



CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
OPTIMIZACIÓN INTELIGENTE
5° "A"

PRÁCTICA 6: CAPTURA DE LOS DATOS DE UN ARCHIVO TSPLIB

Profesor: Aurora Torres Soto

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags., **3** de noviembre de 2020

Práctica 6: Captura de los datos de un archivo TSPLIB

Objetivo:

Con el desarrollo de esta práctica, se implementará un programa que lea el contenido del archivo `bays29.tsp.txt` de la biblioteca TSPLIB y guardará la información de las aristas del problema en una matriz.

Introducción:

Existen dos tipos de archivos: archivos de texto y archivos binarios. Un archivo de texto es una secuencia de caracteres organizada en líneas terminadas por un carácter de nueva línea. Los archivos binarios, por otro lado, no son interpretables por cualquier editor.

Los archivos de texto pueden ser planos o de texto enriquecido. Los archivos de texto plano son aquellos en los que todas las letras tienen el mismo formato y no hay palabras subrayadas, en negrita o letras de distinto tamaño, color o ancho.

Un archivo binario es una secuencia de bytes que tienen una correspondencia uno a uno con un dispositivo externo. Además, el número de bytes escritos (leídos) será el mismo que los encontrados en el dispositivo externo. Ejemplos de estos archivos son fotografías, imágenes, texto con formatos, archivos ejecutables (aplicaciones), etc.

Para manejar archivos en el lenguaje C es necesario determinar el tipo de archivo que requerimos y posteriormente realizar una serie de pasos que nos permitan su creación, direccionamiento, apertura y cerrado.

Para el manejo de los archivos, C posee una serie de funciones que se encuentran en la librería `stdio.h`. La tabla siguiente muestra las principales funciones relacionadas con el manejo de archivos de esta biblioteca.

Nombre	Función
<code>fopen()</code>	Abre un archivo.
<code>fclose()</code>	Cierra un archivo.
<code>fgets()</code>	Lee una cadena de un archivo.
<code>fputs()</code>	Escribe una cadena en un archivo.
<code>fseek()</code>	Busca un byte específico de un archivo.
<code>fprintf()</code>	Escribe una salida con formato en el archivo.
<code>fscanf()</code>	Lee una entrada con formato desde el archivo.
<code>feof()</code>	Devuelve cierto si se llega al final del archivo.
<code>ferror()</code>	Devuelve cierto si se produce un error.
<code>rewind()</code>	Coloca el localizador de posición del archivo al principio del mismo.
<code>remove()</code>	Borra un archivo.
<code>fflush()</code>	Vacia un archivo.

En primer lugar, es necesario que realicemos la declaración de una variable de tipo apuntador a un archivo, pues mediante esta variable se podrán realizar las

operaciones de lectura y escritura. La declaración de este tipo de variable tiene la siguiente sintaxis:

```
FILE *Nombre_de_variable;
```

A continuación, para realizar la apertura de un archivo, se emplea la función `fopen()` que asocia la variable definida con un archivo físico.

```
Nombre_de_variable = fopen(const char nombre_archivo, const char modo);
```

Donde `nombre_archivo` es una cadena de caracteres que representan un nombre valido del archivo y puede incluir una especificación del directorio; y `modo` determina el modo como se abrirá el archivo, es decir si se abre para escritura, para lectura, para inserción, etc.

La siguiente tabla muestra los valores permitidos para `modo`.

Modo	Significado
r	Abre un archivo de texto para lectura.
w	Crea un archivo de texto para escritura.
a	Abre un archivo de texto para añadir.
rb	Abre un archivo binario para lectura.
wb	Crea un archivo binario para escritura.
ab	Abre un archivo binario para añadir.
r+	Abre un archivo de texto para lectura / escritura.
w+	Crea un archivo de texto para lectura / escritura.
a+	Añade o crea un archivo de texto para lectura / escritura.
r+b	Abre un archivo binario para lectura / escritura.
w+b	Crea un archivo binario para lectura / escritura.
a+b	Añade o crea un archivo binario para lectura / escritura.

La función `fopen()` devuelve un apuntador que debe ser cuidadosamente almacenado, pues éste será el medio para acceder y/o manipular el archivo. Si se produce un error al momento de intentar abrir un archivo `fopen()` devuelve un apuntador nulo.

Si se usa `fopen()` para abrir un archivo para escritura, esto producirá que cualquier archivo preexistente sea borrado y reemplazado por el nuevo archivo. Si no existiera un archivo con ese nombre, entonces se creará. Si se quiere añadir información al final del archivo, entonces se debe usar el modo “a”; sin embargo, al usar “a” con un archivo que no existe, la función producirá un error. La apertura de un archivo para las operaciones de lectura requiere que exista el archivo.

