	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	TISTA GARCÍA EDGAR
Asignatura:	ESTRUCTURA DE DATOS Y ALGORITMOS I
Grupo:	1
No de Práctica(s):	PRÁCTICA #5-6. ESTRUCTURA DE DATOS LINEALES: PILA Y COLA // COLA CIRCULAR Y COLA DOBLE
Integrante(s):	CARRILLO CERVANTES IVETTE ALEJANDRA
No. de Equipo de cómputo empleado:	TRABAJO EN CASA
No. de Lista o Brigada:	9
Semestre:	2021 - 2
Fecha de entrega:	27 JUNIO 2021
Observaciones:	

CALIFICACIÓN: _____

PRACTICA # 5-6. ESTRUCTURAS DE DATOS LINEALES: PILA Y COLA // COLA CIRCULAR Y COLA DOBLE

Objetivo de Laboratorio: (5) Revisarás las definiciones, características, procedimientos y ejemplos de las estructuras lineales pila y cola, con la finalidad de que comprendas sus estructuras y puedas implementarlas. (6) Revisarás las definiciones, características, procedimientos y ejemplos de las estructuras lineales cola circular y cola doble, con la finalidad de que comprendas sus estructuras y puedas implementarlas.

Cada código tanto del previo como los ejercicios realizados estan comentados

Previo

Pila.

La pila es un tipo abstracto de dato de tipo LIFO (last in, first out). La estructura pila contiene una variable de tipo entero que indica el tope y una lista de 100 elementos.

```
struct Pila{
    int tope;
    int lista[100];
};
```

Las operaciones que se realizan sobre una pila son:

```
Pila crearPila();
```

Esta función tiene como objetivo crear una pila “p”, toma en cuenta que el tope de dicha pila es igual a 0.
No recibe parámetros, pero devuelve la pila que se creó.

```
int isEmpty(Pila);
```

Es una función utilizada por paso por valor, ya que no se va a modificar nada a la pila, solo va a checar si la pila esta vacía o no; recibe un parámetro que es la pila creada y como se mencionó antes, tiene como objetivo verificar si la pila está vacía o no, (recordemos que al crear una pila el tope de la pila es igual a 0) en caso de que el tope de la pila sea igual a cero, la función va a retornar un 1, en caso contrario la función retornará un 0.

```
void push(Pila*,int);
```

La función “push” es una función utilizada por paso por referencia, ya que se va a modificar la pila, en este caso el objetivo de esta función es agregar elementos a la pila; recibe como parámetros el apuntador de la pila (a donde va a ingresar el elemento) y el valor que quieres ingresar a la pila, lo que hace esta función es acceder a la lista en el índice tope y le asigna el valor que se ingresó en los parámetros, posterior a ello, cambia el valor del tope (le aumenta 1). Como solo ingresa un valor, no retorna nada.

```
int pop(Pila*);
```

La función “pop” es una función utilizada por paso por referencia, ya que se va a modificar la pila, en este caso el objetivo de esta función es eliminar elementos de la pila (el último elemento); recibe como parámetro el apuntador de la pila, manda a llamar a la función “isEmpty” para saber si está vacía o no, ya que si esta vacía te retornará el valor de -1, de lo contrario te creará una variable auxiliar la cual será igual a la lista en el índice tope menos uno y se va a disminuir el tope (ya que se va a quitar un elemento de la pila), finalmente se va a retornar la variable auxiliar.

```
int top(Pila);
```

La función “top” es una función utilizada por paso por valor (no se va a modificar la pila), ya que tiene como objetivo ver el último elemento ingresado; tiene como parámetro la pila, checa si está vacía o no, en caso de estarlo retorna un -1, de lo contrario retorna el ultimo valor ingresado.

Cola.

La cola es un tipo de dato abstracto de tipo FIFO (first in, first out). La estructura cola contiene una variable de tipo entero que indica el índice primero, otra igual de tipo entero que indica el índice ultimo y una lista de 100 elementos.

```
struct Cola{
    int primero;
    int ultimo;
    int lista[100];
};
```

Las operaciones que se realizan sobre una cola son:

```
Cola crearCola();
```

Esta función tiene como objetivo crear una Cola “c”, la cual su índice primero se inicializa en 1 (el cual se encarga de hacer el desencolamiento) y su índice ultimo en 0 (el que será el elemento ingresado). No recibe parámetros, pero devuelve la cola que se creó.

```
int isEmpty(Cola);
```

