```
1
2   #include "listaHabitaciones.h"
3
4
5   extern bool existeCliente(const tId dni, tRegistroClientes ocupacion){
6
7      bool encontrado = false;
8      int compara;
9      int n = 0;
10
11     while ((n < ocupacion.numeroClientes)&&(!encontrado)){
12        compara = strcmp(ocupacion.clientes[n].dni, dni);
13        if (compara == 0){
14           encontrado = true;
15        }
16        n++;
17     }
18
19     return encontrado;
20  }
21
22  extern bool habitacionLlena (const tRegistroClientes ocupacion){
23
24      bool llena = false;
25
26
27      if(ocupacion.numeroClientes < MAX_CLIENTES){
28          llena = false;
29      }
30      else{
31           llena = true;
32      }
33
34      return llena;
35  }
36
37  void vaciarHabitacion (tRegistroClientes *ocupacion){
38
39          ocupacion->numeroClientes = 0;
40  }
41
42  extern int altaCliente (tId dni, tNombre nombre, bool allInclusive, tRegistroClientes
    *ocupacion){
43
44      int error;
45      bool encontrado = false;
46
47      leerTexto("Cliente-DNI: ", dni, MAX_ID);
48      leerTexto("Cliente-Nombre: ", nombre, MAX_NOMBRE_CLIENTE);
49      allInclusive = leerBooleano("El cliente tiene todo incluido? (s/n): ");
50
51
52      encontrado = existeCliente(dni, *ocupacion);
53       if(encontrado == false){
54           if (!habitacionLlena(*ocupacion)) {
55
56               int posicionHueco = ocupacion->numeroClientes;
57               strncpy(ocupacion->clientes[posicionHueco].dni, dni, MAX_ID);
58               strncpy(ocupacion->clientes[posicionHueco].nombre, nombre, MAX_NOMBRE_CLIENTE);
59               ocupacion->clientes[posicionHueco].todoIncluido = allInclusive;
60               error = 0;
61
62          }else{
63              error = 2;
64          }
65      }else{
66          error = 1;
67      }
68
69      return error;
70  }
71
72  extern void listarClientes (const tRegistroClientes ocupacion){
73
74      tId dniCliente;
75     int i;
76     bool Incluido;
77
78      for(i = 0; i < MAX_CLIENTES; i++){
79              if(existeCliente(dniCliente, ocupacion)){
80                  Incluido = ocupacion.clientes[i].todoIncluido;
81
82                  if (Incluido == true){
83                      printf("***Cliente %d: %s - Todo incluido \n", i+1,
```

```
    ocupacion.clientes[i].dni, ocupacion.clientes[i].nombre);
84                  }else{
85                      printf("***Cliente %d: %s - Solo desayuno \n", i+1,
    ocupacion.clientes[i].dni, ocupacion.clientes[i].nombre);
86                  }
87              }
88      }
89  }
90
91
92  extern bool habitacionRegistrada (int numero, const tListaHabitaciones habitaciones, int
    *pos){
93
94      bool encontrado = false;
95
96
97      while ((*pos) < MAX_HABITACIONES  && encontrado == false){
98
    if((!habitaciones[*pos].posicionLibre)&&(habitaciones[*pos].habitacion.numeroHabitacion ==
    numero)){
99              encontrado = true;
100         }else{
101          *pos =  *pos + 1;
102         }
103     }
104     return encontrado;
105 }
106
107
108  extern bool hayEspacioEnHotel (const tListaHabitaciones habitaciones, int *pos){
109
110      bool haySitio = false;
111     (*pos)=0;
112
113     while (*pos < MAX_HABITACIONES && !haySitio) {
114         if(habitaciones[*pos].posicionLibre){
115             haySitio = true;
116
117         }else{
118           (*pos)++;
119         }
120     }
121 return haySitio;
122 }
123
124 void abrirHotel (tListaHabitaciones *habitaciones){
125
126     int i;
127     printf("***Hotel abierto*** \n");
128     for(i = 0; i < MAX_HABITACIONES; i++){
129
130         (*habitaciones)[i].habitacion.ocupacion.numeroClientes = 0;
131         (*habitaciones)[i].posicionLibre = true;
132
133     }
134 }
135
136 extern int annadirHabitacion (int numero, tTextoTipo tipo, tListaHabitaciones
    *habitaciones){
137
138     int error;
139     int posicionExiste=0;
140     int posicionHueco;