Una vez abierto un archivo, podremos utilizar las siguientes funciones para introducir u obtener datos de un archivo:

```
fprintf() y fscanf().
```

Estas funciones se comportan exactamente como `printf()` y `scanf()`, excepto que operan sobre archivo. Sus prototipos son:

```
int fprintf(FILE *F, const char *cadena_de_control, .....);
```

```
int fscanf(FILE *F, const char *cadena_de_control, .....);
```

Donde `F` es un puntero al archivo devuelto por una llamada a `fopen()`; `fprintf()` y `fscanf()` dirigen sus operaciones de entrada y salida al archivo al que apunta `F`.

Las funciones `fgets()` y `fputs()` pueden leer y escribir cadenas hacia o desde los archivos, pues su comportamiento es muy similar al de las funciones `gets()` y `puts()`; excepto porque se debe agregar como parámetro el apuntador al archivo.

```
char *fputs(char *str, FILE *F);
```

```
char *fgets(char *str, int long, FILE *F);
```

Si se produce un EOF (*End of File*) la función `gets` retorna un `NULL`.

Para verificar cuando un archivo ha alcanzado el EOF (*Fin del archivo*), C posee la función `feof()`, que devuelve verdadero cuando se ha alcanzado esta situación o 0 en cualquier otro caso.

La función `ferror()` determina si se ha producido un error en una operación sobre un archivo. Su prototipo es:

```
int ferror(FILE *F);
```

Donde `F` es un puntero a un archivo válido. Esta función devuelve verdadero si se ha producido un error durante la última operación sobre el archivo o falso en caso contrario.

La función `remove()` borra el archivo especificado. Su prototipo es el siguiente:

```
int remove(char *nombre_archivo);
```

Devuelve cero si tiene éxito. Si no un valor distinto de cero.

Una vez que se ha trabajado con los archivos, es muy importante cerrarlos, debido a que algunos sistemas de archivos fallan o bloquean el archivo si esta operación no es ejecutada; también esto ayuda a limpiar los `BUFFER` en lenguajes de más alto nivel.

Para realizar el cierre de un archivo, se usa la función `fclose()`;

Esta función cierra una secuencia que fue abierta mediante una llamada a `fopen()`. Escribiendo toda la información que todavía se encuentre en el buffer en el disco y realizando un cierre formal del archivo a nivel del sistema operativo.

El prototipo de esta función es:

```
int fclose(FILE *F);
```

Donde F es el puntero al archivo devuelto por la llamada a `fopen()`. Si se devuelve un valor cero significa que la operación de cierre ha tenido éxito. Generalmente, esta función solo falla cuando un disco se ha retirado antes de tiempo o cuando no queda espacio libre en el mismo.

Pregunta de Investigación:

¿Qué procedimientos se necesitan para abrir un archivo local y extraer su información para tratarla en lenguaje C?

Predicción:

Me parece que los procedimientos que se expresan en la introducción son suficientes para realizar el trabajo.

Materiales:

Una computadora con compilador de C.	Dos hojas de papel.
Una calculadora.	Lápiz o plumas.

Método (Variables):

Dependiente: El programa con la información en él.

Independiente: El algoritmo que uno como alumno se focalizará en elaborar.

Controlada: El procedimiento de lectura de archivos.

Seguridad:

Realmente no se trabajó en campo, por lo que no se corren riesgos al elaborar el experimento.

Procedimiento:

- 1.- Se localizó la biblioteca de casos de prueba TSPLIB <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsplib.html>.
- 2.- Se descargó el archivo `bays29.tsp.txt` de manera local.
- 3.- Se escribió un programa que leyera el contenido de la sección `EDGE_WEIGHT_SECTION` y se almacenara en un arreglo de 29×29 .
- 4.- Se mostró el contenido del arreglo.

