Lista ordenada){

114 Nodo actual;

115 Nodo siguiente;

116 actual=lista->inicio;

117

118 Nodo auxNodo;

119 **int** nro;

120

121 **while** (actual!=NULL )

122 {

123 nro=actual->nro;

124 siguiente=actual->siguiente;

125 Bodega auxMaximo=(Bodega)actual->dato;

126 **while**( siguiente !=NULL )

127 {

128 Bodega rankig13=(Bodega)siguiente->dato;

129 **if**(auxMaximo->cant<rankig13->cant)

130 {

131 auxMaximo=rankig13;

132 nro=siguiente->nro;

133 }

134 siguiente=siguiente->siguiente;

135

136 }

137

138 Bodega nuevo=**new** BodegaStruct;

139 auxNodo=nodoSeleccionado(lista,nro);

140 Bodega bodegaAnt=(Bodega)auxNodo->dato;

141 nuevo->cant=bodegaAnt->cant;

142 strcpy(nuevo->nombreBodega,bodegaAnt->nombreBodega);

143

144 agregarNodo(ordenada,nuevo);

145 eliminarNodo(lista,nro);

146

147 actual=lista->inicio;

148

149

150 }

151

152

153

154

155 }

1 #include <iostream>

2 #include "RankingVinos.h"

3 #include "Lista.h"

4 #include "infoMensual.h"

5 #include "vino.h"

6 #include "cliente.h"

7 #include "RankingVarietales.h"

8 #include "string.h"

9 #include <iomanip>

10

11 **using namespace std**;

12

13 **void** rankingDeVarietales(Lista seleccionados, Lista clientes, Lista vinos){

14 Lista rankingRango1 = rankingPorRangoEtario(seleccionados, clientes, vinos, 0,

30);

15 Lista ordenada=crearLista();

16 ordenarListaAuxVarietales(rankingRango1,ordenada);

17 **cout**<<"Ranking de varietales de menores de 30:"<<**endl**;

18 mostrarRanking(ordenada);

19 Lista rankingRango2 = rankingPorRangoEtario(seleccionados, clientes, vinos, 30,

50);

20 Lista ordenada1=crearLista();

21 ordenarListaAuxVarietales(rankingRango2,ordenada1);

22 **cout**<<"Ranking de varietales de entre 30 y 50:"<<**endl**;

23 mostrarRanking(ordenada1);

24

25

26 Lista rankingRango3 = rankingPorRangoEtario(seleccionados, clientes, vinos, 50,

1000);

27 Lista ordenada2=crearLista();

28 ordenarListaAuxVarietales(rankingRango3,ordenada2);

29 **cout**<<"Ranking de varietales de mayores de 50:"<<**endl**;

30 mostrarRanking(ordenada2);

31

32 }

33 //rankingPorRangoEtario ?

34 Lista rankingPorRangoEtario(Lista seleccionados, Lista clientes,Lista vinos, **int**

valorEtarioInf, **int** valorEtarioSup) {

35 Lista listaResultante = crearLista();

36 Nodo nodoActual = seleccionados->inicio;

37

38 **while** (nodoActual != NULL){

39 infoMesual seleccionActual = (infoMesual) nodoActual->dato;

40 **int** edadClienteActual = traerEdadCliente(clientes, getsIdUsuarioInfo(

seleccionActual));

41

42 **if** (edadClienteActual > valorEtarioInf && edadClienteActual < valorEtarioSup

){

43

44 Vino vino1 = traerVino(vinos, getsVino1(seleccionActual));

45 Vino vino2 =traerVino(vinos, getsVino2(seleccionActual));

46 Vino vino3 =traerVino(vinos, getsVino3(seleccionActual));

47 Vino vino4 =traerVino(vinos, getsVino4(seleccionActual));

48 Vino vino5 =traerVino(vinos, getsVino5(seleccionActual));

49 Vino vino6 =traerVino(vinos, getsVino6(seleccionActual));

50

51 agregarVarietal(listaResultante,getsVarietal(vino1));

52 agregarVarietal(listaResultante, getsVarietal(vino2));

53 agregarVarietal(listaResultante,getsVarietal(vino3));

54 agregarVarietal(listaResultante, getsVarietal(vino4));

55 agregarVarietal(listaResultante, getsVarietal(vino5));

56 agregarVarietal(listaResultante, getsVarietal(vino6));

