

Universidad de Costa Rica
Escuela de Ingeniería Eléctrica
Programación Bajo Plataformas Abiertas
MSc. Andrés Mora Zúñiga
I Ciclo 2020
IE-0117

Práctica # 7:C: Entrada y Salida de Archivos

Primeramente cabe mencionar que se importan los archivos triangulos.c y triangulos.h del laboratorio6 para trabajar en este laboratorio.

1. Archivo impexp.c

El código del archivo impexp.c corresponde a:

```
1  #include <stdio.h>
2  #include "impexp.h"
3  #include "triangulos.h"
4  #include <math.h>
5  #include <stdlib.h>
6  #include <time.h>
7
8
9  void exportTristoFile(FILE* archivo, int cantidadtri, tri* bloque){
10     for (int i = 0; i < cantidadtri; i++)
11     {
12         fprintf(archivo, "Triangulo %d \n A: %.1f %.1f \n B: %.1f %.1f \n C:
13             %.1f %.1f \n Area: %.1f \n \n",
14             i+1, bloque[i].A.x, bloque[i].A.y, bloque[i].B.x, bloque[i].B.y, bloque[i]
15             ].C.x, bloque[i].C.y, calcArea(bloque[i]));
16     }
17 }
18
19 int importTrisFromFile(FILE* archivo, tri* bloquenuevo){
20     char buffer[1000];
21     double bufferNumeroTri;
22     double bufferNumeroAx;
23     double bufferNumeroAy;
24     double bufferNumeroBx;
25     double bufferNumeroBy;
26     double bufferNumeroCx;
27     double bufferNumeroCy;
28     double bufferArea;
29     int i=0;
30     while(fscanf(archivo, "%s %d %s %d %d %s %d %d %s %d %d %s %d",
31         &buffer, &bufferNumeroTri, &buffer, &bufferNumeroAx, &bufferNumeroAy, &
32         buffer, &bufferNumeroBx,
33         &bufferNumeroBy, &buffer, &bufferNumeroCx, &bufferNumeroCy, &buffer, &
34         bufferArea) != -1){
35
36         bloquenuevo[i].A.x=bufferNumeroAx;
37         bloquenuevo[i].A.y=bufferNumeroAy;
```

```

34     bloquenuevo[i].B.x=bufferNumeroBx;
35     bloquenuevo[i].B.y=bufferNumeroBy;
36     bloquenuevo[i].C.x=bufferNumeroCx;
37     bloquenuevo[i].C.y=bufferNumeroCy;
38     i++;
39 }
40 return i;
41
42 }
```

Dentro de este archivo se incluye el archivo triangulos.h del laboratorio6. Este archivo consiste en dos funciones. La función de la línea 9 no devuelve nada y corresponde, como su nombre lo dice, a exportar los datos de los triángulos creados del laboratorio6 a un archivo. Esta función recibe como parámetro el archivo a donde se quieren exportar los archivos, la cantidad de triángulos que se quieren exportar y el puntero que contiene la dirección de memoria donde se encuentran los triángulos. El for de la línea 10 es el que realiza la exportación de cada segmento del triángulo. En la línea 12 se puede observar que por medio del `fprintf` es que se exportan los datos, donde se muestra que cada segmento del triángulo posee la coordenada en cada vértice A,B y C, además de su respectiva área.

La siguiente función que se ve en la línea 17 corresponde a como su nombre lo dice, a importar los triángulos del archivo correspondiente. Para esto se recibe la dirección del archivo y el puntero que contiene la dirección con la memoria reservada para esos triángulos que se quieren importar. En la línea 18 se muestra el buffer que corresponde a cuando se presenten variables tipo char, o sea letras. Los demás buffers son para asignar respectivamente cada coordenada de cada vértice y el área. En la línea 28 por medio del while y el `fscanf` se logran obtener los datos del archivo. Este método fue útil debido a que ya se sabía como era la estructura del archivo. Por último dentro del while se ingresan todos los datos a la memoria reservada del bloquenuevo, cabe mencionar que no se extrajo el área debido a que se vuelve a calcular con la función del archivo triangulos.c.

2. Archivo impexp.h

En este archivo simplemente se declaran las variables, el código sería:

```

1
2 #if !defined(FUNCTIONS2)
3 #define FUNCTIONS2
4 #include <stdio.h>
5 #include <math.h>
6 #include <stdlib.h>
7 #include <time.h>
8 #include "triangulos.h"
9
10 void exportTristoFile(FILE* archivo, int cantidadtri, tri* bloque);
11 int importTrisFromFile(FILE* archivo, tri* bloquenuevo);
12 #endif // FUNCTIONS2
```

