

**Programación II**

# **Practica 4:**

**Tipos de Datos Abstractos no lineales**

**David Piñuel Bosque**



2023

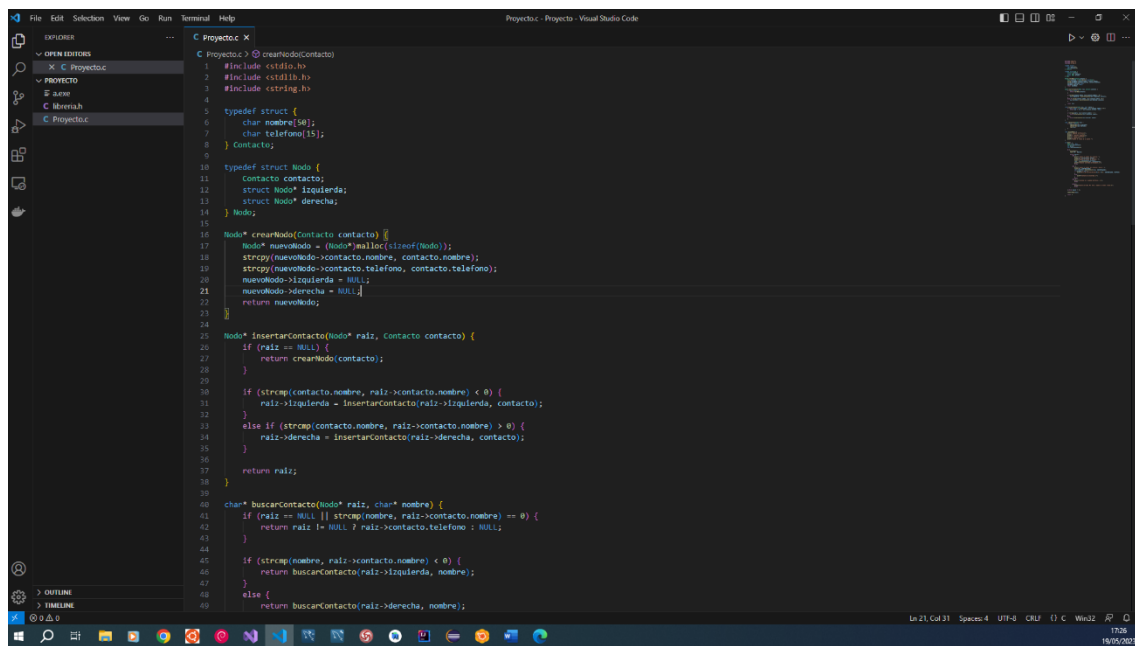
# Índice

|   |    |
|---|----|
| 1 Código Fuente del Programa de Tipos de Datos Abstractos no lineales (Ejercicio 1).....  | 2  |
| 2. Código Fuente del Programa de Tipos de Datos Abstractos no lineales (Ejercicio 2)..... | 4  |
| 3. Ejemplo de prueba del programa compilado y ejecutado (Ejercicio 1) .....               | 7  |
| 4. Ejemplo de prueba del programa compilado y ejecutado (Ejercicio 2) .....               | 9  |
| 5. Aclaraciones y comentarios. ....   | 10 |

# Índice Ilustración

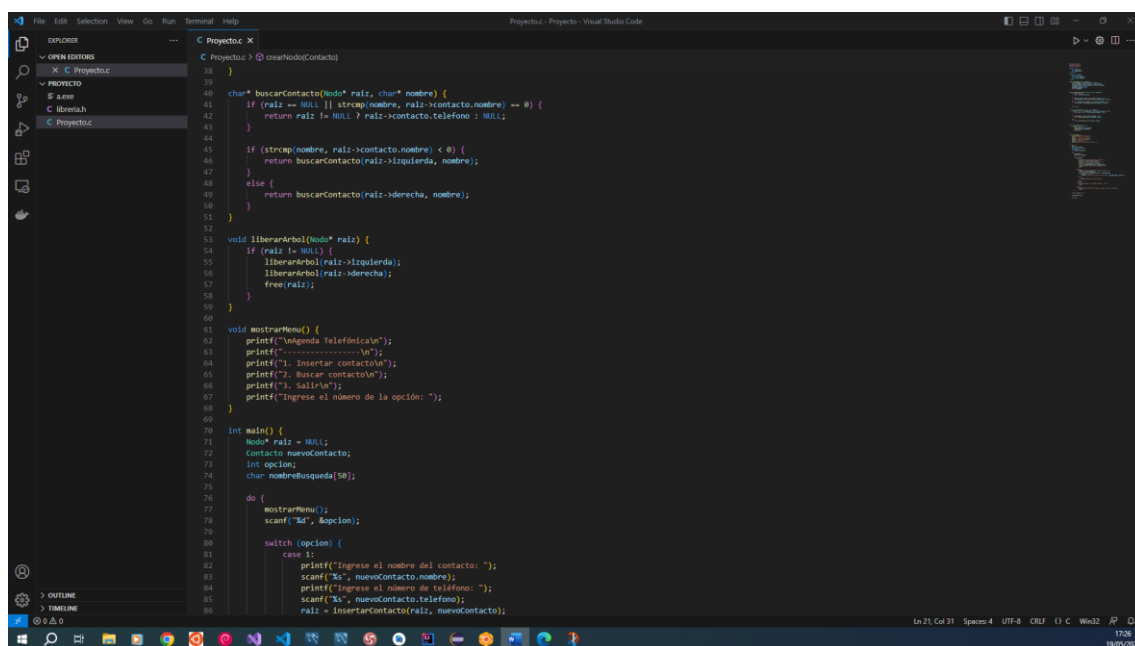
|  |           |
|--|-----------|
| <i>Ilustración 1: Código Fuente Ejercicio 1 Parte 1 .....</i>            | <i>3</i>  |
| <i>Ilustración 2: Código Fuente Ejercicio 1 Parte 2 .....</i>            | <i>3</i>  |
| <i>Ilustración 3: Código Fuente Ejercicio 1 Parte 3 .....</i>            | <i>4</i>  |
| <i>Ilustración 4: Código Fuente Ejercicio 2 Parte 1 .....</i>            | <i>4</i>  |
| <i>Ilustración 5: Código Fuente Ejercicio 2 Parte 2 .....</i>            | <i>5</i>  |
| <i>Ilustración 6: Código Fuente Ejercicio 2 Parte 3 .....</i>            | <i>5</i>  |
| <i>Ilustración 7: Código Fuente Ejercicio 2 Parte 4 .....</i>            | <i>6</i>  |
| <i>Ilustración 8: Código Fuente Ejercicio 2 Parte 5 .....</i>            | <i>6</i>  |
| <i>Ilustración 9: Código Fuente Ejercicio 2 Parte 6 .....</i>            | <i>7</i>  |
| <i>Ilustración 10: Ejemplo Prueba Programa Ejercicio 1 Parte 1 .....</i> | <i>7</i>  |
| <i>Ilustración 11: Ejemplo Prueba Programa Ejercicio 1 Parte 2 .....</i> | <i>8</i>  |
| <i>Ilustración 12: Ejemplo Prueba Programa Ejercicio 1 Parte 3 .....</i> | <i>8</i>  |
| <i>Ilustración 13: Ejemplo Prueba Programa Ejercicio 1 Parte 4 .....</i> | <i>9</i>  |
| <i>Ilustración 14: Ejemplo Prueba Programa Ejercicio 2 Parte 1 .....</i> | <i>9</i>  |
| <i>Ilustración 15: Ejemplo Prueba Programa Ejercicio 2 Parte 2 .....</i> | <i>10</i> |