141
142
143
144     if (habitacionRegistrada(numero, *habitaciones, &posicionExiste)){
145             error = 1;
146
147     }else{
148
149         if (!hayEspacioEnHotel(*habitaciones, &posicionHueco)){
150             error = 2;
151         }else{
152             (*habitaciones)[posicionHueco].habitacion.numeroHabitacion = numero;
153             strncpy((*habitaciones)[posicionHueco].habitacion.tipo, tipo, MAX_TIPO);
154             (*habitaciones)[posicionHueco].posicionLibre = false;
155            error = 0;
156         }
157     }
158
159     return error;
160 }
161
```

```
162  extern bool borrarHabitacion (int numero, tListaHabitaciones *habitaciones){
163
164      bool eliminado = false;
165      int posicion=0;
166
167      if (habitacionRegistrada(numero, *habitaciones, &posicion)) {
168          (*habitaciones)[posicion].posicionLibre = true;
169          (*habitaciones)[posicion].habitacion.numeroHabitacion = 0;
170
171          (*habitaciones)[posicion].habitacion.ocupacion.numeroClientes = 0;
172          eliminado = true;
173      }
174
175      return eliminado;
176  }
177
178  extern void listarHabitacionesOcupadas (tListaHabitaciones habitaciones){
179
180          for(int i = 0; i < MAX_HABITACIONES; i++){
181                  if (habitaciones[i].posicionLibre == false){
182                      printf("-Habitacion %d (%s). Numero de ocupantes: %d. Listado:
    \n",habitaciones[i].habitacion.numeroHabitacion, habitaciones[i].habitacion.tipo,
    habitaciones[i].habitacion.ocupacion.numeroClientes);
183
184                  }
185          }
186  }
187
188
189
190
191  /** CÓDIGO DE FUNCIONES FASE 3*/
192
193  extern bool buscarCliente (const tId dni, const tListaHabitaciones habitaciones, int
    *habitacion){
194
195      bool encontrado = false;
196      int i;
197      tRegistroClientes ocupacionHabitacion;
198      int pos = 0;
199
200      for( i = 0; i < MAX_HABITACIONES; i++){
201                  if(existeCliente(dni, ocupacionHabitacion) &&
    habitacionRegistrada(*habitacion, habitaciones, &pos)){
202                      encontrado = true;
203                      pos = i;
204                      }
205
206      }
207
208      return encontrado;
209  }
210
211  extern int totalClientesEnHotel (const tListaHabitaciones habitaciones){
212
213      int numeroClientes = 0;
214
215      for (int i=0; i < MAX_HABITACIONES;i++){
216          numeroClientes = numeroClientes +
    habitaciones[i].habitacion.ocupacion.numeroClientes;
217      }
218
219
220  return numeroClientes;
221  }
222
223
224  /** CÓDIGO DE FUNCIONES FASE 4: */
225  extern void MenuRegistraHabitacion (tListaHabitaciones *habitaciones){
226
227
228      tTextoTipo tipoHabitacion;
229      int errorRegistro;
230
231      int numHabitacion = leerEnteroRango("Numero de habitacion: ",NUM_HABITACION_INF,
    NUM_HABITACION_SUP);
232      leerTexto("Tipo de habitacion: ",tipoHabitacion, MAX_TIPO);
233
234
235      errorRegistro = annadirHabitacion(numHabitacion, tipoHabitacion, habitaciones);
236
237        if (errorRegistro == 0){
238           printf("***Habitacion %d dada de alta correctamente***\n", numHabitacion);
239        }
```

```
240         else{
241            if (errorRegistro == 1)
242               printf("***Error, no se pudo realizar el alta: La habitacion %d ya fue
    registrada***\n", numHabitacion);
243            else
244               printf("***Error, no se pudo realizar el alta: El hotel no tiene permisos para
    abrir mas habitaciones***\n");
245         }
246      }
247
248   extern void MenuEliminaHabitacion (tListaHabitaciones *habitaciones){
249
250       int numHabitacion;
251
252      numHabitacion = leerEnteroRango("Numero de habitacion a borrar: ",NUM_HABITACION_INF,
    NUM_HABITACION_SUP);
253
254       if (borrarHabitacion(numHabitacion, &(*habitaciones))){
255
256               printf("***La habitacion %d ha sido borrada se su sitema***\n", numHabitacion);
