



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESTRUCTURAS DE DATOS

PRÁCTICA 2

**CONVERSIÓN DE UNA EXPRESIÓN EN NOTACIÓN
INFIX A UNA EXPRESIÓN EN NOTACIÓN
POSTFIXA.**

IMPLEMENTACIÓN DE UNA BICOLA

HERNÁNDEZ CASTELLANOS CÉSAR URIEL

MARTÍNEZ ISLAS MAURICIO JOEL

1CV7

28/02/2017

LUNA BENOSO BENJAMIN



INDICE

Resumen.....	3
Introducción.	3
Experimentación y resultados.....	5
Pseudocódigo.....	11
Infija a postfija.	11
Bicola	14
Conclusiones.....	17
Referencias.....	18

Resumen.

En el presente reporte se muestra la documentación de la práctica, cuyas funciones son las siguientes:

- 1- Convertir una expresión en notación interfija a una en notación postfija que guarde su respectivo resultado en un fichero, mediante el uso de pilas.
- 2- Implementar una bicola con ciertas funciones.

Introducción.

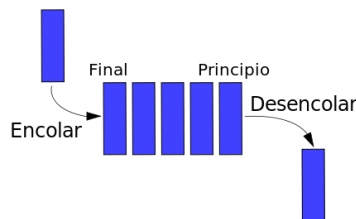
Una computadora es una máquina con la capacidad de manipular un gran volumen de información. Por tanto, es de suma importancia comprender los conceptos de organización y manipulación de la información que se realiza mediante las estructuras de datos que no es más que la forma de organizar de una manera particular los datos en una computadora con el propósito de ser utilizado de manera eficiente.

Variadas estructuras de datos se adecuan para diferentes situaciones, algunas otras son altamente especializadas para ciertas tareas.

Las pilas y colas son dos de las estructuras más utilizadas debido a su amplio ámbito de aplicación. En esta práctica se estudiarán principalmente las colas, su utilización y su implementación básica en el lenguaje C.

Uno de los conceptos más fundamentales y útiles en las ciencias de la computación junto con la pila es el de la cola.

Una cola es un conjunto ordenado de elementos del que se pueden suprimir elementos de un extremo (llamado la parte delantera de la cola)



Representación de una cola.

A una cola se le denomina como una lista FIFO (El primero en entrar, el primero en salir) que es lo contrario de una pila la cual es una lista LIFO (El último en entrar, el primero en salir).

En la nuestra vida cotidiana encontramos innumerables ejemplos de colas, lo podemos ver en una caseta de peaje, en la fila del banco, etc.

Se le aplican tres operaciones fundamentales a una cola. La operación $\text{insertar}(q,x)$, inserta el elemento "x" en la parte posterior de la cola q y establece su contenido en x. La tercera operación $\text{empty}(q)$ retorna falso o verdadero, dependiendo si la cola contiene elementos.

La operación insertar puede ejecutarse siempre, ya que no existe un límite en la cantidad de elementos que puedan contenerse en una cola. Pero, la operación remover solamente puede aplicarse si la cola no se encuentra vacía.

El resultado de intentar remover un elemento de una cola vacía se le ha denominado como subdesbordamiento, por otro lado la operación empty siempre se puede aplicar.

Operaciones básicas.

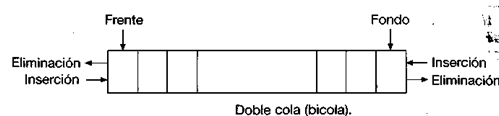
- Crear: Se crea la cola vacía.
- Encolar: (añadir, entrar, insertar): Se añade un elemento a la cola.
- Desencolar: (sacar, salir, eliminar): Se elimina el elemento frontal de la cola, es decir, el primer elemento que entró.
- Frente: (Consultar): Devuelve el elemento frontal de la cola, es decir el primero en ingresar a la cola.

Las colas se usan con frecuencias para simular situaciones del mundo real, como puede ser en el sector transporte u operaciones de investigación (etc), dónde los objetos, eventos o personas son almacenados en una cola como datos para su posterior procesamiento.

Por su parte, la pila, otra estructura que usamos en la práctica la podemos definir como un conjunto ordenado de elementos, en la cual únicamente se pueden agregar y eliminar elementos de ella por un extremo, a la cual llamaremos tope, también es una estructura de tipo LIFO que traducido del inglés significa último en entrar primero en salir.

Colas dobles (Bicola)

La cola doble es una cola bidimensional en la que el ingreso y eliminación pueden ser realizadas por cualquiera de los dos extremos.



Representación de una bicola.

Notación infija.

Es la notación común de fórmulas aritméticas y lógicas, en la cual se escriben los operadores entre los operandos, en los que se encuentre actuando. Este tipo de notación no es tan sencilla de leer para la computadora como lo es la notación prefija o postfija.

Ejemplo: $5 + 5$ es 10

Notación postfija

Es un método algebraico alternativo de introducción de datos. Se refiere a que el operador ocupa la posición después de los operandos sus características principales son: el orden de los operando se conserva igual que en una expresión infija equivalente. Su principio es el de evaluar datos directamente cuando se introducen y manejarlos dentro de una estructura LIFO, lo que optimiza los procesos.