Obtención y Procesamiento de Datos:

Se leyó el archivo en cuestión, donde en un principio, se buscaba verificar que se leyera correctamente el archivo en cuestión. El resultado fue el siguiente:

```

C:\Users\holl\Desktop>dir Archivio\Archivos Extra\Archivos Académicos\Universidad\Ondine - Universidad Autónoma de Aguascalientes\Archivos Académicos\Universidad\T.O. Ondine\Practica 8\Practica 8\Integraciona.exe
===== LECTURA DE ARCHIVOS =====
NAME: days29
TYPE: TSP
COMMENT: 29 cities in Bavaria, street distances (Groetschel, Juenger, Reinelt)
DIMENSION: 29
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
DISPLAY_DATA_TYPE: TWO_DISPLAY
EDGE_WEIGHT_SECTION
0 107 241 190 124 88 316 76 152 157 283 133 113 297 228 129 348 276 188 150 65 341 184 67 221 169 108 45 167
107 0 148 137 88 127 336 183 134 95 254 180 181 234 175 176 265 199 182 67 42 278 271 146 251 105 191 139 79
241 148 0 374 171 259 589 317 217 232 491 312 286 391 412 349 422 356 355 284 182 435 417 192 424 116 337 273 77
190 137 374 0 282 234 222 192 248 42 117 387 79 107 38 121 152 86 68 78 137 151 239 135 137 242 165 228 205
124 88 171 282 0 61 392 282 46 160 319 112 163 322 240 232 314 287 238 155 65 366 380 175 307 57 228 121 97
88 127 259 234 61 0 386 141 72 167 351 55 157 331 272 226 362 296 232 164 85 375 249 147 381 118 188 60 185
190 336 589 222 382 386 0 233 438 254 282 439 235 254 210 187 313 266 154 282 321 298 168 249 95 437 198 314 435
76 183 137 192 282 141 233 0 213 188 272 193 131 382 233 98 344 289 177 216 141 346 188 57 190 245 43 81 243
152 134 217 248 46 72 438 213 0 206 365 89 289 368 286 278 360 333 284 281 111 412 321 221 353 72 266 132 111
157 95 232 42 188 167 254 188 286 0 159 220 57 149 88 122 193 127 180 28 95 192 241 231 169 280 161 189 163
183 254 491 117 319 351 282 272 365 159 0 484 176 106 79 161 165 141 95 187 254 103 279 215 117 359 216 388 322
133 188 312 287 112 55 439 193 89 220 484 0 210 384 325 279 415 349 285 217 138 428 310 280 354 169 241 112 238
113 181 288 79 163 157 235 131 209 57 276 218 0 186 117 75 231 165 81 85 92 238 184 74 158 288 104 158 286
197 234 391 187 322 331 254 382 368 149 186 384 186 0 69 191 95 35 125 167 255 44 389 245 169 327 246 335 288
228 175 412 38 240 272 218 233 286 80 79 325 117 69 0 122 122 56 56 188 175 113 240 176 125 280 177 266 243
129 176 349 121 232 226 187 98 278 132 161 279 75 191 122 0 244 178 66 160 161 235 118 61 92 277 95 155 275
148 265 422 152 314 362 313 344 368 193 165 415 231 59 122 244 0 66 178 198 286 77 362 287 228 358 298 389 319
276 199 356 86 287 296 266 289 333 127 141 349 165 35 56 178 66 0 112 132 220 79 296 232 181 292 233 314 253
188 182 355 68 238 232 154 177 284 180 95 285 81 125 56 66 178 112 0 128 167 169 179 120 69 283 121 213 281
150 67 284 76 155 164 282 216 281 20 187 217 65 167 188 168 198 132 128 0 88 211 269 159 197 172 189 182 335
45 42 182 137 65 85 321 141 111 95 254 138 92 255 175 161 286 220 167 88 0 299 229 104 236 110 149 97 388
341 278 435 151 366 375 298 346 412 193 183 428 230 44 113 235 77 79 169 211 299 0 353 289 213 371 298 379 332
184 271 417 239 388 249 168 188 221 241 279 318 184 389 248 188 362 286 179 269 229 353 0 121 162 345 80 169 242
67 146 292 135 175 147 249 57 221 311 215 280 74 245 176 62 287 232 128 159 184 289 121 0 154 228 41 93 218
121 251 424 137 387 381 95 190 353 169 117 354 150 169 125 92 228 181 69 197 236 213 162 154 0 352 147 247 358
169 185 116 242 57 118 437 245 72 280 159 169 288 127 288 277 358 292 283 172 118 371 345 228 352 0 265 178 39
188 191 337 165 228 188 198 43 216 161 216 241 184 246 177 55 299 233 121 189 149 298 88 41 147 265 0 124 263
45 139 273 228 121 68 314 81 132 189 388 112 158 335 266 155 388 314 213 182 97 379 189 93 247 178 124 0 199
167 79 77 285 97 185 435 243 111 163 322 238 286 288 243 275 319 253 281 135 188 332 342 218 350 39 263 199 0
DISPLAY_DATA_SECTION
1 1150.0 1760.0
2 630.0 1660.0
3 48.0 2898.0
4 750.0 1180.0
5 750.0 2038.0
6 1830.0 0
7 1650.0 650.0
8 1490.0 1630.0
9 790.0 2260.0
10 710.0 1310.0
11 840.0 550.0
12 1170.0 2380.0
13 970.0 1340.0
14 510.0 780.0
15 750.0 980.0
16 3280.0 1280.0
17 230.0 590.0
18 460.0 860.0
19 1840.0 950.0
20 580.0 1380.0
21 830.0 1778.0
22 490.0 580.0