La función “isEmpty” es una función utilizada por paso por valor, ya que no se va a modificar nada a la cola, solo va a checar si la cola está vacía o no; recibe un parámetro que es la pila creada, y tiene una condicionante, la cual es que si el primer elemento es igual al último más uno (si la cola esta vacía) retorna un 1, de lo contrario (si encuentra algún elemento) retorna un 0.

```
void encolar(Cola*,int);
```

La función “encolar” es una función utilizada por paso por referencia, ya que la cola se va a modificar por qué se va a ingresar un valor; tiene como parámetros el apuntador de la cola creada y el valor que se ingresará a la cola (cabe recalcar que esta función no regresa ningún valor), lo que hace esta función es poner el valor ingresado en su índice ultimo y después aumenta el último elemento a 1.

```
int desencolar(Cola*);
```

La función “desencolar” extrae el elemento del inicio de la cola, tiene como parámetro la cola creada con su apuntador y verifica con la función “isEmpty” si esta vacía o no, en caso de estarlo imprime un mensaje diciendo que la cola está vacía, por el contrario (si la cola no esta vacía), crea una variable auxiliar la cual será igual a la cola “c” en la lista en su índice primero menos uno y se realizará otra vez la verificación para saber si esta vacía la cola, en caso de estarlo se crea una cola con la función “crearCola” y finalmente retorna la variable auxiliar.

```
int first(Cola);
```

La función “first” recibe como parámetro la cola creada y devuelve una copia del primer elemento de la cola, pero no lo elimina.

Cola circular (doble).

La cola circular doble es un tipo de dato abstracto de datos en donde cada elemento puede ser ingresado y recuperado por ambos lados de la estructura. Puede ser de tipo FIFO (last in, first out), o LIFO (last in, first out). La estructura Cola, tendrá como miembros el índice primero, índice último, tamaño de la cola, espacios disponibles de la cola y una lista como apuntador.

```
struct Cola{
    int primero;
    int ultimo;
    int tamano;
    int disponibles;
    int* lista;
};
```

Las operaciones que se realizan sobre una cola circular doble son:

```
Cola crearCola(int);
```

Tiene como función crear una cola, recibe como parámetro el tamaño que va a tener la cola creada, inicializa la cola en su índice primero en 1, en su índice ultimo en 0, en su tamaño al tamaño que la función recibió, como disponibles al tamaño (ya que apenas se creo la cola), y finalmente se asigna lista a memoria dinamica con la función calloc y devuelve la cola creada.

```
int isEmpty(Cola);
```

Esta función recibe la cola creada y se verifica si está vacía o no, en cambio de pila y cola simple esta función debe de cumplir con dos condiciones, si la cola está vacía y a su vez si los espacios disponibles son igual que el tamaño, entonces se cumple y devuelve 1, en caso contrario devuelve 0.

```
void encolarInicio(Cola*,int);
```

Tiene como parámetros, el apuntador de la cola y el valor de “x” ingresado por el usuario, se verifica si la cola en su índice ultimo es igual a cero, encolamos al final, pero en caso contrario, verificamos si hay espacios disponibles en la cola, en caso de que no, aparecerá un anuncio al respecto, y en caso de que si haya espacios disponibles, se encola al inicio.

```
void encolarFinal(Cola*,int);
```

Esta función tiene como objetivo encolar al final de la lista, recibe como parámetros el apuntador de la cola y el valor que se le asignará a “x”, si ya no hay espacios disponibles no se podrá encolar y saldrá un anuncio al respecto; pero si hay espacios disponibles, entonces se podrá encolar al final (el valor “x” puesto por el usuario) en la lista en su índice ultimo menos 1. Esta función no devuelve nada, ya que solo se ingresan valores.

```
int desencolarInicio(Cola*);
```

Esta función tiene como objetivo desencolar al inicio de una lista, recibe el como parámetro el apuntador de la cola creada verifica si esta vacía o no, en caso de estarla aparece un anuncio al respecto, en caso contrario, aumenta el lugar de espacios disponibles y después crea una variable auxiliar, la cual será igual a la lista en su índice primero menos uno. Se hace una condición, si el índice primero es diferente a el ultimo, entonces su índice primero es igual al resultado de su modulo más uno (para que no se sobrescriba ningún dato), en dado caso que no se cumpla aumentamos el tamaño del primero y realizamos otro if el cual dice que si la cola vacía, se creara una cola nueva, finalmente se retornará la variable auxiliar

```
void mostrarValores(Cola);
```

Esta función recibe como parámetro la cola creada y no regresa valores, ya que solo se muestran valores, en este caso se hace un ciclo for según el tamaño de la cola y se imprime la posición y calor de cada elemento que se agrega en la cola.

```
void mostrarIndices(Cola);
```

Esta función recibe como parámetro la cola creada y no regresa valores, ya que solo se imprime el indicé primero y último, así como el espacio disponible de elementos que tiene la cola.