3. Archivo main.c

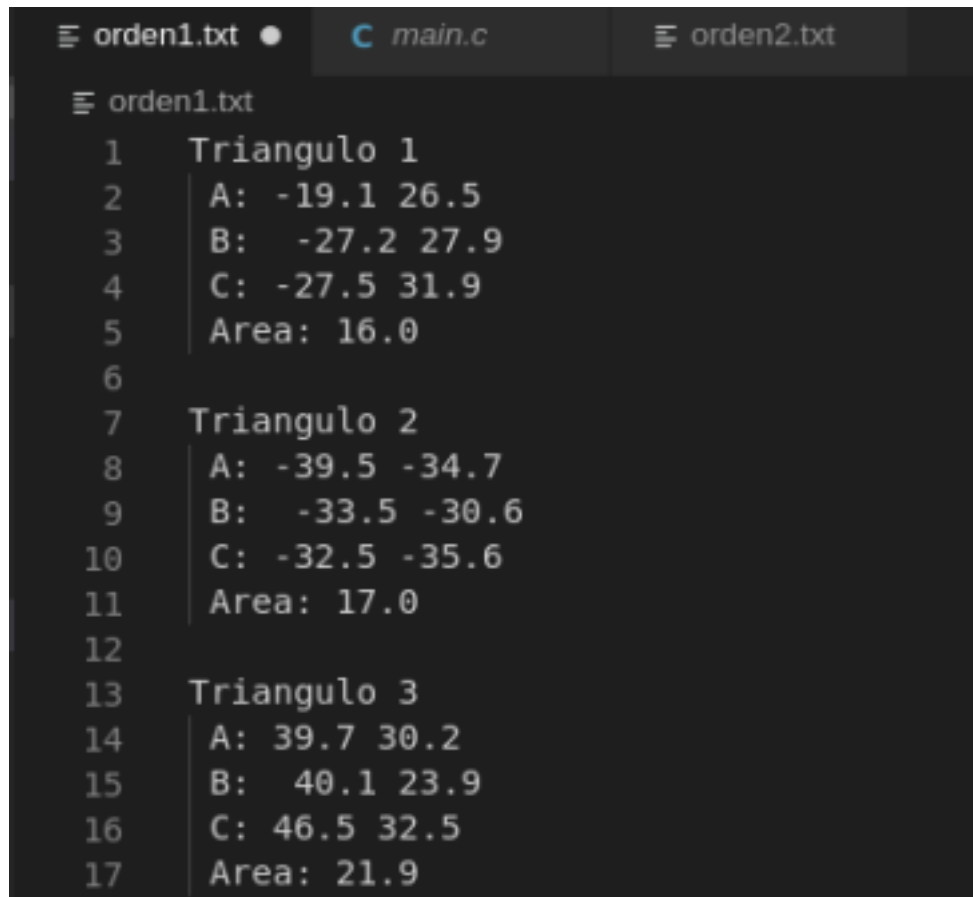
El código corresponde a:

```

1
2 #include <stdio.h>
3 #include <math.h>
4 #include <stdlib.h>
5 #include <time.h>
6 #include "triangulos.h"
7 #include "impexp.h"
8
9 int main(int argc, char const *argv[])
10 {
11     int cantidadtri= atoi(argv[1]);
12     tri* bloque=reserveTri(cantidadtri);
13     initTri(bloque, cantidadtri);
14     FILE* archivo1=fopen("desorden.txt","w");
15     FILE* archivo2=fopen("orden1.txt","w");
16     exportTristoFile(archivo1,cantidadtri,bloque);
17     sortTri(bloque,cantidadtri);
18     exportTristoFile(archivo2,cantidadtri,bloque);
19     free(bloque);
20     fclose(archivo1);
21     fclose(archivo2);
22     FILE* archivo3=fopen("desorden.txt","r");
23     FILE* archivo4=fopen("orden2.txt","w");
24     tri* bloquenuevo=reserveTri(cantidadtri);
25     int cantidadtrinuevo=importTrisFromFile(archivo3,bloquenuevo);
26     sortTri(bloquenuevo,cantidadtrinuevo);
27     exportTristoFile(archivo4,cantidadtrinuevo,bloquenuevo);
28     free(bloquenuevo);
29     fclose(archivo3);
30     fclose(archivo4);
31
32     return 0;
33 }
```