1. Código Fuente del Programa de Tipos de Datos Abstractos no lineales (Ejercicio 1).



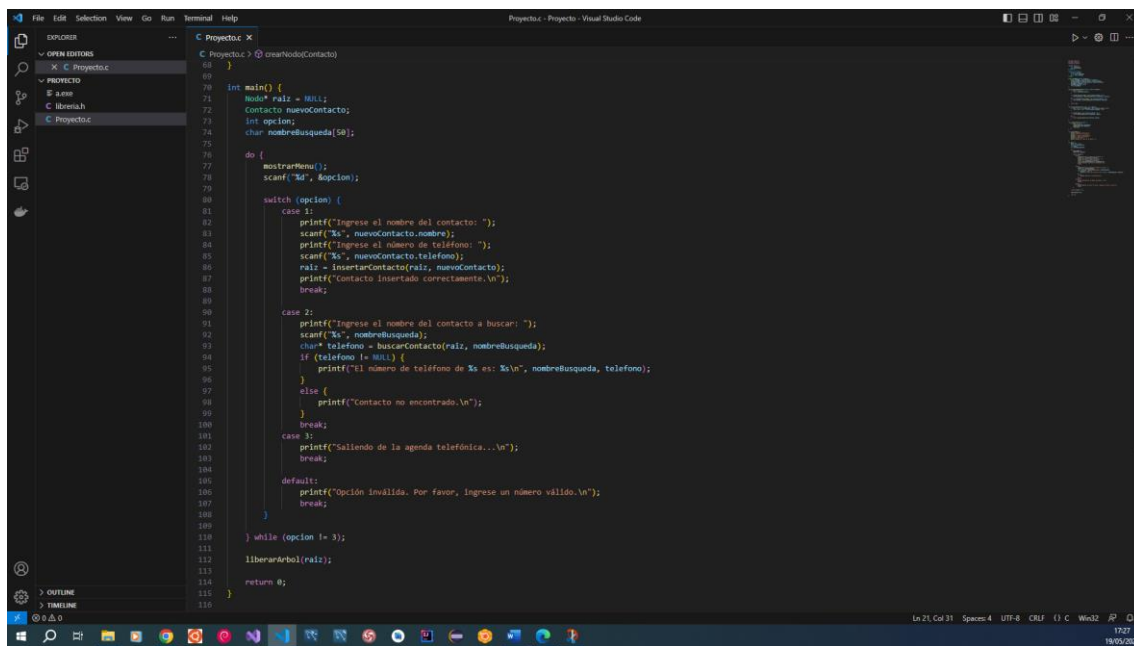
```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 typedef struct {
6     char nombre[50];
7     char telefono[15];
8 } Contacto;
9
10 typedef struct Nodo {
11     Contacto contacto;
12     struct Nodo* izquierda;
13     struct Nodo* derecha;
14 } Nodo;
15
16 Nodo* crearNodo(Contacto contacto) {
17     Nodo* nuevoNodo = (Nodo*) malloc(sizeof(Nodo));
18     strcpy(nuevoNodo->contacto.nombre, contacto.nombre);
19     strcpy(nuevoNodo->contacto.telefono, contacto.telefono);
20     nuevoNodo->izquierda = NULL;
21     nuevoNodo->derecha = NULL;
22     return nuevoNodo;
23 }
24
25 Nodo* insertarContacto(Nodo* raiz, Contacto contacto) {
26     if (raiz == NULL) {
27         return crearNodo(contacto);
28     }
29
30     if (strcmp(contacto.nombre, raiz->contacto.nombre) < 0) {
31         raiz->izquierda = insertarContacto(raiz->izquierda, contacto);
32     }
33     else if (strcmp(contacto.nombre, raiz->contacto.nombre) > 0) {
34         raiz->derecha = insertarContacto(raiz->derecha, contacto);
35     }
36
37     return raiz;
38 }
39
40 char* buscarContacto(Nodo* raiz, char* nombre) {
41     if (raiz == NULL || strcmp(nombre, raiz->contacto.nombre) == 0) {
42         return raiz != NULL ? raiz->contacto.telefono : NULL;
43     }
44
45     if (strcmp(nombre, raiz->contacto.nombre) < 0) {
46         return buscarContacto(raiz->izquierda, nombre);
47     }
48     else {
49         return buscarContacto(raiz->derecha, nombre);
50     }
51 }
```