Modulo en algunas operaciones de colaDoble

El modulo en las operaciones de colaDoble se utiliza para asegurar que lo que se encola no se sobrescriba con lo que ya esta y asegurar que, si cabe en la cola, aparte de poder tener el control de los índices.

EJEMPLO DEL MODULO EN ENCOLAR INICIO

```c
void encolarInicio(Cola *c,int x){

// SUPONGAMOS QUE EL VALOR DEL TAM DE LA COLA ES 8, Y SU INDICE PRIMERO ES 9

    if(c->ultimo == 0){
        encolarFinal(c,x);
    }
    else{
        if(c->disponibles == 0){
            printf("ya esta a su maxima capacidad \n");
        }
        else{
            if(c->primero==1){
            // SUPONEMOS QUE C->PRIMERO = 9, POR LO TANTO NO CUMPLE CON ESTA CONDICION
                c->primero+=c->tamano-1;
                c->lista[c->primero-1]=x;
                c->disponibles--;
            }
            else{ // COMO C->PRIMERO = 9, HACEMOS LAS OPERACIONES SIGUIENTES
                c->primero=(c->primero+c->tamano)%((c->tamano)+1);
                // C->PRIMERO = (9 + 8) % (8 + 1)    >>    C->PRIMERO = (17) % (9)    >>    C->PRIMERO = 8

                c->lista[c->primero-1]=x;
                // C->LISTA[8-1] = X    >>    C->LISTA[7] = X    >>    C->PRIMERO = 7

                c->disponibles--;
            }
            // REPETIMOS ELSE HASTA QUE C->PRIMERO=1 Y ENTRE EN LA FUNCION IF
        }
    }
}
```

Ejemplo modulo.c

Previo1.c

Para este ejercicio, lo que hice fue ir ingresando y quitando elementos a la pila con las funciones del archivo “pila.c” y “cola.c” anteriormente vistas, siguiendo los parámetros y tipos de funciones que tiene cada una de ellas.

```c
    Pila previo = crearPila();

    // INGRESAMOS ELEMENTOS
    printf("\nNO INGRESAMOS NINGUN VALOR\n");
    printf("Valor dentro del Tope es: %d \n", top(previo));

    push(&previo, 6); //
    printf("\nINGRESAMOS EL VALOR 6 A LA PILA\n");
    printf("VALOR DEL TOPE: %d \n", previo.tope);
    printf("VALOR DENTRO DEL TOPE: %d \n\n", top(previo));
```

```c
    printf("\nVALOR DENTRO DE LA COLA AL DESENCOLAR: %d \n", desencolar(&previoC));

    encolar(&previoC, 52);
    printf("\nINGRESAMOS EL VALOR 52 A LA COLA\n");
```

Imprime en pantalla el valor del tope y que valor tiene ese tope

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>previo1.exe

__________PILA_____________________________

NO INGRESAMOS NINGUN VALOR
La pila ya está vacia.
Valor dentro del Tope es: -1

INGRESAMOS EL VALOR 6 A LA PILA
VALOR DEL TOPE: 1
VALOR DENTRO DEL TOPE: 6


INGRESAMOS EL VALOR 16 A LA PILA
VALOR DEL TOPE: 2
VALOR DENTRO DEL TOPE: 16


INGRESAMOS EL VALOR 4 A LA PILA
VALOR DEL TOPE: 3
VALOR DENTRO DEL TOPE: 4


ELIMINAMOS UN VALOR A LA PILA
VALOR DEL TOPE: 2
VALOR DENTRO DEL TOPE: 16


ELIMINAMOS UN VALOR A LA PILA
VALOR DEL TOPE: 1
VALOR DENTRO DEL TOPE: 6


INGRESAMOS EL VALOR 9 A LA PILA
VALOR DEL TOPE: 2
VALOR DENTRO DEL TOPE: 9


__________COLA_____________________________

EL PRIMER ELEMENTO DE LA COLA ES: 1
EL ULTIMO ELEMENTO DE LA COLA ES: 0

INGRESAMOS EL VALOR 16 A LA COLA

INGRESAMOS EL VALOR 5 A LA COLA

INGRESAMOS EL VALOR 23 A LA COLA

VALOR DENTRO DE LA COLA AL DESENCOLAR: 16

INGRESAMOS EL VALOR 52 A LA COLA

VALOR DENTRO DE LA COLA AL DESENCOLAR: 5

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>
```

Previo1.c

Previo2.c

Para este ejercicio, lo que hice fue ir ingresando y quitando elementos a la cola con las funciones del archivo “colaDoble.c” anteriormente vistas.