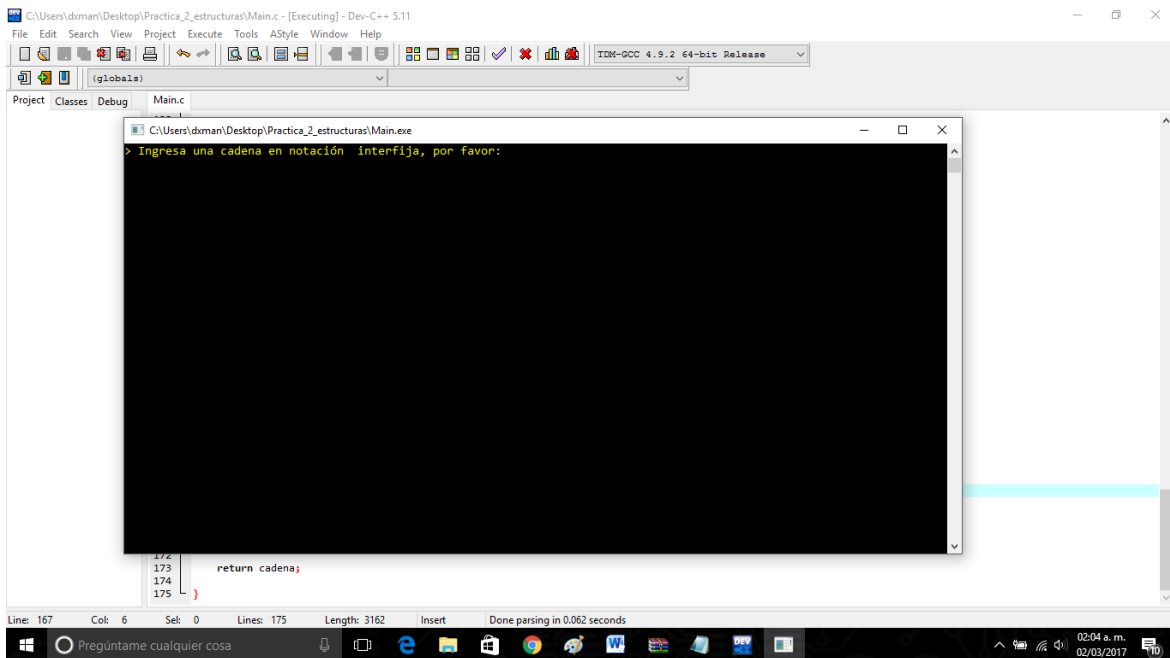
Ejemplo: $123*+$ es 7

Experimentación y resultados.

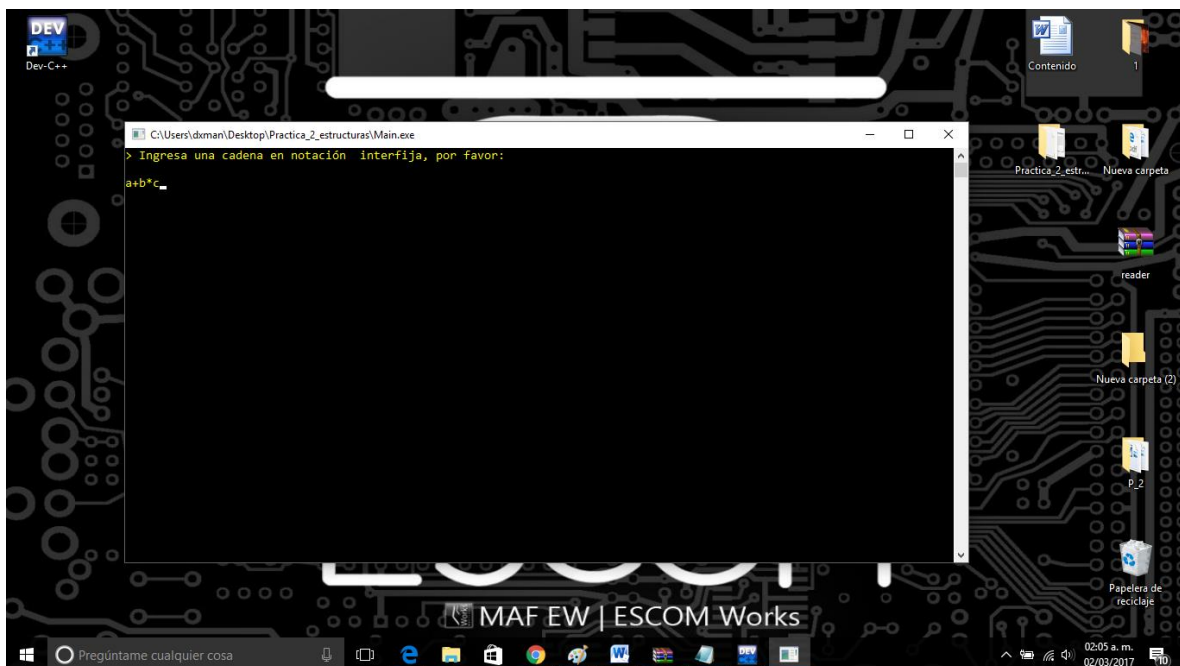
La práctica consiste en llevar a cabo dos aplicaciones, las cuales tienen como propósito el de convertir una expresión en notación infija a una en notación postfija, para posteriormente guardar ambas en un fichero, la segunda aplicación consiste en implementar una bicola con las siguientes funcionalidades.

- Ingresar frente.
- Ingresar final.
- Eliminar frente.
- Eliminar final.
- Leer elemento del frente.
- Leer elemento del final.
- Imprimir toda la bicola (sacando todos los elementos por el frente)

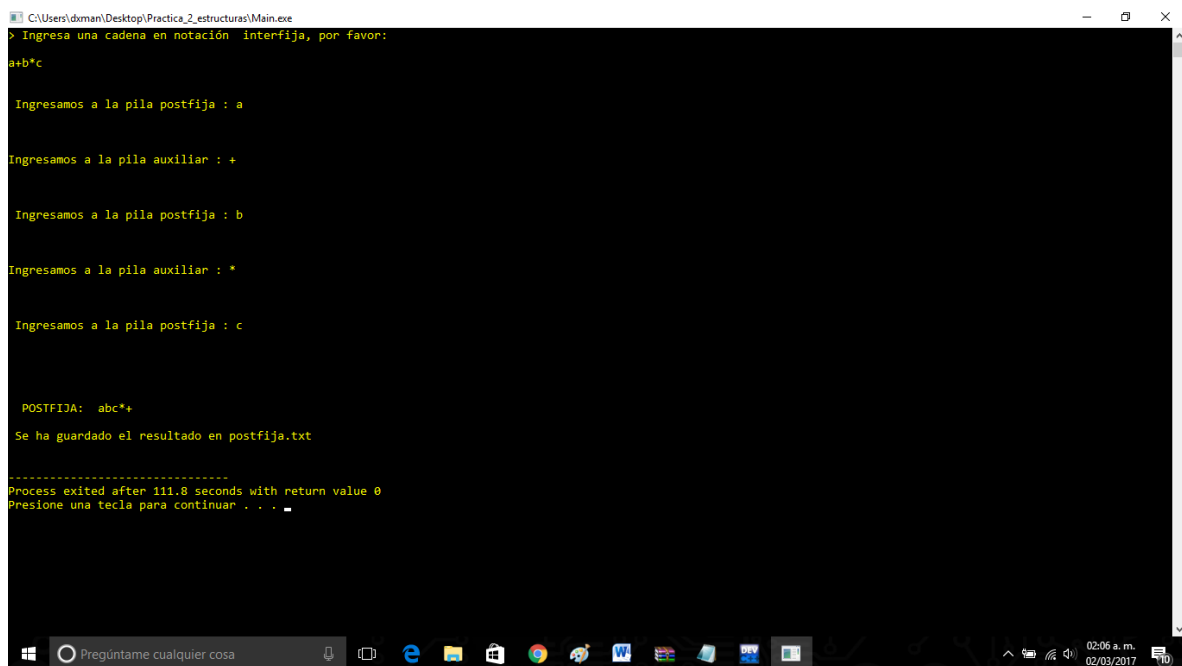
Conversión de notación infija a postfija.



