CAPITULO 1

Ejercicios Propuestos

´Indice

1.Enunciados Ficheros

Enunciado agenda.c

Hacer una agenda telefonica con listas simples con la siguiente estructura:

Listing 1.1:

```
struc entrada {
char *nombre;
char *direccion;
char  *telefono;
};

struct nodoagenda { struct
 entrada datos ; struct
 nodoagenda *sig;
};

typedef  struct  nodoagenda  *tipoagenda;
```

El menu principal será:

Listing 1.2:

```
printf("Menú:\n");
printf("1) Ver contenido completo de la agenda.\n");
printf("2) Dar de alta una persona.\n");
printf("3) Buscar teléfonos de una persona.\n");
printf("4)  Salir.\n");
```

- Al inicio del programa deberá leer de un fichero.

Listing 1.3:

```
1 Tipoagenda lee_agenda ( char nombre_fichero[] )
```

- Antes de terminar el programa deberá guardarse la agenda.

Listing 1.4:

```
1 void escribe_agenda((Tipo_agenda agenda, char nombre_fichero[]);
```

- Las operaciones del menú se harán con punteros.

CAPITULO 2

Soluciones ejercicios propuestos

Índice

1.Soluciones de Ficheros

Solución agenda.c

Listing 2.1: ../cejerciciosp/ficheros/agenda.c

```
1  /*
2  Agenda con listas simples y ficheros
3  */
4
5  #include <stdio.h>
6  #include <stdlib.h>
7  #include <string.h>
8
9  #define MAXCADENA 200
10
11 enum { Ver=1, Alta, Buscar, Salir };
12
13 struct Entrada {
14   char * nombre;
15   char * direccion;
16   char * telefono;
17 };
18
19 struct Nodo Agenda {
20   struct Entrada datos;
21   struct Nodo Agenda * sig;
22 };
23
24 typedef struct NodoAgenda * TipoAgenda;
25
26 void quita_fin_de_linea(char linea[])
27 {
```

```
28   int i;
29   for (i=0; linea[i] != '\0'; i++)
30     if ( linea [ i] == '\ n'){
31       linea[i] = '\O';
32       break;
33     }
34 }
35
36 void muestra_entrada(struct NodoAgenda * e)
37   // Podríamos haber pasado e por valor, pero resulta más eficiente
38   // /y no mucho más
39   // incómodo) hacerlo por referencia:
40   //pasamos así sólo 4 bytes en lugar de 12.
41 {
42   printf("Nombre   : %s\n", e->datos.nombre);
43   printf("Dirección: %s\n", e->datos.direccion);
44   printf("Teléfono : %s\n", e->datos.telefono);
45 }
46
47 void libera_entrada(struct NodoAgenda * e)
48 {
49   int i;
50
51   free(e->datos.nombre);
52   free (e-> datos . direccion );
53   free (e-> datos . telefono );
54   free (e);
55 }
56
57
58 TipoAgenda crea_agenda(void)
59 {
60   return NULL;
61 }
62
63 struct NodoAgenda * buscar_entrada_por_nombre(TipoAgenda agenda, char nombre[])
64 {
65   struct NodoAgenda * aux;
66
67   for ( aux = agenda ; aux != NULL ; aux = aux -> sig)
68     if (strcmp(aux->datos.nombre, nombre) == 0)
69       return aux;
70
71   return NULL;
72 }
73
74
75 TipoAgenda anyadir_entrada(  TipoAgenda agenda, char nombre[],
76                              char direccion[], char telefono[])
77 {
78   struct Nodo Agenda  * aux , * e;
79
80   /* Averiguar si ya tenemos una persona con ese nombre */
81   if (buscar_entrada_por_nombre(agenda, nombre)  != NULL)
82     return agenda;
83
84   /* Si llegamos aquí, es porque no teníamos registrada a esa persona. */
85   e = malloc(sizeof(struct NodoAgenda));
86   e->datos.nombre = malloc((strlen(nombre)+1)*sizeof(char));
87   strcpy(e->datos.nombre, nombre);
88   e->datos.direccion = malloc((strlen(direccion)+1)*sizeof(char));
89   strcpy(e->datos.direccion,direccion);
90   e->datos.telefono = malloc((strlen(telefono)+1)*sizeof(char));
91   strcpy(e->datos.telefono, telefono);
92   e->sig = agenda;
```

```