```

Una vez que se verificó que la lectura era correcta, se localizó la sección en donde estaba la matriz y se empezó a escanear. Vemos la matriz en el programa a continuación.

```

C:\Users\holl\Desktop>dir Archivio\Archivos Extra\Archivos Académicos\Universidad\Ondine - Universidad Autónoma de Aguascalientes\Archivos Académicos\Universidad\T.O. Ondine\Practica 8\Practica 8\Integraciona.exe
===== LECTURA DE ARCHIVOS =====
NAME: days29
TYPE: TSP
COMMENT: 29 cities in Bavaria, street distances (Groetschel, Juenger, Reinelt)
DIMENSION: 29
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: FULL_MATRIX
DISPLAY_DATA_TYPE: TWO_DISPLAY
EDGE_WEIGHT_SECTION
0 107 241 190 124 88 316 76 152 157 283 133 113 297 228 129 348 276 188 150 65 341 184 67 221 169 108 45 167
107 0 148 137 88 127 336 183 134 95 254 180 181 234 175 176 265 199 182 67 42 278 271 146 251 105 191 139 79
241 148 0 374 171 259 589 317 217 232 491 312 286 391 412 349 422 356 355 284 182 435 417 192 424 116 337 273 77
190 137 374 0 282 234 222 192 248 42 117 387 79 107 38 121 152 86 68 78 137 151 239 135 137 242 165 228 205
124 88 171 282 0 61 392 282 46 160 319 112 163 322 240 232 314 287 238 155 65 366 380 175 307 57 228 121 97
88 127 259 234 61 0 386 141 72 167 351 55 157 331 272 226 362 296 232 164 85 375 249 147 381 118 188 60 185
190 336 589 222 382 386 0 233 438 254 282 439 235 254 210 187 313 266 154 282 321 298 168 249 95 437 198 314 435
76 183 137 192 282 141 233 0 213 188 272 193 131 382 233 98 344 289 177 216 141 346 188 57 190 245 43 81 243
152 134 217 248 46 72 438 213 0 206 365 89 289 368 286 278 360 333 284 281 111 412 321 221 353 72 266 132 111
157 95 232 42 188 167 254 188 286 0 159 220 57 149 88 122 193 127 180 28 95 192 241 231 169 280 161 189 163
183 254 491 117 319 351 282 272 365 159 0 484 176 106 79 161 165 141 95 187 254 103 279 215 117 359 216 388 322
133 188 312 287 112 55 439 193 89 220 484 0 210 384 325 279 415 349 285 217 138 428 310 280 354 169 241 112 238
113 181 288 79 163 157 235 131 209 57 276 218 0 186 117 75 231 165 81 85 92 238 184 74 158 288 104 158 286
197 234 391 187 322 331 254 382 368 149 186 384 186 0 69 191 95 35 125 167 255 44 389 245 169 327 246 335 288
228 175 412 38 240 272 218 233 286 80 79 325 117 69 0 122 122 56 56 188 175 113 240 176 125 280 177 266 243
129 176 349 121 232 226 187 98 278 132 161 279 75 191 122 0 244 178 66 160 161 235 118 61 92 277 95 155 275
148 265 422 152 314 362 313 344 368 193 165 415 231 59 122 244 0 66 178 198 286 77 362 287 228 358 298 389 319
276 199 356 86 287 296 266 289 333 127 141 349 165 35 56 178 66 0 112 132 220 79 296 232 181 292 233 314 253
188 182 355 68 238 232 154 177 284 180 95 285 81 125 56 66 178 112 0 128 167 169 179 120 69 283 121 213 281
150 67 284 76 155 164 282 216 281 20 187 217 65 167 188 168 198 132 128 0 88 211 269 159 197 172 189 182 335
45 42 182 137 65 85 321 141 111 95 254 138 92 255 175 161 286 220 167 88 0 299 229 104 236 110 149 97 388
341 278 435 151 366 375 298 346 412 193 183 428 230 44 113 235 77 79 169 211 299 0 353 289 213 371 298 379 332
184 271 417 239 388 249 168 188 221 241 279 318 184 389 248 188 362 286 179 269 229 353 0 121 162 345 80 169 242
67 146 292 135 175 147 249 57 221 311 215 280 74 245 176 62 287 232 128 159 184 289 121 0 154 228 41 93 218
121 251 424 137 387 381 95 190 353 169 117 354 150 169 125 92 228 181 69 197 236 213 162 154 0 352 147 247 358
169 185 116 242 57 118 437 245 72 280 159 169 288 127 288 277 358 292 283 172 118 371 345 228 352 0 265 178 39
188 191 337 165 228 188 198 43 216 161 216 241 184 246 177 55 299 233 121 189 149 298 88 41 147 265 0 124 263
45 139 273 228 121 68 314 81 132 189 388 112 158 335 266 155 388 314 213 182 97 379 189 93 247 178 124 0 199
167 79 77 285 97 185 435 243 111 163 322 238 286 288 243 275 319 253 281 135 188 332 342 218 350 39 263 199 0
DISPLAY_DATA_SECTION
1 1150.0 1760.0
2 630.0 1660.0
3 48.0 2898.0
4 750.0 1180.0
5 750.0 2038.0
6 1830.0 0
7 1650.0 650.0
8 1490.0 1630.0
9 790.0 2260.0
10 710.0 1310.0
11 840.0 550.0
12 1170.0 2380.0
13 970.0 1340.0
14 510.0 780.0
15 750.0 980.0
16 3280.0 1280.0
17 230.0 590.0
18 460.0 860.0
19 1840.0 950.0
20 580.0 1380.0
21 830.0 1778.0
22 490.0 580.0

