



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

ESTRUCTURAS DE DATOS

**LISTAS CIRCULARES SIMPLEMENTE
ENLAZADAS**

PRÁCTICA 6

**HERNÁNDEZ CASTELLANOS CÉSAR URIEL
MARTÍNEZ ISLAS MAURICIO JOEL**

1CV7

22/04/2017

BENJAMIN LUNA BENOSO



Índice

Resumen.....	3
Introducción.....	3
Experimentación y resultados.....	8
Pseudocódigo.....	9
Conclusiones.....	10
Bibliografía.....	12

Resumen.

En el presente reporte se muestra la documentación de la práctica número seis, cuya función principal es la siguiente:

Dar solución al problema de Josephus, el cual establece que un grupo de soldados se encuentra rodeado por la fuerza enemiga. No hay esperanza de victoria sin refuerzos, pero solamente hay un caballo para escapar. Los soldados hacen un pacto para determinar cuál de ellos va escapar para pedir ayuda. Forman un círculo y se elige un número n y un nombre. Iniciando en nombre, y en sentido de las manecillas del reloj, se cuenta hasta n y el soldado se retira del circulo, y la cuenta continua en el soldado siguiente. El proceso sigue hasta que solo quede el soldado que escapa.

Introducción.

Las listas enlazadas fueron desarrolladas en 1956 por Cliff Shaw y Herbert Simon en RAND Corporation como una estructura de datos de su lenguaje de procesamiento de la información (IPL), las listas fueron usados por los autores para el desarrollo de distintos programas de inteligencia artificial

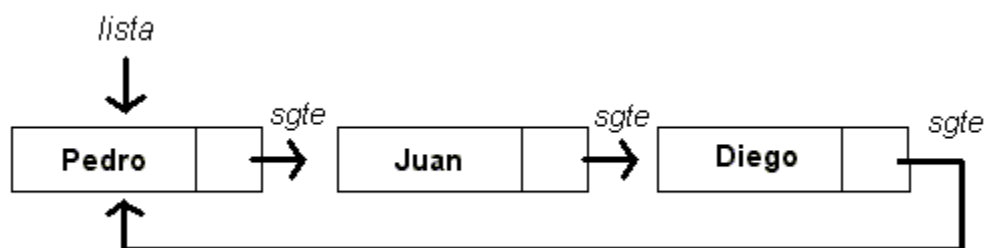
Las listas enlazadas se tratan de estructuras de datos similares a los array, sin embargo el acceso en las listas enlazadas se realiza con un puntero y no con un índice, la asignación de la memoria se realiza durante la ejecución del programa.

En cambio, mientras que en un array los elementos están de manera secuencial en la memoria en una lista se encuentran dispersos, el enlace entre los nodos se realiza por un puntero.

El puntero siguiente del último elemento apunte hacia NULL, lo que indica el final de la lista.

Tipos de listas enlazadas:

- Listas simplemente enlazadas.
- Listas doblemente enlazadas.
- Listas enlazadas circulares.
- Listas enlazadas simples circulares.
- Listas enlazadas doblemente circulares.



Lista circular simplemente enlazada.

Para poder acceder a un elemento específico de la lista, la lista es recorrida comenzando por el inicio, el puntero siguiente permite el cambio hacia el próximo elemento.

El desplazamiento únicamente se realiza en una dirección del primero al último elemento, esto para las LSE, en cambio si queremos recorrer la lista hacia delante y hacia atrás tendremos que recurrir a las listas doblemente enlazadas.

Aplicaciones de las listas enlazadas.

Las listas enlazadas se usan como módulos para otras estructuras de datos, como lo son las pilas, colas y sus variaciones.

En algunas ocasiones, las listas enlazadas se usan para implementarse con árboles binarios de búsqueda equilibrados, Sin embargo, algunas veces la lista enlazada es dinámicamente creada fuera de un subconjunto propio de nodos semejante a un árbol, y son más eficientes para recorrer diversos datos.