93    agenda = e;
94    return agenda;
95  }
96
97  void muestra_agenda(TipoAgenda agenda)
98  {
99    struct NodoAgenda * aux;
100
101   for ( aux = agenda ; aux != NULL ; aux = aux -> sig)
102     muestra_entrada(aux);
103 }
104
105
106 void libera_agenda(TipoAgenda agenda)
107 {
108   struct Nodo Agenda * aux , * siguiente ;
109
110   aux = agenda;
111   while ( aux != NULL ) {
112     siguiente = aux->sig;
113     libera_entrada(aux);
114     aux = siguiente ;
115   }
116 }
117
118 void escribe_agenda(TipoAgenda agenda, char nombre_fichero[])
119 {
120   struct Nodo Agenda * aux;
121   FILE * fp;
122
123   fp = fopen(nombre_fichero, "w");
124   for (aux=agenda; aux!=NULL; aux=aux->sig)
125     fprintf(fp, "%s\n%s\n%s\n",    aux->datos.nombre,
126                                    aux->datos.direccion,
127                                    aux->datos.telefono);
128   fclose(fp);
129 }
130
131 TipoAgenda lee_agenda(char nombre_fichero[])
132 {
133   TipoAgenda agenda;
134   struct Entrada * entrada_ leida ;
135   FILE * fp;
136   char nombre[MAXCADENA+1], direccion[MAXCADENA+1], telefono[MAXCADENA+1];
137   int longitud ;
138
139   agenda = crea_agenda();
140
141   fp = fopen(nombre_fichero , "r");
142   if ( fp != NULL ) // Si hay fichero
143   {
144   while (1) {
145     fgets(nombre, MAXCADENA, fp);
146     if ( feof ( fp )) break ; // Si se acab´o el fichero,  acabar la lectura .
147     quita_fin_de_linea(nombre);
148
149     fgets(direccion, MAXCADENA, fp);
150     quita_fin_de_linea(direccion);
151
152     fgets(telefono, MAXCADENA, fp);
153     quita_fin_de_linea(telefono);
154
155     agenda = anyadir_entrada(agenda, nombre, direccion, telefono);
156   }
157   fclose(fp);
```

```
158   } // if
159   return agenda;
160 }
161
162
163 /*   ************************************************************************
164  * Programa principal
165  ************************************************************************   */
166
167 int main(void)
168 {
169   TipoAgenda miagenda;
170   struct Nodo Agenda * encontrada;
171   int opcion;
172   char nombre[MAXCADENA+1];
173   char direccion[MAXCADENA+1];
174   char telefono[MAXCADENA+1];
175   char linea[MAXCADENA+1];
176
177   miagenda = lee_agenda("agenda.txt");
178   do {
179     printf("Menú:\n");
180     printf("1) Ver contenido completo de la agenda.\n");
181     printf("2) Dar de alta una persona.\n");
182     printf("3) Buscar teléfonos de una persona.\n");
183     printf("4)  Salir.\n");
184     printf("Opción: ");
185     gets(linea); sscanf(linea, "%d", &opcion);
186
187     switch(opcion) {
188
189       case Ver:
190         muestra_agenda(miagenda);
191         break;
192
193       case Alta:
194         printf("Nombre    : "); gets(nombre);
195         printf("Dirección: "); gets(direccion);
196         printf("Teléfono : "); gets(telefono);
197         miagenda = anyadir_entrada(miagenda, nombre, direccion, telefono);
198         break;
199
200       case Buscar:
201         printf("Nombre: "); gets(nombre);
202         encontrada = buscar_entrada_por_nombre(miagenda, nombre);
203         if ( encontrada == NULL )
204           printf("No hay nadie llamado %s en la agenda.\n", nombre);
205         else
206           muestra_entrada(encontrada);
207         break;
208     }
209   } while (opcion != Salir);
210
211   escribe_agenda(miagenda, "agenda.txt");
212   libera_agenda(miagenda);
213
214   return 0;
215 }
216
```