Ilustración 1: Código Fuente Ejercicio 1 Parte 1



```
52
53 void liberarArbol(Nodo* raiz) {
54     if (raiz != NULL) {
55         liberarArbol(raiz->izquierda);
56         liberarArbol(raiz->derecha);
57         free(raiz);
58     }
59 }
60
61 void mostrarMenu() {
62     printf("\nMenu telefonico\n");
63     printf("1. Insertar contacto\n");
64     printf("2. Buscar contacto\n");
65     printf("3. Salir\n");
66     printf("Ingrese el número de la opción: ");
67 }
68
69 int main() {
70     Nodo* raiz = NULL;
71     Contacto nuevoContacto;
72     int opcion;
73     char nombreBusqueda[50];
74
75     do {
76         mostrarMenu();
77         scanf("%d", &opcion);
78
79         switch (opcion) {
80             case 1:
81                 printf("Ingrese el nombre del contacto: ");
82                 scanf("%s", nuevoContacto.nombre);
83                 printf("Ingrese el número de teléfono: ");
84                 scanf("%s", nuevoContacto.telefono);
85                 raiz = insertarContacto(raiz, nuevoContacto);
86             break;
87             case 2:
88                 printf("Ingrese el nombre del contacto a buscar: ");
89                 scanf("%s", nombreBusqueda);
90                 char* telefono = buscarContacto(raiz, nombreBusqueda);
91                 if (telefono != NULL) {
92                     printf("El teléfono de %s es %s\n", nombreBusqueda, telefono);
93                 } else {
94                     printf("No se encontró el contacto.\n");
95                 }
96             break;
97             case 3:
98                 return 0;
99             default:
100                 printf("Opción no válida.\n");
101         }
102     } while (opcion != 3);
103
104     liberarArbol(raiz);
105     return 0;
106 }
```

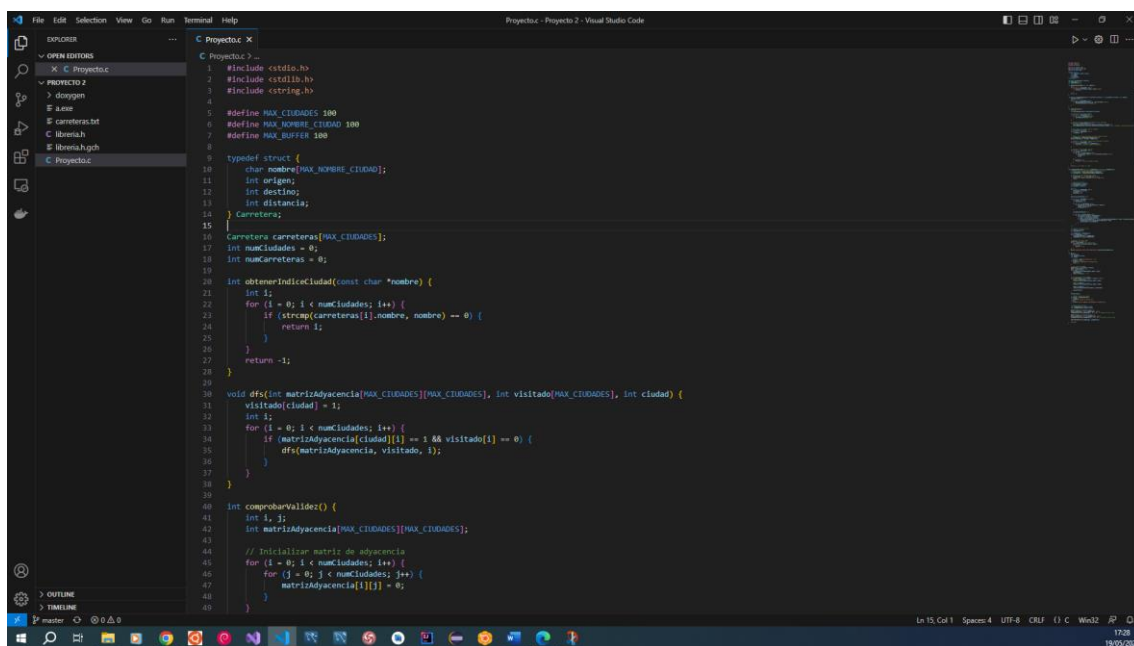
Ilustración 2: Código Fuente Ejercicio 1 Parte 2



```
1 // Proyecto 1 - Ejercicio 1 Parte 3
2 // Autor: [Nombre]
3 // Fecha: [Fecha]
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 #define MAX_CONTACTOS 100
10 #define MAX_NOMBRE 50
11 #define MAX_TELEFONO 20
12
13 typedef struct Contacto {
14     char nombre[MAX_NOMBRE];
15     char telefono[MAX_TELEFONO];
16     struct Contacto *siguiente;
17 } Contacto;
18
19 Contacto *raiz = NULL;
20
21 void mostrarMenu() {
22     printf("Menu de opciones:\n");
23     printf("1. Ingresar contacto\n");
24     printf("2. Buscar contacto\n");
25     printf("3. Eliminar contacto\n");
26     printf("4. Salir\n");
27 }
28
29 void ingresarContacto() {
30     Contacto nuevoContacto;
31     char nombreBusqueda[50];
32
33     do {
34         mostrarMenu();
35         scanf("%d", &opcion);
36
37         switch (opcion) {
38             case 1:
39             {
40                 printf("Ingrese el nombre del contacto: ");
41                 scanf("%s", nuevoContacto.nombre);
42                 printf("Ingrese el número de teléfono: ");
43                 scanf("%s", nuevoContacto.telefono);
44                 raiz = insertarContacto(raiz, nuevoContacto);
45                 printf("Contacto insertado correctamente.\n");
46                 break;
47             }
48             case 2:
49             {
50                 printf("Ingrese el nombre del contacto a buscar: ");
51                 scanf("%s", nombreBusqueda);
52                 char *telefono = buscarContacto(raiz, nombreBusqueda);
53                 if (telefono != NULL) {
54                     printf("El número de teléfono de %s es: %s\n", nombreBusqueda, telefono);
55                 }
56                 else {
57                     printf("Contacto no encontrado.\n");
58                 }
59                 break;
60             }
61             case 3:
62             {
63                 printf("Saliedo de la agenda telefónica...\n");
64                 break;
65             }
66             default:
67             {
68                 printf("Opción inválida. Por favor, ingrese un número válido.\n");
69                 break;
70             }
71         }
72     } while (opcion != 3);
73 }
74
75 void liberarArbol(Contacto *raiz) {
76     if (raiz == NULL) return;
77     liberarArbol(raiz->siguiente);
78     free(raiz);
79 }
80
81 int main() {
82     ingresarContacto();
83     liberarArbol(raiz);
84     return 0;
85 }
```