57

58 }

59 nodoActual = nodoActual->siguiente;

60 }

61

62 **return** listaResultante;

63 }

64

65 **void** agregarVarietal(Lista lista,**char** varietalSeleccionado[]){

66 Nodo nodoActual = lista->inicio;

67 **bool** encontrado = **false**;

68

69 **while**(nodoActual != NULL && !encontrado){

70 RankingVarietales ranking = (RankingVarietales) nodoActual->dato;

71

72 **if**(strcmp(ranking -> varietal, varietalSeleccionado )== 0 ){

73 encontrado = **true**;

74 ranking->cantidadSelecciones +=1;

75 }**else**{

76 nodoActual = nodoActual->siguiente;

77 }

78 }

79 **if**(!encontrado){

80 RankingVarietales nuevoVarietal = **new** RankingVarietalesStruct();

81 nuevoVarietal->cantidadSelecciones = 1;

82 strcpy(nuevoVarietal->varietal,varietalSeleccionado);

83 agregarNodo(lista, nuevoVarietal);

84 }

85 }

86

87 **int** traerEdadCliente(Lista clientes,**int** idBuscado){

88 Nodo nodoActual = clientes->inicio;

89 Cliente encontrado = NULL;

90 **while**(nodoActual != NULL && encontrado == NULL){

91 Cliente clienteActual = (Cliente) nodoActual->dato;

92 **if**(getsIdUsuario(clienteActual) == idBuscado){

93 encontrado = clienteActual;

94 }

95 nodoActual = nodoActual->siguiente;

96 }

97 **return** getsEdad(encontrado);

98 }

99

100 Vino traerVino(Lista vinos,**int** idBuscado){

101 Nodo nodoActual = vinos->inicio;

102 Vino encontrado = NULL;

103 **while**(nodoActual != NULL && encontrado == NULL){

104 Vino vinoActual = (Vino) nodoActual->dato;

105 **if**(getsVino(vinoActual)== idBuscado){

106 encontrado = vinoActual;

107 }

108 nodoActual = nodoActual->siguiente;

109 }

110 **return** encontrado;

111 }

112

113 //void ordenarRanking(Lista lista){

114 //

115 //}

116

117 **void** mostrarRanking(Lista ranking){

118

119 Nodo nodoActual = ranking->inicio;

120 **cout** << **setw**( 70 ) << setfill( '-' ) << '\n' << setfill( ' ' );

121 **cout** << "| "<< left << **setw**( 14 ) << "Posicion"<< "|" << right << **setw**( 1) << "

Varietal" << right << **setw**( 21 )<< "|"<< "Cantidad Selecionado" << " |"<<**endl**;

122 **cout** << **setw**( 70 ) << setfill( '-' ) << '\n' << setfill( ' ' ) << '\n';

123 **while**(nodoActual != NULL){

124 RankingVarietales puestoActual = (RankingVarietales) nodoActual->dato;

125 **int** resultado=strlen(puestoActual->varietal);

126 **cout**<<"Nro en lista: "<<nodoActual->nro<<" varietal: "<<puestoActual->

varietal<<right << **setw**(38-resultado)<<" Cant.selecciones: "<<puestoActual->

cantidadSelecciones<<**std**::**endl**;

127

128 nodoActual = nodoActual->siguiente;

129 }

130 **cout**<<"\n"<<**endl**;

131

132 }

133

134 **void** ordenarListaAuxVarietales(Lista lista, Lista ordenada){

135 Nodo actual;

136 Nodo siguiente;

137 actual=lista->inicio;

138 Nodo auxNodo;

139 **int** nro;

140

141 **while** (actual!=NULL )

142 {

143 nro=actual->nro;

144 siguiente=actual->siguiente;

145 RankingVarietales auxMaximo=(RankingVarietales)actual->dato;

146 **while**( siguiente !=NULL )

147 {

148 RankingVarietales rankig13=(RankingVarietales)siguiente->dato;

149 **if**(auxMaximo->cantidadSelecciones<rankig13->cantidadSelecciones)

150 {

151 auxMaximo=rankig13;

152 nro=siguiente->nro;

153 }

154 siguiente=siguiente->siguiente;

155

156 }

157 RankingVarietales nuevo = **new** RankingVarietalesStruct;

158 auxNodo=nodoSeleccionado(lista,nro);