257
258       }else{
259           printf("***Error, la habitacion %d no consta como registrada en su sistema***\n",
    numHabitacion);
260       }
261   }
262
263   extern void MenuRegistraClientes (tListaHabitaciones *habitaciones){
264
265       tId dniCliente;
266       tNombre nombreCliente;
267       tRegistroClientes ocupacionHabitacion;
268
269       int errorAltaCliente;
270       bool todoIncluido = false;
271       bool respuesta;
272       int posicion;
273       int pos=0;
274       int error = 0;
275
276
277       int  numHabitacion = leerEnteroRango("Numero de habitacion en la que registrar
    cliente: ",NUM_HABITACION_INF, NUM_HABITACION_SUP);
278       bool registrado = habitacionRegistrada(numHabitacion, *habitaciones, &pos);
279
280       do{
281
282       if(registrado == true ){
283
284           for(int i = 0; i < MAX_CLIENTES; i++){
285
    strncpy(ocupacionHabitacion.clientes[i].dni,(*habitaciones)[pos].habitacion.ocupacion.client
    es[i].dni , MAX_ID);
286           }
287
288           ocupacionHabitacion.numeroClientes =
    (*habitaciones)[pos].habitacion.ocupacion.numeroClientes;
289           errorAltaCliente = altaCliente(dniCliente, nombreCliente, todoIncluido,
    &ocupacionHabitacion);
290
291           if (errorAltaCliente == 0){
292
293               posicion = ocupacionHabitacion.numeroClientes;
294
295               strncpy((*habitaciones)[pos].habitacion.ocupacion.clientes[posicion].dni,
    ocupacionHabitacion.clientes[posicion].dni,MAX_ID);
296               strncpy((*habitaciones)[pos].habitacion.ocupacion.clientes[posicion].nombre,
    ocupacionHabitacion.clientes[posicion].nombre, MAX_NOMBRE_CLIENTE);
297                (*habitaciones)[pos].habitacion.ocupacion.clientes[posicion].todoIncluido =
    ocupacionHabitacion.clientes[posicion].todoIncluido;
298
299               (*habitaciones)[pos].habitacion.ocupacion.numeroClientes++;
300               printf("***Cliente dado de alta correctamente***\n");
301
302           }else{
303               if (errorAltaCliente == 1){
304                       printf("***Error, no se pudo realizar el alta: El DNI %s ya fue
    registrado previamente***\n", dniCliente);
305               }else{
306                   printf("***Error, no se pudo realizar el alta: La habitacion %d esta
    llena***\n", numHabitacion);
307                   }
308           }
309       respuesta = leerBooleano("\nDesea introducir mas clientes en la habitacion? (s/n): ");
```

```
310
311         }else{
312
313             printf("***Error, la habitacion %d no aparece como registrada en su sistema***\n",
    numHabitacion);
314              error = 1;
315         }
316
317        }while(respuesta == true && error != 1);
318    }
319
320    extern void MenuListaTotal (tListaHabitaciones habitaciones){
321
322        int i;
323        bool Incluido;
324        bool posicionLibre = false;
325        bool registrado = false;
326        int numClientes;
327
328            for( i = 0; i < MAX_HABITACIONES; i++){
329                    posicionLibre = habitaciones[i].posicionLibre;
330
331                    if(posicionLibre == false){
332                        printf("-Habitacion %i (%s). Numero de ocupantes: %i. Listado:
    \n",habitaciones[i].habitacion.numeroHabitacion, habitaciones[i].habitacion.tipo,
    habitaciones[i].habitacion.ocupacion.numeroClientes);
333                        numClientes = habitaciones[i].habitacion.ocupacion.numeroClientes;
334                        registrado = true;
335
336                        for(int j = 0; j < numClientes && numClientes !=0 ; j++){
337                            Incluido =
    habitaciones[i].habitacion.ocupacion.clientes[j].todoIncluido;
338                                if (Incluido == true){
339                                    printf("***Cliente %d: %s - %s - Todo incluido \n", j+1,