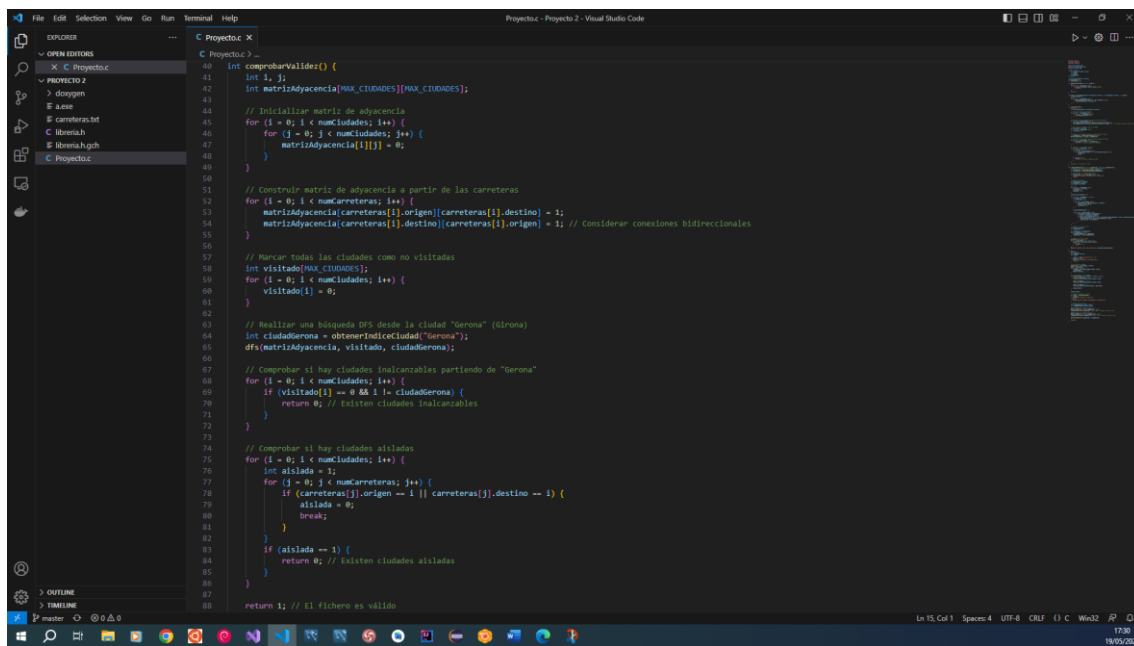
Ilustración 3: Código Fuente Ejercicio 1 Parte 3