Solicitud de la expresión en notación infija.



Ingreso de la expresión en notación infija.



```
C:\Users\dxman\Desktop\Practica_2_estructuras\Main.exe
> Ingresa una cadena en notación interfija, por favor:
a+b*c

Ingresamos a la pila postfija : a

Ingresamos a la pila auxiliar : +

Ingresamos a la pila postfija : b

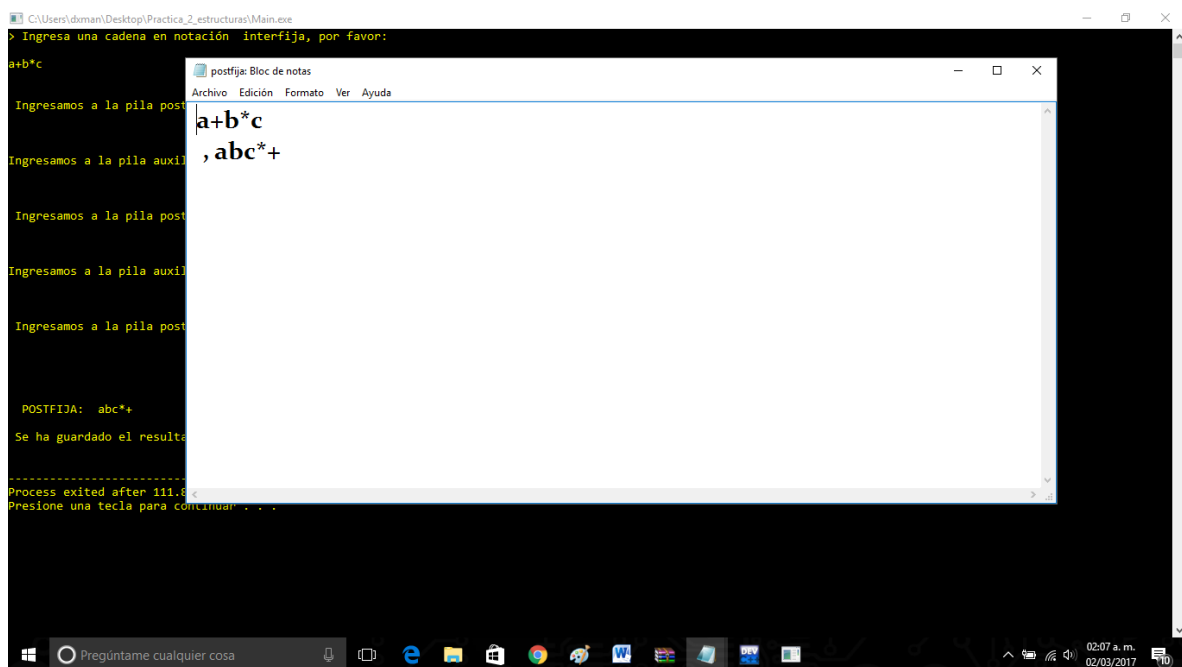
Ingresamos a la pila auxiliar : *

Ingresamos a la pila postfija : c

POSTFIJA: abc*+
Se ha guardado el resultado en postfija.txt

-----
Process exited after 111.8 seconds with return value 0
Presione una tecla para continuar . . .
```