```

Conclusiones:

Me pareció una práctica muy sencilla, pues esta práctica será una de ayuda para proyectos posteriores. Creo que fue un buen punto de arranque para actividades posteriores.

Referencias:

Purcell, E. (2007). *Cálculo*. Londres: Pearson Education.

Talbi, E. (2009). *Metaheuristics from design to implementation*. New Jersey: John Wiley & Sons Publication.

Torres, A. (2020). *Apuntes: Optimización Inteligente*. 5° ICI. México: Universidad Autónoma de Aguascalientes.

Anexos:

Anexo 1: Código del programa en lenguaje C:

```
/*  
    Universidad Autónoma de Aguascalientes  
  
        Centro de Ciencias Básicas  
    Departamento de Ciencias de la Computación  
        Optimización Inteligente  
  
        5° "A"
```

Práctica 6: Captura de los datos de un archivo TSPLIB

Doctora Aurora Torres Soto

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: 6 de noviembre del 2020

Descripción:

```
*/  
//Cargamos las librerías  
#include <stdio.h>  
#include <locale.h>  
#include <string.h>  
  
main()  
{  
    setlocale(LC_ALL, "");
```

```

//1. Declaramos las variables que usaremos
/*
    car: Variable que lee caracter por caracter en una línea
    line: Vector que extrae una línea del archivo
    filetxt: Variable FILE de tipo apuntador a un archivo
    qv: Cantidad de vértices del grafo del archivo
    reachedgraph: Detecta la línea en la que la matriz
comienza
*/
int h = 0,i,j,k = 0,qv, reachedgraph = 0;
FILE *filetxt;
char line[200],car;

printf("===== LECTURA DE ARCHIVOS
=====\\n");

//2. Abrimos el archivo
filetxt = fopen("bays29.tsp.txt","r");

int graph[29][29];
int n;

//3. Procesamos el archivo
if(filetxt == NULL)
{
    printf("Problemas para abrir el archivo");
    return 1;
}

//4. Leemos el archivo hasta llegar al EOF (End Of File)
while(feof(filetxt) == 0)
{
    if(reachedgraph == 0)
    {
        fscanf(filetxt,"%s",line);

        if(strcmp(line, "EDGE_WEIGHT_SECTION")==0)
        {
            reachedgraph = 1;
        }
    }
}