## 2. Código Fuente del Programa de Tipos de Datos Abstractos no lineales (Ejercicio 2)



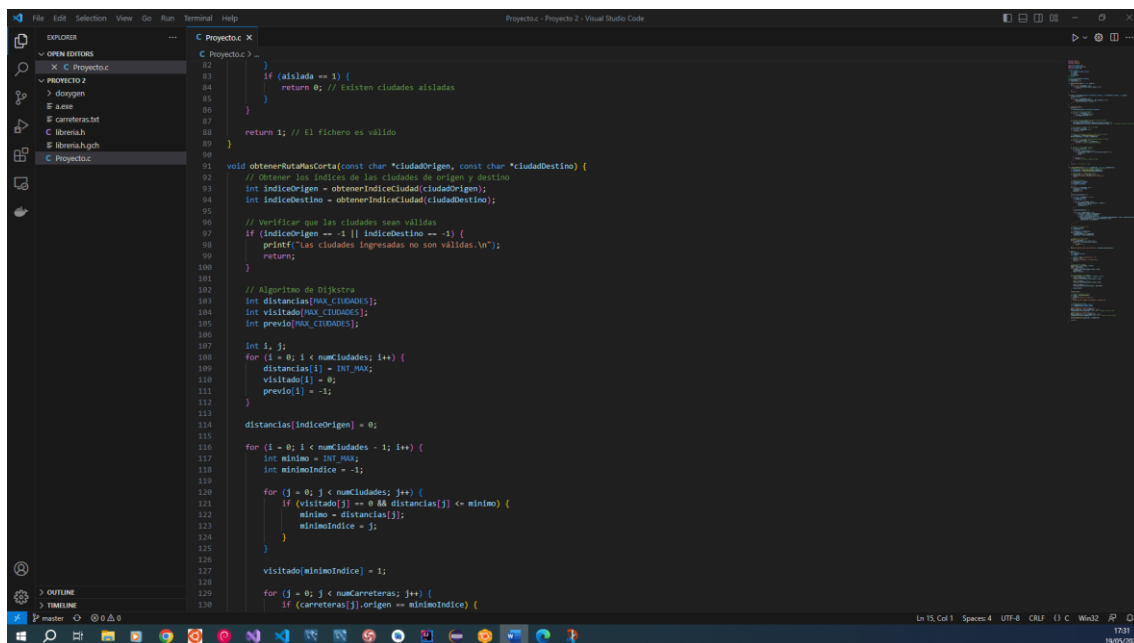
```
1 // Proyecto 2 - Ejercicio 2 Parte 1
2 // Autor: [Nombre]
3 // Fecha: [Fecha]
4
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 #define MAX_CUADROS 100
10 #define MAX_NOMBRES_CUADRO 50
11 #define MAX_VERTICES 100
12
13 typedef struct Cuadro {
14     char nombre[MAX_NOMBRES_CUADRO];
15     int origen;
16     int destino;
17     int distancia;
18 } Cuadro;
19
20 Cuadro *caracteres[MAX_CUADROS];
21 int numCuadros = 0;
22
23 void obtenerIndiceCuadro(const char *nombre) {
24     int i;
25     for (i = 0; i < numCuadros; i++) {
26         if (strcmp(caracteres[i].nombre, nombre) == 0) {
27             return i;
28         }
29     }
30     return -1;
31 }
32
33 void dfs(int matrizAdyacencia[MAX_CUADROS][MAX_CUADROS], int visitado[MAX_CUADROS], int ciudad) {
34     visitado[ciudad] = 1;
35     int i;
36     for (i = 0; i < numCuadros; i++) {
37         if (matrizAdyacencia[ciudad][i] == 1 && visitado[i] == 0) {
38             dfs(matrizAdyacencia, visitado, i);
39         }
40     }
41 }
42
43 int comprobarValidar() {
44     int i, j;
45     int matrizAdyacencia[MAX_CUADROS][MAX_CUADROS];
46
47     // Inicializar matriz de adyacencia
48     for (i = 0; i < numCuadros; i++) {
49         for (j = 0; j < numCuadros; j++) {
50             matrizAdyacencia[i][j] = 0;
51         }
52     }
53 }
```

Ilustración 4: Código Fuente Ejercicio 2 Parte 1



```
40 int comprobarValidar() {
41     int i, j;
42     int matrizAdyacencia[MAX_CIUDADES][MAX_CIUDADES];
43
44     // Inicializar matriz de adyacencia
45     for (i = 0; i < numCiudades; i++) {
46         for (j = 0; j < numCiudades; j++) {
47             matrizAdyacencia[i][j] = 0;
48         }
49     }
50
51     // Construir matriz de adyacencia a partir de las carreteras
52     for (i = 0; i < numCarreteras; i++) {
53         matrizAdyacencia[carreteras[i].origen][carreteras[i].destino] = 1;
54         matrizAdyacencia[carreteras[i].destino][carreteras[i].origen] = 1; // Considerar conexiones bidireccionales
55     }
56
57     // Marcar todas las ciudades como no visitadas
58     int visitado[MAX_CIUDADES];
59     for (i = 0; i < numCiudades; i++) {
60         visitado[i] = 0;
61     }
62
63     // Realizar una búsqueda DFS desde la ciudad "Gerona" (Girona)
64     int ciudadGerona = obtenerIndiceCiudad("Gerona");
65     dfs(matrizAdyacencia, visitado, ciudadGerona);
66
67     // Comprobar si hay ciudades inalcanzables partiendo de "Gerona"
68     for (i = 0; i < numCiudades; i++) {
69         if (visitado[i] == 0 && i != ciudadGerona) {
70             return 0; // Existen ciudades inalcanzables
71         }
72     }
73
74     // Comprobar si hay ciudades aisladas
75     for (i = 0; i < numCiudades; i++) {
76         int aislada = 1;
77         for (j = 0; j < numCarreteras; j++) {
78             if (carreteras[j].origen == i || carreteras[j].destino == i) {
79                 aislada = 0;
80                 break;
81             }
82         }
83         if (aislada == 1) {
84             return 0; // Existen ciudades aisladas
85         }
86     }
87
88     return 1; // El fichero es válido
89 }
```

Ilustración 5: Código Fuente Ejercicio 2 Parte 2



```
90
91 void obtenerRutaMasCorta(const char *ciudadOrigen, const char *ciudadDestino) {
92     // Obtener los índices de las ciudades de origen y destino
93     int indiceOrigen = obtenerIndiceCiudad(ciudadOrigen);
94     int indiceDestino = obtenerIndiceCiudad(ciudadDestino);
95
96     // Verificar que las ciudades sean válidas
97     if (indiceOrigen == -1 || indiceDestino == -1) {
98         printf("Las ciudades ingresadas no son válidas.\n");
99         return;
100     }
101
102     // Algoritmo de Dijkstra
103     int distancias[MAX_CIUDADES];
104     int visitado[MAX_CIUDADES];
105     int previo[MAX_CIUDADES];
106
107     int i, j;
108     for (i = 0; i < numCiudades; i++) {
109         distancias[i] = INT_MAX;
110         visitado[i] = 0;
111         previo[i] = -1;
112     }
113
114     distancias[indiceOrigen] = 0;
115
116     for (i = 0; i < numCiudades - 1; i++) {
117         int minimo = INT_MAX;
118         int minimoIndice = -1;
119
120         for (j = 0; j < numCiudades; j++) {
121             if (visitado[j] == 0 && distancias[j] <= minimo) {
122                 minimo = distancias[j];
123                 minimoIndice = j;
124             }
125         }
126
127         visitado[minimoIndice] = 1;
128
129         for (j = 0; j < numCarreteras; j++) {
130             if (carreteras[j].origen == minimoIndice) {
```