Proceso y resultado de la conversión a postfija.



```
C:\Users\dxman\Desktop\Practica_2_estructuras\Main.exe
> Ingresa una cadena en notación interfija, por favor:
a+b*c

Ingresamos a la pila postfija : a

Ingresamos a la pila auxiliar : +

Ingresamos a la pila postfija : b

Ingresamos a la pila auxiliar : *

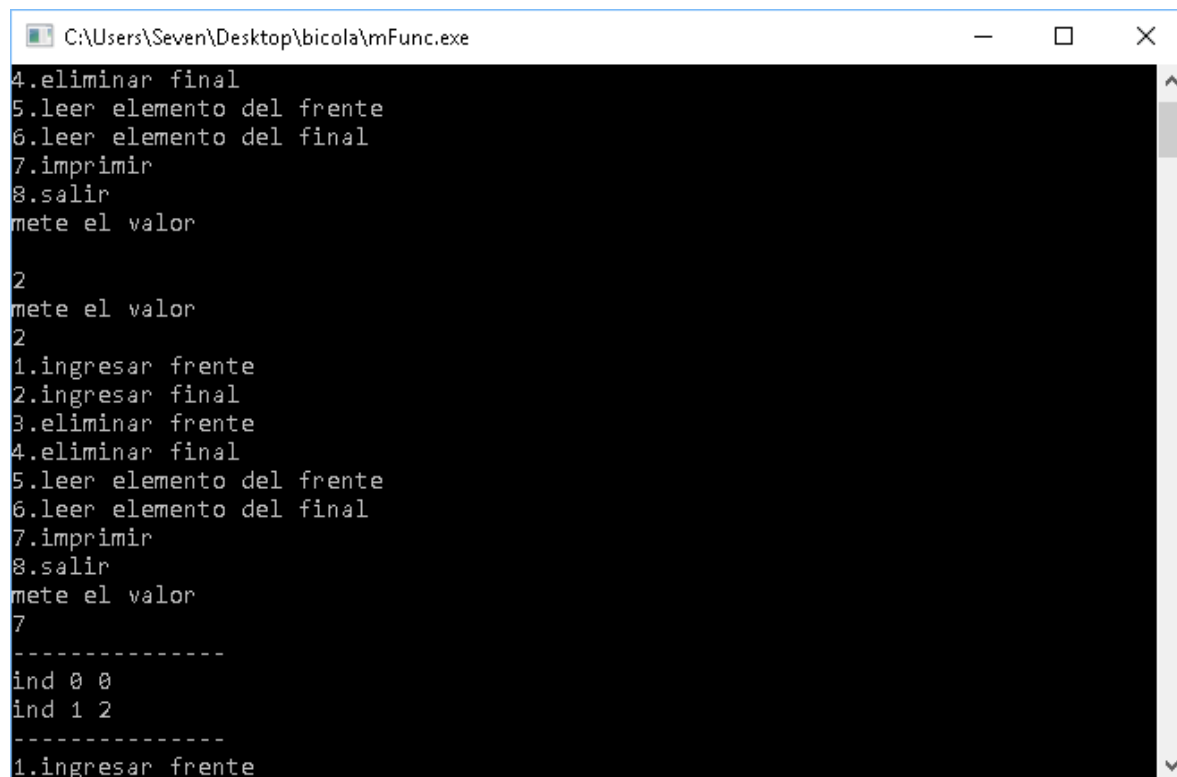
Ingresamos a la pila postfija : c

POSTFIJA: abc*+
Se ha guardado el resultado en postfija.txt

-----
Process exited after 111.8 seconds with return value 0
Presione una tecla para continuar . . .
```