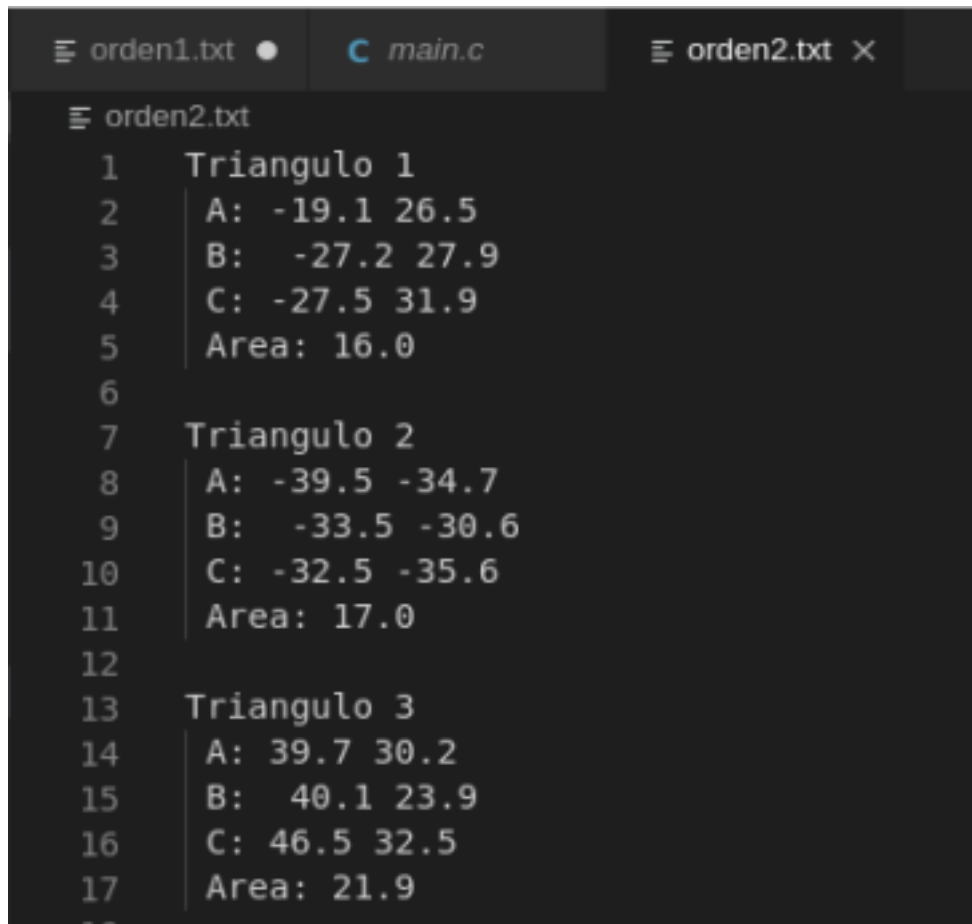
Para el main se importó triangulos.h el cual hace y ordena los triángulos, además de impexp.h el cual exporta los triángulos a un archivo y también los importa. En la línea 11 primeramente se extrae y transforma el datos ingresado desde la línea de comandos. En la línea 12 se reserva la memoria para la cantidad deseada de triángulos mediante la función reserveTri, y guarda la ubicación en bloque. Se llena la memoria reservada a a que apunta bloque con triángulos aleatorios mediante la función initTri. Se abren los archivos de desorden.txt y orden1.txt. Mediante la función exportTristoFile se exportan los triángulos de la memoria reservada y se colocan en el archivo desorden.txt. Se acomodan los triángulos de manera ascendente respecto al área con SorTri. Se exportan de nuevo los triángulos con la función exportTristoFile. Luego se libera la memoria de bloque y se cierran los archivos desorden y orden1. Posteriormente como se ve en la línea 22 se vuelve a abrir el archivo desorden.txt pero a diferencia del anterior este es en tipo read, y se abre el archivo orden2.txt tipor write. Se reserva una nueva casilla de memoria con la ubicación en el puntero bloque-nuevo. En la línea 25 se puede observar que mediante la función importTrisFromFile se logran importar los triángulos del archivo desorden.txt y devuelve la cantidad de de triángulos ex-

portados y lo guarda en la variable cantidadtrinuevo. Se ordenan los triángulos mediante la función `sortTri` como se realizó anteriormente. Se exportan al documento `orden2.txt` mediante la función `exportTrioToFile`. Por último se libera la memoria del bloque nuevo y se cierran ambos archivos(`desorden.txt`,`orden2.txt`). Como resultado se tiene el mismo documento en `orden1.txt` y en `orden2.txt` como se ve en las siguientes figuras:



```
orden1.txt ●  C main.c  orden2.txt
orden1.txt
1  Triangulo 1
2  A: -19.1 26.5
3  B: -27.2 27.9
4  C: -27.5 31.9
5  Area: 16.0
6
7  Triangulo 2
8  A: -39.5 -34.7
9  B: -33.5 -30.6
10 C: -32.5 -35.6
11 Area: 17.0
12
13 Triangulo 3
14 A: 39.7 30.2
15 B: 40.1 23.9
16 C: 46.5 32.5
17 Area: 21.9
```

Figura 1: Archivo `orden1.txt`



```
orden2.txt
1  Triangulo 1
2  A: -19.1 26.5
3  B: -27.2 27.9
4  C: -27.5 31.9
5  Area: 16.0
6
7  Triangulo 2
8  A: -39.5 -34.7
9  B: -33.5 -30.6
10 C: -32.5 -35.6
11 Area: 17.0
12
13 Triangulo 3
14 A: 39.7 30.2
15 B: 40.1 23.9
16 C: 46.5 32.5
17 Area: 21.9
```

Figura 2: Archivo orden1.txt

4. Archivo Makefile

Se realiza la automatización con los targets all y clean. El código corresponde a:

```
1 all:
2     gcc -o triangulos.x main.c triangulos.c impexp.c -lm
3 clean:
4     rm -f *.o *.x *.txt
```