    habitaciones[i].habitacion.ocupacion.clientes[j].dni,
    habitaciones[i].habitacion.ocupacion.clientes[j].nombre);
340                                }else{
341                                    printf("***Cliente %d: %s - %s - Solo desayuno \n", j+1,
    habitaciones[i].habitacion.ocupacion.clientes[j].dni,
    habitaciones[i].habitacion.ocupacion.clientes[j].nombre);
342                                }
343                        }
344                        if(numClientes == 0 ){
345                                printf("***No hay clientes en la habitacion \n");
346                        }
347
348                }
349            }
350            if(registrado == false){
351                printf("***No hay ninguna habitacion registrada en su hotel \n");
352            }
353
354    }
355
356    extern void MenuListaHabitacion (tListaHabitaciones habitaciones){
357
358
359        int i;
360        int j;
361        int numHabitacion;
362        int numClientes;
363        bool registrado = false;
364        bool Incluido = false;
365        printf("Introduzca el numero de la habitacion: ");
366        scanf("%d",&numHabitacion);
367
368        for(i=0; i<MAX_HABITACIONES && !registrado; i++){
369               registrado =  habitacionRegistrada(numHabitacion, habitaciones, &i);
370               numClientes = habitaciones[i].habitacion.ocupacion.numeroClientes;
371
372               if(registrado == true){
373                    for(j=0; j < numClientes && numClientes != 0;  j++){
374                        Incluido =
    habitaciones[i].habitacion.ocupacion.clientes[j].todoIncluido;
375                        if (Incluido == true){
376                                printf("***Cliente %d: %s - %s - Todo incluido \n", j+1,
    habitaciones[i].habitacion.ocupacion.clientes[j].dni,
    habitaciones[i].habitacion.ocupacion.clientes[j].nombre);
377                                }else{
378                                    printf("***Cliente %d: %s - %s - Solo desayuno \n", j+1,
    habitaciones[i].habitacion.ocupacion.clientes[j].dni,
    habitaciones[i].habitacion.ocupacion.clientes[j].nombre);
379                                }
380                        }
```

```
381                     }
382         }
383
384         if(registrado == false ){
385                 printf("***Error, la habitacion %d no aparece como registrada en su
    sistema***\n", numHabitacion);
386         }
387
388         if(numClientes == 0){
389             printf("***No hay clientes en la habitacion \n");
390         }
391     }
392
393     extern void MenuBuscaCliente (tListaHabitaciones habitaciones){
394
395          bool encontrado = false;
396         int i;
397         int j;
398         tId dniCliente;
399
400         leerTexto("Indiqueme el DNI del cliente a buscar: ", dniCliente, MAX_ID);
401
402         for(i=0; i < MAX_HABITACIONES && !encontrado;  i++){
403                 for(j=0; j <  ( habitaciones[i].habitacion.ocupacion.numeroClientes);  j++)   {
404
    if(strcmp(dniCliente,habitaciones[i].habitacion.ocupacion.clientes[j].dni) == 0 &&
    habitaciones[i].habitacion.numeroHabitacion !=0){
405                             encontrado = true;
406                     }
407                 }
408                 if(encontrado){
409                         printf("***El cliente se encuentra en la habitacion numero
    %i***",habitaciones[i].habitacion.numeroHabitacion);
410                 }
411         }
412         if(encontrado == false){
413                 printf("***El cliente indicado no se encuentra alojado en el hotel***");
414
415         }
416     }
417
418     extern void MenuCuentaClientes (tListaHabitaciones habitaciones){
419
420         int numeroTotal = totalClientesEnHotel (habitaciones);
421
422         printf("El numero total de clientes alojados actualmente en el hotel es de %d
    personas", numeroTotal);
423
424     }
425
426
427
```