```
22      encolarInicio(&previo2, 28);
23      printf("\nINGRESAMOS EL VALOR 28 A LA COLA\n");
24
25      desencolarFinal(&previo2);
26      printf("\nVALOR DESPUES DE DESENCOLAR: %d\n", desencolarFinal(&previo2));
```

Imprime el valor ingresado y el valor después de desencolar.

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>previo2.exe

INGRESAMOS EL VALOR 5 A LA COLA

INGRESAMOS EL VALOR 6 A LA COLA

INGRESAMOS EL VALOR 16 A LA COLA

INGRESAMOS EL VALOR 12 A LA COLA

INGRESAMOS EL VALOR 23 A LA COLA

INGRESAMOS EL VALOR 28 A LA COLA

VALOR DESPUES DE DESENCOLAR: 6

VALOR DESPUES DE DESENCOLAR: 12

VALOR DESPUES DE DESENCOLAR: 28

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>
```

Previo2.c

ACTIVIDADES PARA EL DESARROLLO DE LA PRÁCTICA

Ejercicio 1

Este ejercicio lo realice a través del concepto visto en clase de “pila”

Este programa tiene como función principal crear una pila, pedirle al usuario ingresar 10 valores para la pila y de estos 10 valores determinar el mayor de los elementos que el usuario ingreso. Para ello se crearon 3 pilas

```
10      Pila datos = crearPila(); // ALMACENAR LOS DATOS QUE INGRESO EL USUARIO
11      Pila mayor = crearPila(); // ALMACENAR EL NUMERO MAYOR
12      Pila aux = crearPila(); // PILA AUXILIAR PARA PASAR DATOS A LA OTRA
```

Pilas creadas

Después se realizó un ciclo for para ir agregando todos los valores ingresados por el usuario a la primera pila “datos”, dentro de este ciclo for se creó una variable donde se va almacenando el dato escrito por el usuario y se agregar a la pila con la función push `push (&datos, valor);` (este ciclo se repetirá 10 veces para ir ingresando valor por valor a la pila).

Luego determinamos el mayor de los datos ingresados por el usuario mediante un ciclo while, mientras que la pila no esté vacía se hará un push en la pila “mayor” con los datos sacados de la pila “datos”, pero siendo de tipo LIFO, el ultimo valor de la pila datos, será el primero que ingrese a la pila mayor. Con otro ciclo while, mientras que la pila no esté vacía, si el tope de la pila “datos” es menor que el tope de la pila “mayor”, se hará un push igual que el anterior pero ahora con la pila “aux” con los datos sacados de la pila “datos”, pero si no se cumple con la condición, entonces se hará un push de la pila “aux” con los datos sacados de la pila mayor, seguido de otro push de la pila “mayor” con los datos sacados de la pila “datos”.

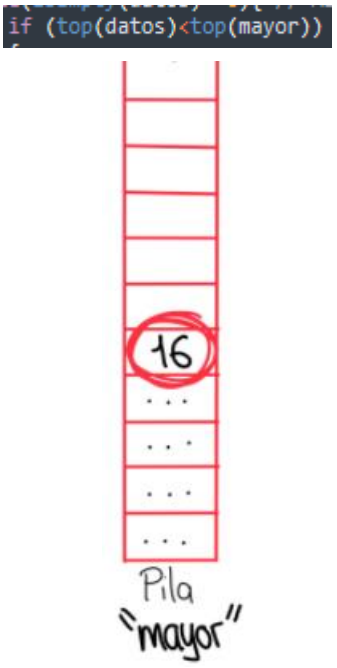
Al imprimir el valor mayor de los números ingresados por el usuario, será el valor al que al utilizar la función pop en la pila “mayor” devuelva

Al principio de este ejercicio me costo un poco de trabajo diferenciar las diferentes pilas, pero al momento de dibujarlas, fue mucho más sencillo entenderlo y termine este primer ejercicio en clase.