159 RankingVarietales varietalesAnt=(RankingVarietales)auxNodo->dato;

160 nuevo->cantidadSelecciones=varietalesAnt->cantidadSelecciones;

161 strcpy(nuevo->varietal,varietalesAnt->varietal);

162

163 agregarNodo(ordenada,nuevo);

164 eliminarNodo(lista,nro);

165

166 actual=lista->inicio;

167

168

169 }

170 }

include <iostream>

2 #include "rankingVinos.h"

3 #include "lista.h"

4 #include "infoMensual.h"

5 #include "vino.h"

6 #include "RankingVarietales.h"

7 #include <iomanip>

8 #include <string.h>

9

10

11 **using namespace std**;

12

13 **void** rankingPorVinos (Lista seleccionados,Lista vinos)

14 {

15 Lista ranking = crearLista();

16 **int** ultimo = ultimoAnio();

17

18 **for**(**int** i=1; i<= seleccionados->tamanio ; i++)

19 {

20 Nodo nodoInforMensual=nodoSeleccionado(seleccionados,i);

21 infoMesual info = (infoMesual) nodoInforMensual->dato;

22 **if**(getsAnioSeleccion(info) == ultimo){

23 Vino vino1 = traerVino(vinos, getsVino1(info));

24 Vino vino2 =traerVino(vinos, getsVino2(info));

25 Vino vino3 =traerVino(vinos, getsVino3(info));

26 Vino vino4 =traerVino(vinos, getsVino4(info));

27 Vino vino5 =traerVino(vinos, getsVino5(info));

28 Vino vino6 =traerVino(vinos, getsVino6(info));

29

30 agregarVinos(ranking,vino1);

31 agregarVinos(ranking,vino2);

32 agregarVinos(ranking,vino3);

33 agregarVinos(ranking,vino4);

34 agregarVinos(ranking,vino5);

35 agregarVinos(ranking,vino6);

36

37 }

38 }

39

40 Lista ordenada=crearLista();

41 ordenarListaAux(ranking,ordenada);

42

43 mostrarListaRankigVino(ordenada);

44 }

45

46 RankingVinos buscarVino(Lista lista,**int** idVino)

47 {

48 Nodo aux=lista->inicio;

49 **int** encontrado=0;

50 **while**(aux!=NULL && encontrado==0)

51 {

52 RankingVinos rankig=(RankingVinos) aux->dato;

53 Vino vino=rankig->vino;

54 **if**(getsVino(vino)==idVino)

55 {

56 encontrado=1;

57 }

58 **else**

59 {

60 aux=aux->siguiente;

61 }

62 }

63 **if**(encontrado==1)

64 {

65 **return** (RankingVinos)aux->dato;

66 }

67 **else**

68 {

69 **return** NULL;

70 }

71 }

72

73 **void** agregarVinos(Lista ranking, Vino vino){

74

75 RankingVinos rankingDato=buscarVino(ranking,getsVino(vino));

76 **if** (rankingDato!=NULL)

77 {

78 rankingDato->cantidadSelecciones+=1;

79 }

80 **else**

81 {

82 rankingDato=**new** RankingVinosStruct();

83 rankingDato->cantidadSelecciones=1;

84 rankingDato->vino=vino;

85 agregarNodo(ranking,rankingDato);

86 }

87

88

89 }

90

91 **void** ordenarListaAux(Lista lista, Lista ordenada)

92 {

93 Nodo actual;

94 Nodo siguiente;

95 actual=lista->inicio;

96

97 Nodo auxNodo;

98

99 **int** nro;

100 **int** i=1;

101

102 **while** (actual!=NULL )

103 {

104 nro=actual->nro;

105 siguiente=actual->siguiente;

106 RankingVinos rankig1=(RankingVinos)actual->dato;

107 **while**( siguiente !=NULL )

108 {

109

110 RankingVinos rankig2=(RankingVinos)siguiente->dato;

111

112 **if**(rankig1->cantidadSelecciones<rankig2->cantidadSelecciones)

113 {

114 rankig1=rankig2;

115 nro=siguiente->nro;

116 }

117 siguiente=siguiente->siguiente;

118

119 }

120 RankingVinos nuevo=**new** RankingVinosStruct;

121 auxNodo=nodoSeleccionado(lista,nro);