Ilustración 6: Código Fuente Ejercicio 2 Parte 3

```

114 distancias[indiceOrigen] = 0;
115
116 for (i = 0; i < numCiudades - 1; i++) {
117     int minimo = INT_MAX;
118     int minimoIndice = -1;
119
120     for (j = 0; j < numCiudades; j++) {
121         if (visitado[j] == 0 && distancias[j] <= minimo) {
122             minimo = distancias[j];
123             minimoIndice = j;
124         }
125     }
126     visitado[minimoIndice] = 1;
127
128     for (j = 0; j < numCarreteras; j++) {
129         if (carreteras[j].origen == minimoIndice) {
130             int vecino = carreteras[j].destino;
131             int peso = carreteras[j].distancia;
132             if (distancias[minimoIndice] != INT_MAX && distancias[minimoIndice] + peso < distancias[vecino]) {
133                 distancias[vecino] = distancias[minimoIndice] + peso;
134                 previo[vecino] = minimoIndice;
135             }
136         }
137     }
138 }
139
140 // Construir la ruta más corta
141 int ruta[MAX_CIUDADES];
142 int longitudRuta = 0;
143
144 int ciudadActual = indiceDestino;
145 while (ciudadActual != -1) {
146     ruta[ciudadActual] = ciudadActual;
147     ciudadActual = previo[ciudadActual];
148 }
149
150 // Imprimir la ruta más corta
151 printf("Ruta más corta: ");
152 for (i = longitudRuta - 1; i >= 0; i--) {
153     printf("%s", carreteras[ruta[i]].nombre);
154     if (i != 0) {
155         printf(" -> ");
156     }
157 }
158
159 printf("\nDistancia total: %d kilómetros\n", distancias[indiceDestino]);
160
161 int main() {

```

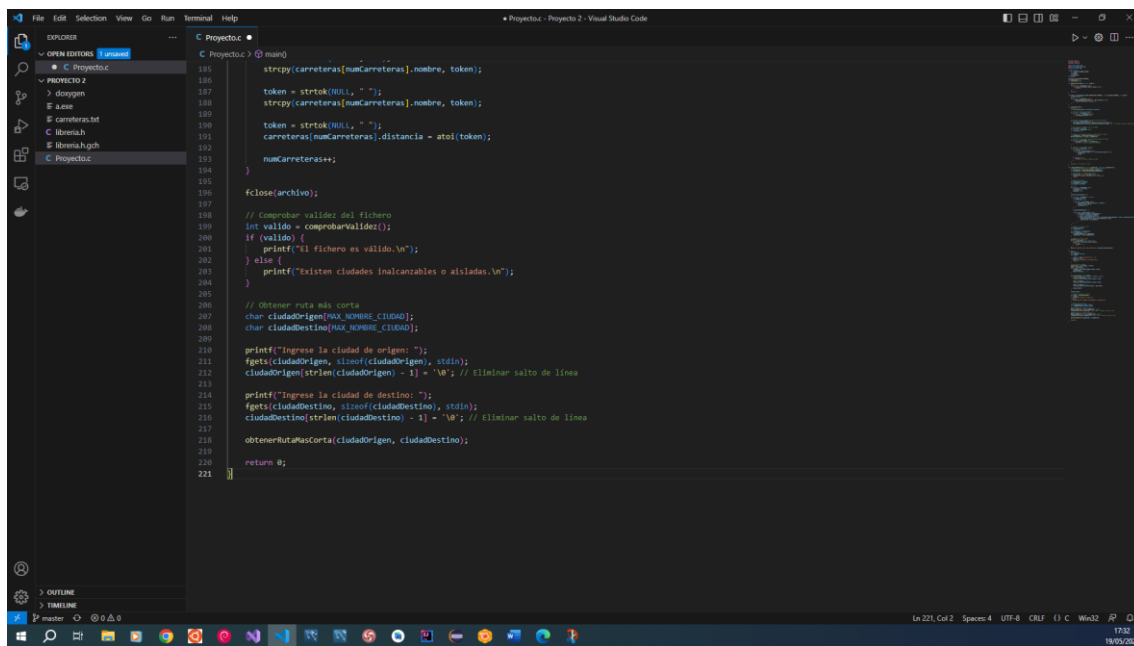
**Ilustración 7: Código Fuente Ejercicio 2 Parte 4**

```

162 int main() {
163     FILE *archivo;
164     char buffer[MAX_BUFFER];
165     char *token;
166
167     archivo = fopen("carreteras.txt", "r");
168     if (archivo == NULL) {
169         printf("No se pudo abrir el archivo.\n");
170         return 1;
171     }
172
173     // Leer lista de ciudades
174     fgets(buffer, sizeof(buffer), archivo);
175     token = strtok(buffer, ",");
176     while (token != NULL) {
177         strcpy(carreteras[numCiudades].nombre, token);
178         numCiudades++;
179         token = strtok(NULL, ",");
180     }
181
182     // Leer distancias entre ciudades
183     while (fgets(buffer, sizeof(buffer), archivo) != NULL) {
184         token = strtok(buffer, " ");
185         strcpy(carreteras[numCarreteras].nombre, token);
186
187         token = strtok(NULL, " ");
188         strcpy(carreteras[numCarreteras].destino, token);
189
190         token = strtok(NULL, " ");
191         carreteras[numCarreteras].distancia = atoi(token);
192
193         numCarreteras++;
194     }
195
196     fclose(archivo);
197
198     // Comprobar validez del fichero
199     int valido = comprobarValidez();
200     if (valido) {
201         printf("El fichero es válido.\n");
202     } else {
203         printf("Existen ciudades inalcanzables o aisladas.\n");
204     }
205
206     // Obtener ruta más corta
207     char ciudadOrigen[MAX_NOMBRE_CIUDAD];
208     char ciudadDestino[MAX_NOMBRE_CIUDAD];
209