Impresión en pantalla

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>ejercicio1.exe

ESCRIBE EL VALOR [1] DE LA PILA:  4

ESCRIBE EL VALOR [2] DE LA PILA:  5

ESCRIBE EL VALOR [3] DE LA PILA:  6

ESCRIBE EL VALOR [4] DE LA PILA:  5

ESCRIBE EL VALOR [5] DE LA PILA:  5

ESCRIBE EL VALOR [6] DE LA PILA:  16

ESCRIBE EL VALOR [7] DE LA PILA:  12

ESCRIBE EL VALOR [8] DE LA PILA:  9

ESCRIBE EL VALOR [9] DE LA PILA:  3

ESCRIBE EL VALOR [10] DE LA PILA:  2

EL VALOR MAYOR DE LOS NUMEROS INGRESADOS POR EL USUARIO ES 16

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>
```

Ejemplo




```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>ejercicio1.exe

ESCRIBE EL VALOR [1] DE LA PILA:  50

ESCRIBE EL VALOR [2] DE LA PILA:  120

ESCRIBE EL VALOR [3] DE LA PILA:  18

ESCRIBE EL VALOR [4] DE LA PILA:  49

ESCRIBE EL VALOR [5] DE LA PILA:  14

ESCRIBE EL VALOR [6] DE LA PILA:  65

ESCRIBE EL VALOR [7] DE LA PILA:  25

ESCRIBE EL VALOR [8] DE LA PILA:  75

ESCRIBE EL VALOR [9] DE LA PILA:  35

ESCRIBE EL VALOR [10] DE LA PILA:  76

EL VALOR MAYOR DE LOS NUMEROS INGRESADOS POR EL USUARIO ES 120

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V1>
```

Ejercicio1.c

Ejercicio 2

Este ejercicio lo realice a través del concepto visto en clase de “cola”

Este ejercicio tiene como función principal crear una estructura llamada “Documento” el que tendrá como miembros: [nombre, autor, numero de páginas, tamaño], para después solicitar al usuario el número de documentos que quiera registrar y llenar cada uno de ellos, (cabe recalcar que se deberá almacenar en un arreglo dinámico de documentos), luego cuando el usuario termine de indicar los datos el programa deberá comenzar a “imprimir” los documentos que están en la cola tomando en cuenta que cada página del documento se tarda 4 segundos en imprimir, finalmente se tendrá que imprimir en pantalla el nombre del documento, así como el tiempo aproximado de lectura en minutos.

Para este ejercicio, realice un programa que incluye como head un archivo llamado “Documentos,h” donde se encuentran las operaciones que tiene la cola pero modificadas para que en vez de recibir un dato de tipo entero, reciba un documento.
Después de incluir a el archivo “Documento.h”, se le pide al usuario cuantos documentos quiere ingresar y se crea una función malloc que es la que va a almacenar los documentos ingresados, luego se crea un ciclo for para llenar los datos que tiene cada documento y se mandan a encolar

