Universidad de Costa Rica
Facultad de Ingeniería
Escuela de Ingeniería Eléctrica

IE-0117 Programación Bajo Plataformas Abiertas

MSc. Andrés Mora Zúñiga - I Ciclo 2020

Laboratorio # 7

C: Entrada y Salida de Archivos

Instrucciones Generales:
Los laboratorios se deben realizar de manera individual.
El laboratorio debe entregarse antes del 10 de Julio a las 23:59.
Entregue un archivo comprimido que incluya un directorio llamado `informe` con los archivos necesarios para generar el PDF del informe (.tex, imágenes, código, entre otros) y un directorio `src` con los archivos `.h`, y `.c` que lleven a la solución de cada ejercicio. Cualquier otro formato o entrega tardía no se revisará y el laboratorio tendrá una nota de cero.

1. Introducción

El siguiente laboratorio tiene como objetivo comprobar los conocimientos adquiridos durante el curso. Específicamente se evaluará el buen manejo del lenguaje de programación C. Los temas a evaluar son: entrada y salida de archivos.

Para ello se presente como ejercicio académico la implementación de un programa en C capaz de extender la funcionalidad del Laboratorio # 6 y permitir que los tríangulos sean importados a partir de archivos de texto plano, y que además triángulos dentro del programa también puedan ser exportados en archivos de texto plano.Además se deberá de automatizar la compilación del programa haciendo uso de un Makefile.

2. Creación de un archivo de encabezado (.h)

Cree un archivo de encabezado llamado `impexp.h` en el cual deberá incluir todas las bibliotecas (por ejemplo `stdlib.h`) que utilizará. Además debe de declarar las firmas/prototipos de las funciones que va a implementar.

2.1. Prototipos de funciones (10 pts)

Declare como mínimo los siguientes prototipos de funciones:

- `importTrisFromFile`: esta función recibe la ruta del archivo de donde se desean cargar los `tri` y una variable en la cual se pueda almacenar la dirección de memoria del bloque en donde van a estar los `tri` luego de la importación. La función además regresa un entero con la cantidad de `tri` importados.
- `exportTrisToFile`: esta función recibe el nombre o ruta del archivo en el que se desea exportar los `tri` y la cantidad de `tri` que se van a exportar. La función no regresa nada.

Nota 1: preste especial atención a que los tipos de las variables no están especificados. Sin embargo si se establece qué debería de recibir y retonar la función de manera lógica. Es responsabilidad del estudiante hacer de manera adecuada la selección de tipos de datos y el formato en el cual desea exportar e importar los `tri`.

Nota 2: se recomienda crear otras funciones intermedias para que la implementación completa sea más ordenada y modular. Sin embargo, esto queda a discreción del estudiante.

Nota 3: recuerde por cada `malloc` hay un `free` y por cada `fopen` hay un `fclose`.

3. Creación de un archivo de implementación (.c) (60 pts)

Cree un archivo llamado `impexp.c` en el cual se incluya el encabezado `impexp.h` y se implemente el cuerpo de todas las funciones allí contenidas.

4. Programa Principal (20 pts)

El archivo `main.c` debe de realizar lo siguiente:

1. Incluir el encabezado de `triangulos.h` y `impexp.h` para hacer uso de los tipos de datos creados y todas sus funciones.
2. Recibir por parámetro de línea de comandos la cantidad de triángulos que se desea crear.
3. Crear una variable llamada `bloq` del tipo `tri*` y hacer que apunte a un bloque de memoria en heap que pueda almacenar la cantidad de triángulos especificados por el parámetro de línea de comandos.
4. Llamar a la función `initTri` enviando la variable `bloq` como parámetro para que se llene de triángulos.
5. Llamar a la función `exportTrisToFile` con los parámetros correctos para que se exporten todos los `tri` a un archivo llamado `desorden.txt`.
6. Llamar a la función `sortTri` enviando la variable `bloq` como parámetro para ordenar los triángulos según el tamaño de su área.
7. Llamar a la función `exportTrisToFile` con los parámetros correctos para que se guarden en un archivo llamado `orden1.txt` todos los `tri` ordenados.
8. Liberar la memoria reservada en heap por medio del puntero `bloq`.
9. Llamar a la función `importTrisFromFile` con los argumentos correctos de forma que después de llamar a la función se tenga un puntero válido a los `tri` almacenados en `desorden.txt` y otra variable con la cantidad de `tri` importados.
10. Llamar a la función `sortTri` enviando el nuevo puntero como parámetro para ordenar los triángulos según el tamaño de su área.
11. Llamar a la función `exportTrisToFile` con los parámetros correctos para que se guarden en un archivo llamado `orden2.txt` todos los `tri` ordenados.
12. Liberar la memoria que fue reservada en heap por la función `importTrisFromFile`.
13. Fin

Nota: `orden1.txt` y `orden2.txt` deberían de verse exactamente igual si su programa funciona de manera correcta y consistente.

5. Automatización de la compilación (10 pts)

Finalmente cree un Makefile para este proyecto. El Makefile debe de contener al menos los siguientes targets:

- all: compila los binarios y crea el ejecutable triangulos.x el cual ejecutaría el programa principal.
- clean: elimina todos los archivos temporales e intermedios, así como el ejecutable final.

6. Informe final del laboratorio

Finalmente, luego de concluir con la implementación y automatización de la compilación, debe de realizar un informe técnico con los detalles de la elaboración del código fuente y capturas de pantalla que evidencien la funcionalidad de su código. Si no se presenta el informe escrito se le asignará una nota de cero al laboratorio. Además se debe proveer todo el código fuente generado para resolver el laboratorio y su respectivo Makefile siguiendo el orden solicitado en las instrucciones generales.