```

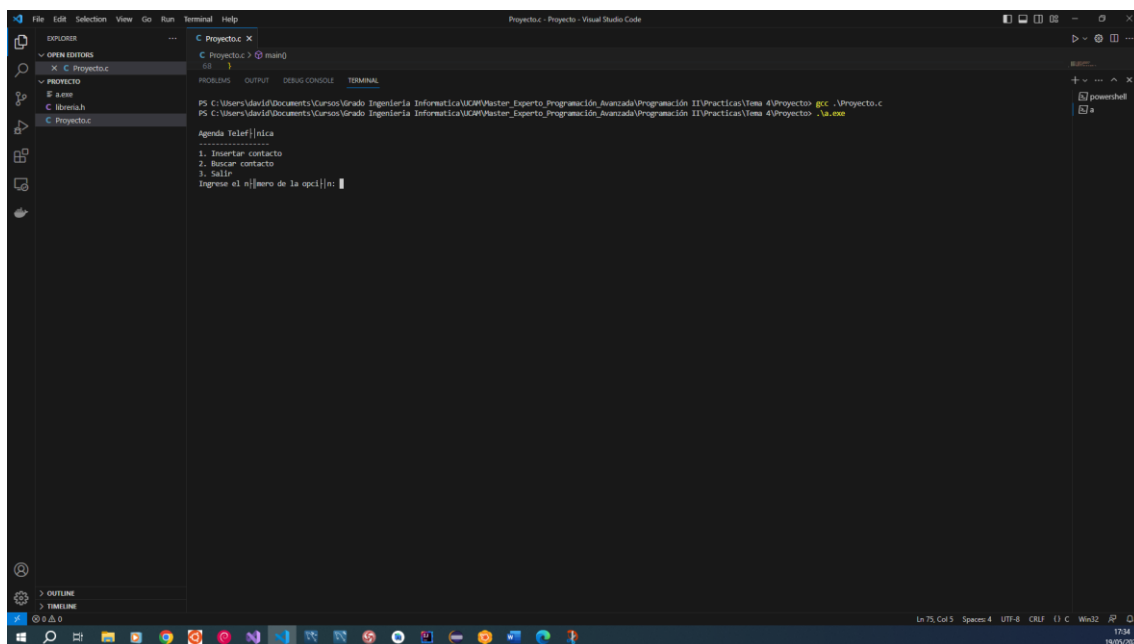
**Ilustración 8: Código Fuente Ejercicio 2 Parte 5**



```
185 strcpy(carreteras[numCarreteras].nombre, token);
186
187 token = strtok(NULL, " ");
188 strcpy(carreteras[numCarreteras].nombre, token);
189
190 token = strtok(NULL, " ");
191 carreteras[numCarreteras].distancia = atoi(token);
192
193 numCarreteras++;
194
195 }
196
197 fclose(archivo);
198
199 // Comprobar validez del fichero
200 int valido = comprobarValidez();
201 if (valido) {
202     printf("El fichero es válido.\n");
203 } else {
204     printf("Existen ciudades inalcanzables o aisladas.\n");
205 }
206
207 // Obtener ruta más corta
208 char ciudadOrigen[MAX_NOMBRES_CIUDADES];
209 char ciudadDestino[MAX_NOMBRES_CIUDADES];
210
211 printf("Ingrese la ciudad de origen: ");
212 fgets(ciudadOrigen, sizeof(ciudadOrigen), stdin);
213 ciudadOrigen[strlen(ciudadOrigen) - 1] = '\0'; // Eliminar salto de línea
214
215 printf("Ingrese la ciudad de destino: ");
216 fgets(ciudadDestino, sizeof(ciudadDestino), stdin);
217 ciudadDestino[strlen(ciudadDestino) - 1] = '\0'; // Eliminar salto de línea
218
219 obtenerRutaMasCorta(ciudadOrigen, ciudadDestino);
220
221 return 0;
222 }
```

Ilustración 9: Código Fuente Ejercicio 2 Parte 6

### 3. Ejemplo de prueba del programa compilado y ejecutado (Ejercicio 1)



```
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCAM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto 1> gcc .\Proyecto_1.c
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCAM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto 1> .\a.exe
Agenda Telefónica
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 1
Ingrese el número de la opción: 1
```

Ilustración 10: Ejemplo Prueba Programa Ejercicio 1 Parte 1

```
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto> gcc -o Proyecto.c
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto> .\a.exe

Agenda Telefonica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 1
Ingrese el nombre del contacto: David
Ingrese el número de teléfono: 661275932
Contacto insertado correctamente.

Agenda Telefonica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 
```