```
// TODOS LOS DATOS SE MANDAN A ENCOLAR
encolar(&DocCola, ap[i]);
```

Finalmente se utiliza otro ciclo de repetición, igual tipo for, en el cual se crea una variable tipo documento llamada CRegresada en la cual se almacenará los documentos desencolados para que después se imprima en pantalla el tiempo de “impresión” de cada documento, así como su nombre, aparte del tiempo en total de impresión de los documentos realizados.

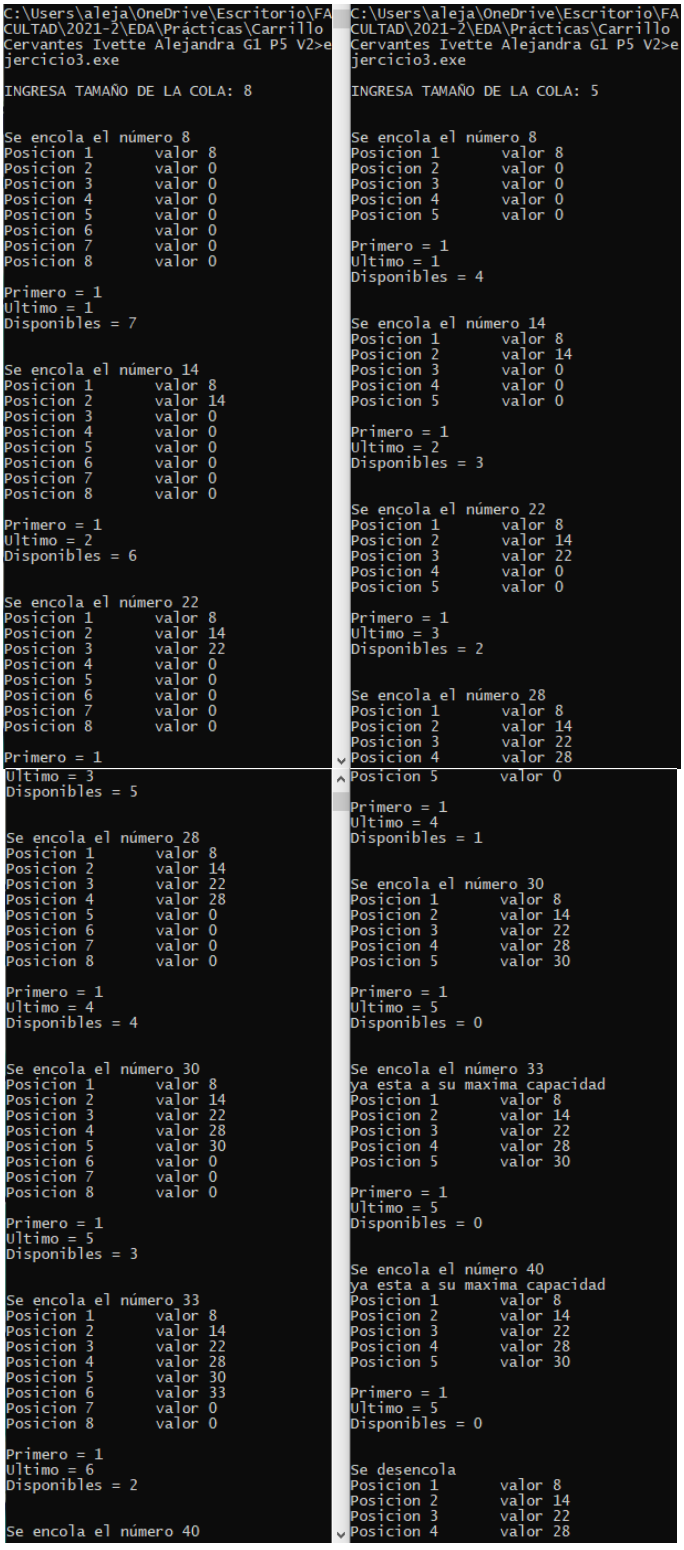
Este ejercicio me llevo un poco de tiempo, ya que no sabía como se podía imprimir los documentos, pero me di cuenta que debía de desencolar los documentos para poder imprimirlos ya que es de tipo FIFO (no se si haya otra manera, pero así se pudo), de ahí en fuera la conversión de segundos a minutos y la suma estuvo sencillo.

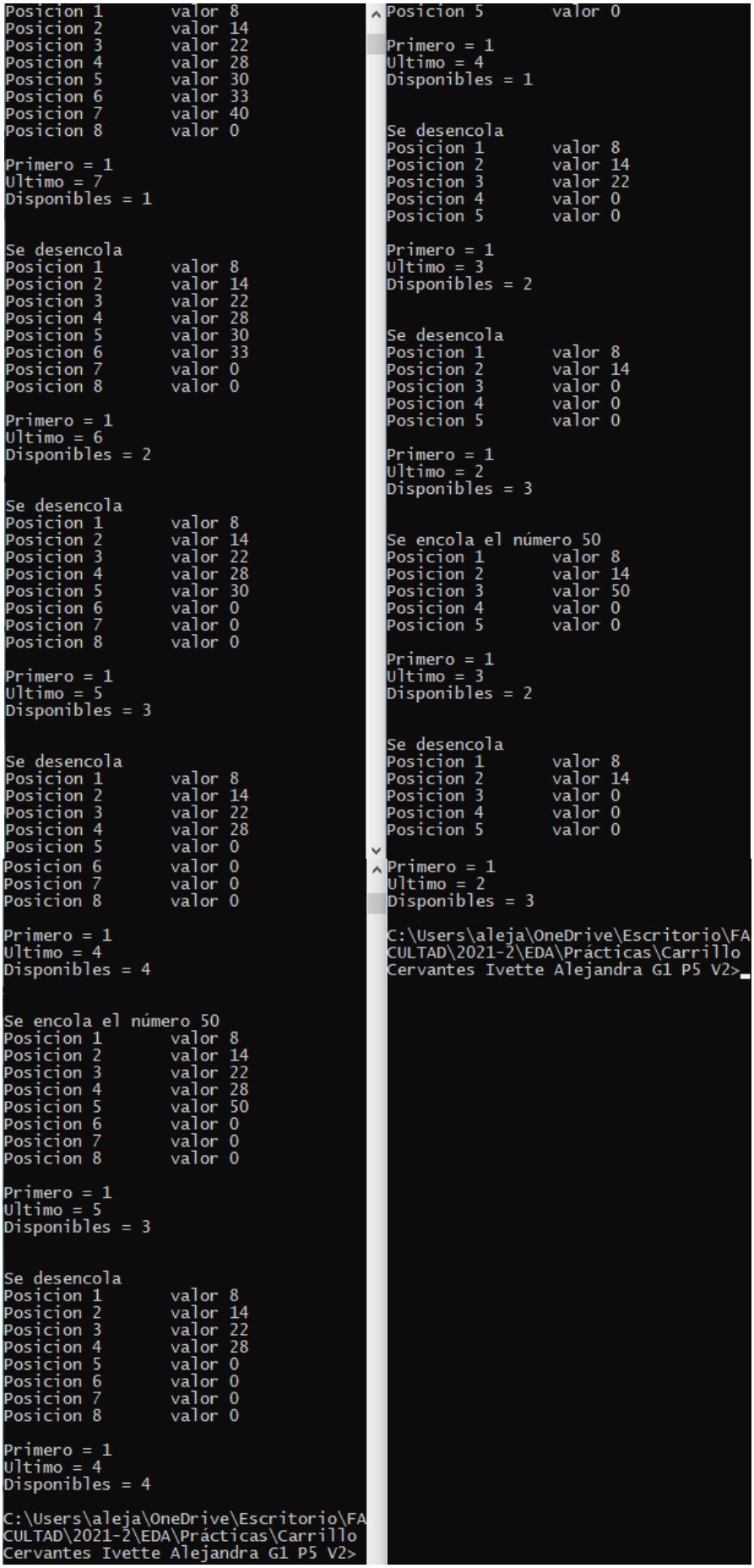
```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V2>ejercicio2.exe