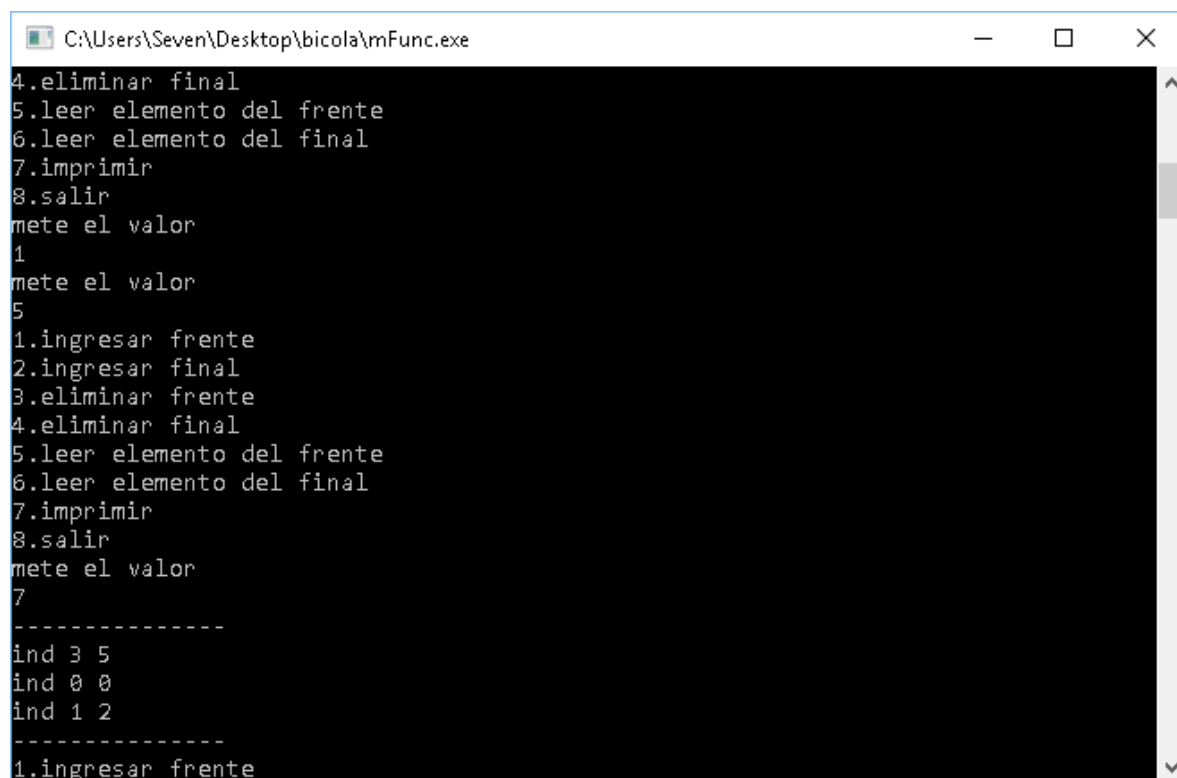
Resultado en el fichero postfija.txt

Bicola



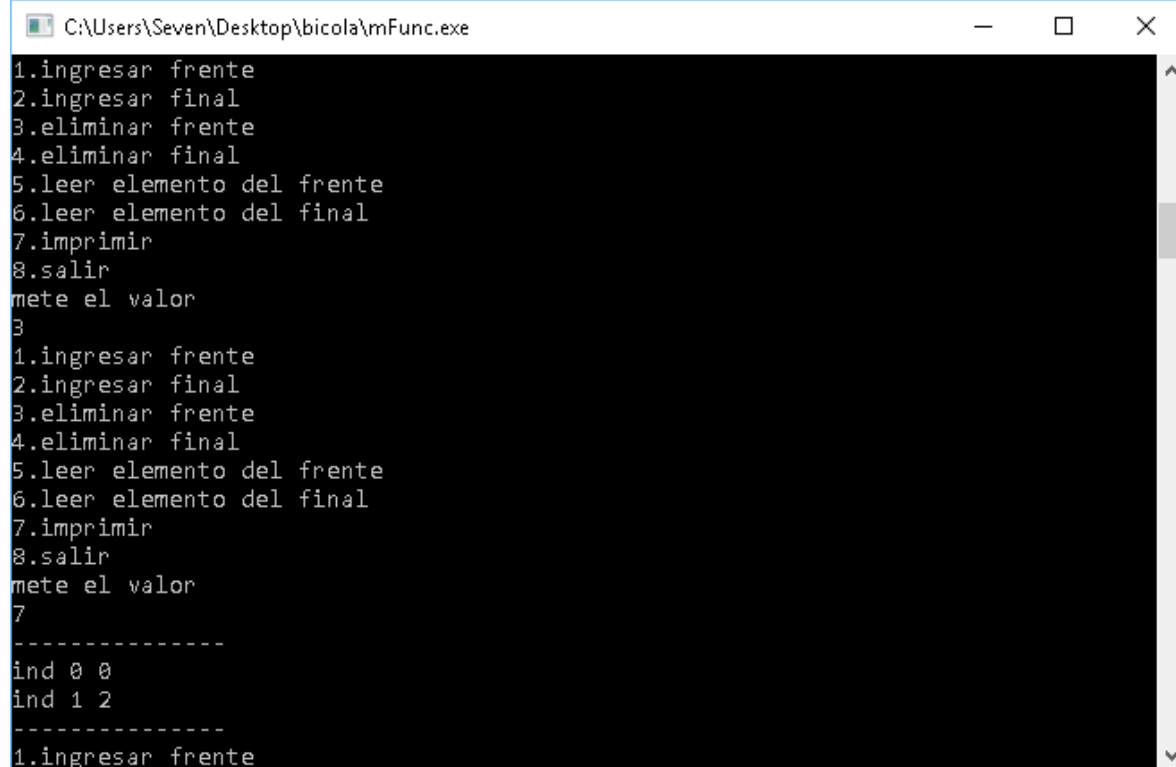
```
C:\Users\Seven\Desktop\bicola\mFunc.exe
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
2
mete el valor
2
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
7
-----
ind 0 0
ind 1 2
-----
1.ingresar frente
```

Metiendo valor por final e imprimiendo



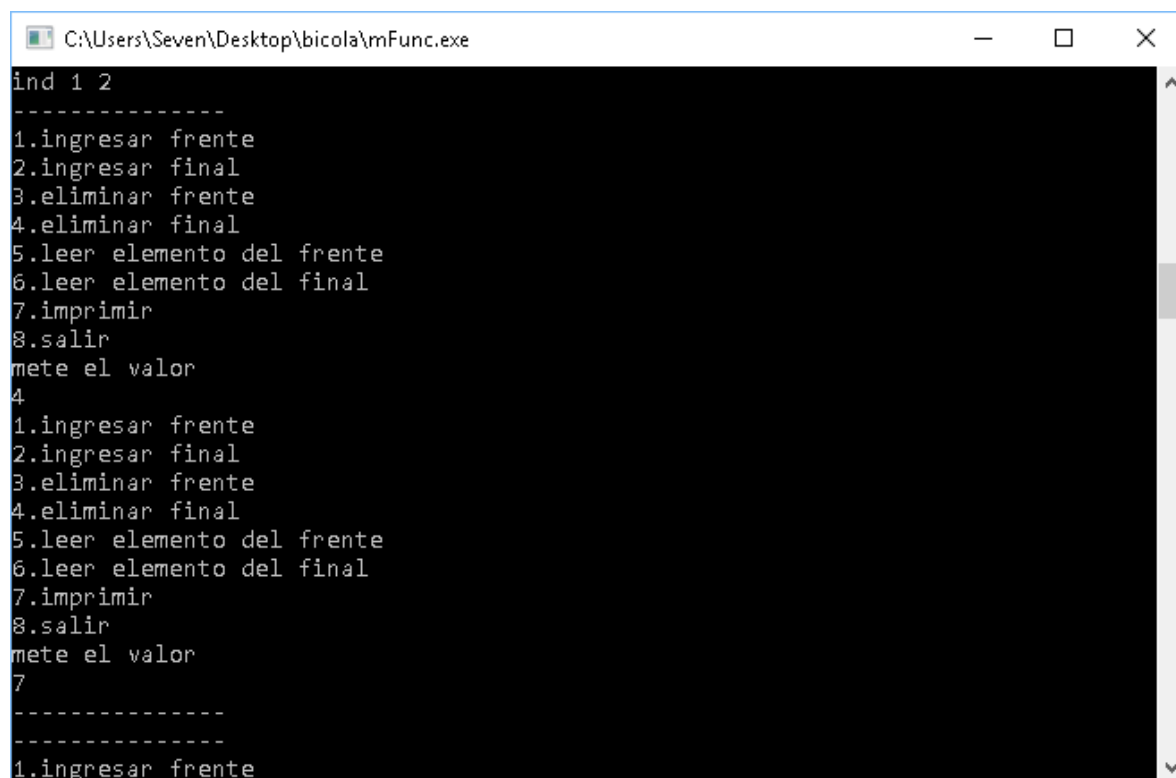
```
C:\Users\Seven\Desktop\bicola\mFunc.exe
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
1
mete el valor
5
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
7
-----
ind 3 5
ind 0 0
ind 1 2
-----
1.ingresar frente
```