```



```

else
{
    if(h != 29)
    {
        fscanf(filetxt,"%d",&n);
        graph[h][j] = n;
        j++;

        if(j == 29)
        {
            j = 0;
            h++;
        }
    }
    else
    {
        fscanf(filetxt,"%s",line);
    }
}

for(i = 0; i < 29; i++)
{
    for(j = 0; j < 29; j++)
    {
        printf("[%d] ",graph[i][j]);
    }
    printf("\n");
}

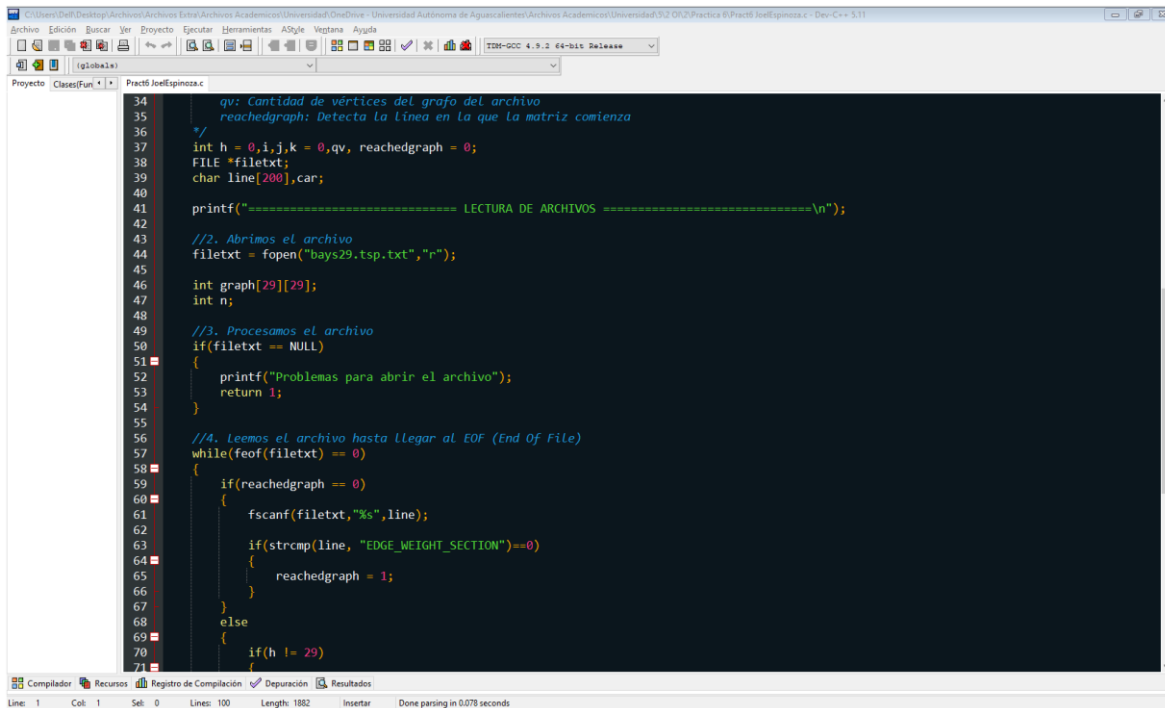
fclose(filetxt);
getchar();
}