122 RankingVinos vinoAnt=(RankingVinos)auxNodo->dato;

123 nuevo->cantidadSelecciones=vinoAnt->cantidadSelecciones;

124 nuevo->vino=vinoAnt->vino;

125

126 agregarNodo(ordenada,nuevo);

127 eliminarNodo(lista,nro);

128

129 actual=lista->inicio;

130 i++;

131

132 }

133

134 }

135

136

137

138

139 **void** mostrarListaRankigVino(Lista lista){

140 **int** j=1;

141 **cout** << **setw**( 80 ) << setfill( '-' ) << '\n' << setfill( ' ' );

142 **cout** << "| "<< left << **setw**( 14 ) << "Posicion"<< "|" << right << **setw**( 1) << "

Vino" << right << **setw**( 9 )

143 << " | " << "Etiqueta"<<right << **setw**(7)<< "|" << "Anio"<<right << **setw**(7) <<

"|"<< "Cantidad Selecionado" << " |"<<**endl**;

144 **cout** << **setw**( 80 ) << setfill( '-' ) << '\n' << setfill( ' ' ) << '\n';

145 **while**(j<=lista->tamanio){

146

147 Nodo nodoRanking=nodoSeleccionado(lista,j);

148 RankingVinos rankingDato=(RankingVinos)nodoRanking->dato;

149 Vino vino1=rankingDato->vino;

150 **int** resultado=strlen(getsetiqueta(vino1)); //cuento El tamaño de la cadena

para despues poder restar

151

152 **cout** <<"Nro en lista: "<<nodoRanking->nro<<" -Vino "<<getsVino(vino1)<< " :

"<<getsetiqueta(vino1)<<right << **setw**(16-resultado)<< ""<<getsAnioCosecha(vino1)<<right

<< **setw**(7) << ""<<" Cant seleccionada: "<<rankingDato->cantidadSelecciones<<**endl**;

153 j++;

154 }

155

156 **cout**<<"\n"<<**endl**;

157

158 }

1 #include <iostream>

2 #include <stdlib.h>

3 #include <string.h>

4 #include "vino.h"

5

6 **using namespace std**;

7

8 Vino crearVino(){

9 Vino e = **new** VinoEstructura;

10 **return** e;

11 }

12

13 **int** getsVino(Vino a){

14 **return** a->idVino;

15 }

16

17 **char** \* getsetiqueta(Vino a){

18 **return** a->etiqueta;

19 }

20

21 **char** \* getsNombreBodega(Vino a){

22 **return** a->bodega;

23 }

24 **char** \* getsVarietal(Vino a){

25 **return** a->varietal;

26 }

27

28 **int** getsAnioCosecha(Vino a){

29 **return** a->anioCosecha;

30 }

31

32 **void** setsIdVino(Vino a, **int** nuevoIdvino){

33 a->idVino=nuevoIdvino;

34 }

35 **void** setsEtiqueta(Vino a, **char** nuevaEtiqueta[50]){

36 strcpy(a->etiqueta,nuevaEtiqueta);

37 }

38 **void** setsBodega(Vino a, **char** nuevoBodega[50]){

39 strcpy(a->bodega,nuevoBodega);

40 }

41 **void** setsSegmentoVino(Vino a, **char** nuevoSegmentoVino[50]){

42 strcpy(a->segmento\_del\_vino,nuevoSegmentoVino);

43 }

44 **void** setsVarietal(Vino a, **char** nuevoVarietal[50]){

45 strcpy(a->varietal,nuevoVarietal);

46 }

47 **void** setsAnioCosecha(Vino a, **int** nuevoAnioCosecha){

48 a->anioCosecha=nuevoAnioCosecha;

49 }

50 **void** setsTerroir(Vino a, **char** nuevoTerroir[50]){

51 strcpy(a->terroir,nuevoTerroir);

52 }

1 #ifndef FUNCIONESCARGARARCHIVOS\_H\_INCLUDED

2 #define FUNCIONESCARGARARCHIVOS\_H\_INCLUDED

3

4 #include "infoMensual.h"

5 #include "vino.h"

6 #include "cliente.h"

7 #include "lista.h"

8 #include <string.h>

9

10

11 **using namespace std**;

12

13 **void** leerYpasarAusuarios(Lista clien);

14 **void** leerYpasarAseleccion(Lista selec);