Metiendo valor por frente e imprimiendo



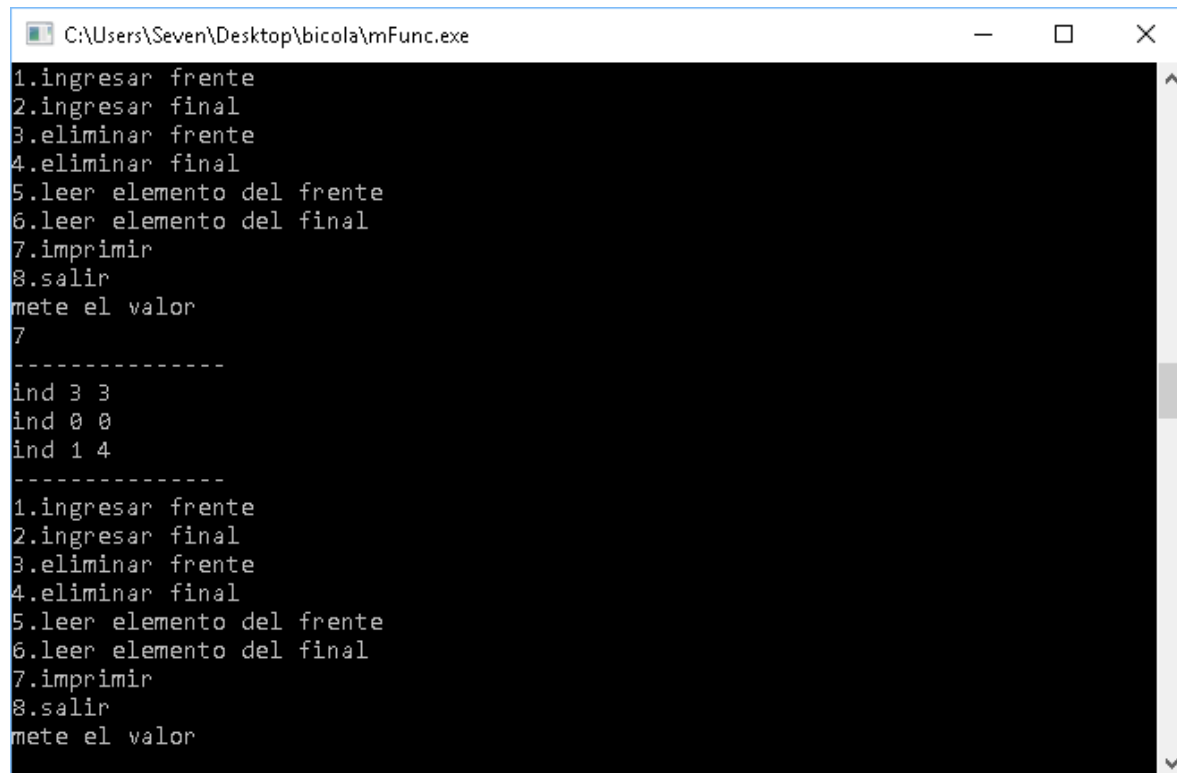
```
C:\Users\Seven\Desktop\bicola\mFunc.exe
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
3
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
7
-----
ind 0 0
ind 1 2
-----
1.ingresar frente
```

Eliminando frente e imprimiendo



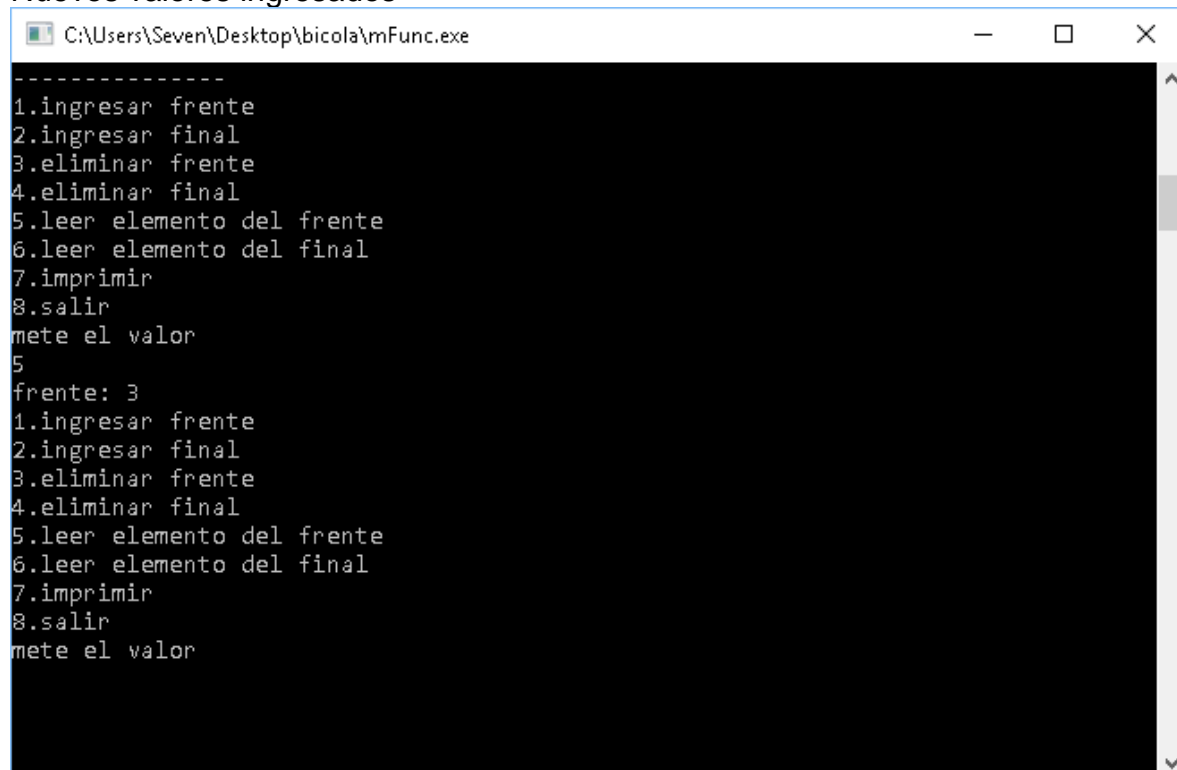
```
C:\Users\Seven\Desktop\bicola\mFunc.exe
ind 1 2
-----
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
4
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
7
-----
-----
1.ingresar frente
```