CUANTOS DOCUMENTOS QUIERES REGISTRAR? 2


        DOCUMENTO [1]

Nombre del Documento [1]: El prisionero de Azkaban

Autor [1]: J. K. Rowling

Número de paginas [1]: 359

Tamaño del documento [1]: 407


        DOCUMENTO [2]

Nombre del Documento [2]: Serie Álgebra

Autor [2]: Alejandra Carrillo

Número de paginas [2]: 20

Tamaño del documento [2]: 25

_ _ _ _ _ _ IMPRIMIENDO _ _ _ _ _ _

Doc 1:  Nombre: El prisionero de Azkaban
        Tiempo de impresión: 23.93 minutos // 359 páginas x 4 seg = 1436.00 segundo
s

Doc 2:  Nombre: Serie Álgebra
        Tiempo de impresión: 1.33 minutos // 20 páginas x 4 seg = 80.00 segundos


Tiempo total de la impresion: 25.27 minutos

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V2>
```

Ejercicio 3

Este ejercicio lo realice a través del concepto visto en clase de “cola circular”

Este programa tiene como función principal realizar operaciones de una cola circular y ver en pantalla como funciona esta. Se debe de ver en pantalla las operaciones para una “cola circular” de 8 y 5 elementos, para ello se le pregunta al usuario el tamaño que quiere que la cola circular tenga; además se muestra en pantalla el contenido de la estructura conforme se va avanzando en cada operación, su índice “primero” y su índice “ultimo”, así como cuántos elementos vacíos o disponibles hay, para ello se utilizaron las funciones de una cola circular “encolarfinal, mostrarvalores, mostraríndices”

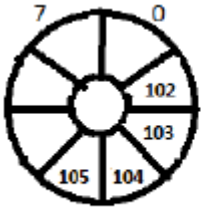




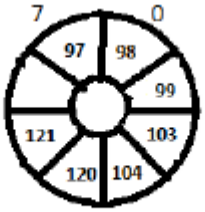
Ejercicio3.c

Ejercicio 4

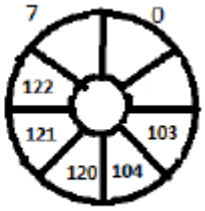
Este ejercicio tiene como objetivo, realizar las operaciones necesarias para que, dada una cola doble, contega los elementos que se indican en las figuras, utilizando solo un bloque de instrucciones para llegar a esos estados.



Estado 1



Estado2



Estado 3

Para realizar estos estados, tome en cuenta que las posiciones comenzaban desde el indice 0 e hice las siguientes operaciones (Se encuentran en el codigo)

```c
printf("\n\n\t_ESTADO 1_\n");
//printf("\nSe encola al final el n%cmero 101\n", 163);
encolarFinal(&queue,101);

//printf("Se encola al final el n%cmero 102\n", 163);
encolarFinal(&queue,102);

//printf("Se encola al final el n%cmero 103\n", 163);
encolarFinal(&queue,103);

//printf("Se encola al final el n%cmero 104\n", 163);
encolarFinal(&queue,104);