15 **void** leerYpasaraVinos(Lista vin);

16

17

18 Cliente pasarCharAStructConStringUsuario(**string** texto);

19 Vino pasarCharAStructConStringVino(**string** texto);

20 infoMesual pasarCharAstructSeleccion(**string** texto);

21

22

23 #endif // FUNCIONESCARGARARCHIVOS\_H\_INCLUDED

1 #ifndef INFOMENSUAL\_H\_INCLUDED

2 #define INFOMENSUAL\_H\_INCLUDED

3

4 /\*

5 Definicion de tipo de dato InfoMensual para representar los datos de las selecciones

de los Clientes

6 Atributos:

7 idUsuario

8 mesSeleccion

9 anioSeleccion

10 idVino1

11 idVino2

12 idVino3

13 idVino4

14 idVino5

15 idVino6

16 Axiomas

17 idUsuario: mayor a 0

18 mesSeleccion: entre 1 y 12

19 anioSeleccion: entre 0 y 2147483647

20 idVino1: entre 0 y 2147483647

21 idVino2: entre 0 y 2147483647

22 idVino3: entre 0 y 2147483647

23 idVino4: entre 0 y 2147483647

24 idVino5: entre 0 y 2147483647

25 idVino6: entre 0 y 2147483647

26 \*/

27

28 /\*Tipo de estructura\*/

29

30 **struct** infoMesualEstructura{

31 **int** idUsuario;

32 **int** mesSeleccion;

33 **int** anioSeleccion;

34 **int** idVino1;

35 **int** idVino2;

36 **int** idVino3;

37 **int** idVino4;

38 **int** idVino5;

39 **int** idVino6;

40 };

41 **typedef struct** infoMesualEstructura \* infoMesual;

42

43 **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

44

45

46 /\*getters y setters\*/

47

48 **int** getsIdUsuarioInfo(infoMesual a);

49 **int** getsAnioSeleccion(infoMesual a);

50 **int** getsVino1(infoMesual a);

51 **int** getsVino2(infoMesual a);

52 **int** getsVino3(infoMesual a);

53 **int** getsVino4(infoMesual a);

54 **int** getsVino5(infoMesual a);

55 **int** getsVino6(infoMesual a);

56

57 **void** setsIdusuario(infoMesual a, **int** nuevoId);

58 **void** setsMesSeleccion(infoMesual a, **int** nuevoMesSeleccion);

59 **void** setsAnioSeleccion(infoMesual a, **int** nuevoAnioSeleccion);

60 **void** setsIdVino1(infoMesual a, **int** nuevoIdVino1);

61 **void** setsIdVino2(infoMesual a, **int** nuevoIdVino2);

62 **void** setsIdVino3(infoMesual a, **int** nuevoIdVino3);

63 **void** setsIdVino4(infoMesual a, **int** nuevoIdVino4);

64 **void** setsIdVino5(infoMesual a, **int** nuevoIdVino5);

65 **void** setsIdVino6(infoMesual a, **int** nuevoIdVino6);

66

67 **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

68

69 /\*

70 precondicion: ninguna

71 postcondicion: crea la InfoMensual

72 \*/

73

74 infoMesual crearInfoMensual();

75

76 #endif // INFOMENSUAL\_H\_INCLUDED

1 #ifndef LISTA\_H\_

2 #define LISTA\_H\_

3

4 **/\*\***

5 **Definición del Tipo de Dato para manejo de Listas.**

6 **Atributos:**

7 **\* nro,**

8 **\* dato,**

9 **\* nodoSiguiente,**

10

11 **\*/**

12

13 **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

14

15

16 **typedef struct** nodo\* Nodo;

17

18 **typedef struct** listaStruct\* Lista;

19

20 **struct** nodo {

21 **int** nro;

22 **void**\* dato;

23 Nodo siguiente;

24 };

25

26 **struct** listaStruct {

27 **int** tamanio;

28 Nodo inicio;

29 };

30

31 **/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

32 /\* Definicion de Primitivas \*/

33 /\*--------------------------\*/

34

35 /\*

36 PRE : El Lista no debe haber sido creado.

37 POST: El Lista queda creado y preparado para ser usado.

38 \*/

39 Lista crearLista();

40

41 /\*

42 PRE : El Lista debe haber sido creada.

43 POST: Crea nuevo nodo en la lista con el dato enviado por parametro.

44 \*/

45 **void** agregarNodo(Lista lista,**void**\* dato);