**Ilustración 11: Ejemplo Prueba Programa Ejercicio 1 Parte 2**

```
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto> gcc -o Proyecto.c
PS C:\Users\David\Documents\Cursos\Verano Ingeniería Informática\UCM\Master_Esperto_Programación_Avanzada\Programación II\Prácticas\Tema 4\Proyecto> .\a.exe

Agenda Telefonica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 1
Ingrese el nombre del contacto: David
Ingrese el número de teléfono: 661275932
Contacto insertado correctamente.

Agenda Telefonica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 2
Ingrese el nombre del contacto a buscar: David
El número de teléfono de David es: 661275932

Agenda Telefonica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 
```

**Ilustración 12: Ejemplo Prueba Programa Ejercicio 1 Parte 3**



```
File Edit Selection View Go Run Terminal Help
Proyecto - Proyecto - Visual Studio Code

C Proyecto.c X
C Proyecto.c > main()
68 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\David\Documents\Cursos\Verado Ingenieria Informatica\UCRM\Master_Esperto_Programación_Avanzada\Programación II\Practicas\Tema 4\Proyecto> gcc -oProyecto.c
PS C:\Users\David\Documents\Cursos\Verado Ingenieria Informatica\UCRM\Master_Esperto_Programación_Avanzada\Programación II\Practicas\Tema 4\Proyecto> .\a.exe

Agenda Telefónica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 1
Ingrese el nombre del contacto: David
Ingrese el número de tel/fono: 661275932
Contacto insertado correctamente.

Agenda Telefónica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 2
Ingrese el nombre del contacto a buscar: David
El número de tel/fono de David es: 661275932

Agenda Telefónica
-----
1. Insertar contacto
2. Buscar contacto
3. Salir
Ingrese el número de la opción: 3
Saliedo de la agenda telefónica...
PS C:\Users\David\Documents\Cursos\Verado Ingenieria Informatica\UCRM\Master_Esperto_Programación_Avanzada\Programación II\Practicas\Tema 4\Proyecto>
```

**Ilustración 13: Ejemplo Prueba Programa Ejercicio 1 Parte 4**

#### 4. Ejemplo de prueba del programa compilado y ejecutado (Ejercicio 2)

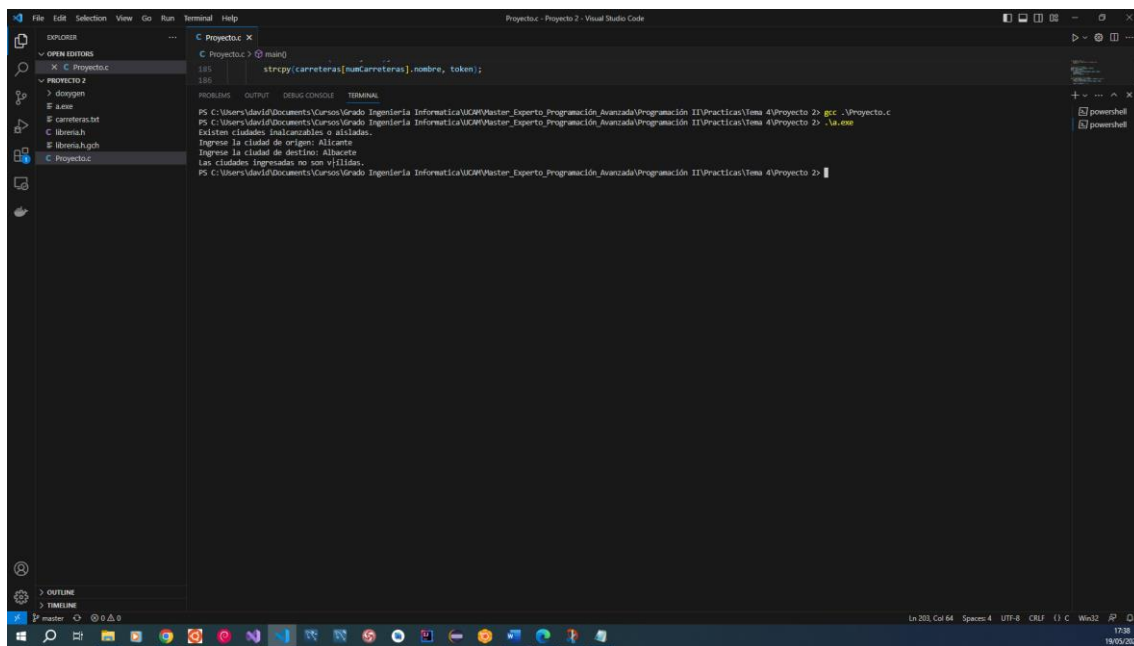
```
File Edit Selection View Go Run Terminal Help
Proyecto - Proyecto 2 - Visual Studio Code

C Proyecto.c X
C Proyecto.c > main()
100 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\David\Documents\Cursos\Verado Ingenieria Informatica\UCRM\Master_Esperto_Programación_Avanzada\Programación II\Practicas\Tema 4\Proyecto 2> gcc -oProyecto.c
PS C:\Users\David\Documents\Cursos\Verado Ingenieria Informatica\UCRM\Master_Esperto_Programación_Avanzada\Programación II\Practicas\Tema 4\Proyecto 2> .\a.exe
Existes ciudades: insalvables o aisladas.
Ingrese la ciudad de origen:
```

**Ilustración 14: Ejemplo Prueba Programa Ejercicio 2 Parte 1**



**Ilustración 15: Ejemplo Prueba Programa Ejercicio 2 Parte 2**

## 5. Aclaraciones y comentarios.

En el ejercicio 1 he realizado la estructura del programa en forma de árbol con un raíz padre. Y en el ejercicio 2 he realizado el programa calculando primero si hay ciudades inalcanzables o no existen en la lista del archivo carretera.txt y ya calcula la distancia entre las dos ciudades.