Eliminando final e imprimiendo



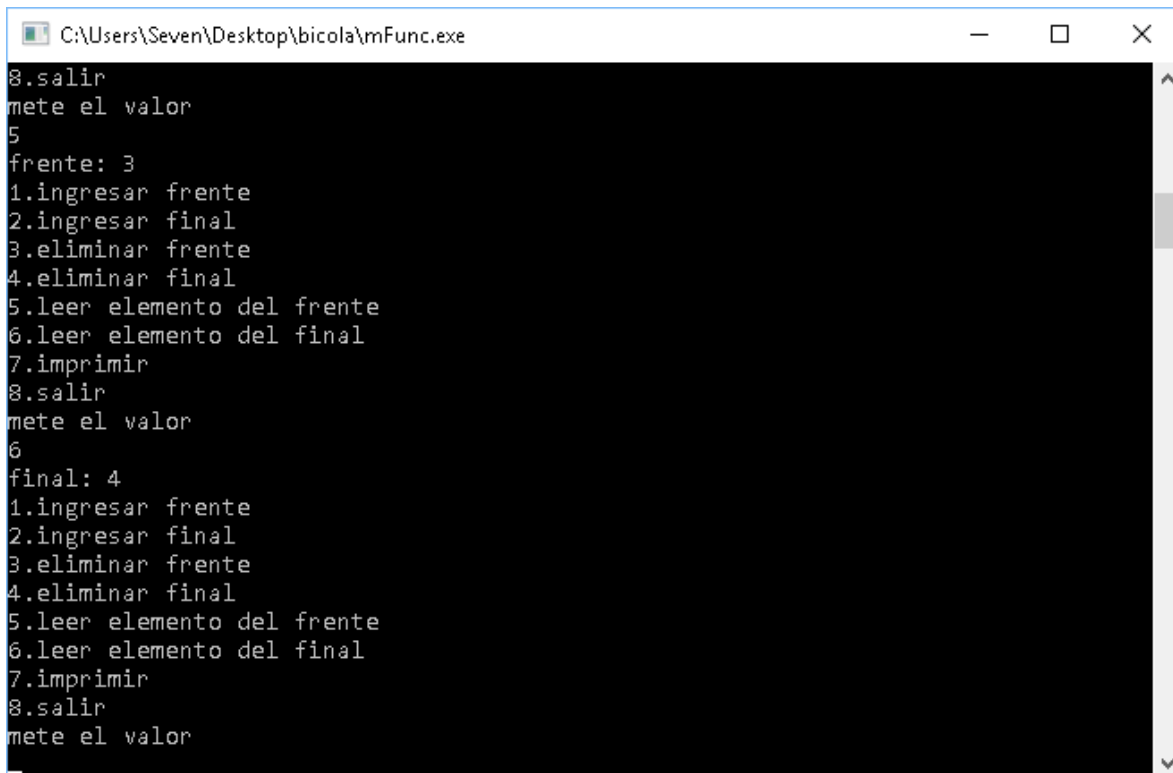
```
C:\Users\Seven\Desktop\bicola\mFunc.exe
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
7
-----
ind 3 3
ind 0 0
ind 1 4
-----
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
```

Nuevos valores ingresados



```
C:\Users\Seven\Desktop\bicola\mFunc.exe
-----
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
5
frente: 3
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
```

Leer frente



```
C:\Users\Seven\Desktop\bicola\mFunc.exe
8.salir
mete el valor
5
frente: 3
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
6
final: 4
1.ingresar frente
2.ingresar final
3.eliminar frente
4.eliminar final
5.leer elemento del frente
6.leer elemento del final
7.imprimir
8.salir
mete el valor
```

Leer final

Pseudocódigo.

Infija a postfija.

Inicio de función principal.

```
PILA pil, postfija, aux
Entero i -> 0
CrearPila( : pil )
CrearPila( : postfija)
CrearPila( : aux)
Cadena cadena
cadena -> regresar (cadena)
```

Desde i -> 0 hasta fin de cadena Hacer

1 Sí (cadena en i -> '+') o (cadena en i -> '-') o (cadena en i ->

“*”) o (cadena en i -> '/') o (cadena en i -> '^') o (cadena en i -
> '(') o (cadena en i -> ')') entonces

2 Si((PilaVacía(pil) -> 0) y (prioridad(de cadena en i) es
menor o igual a prioridad (CimaPila(pil)))) entonces

Mientras (PilaVacía(pil) -> 0 o (prioridad (cadena
en i) es menor o igual a prioridad(CimaPila(pil))))

InsertarPila(:postfija, QuitarPila(:pil))

Fin Mientras

InsertarPila(:pil, cadena en i)

2 En caso contrario

Imprimir “Ingresamos a la pila auxiliar”
InsertarPila(:pil, cadena en i)

2 Fin Si

1 En caso contrario

Imprimir “Ingresamos a la pila postfija”
Insertarpila(:postfija, cadena en i)

Fin Para

Mientras (PilaVacía(pil) sea igual a 0) entonces
InsertarPila(&postfija, QuitarPila(:pil))

Fin Mientras

Mientras(PilaVacía(postfija) sea igual a 0) entonces
InsertarPila(:aux, QuitarPila(:postfija))

Fin Mientras

Si cadenita es nula entonces
Error de salida

Fin si

Copiar_el_contenido_de_la_cadena_fuente_en_la_variable_destino (cadenita, “”)

Mientras (PilaVacía(aux) es igual a 0)

Concatenar_cadena_con_caracter(cadenita, QuitarPila(:aux))

Fin mientras

Imprimir "POSTFIJA se ha guardado en el fichero postfija.txt"

Concatenar(cadena, "")
Concatenar(cadena, cadenita)
Ingresar_a_txt(cadena)

Fin Principal

Entero prioridad(carácter operador)

Entero prioridad -> 0

Si ((operador -> '*') o (operador -> '/')) entonces
 Prioridad -> 2
En caso contrario Si ((operador -> '-') o (operador == '+')) entonces
 Prioridad -> 1
En caso contrario Si (operador -> '^') entonces
 Prioridad -> 3
En caso contrario Si (operador -> '(') o (operador -> ')') entonces
 Prioridad -> 0
En caso contrario
 Prioridad -> 0
Fin Si

Retornar prioridad

Fin función prioridad.