```

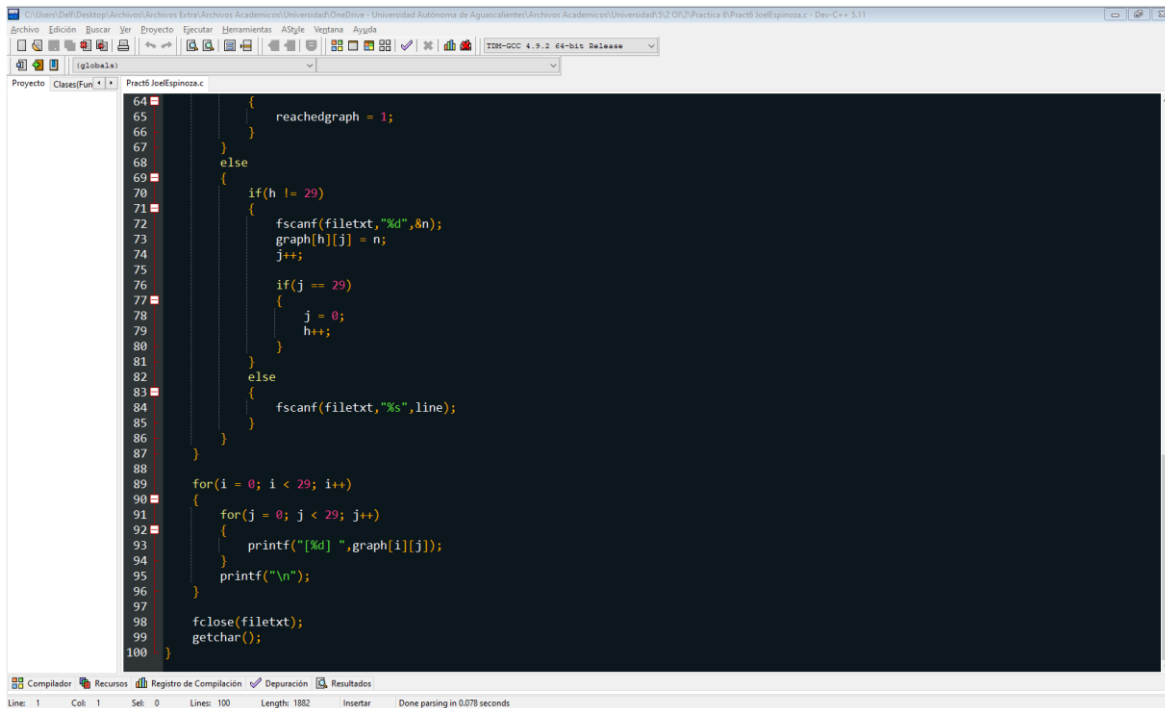
Anexo 2: Impresión de pantalla completa al ejecutar el programa

```
----- LECTURA DE ARCHIVOS -----
[0] [107] [243] [190] [124] [80] [316] [76] [152] [157] [283] [133] [113] [297] [228] [129] [348] [276] [188] [150] [65] [341] [184] [67] [221] [169] [108] [45] [167]
[107] [0] [163] [127] [88] [127] [336] [183] [134] [95] [254] [180] [101] [224] [175] [176] [265] [159] [182] [67] [42] [178] [271] [146] [251] [105] [191] [139] [79]
[241] [148] [0] [374] [171] [259] [509] [317] [217] [232] [401] [312] [280] [391] [412] [349] [422] [356] [555] [204] [182] [435] [417] [292] [424] [116] [317] [273] [77]
[198] [137] [374] [0] [202] [234] [222] [192] [248] [42] [117] [287] [79] [107] [38] [121] [152] [86] [68] [70] [137] [151] [239] [135] [137] [242] [165] [228] [205]
[124] [88] [171] [202] [6] [61] [392] [202] [46] [160] [319] [112] [163] [322] [240] [232] [314] [287] [238] [155] [65] [366] [300] [175] [307] [57] [220] [121] [97]
[0] [127] [269] [234] [63] [0] [386] [141] [72] [167] [353] [55] [157] [333] [272] [226] [362] [296] [232] [144] [88] [375] [246] [147] [304] [118] [188] [40] [185]
[316] [336] [509] [222] [392] [386] [0] [233] [438] [254] [202] [439] [235] [254] [210] [187] [313] [266] [154] [282] [321] [298] [168] [249] [95] [437] [190] [314] [435]
[76] [183] [317] [192] [202] [141] [233] [0] [213] [188] [272] [193] [131] [302] [233] [98] [344] [289] [177] [216] [141] [346] [180] [57] [190] [245] [43] [81] [243]
[152] [134] [127] [249] [46] [72] [438] [212] [0] [206] [365] [89] [280] [368] [286] [278] [360] [233] [204] [201] [111] [412] [221] [223] [353] [72] [266] [132] [111]
[157] [95] [232] [42] [160] [167] [254] [188] [206] [0] [159] [220] [57] [149] [0] [132] [193] [127] [100] [28] [95] [193] [241] [131] [169] [200] [161] [189] [163]
[283] [254] [491] [117] [319] [351] [202] [272] [365] [159] [0] [404] [176] [186] [79] [163] [165] [141] [95] [187] [254] [103] [279] [215] [117] [359] [216] [308] [322]
[133] [180] [212] [287] [112] [55] [439] [192] [89] [220] [404] [0] [210] [184] [325] [279] [415] [349] [205] [217] [138] [428] [310] [200] [354] [169] [241] [112] [238]
[113] [101] [280] [79] [163] [157] [235] [133] [209] [57] [176] [210] [0] [186] [117] [75] [211] [165] [81] [85] [92] [230] [184] [74] [150] [208] [104] [158] [206]
[297] [234] [391] [107] [322] [311] [254] [382] [368] [149] [106] [384] [180] [0] [69] [191] [59] [35] [125] [167] [255] [44] [309] [245] [169] [327] [246] [335] [288]
[228] [175] [412] [18] [248] [272] [218] [233] [280] [80] [79] [125] [117] [69] [0] [122] [122] [56] [56] [300] [175] [113] [248] [176] [125] [280] [177] [260] [243]
[229] [176] [340] [121] [232] [226] [187] [98] [278] [132] [161] [279] [76] [151] [122] [0] [244] [170] [66] [160] [163] [235] [118] [62] [102] [277] [55] [155] [295]
[148] [265] [422] [152] [314] [362] [313] [344] [360] [193] [165] [415] [231] [59] [122] [244] [0] [60] [178] [198] [286] [77] [362] [287] [228] [358] [299] [380] [319]
[276] [199] [356] [86] [287] [296] [266] [289] [333] [127] [141] [349] [165] [35] [56] [178] [66] [0] [112] [132] [220] [79] [296] [232] [181] [292] [233] [314] [253]
[188] [182] [355] [68] [238] [232] [154] [177] [204] [100] [95] [285] [81] [125] [56] [56] [178] [112] [0] [128] [167] [169] [179] [120] [69] [283] [221] [213] [201]
[158] [67] [204] [70] [155] [164] [282] [216] [201] [28] [187] [217] [85] [167] [188] [160] [198] [132] [128] [0] [88] [211] [269] [159] [197] [172] [189] [182] [135]
[65] [42] [182] [137] [65] [85] [321] [141] [111] [95] [254] [138] [92] [255] [175] [161] [286] [220] [167] [88] [0] [299] [229] [184] [236] [118] [149] [97] [188]
[241] [278] [435] [152] [368] [375] [290] [346] [412] [193] [183] [418] [230] [44] [111] [235] [77] [79] [169] [211] [299] [0] [352] [289] [213] [371] [290] [379] [332]
[184] [271] [417] [239] [380] [249] [168] [180] [321] [241] [279] [310] [184] [389] [240] [118] [362] [296] [179] [269] [229] [353] [0] [121] [162] [345] [80] [189] [342]
[67] [146] [292] [135] [175] [147] [249] [57] [221] [131] [215] [200] [74] [245] [176] [62] [287] [232] [120] [159] [104] [289] [121] [0] [154] [220] [41] [93] [218]
[221] [251] [424] [137] [307] [381] [95] [189] [353] [169] [117] [354] [150] [169] [125] [92] [226] [181] [69] [197] [236] [213] [162] [154] [0] [352] [147] [267] [358]
[169] [185] [116] [242] [57] [118] [437] [245] [72] [200] [159] [169] [208] [327] [280] [277] [388] [292] [283] [172] [110] [371] [345] [220] [352] [0] [265] [178] [191]
[108] [191] [337] [165] [220] [188] [190] [43] [268] [161] [216] [241] [104] [246] [177] [55] [299] [233] [121] [189] [149] [290] [80] [41] [147] [265] [0] [124] [263]
[45] [139] [273] [228] [121] [60] [314] [81] [132] [189] [308] [112] [158] [335] [266] [155] [380] [314] [233] [182] [97] [379] [189] [93] [247] [170] [124] [0] [199]
[167] [79] [77] [265] [97] [189] [435] [243] [111] [103] [322] [238] [206] [288] [243] [278] [319] [253] [201] [135] [100] [332] [342] [218] [350] [39] [263] [199] [0]
```