//printf("Se encola al final el n%cmero 105\n", 163);
encolarFinal(&queue,105);

//printf("Se desencolo el inicio el inicio\n\n", 163);
desencolarInicio(&queue);

mostrarValores(queue);
//mostrarIndices(queue);
```

```c
printf("\n\n\t_ESTADO 2:_\n");
//printf("\nSe encola desencolo al final\n");
desencolarFinal(&queue);

//printf("Se encola desencolo el inicio\n");
desencolarInicio(&queue);

//printf("Se encola al inicio el n%cmero 99\n", 163);
encolarInicio(&queue,99);

//printf("Se encola al inicio el n%cmero 98\n", 163);
encolarInicio(&queue,98);

//printf("Se encola al incio el n%cmero 97\n", 163);
encolarInicio(&queue,97);

//printf("Se encola al final el n%cmero 120\n", 163);
encolarFinal(&queue,120);

//printf("Se encola al final el n%cmero 121\n\n", 163);
encolarFinal(&queue,121);

mostrarValores(queue);
//mostrarIndices(queue);
```

```c
printf("\n\n\t_ESTADO 3:_\n");

//printf("\nSe encola desencolo el inicio\n");
desencolarInicio(&queue);

//printf("Se encola desencolo el inicio\n");
desencolarInicio(&queue);

//printf("Se encola desencolo el inicio\n");
desencolarInicio(&queue);

//printf("Se encola al final el n%cmero 122\n\n", 163);
encolarFinal(&queue,122);

mostrarValores(queue);
//mostrarIndices(queue);
```

```
C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V2>ejercicio4.exe


        _ESTADO 1_
Posicion 1       valor 0
Posicion 2       valor 102
Posicion 3       valor 103
Posicion 4       valor 104
Posicion 5       valor 105
Posicion 6       valor 0
Posicion 7       valor 0
Posicion 8       valor 0


        _ESTADO 2:_
Posicion 1       valor 98
Posicion 2       valor 99
Posicion 3       valor 103
Posicion 4       valor 104
Posicion 5       valor 120
Posicion 6       valor 121
Posicion 7       valor 0
Posicion 8       valor 97


        _ESTADO 3:_
Posicion 1       valor 0
Posicion 2       valor 0
Posicion 3       valor 103
Posicion 4       valor 104
Posicion 5       valor 120
Posicion 6       valor 121
Posicion 7       valor 122
Posicion 8       valor 0

C:\Users\aleja\OneDrive\Escritorio\FACULTAD\2021-2\EDA\Prácticas\Carrillo Cervantes
 Ivette Alejandra G1 P5 V2>
```

Ejercicio4.c

Conclusiones

Se cumplieron los objetivos de esta práctica, ya que se reviso paso a paso las características y operaciones que tienen las estructuras lineales lista y cola, al igual que cola circular y cola doble, se realizaron ejercicios acerca de estos temas tanto teóricos, como prácticos.

En esta práctica se logró cubrir el 100% de los ejercicios solicitados en la práctica (4 ejercicios), el ejercicio 1, como comentaba anteriormente, me costó trabajo definir para que se utilizaba cada una de las 3 pilas, pero dibujándolas y viendo que se podía hacer con cada una de ellas, fue más sencilla su implementación en el código. En el segundo ejercicio si me costo un poco más de trabajo, ya que me confundí al momento de imprimir los documentos, hasta que recordé que Cola es de tipo FIFO, entonces ya pude imprimir cada documento. En el tercer y cuarto ejercicio no estaba segura si estaban bien hechos los programas, ya que se me hicieron muy sencillos, pero después de que preguntarán en clase, me di cuenta de que si estaban bien.

Respecto al previo, lo que más me costó trabajo fue encontrar el uso del modulo en las operaciones de cola circular, pero después de investigar un poco y hacer las instrucciones a mano, ya me quedo un poco más clara la idea de para que sire el módulo.

Considero que estos ejercicios si contribuyeron al aprendizaje del concepto visto en clase, ya que vimos cada estructura lineal en práctica, aparte de que aplicamos en esta práctica temas anteriormente vistos como es la memoria dinamica, apuntadores, funciones, etc. Estos ejercicios estuvieron muy bien planteados para la práctica, ya que aparte de que se vio muy bien el uso de las estructuras lineales te hace investigar más acerca del tema para comprenderlo muy bien. Personalmente, siento que este tema es uno de los más importantes en la asignatura, ya que considero que te orienta a lo que son listas ligadas.

Referencias

- Joyanes. Programación en C. Departamento de Lenguajes y Sistemas Informáticos e Ingenie-ría de Software. 2da Edición