Cadena regresar(Cadena x)

Cadena cadena
Entero contador -> 0
Entero i -> 0

Cadena -> longitud de cien
Si (cadena es nula) entonces

 Salir(-1)
Fin Si

Imprimir "Ingresa una cadena en notación interfija, por favor"

Obtenercadena(cadena)

Regresar cadena

Fin función regresar

Bicola

//incluyendo a las funciones de bicola

```
Funcion ingresarFrente ( COLA c, tipoDato a)
    si colaLlena Entonces
        imprimir "no"
    Sino
        c->frente = (c->frente==0) ? c->frente+(TAM-1) : (c->frente-1)%TAM
    FinSi
```

Fin Funcion

```
Funcion ingresarFinal ( COLA c, tipoDato a )
    si colaLlena entonces
        imprimir "no"
    sino
        c->fin = (c->final+1)%TAM
        c->CListaCola[c->final] = a
    FinSi
```

Fin Funcion

```
Funcion eliminarFrente ( COLA* c )
    si colaVacía Entonces
        imprimir "no"
    sino
        c->frente = (c->frente+1)%TAM
    FinSi
```

Fin Funcion

```
Funcion eliminarFinal ( COLA* c )
    si colaVacía Entonces
        imprimir "no"
    Sino
        c->final = (c->final==0) ? c->final+(TAM-1) : (c->final-1)%TAM
    FinSi
```

Fin Funcion

```
Funcion td <- leerFrente ( COLA* c )
    td = c->clistaCola[c->frente]
Fin Funcion
```

```
Funcion td <- leerFinal ( COLA *c )
    td = c->clistaCola[c->final]
Fin Funcion
```

```

Funcion imprimirC ( COLA* c )
    i=c->frente
    Si !colaVacía(*c) entonces
        mientras i<>c->final
            imprimir c->clistaCola[i]
            si i>=TAM-1 Entonces
                i=0
            Sino
                i++;
            FinSi
        FinMientras
        imprimir leerFinal(c)
    FinSi
Fin Funcion

```

```

Funcion tipoDato <- getValue ( )
    Imprimir "mete el valor"
    leer tipoDato n
Fin Funcion

```

```

Funcion printMenu ( )
    imprimir "1. ingresar frente"
    imprimir "2. ingresar final"
    imprimir "3. eliminar frente"
    imprimir "4. eliminar final"
    imprimir "5. leer elemento del frente"
    imprimir "6. leer elemento del final"
    imprimir "7. imprimir"
    imprimir "8. salir"
Fin Funcion

```

```

Algoritmo bicolalImplementación
    Hacer
        printMenu()
        cont = getValue()
        COLA c
        crearCola(&c)
        Segun cont Hacer
            1:
                ingresarFrente(&c, getValue())
            2:
                ingresarFinal(&c, getValue())
            3:
                eliminarFrente(&c)
            4:
                eliminarFinal(&c)

```

```
5:      leerFrente(&c)
6:      leerFinal(&c)
7:      imprimirC(&c)
      Fin Segun
      Hasta Que cont<>8
FinAlgoritmo
```


Conclusiones.

Hernández Castellanos César Uriel.

Como conclusión de la práctica sobre colas como estructuras de datos, es de vital importancia mencionar que se trata de un conjunto ordenado o estructura de datos en la que el modo de acceso a sus elementos es de tipo FIFO (First In First Out) que nos permite almacenar y recuperar datos. La estructura cola puede aplicarse en una gran multitud de ocasiones en la computación, gracias a su simplicidad y ordenación de la misma estructura.

Junto con las pilas, las colas son uno de los conceptos más útiles en la programación, ya que nos auxilian a simplificar ciertas tareas, además poseen un gran campo de aplicación.

Además se pudo percatar que a la computadora le es mucho más sencillo trabajar con expresiones en notación postfija o prefija, por lo que es de gran utilidad la aplicación desarrollada en la presente práctica, para lograr comprender como trabaja en parte el compilador por dentro.

De igual forma se tuvo un acercamiento a un tipo de cola especial, que es la bicola, la cual funcionó de manera eficiente en ciertos rubros, la cual cumple con las características de una bicola convencional que es la de inserción/eliminación de elementos en ambos extremos de la cola, ser representada como un vector y con dos índices.

Martínez Islas Mauricio Joel.

La manera en la que se implementó la bicola no resultó de una manera muy positiva, no se pudo llegar a ninguna manera de resolver el hecho de que en una bicola, de la manera en la que se plantean inicialmente las funciones de la cola (que se puede ver como el padre de la bicola surge un problema, el hecho de que se va a desperdiciar el índice cero, y se va a tener un lugar desperdiciado, por lo que el diseño de la bicola que se utilizó resulta, de cierta manera vano, ya que se puede utilizar todas sus funciones como se tenía en mente, pero se desperdicia un lugar. Éste hecho es algo para pensar, ya que en una cola tradicional sí se puede fijar un valor inicial (aunque eso no soluciona el problema cuando se implementa en la bicola) y decir que cuando verdaderamente se encuentra vacía es cuando, frente ha rebasado a final en un lugar (lo que daría paso a reestablecer los índices en su valor inicial), aun así, en la cola tradicional hay ciertas alternativas que permiten solucionar esto de una manera que todavía se ve eficiente.

Referencias

- [1]Y. Langsam, M. Augenstein and A. Tenenbaum, *Estructuras de datos con C y C++*, 1st ed. Pearson Prentice Hall, 2000.
- [2]"Colas Dobles y Circulares", *Scribd*, 2017. [Online]. Available: <https://es.scribd.com/doc/41076913/Colas-Dobles-y-Circulares>. [Accessed: 02-Mar- 2017].
- [3]"Data Structures - GeeksforGeeks", *GeeksforGeeks*, 2017. [Online]. Available: <http://www.geeksforgeeks.org/data-structures/>. [Accessed: 02- Mar- 2017].
- [4]"Data Structures - University of California, San Diego, Higher School of Economics | Coursera", *Coursera*, 2017. [Online]. Available: <https://es.coursera.org/learn/data-structures>. [Accessed: 02- Mar- 2017].
- [5]"Data Structures - Wikibooks, open books for an open world", *En.wikibooks.org*, 2017. [Online]. Available: https://en.wikibooks.org/wiki/Data_Structures. [Accessed: 02- Mar- 2017].
- [6]"Compilador - Notacion Infija, Postfija,Perfija y Polaca", *Compilador.wikispaces.com*, 2017. [Online]. Available: <https://compilador.wikispaces.com/Notacion+Infija,+Postfija,Perfija+y+Polaca>. [Accessed: 02- Mar- 2017].