Anexo 3: Algunas capturas del código en el IDE



```
34      qv: Cantidad de vértices del grafo del archivo
35      reachedgraph: Detecta la línea en la que la matriz comienza
36
37      */
38      int h = 0, i, j, k = 0, qv, reachedgraph = 0;
39      FILE *filetxt;
40      char line[200], car;
41
42      printf("===== LECTURA DE ARCHIVOS =====\n");
43
44      //2. Abrimos el archivo
45      filetxt = fopen("bays29.tsp.txt", "r");
46
47      int graph[29][29];
48      int n;
49
50      //3. Procesamos el archivo
51      if(filetxt == NULL)
52      {
53          printf("Problemas para abrir el archivo");
54          return 1;
55      }
56
57      //4. Leemos el archivo hasta llegar al EOF (End Of File)
58      while(feof(filetxt) == 0)
59      {
60          if(reachedgraph == 0)
61          {
62              fscanf(filetxt, "%s", line);
63              if(strcmp(line, "EDGE_WEIGHT_SECTION") == 0)
64              {
65                  reachedgraph = 1;
66              }
67          }
68          else
69          {
70              if(h != 29)
71              {
```



```
64          {
65              reachedgraph = 1;
66          }
67          else
68          {
69              if(h != 29)
70              {
71                  fscanf(filetxt, "%d", &n);
72                  graph[h][j] = n;
73                  j++;
74
75                  if(j == 29)
76                  {
77                      j = 0;
78                      h++;
79                  }
80              }
81              else
82              {
83                  fscanf(filetxt, "%s", line);
84              }
85          }
86      }
87
88      for(i = 0; i < 29; i++)
89      {
90          for(j = 0; j < 29; j++)
91          {
92              printf("[%d] ", graph[i][j]);
93          }
94          printf("\n");
95      }
96
97      fclose(filetxt);
98      getchar();
99
100 }
```