46

47 /\*

48 PRE : El Lista debe haber sido creada.

49 POST: Elimina el nodo de nro igual al que se envio por parametro

50 \*/

51 **void** eliminarNodo(Lista lista,**int** nro);

52

53 /\*

54 PRE : El Lista debe haber sido creada.

55 POST: Elimina todos los nodos de la lista liberando la memoria

56 \*/

57 **void** eliminarLista(Lista lista);

58

59 **bool** listaVacia(Lista lista);

60

61 Nodo nodoSeleccionado(Lista lista,**int** nro);

62

63 Nodo ultimoNodo(Lista lista);

64

65 Nodo anteriorNodo(Lista lista,Nodo nodo);

66

67 **void** mostrarLista(Lista lista);

68

69 **int** ultimoAnio();

70 #endif

1 #ifndef MENU\_H\_INCLUDED

2 #define MENU\_H\_INCLUDED

3

4 **void** IniciarMenu();

5

6 #endif

1 #ifndef RANKINGBODEGAS\_H\_INCLUDED

2 #define RANKINGBODEGAS\_H\_INCLUDED

3 #include "lista.h"

4

5 //Estructura que guarda el nombre de una bodega y la lista de vinos que contiene

6 **typedef struct** BodegaStruct{

7 **char** nombreBodega[50];

8 **int** cant;

9 }\*Bodega;

10 //Función que crea una lista de bodegas.

11 //Abre el archivo catalogos.txt y toma el nombre de la bodega del primer registro.

12 //Luego, agrega el vino a la Lista vinos de la bodega recién creada.

13 //Pasa al segundo registro. Si el nombre de la bodega ya existe en la lista,

14 //agrega otro vino a la Lista vinos de la bodega ya creada; sino, crea una nueva,

15 //la agrega a la lstBodegas y hace lo mismo con el vino

16 **void** lstBodegas(Lista catalogo, Lista info);

17

18 Bodega buscarBodega(Lista lista,**char** nombre[50]);

19

20 **void** agregarBodega(Lista bodega, Vino vino);

21

22

23 **void** mostrarListaBodega(Lista ordenada);

24

25 **void** ordenarListaAuxBodega(Lista lista, Lista ordenada);

26 #endif // RANKINGBODEGAS\_H\_INCLUDED

1 #ifndef RANKINGVARIETALES\_H\_INCLUDED

2 #define RANKINGVARIETALES\_H\_INCLUDED

3

4 **struct** RankingVarietalesStruct{

5 **int** posicion;

6 **int** cantidadSelecciones;

7 **char** varietal [50];

8 };

9

10 **typedef struct** RankingVarietalesStruct\* RankingVarietales;

11

12 **int** traerEdadCliente(Lista clientes,**int** idBuscado);

13

14 Vino traerVino(Lista vinos,**int** idBuscado);

15

16 **void** agregarVarietal(Lista lista,**char** varietalSeleccionado [50]);

17

18 **void** rankingDeVarietales(Lista seleccionados, Lista clientes, Lista vinos);

19

20 Lista rankingPorRangoEtario(Lista seleccionados, Lista clientes,Lista vinos, **int**

valorEtarioInf, **int** valorEtarioSup);

21

22 **void** mostrarRanking(Lista ranking);

23

24 **void** ordenarListaAuxVarietales(Lista lista, Lista ordenada);

25

26 #endif // RANKINGVARIETALES\_H\_INCLUDED

1 #ifndef RANKINGVINOS\_H\_INCLUDED

2 #define RANKINGVINOS\_H\_INCLUDED

3 #include "vino.h"

4 #include "lista.h"

5

6 **struct** RankingVinosStruct{

7 **int** cantidadSelecciones;

8 Vino vino;

9 };

10 **typedef struct** RankingVinosStruct\* RankingVinos;

11

12

13 **void** rankingPorVinos (Lista seleccionados,Lista vinos);

14 **void** ordenarListaAux(Lista lista, Lista ordenada);

15 **void** agregarVinos(Lista ranking, Vino vino);

16 RankingVinos buscarVino(Lista lista,**int** idVino);

17 **void** mostrarListaRankigVino(Lista lista);

18 #endif // RANKINGVINOS\_H\_INCLUDED