Ventajas de las listas enlazadas respecto a los arrays.

1- Tamaño dinámico: Lo que implica optimización de la memoria.

Desventajas de las listas enlazadas respecto a los arrays.

- 1- El acceso es secuencial (para llegar a una posición deberemos pasar por todas las anteriores). Esto significa lentitud. Imaginad por un momento el tinglado que tendríamos que montar para ordenar una lista enlazada. Si buscamos un elemento de una lista ordenada también tardaremos más, no vale la búsqueda dicotómica que vimos en la Ampliación 1 de C (métodos de clasificación en memoria dinámica).

- 2- El código se complica.

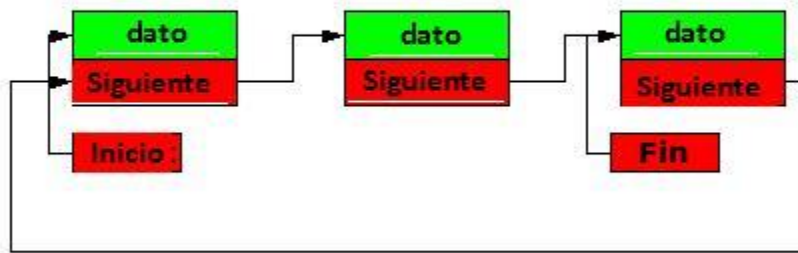
Lista circular.

La lista circular se asemeja a una lista enlazada simple o doblemente enlazada, con una única diferencia para el desplazamiento al interior de la lista: esta no tendrá fin.

Para que sea posible que la lista circular no tenga fin, el puntero siguiente del último elemento apuntará hacia el primer elemento de la lista en lugar de apuntar a nulo, lo que ocurre en las listas simplemente enlazadas y dobles.

En las listas circulares, no se llega a un fin. Cuando se llega al último elemento, el desplazamiento volverá a comenzar.

Lista circular



Lista circular.

¿Por qué utilizar una estructura circular?

En una lista simplemente enlazada, el movimiento siempre fluirá desde la cabeza en dirección hacia el final de la lista, pero ¿Qué ocurre cuando desde el último nodo se necesita operar con el primero?, este es el punto de diferencia entre una estructura abierta y una cerrada.

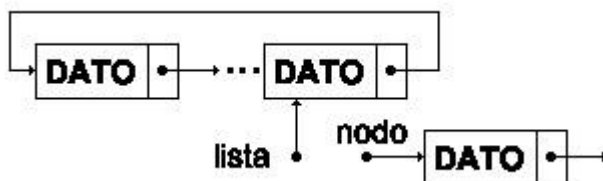
Operaciones básicas con listas circulares.

A todos los efectos, las listas circulares son como las listas abiertas en cuanto las operaciones que se pueden realizar sobre ellas.

- Añadir o insertar elementos.
- Borrar elementos.
- Moverse a través de la lista, siguiente.

Añadir elemento en una lista circular.

Partiremos de un nodo a insertar.



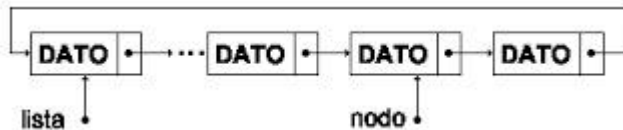
El proceso es el siguiente.

- 1- Hacemos que nodo->siguiete apunte a lista->siguiete.

2- Después que lista->siguiente apunte a nodo.

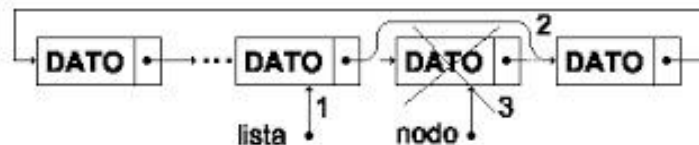


Eliminar un nodo en una lista circular con más de un elemento.



Lista con más de un elemento

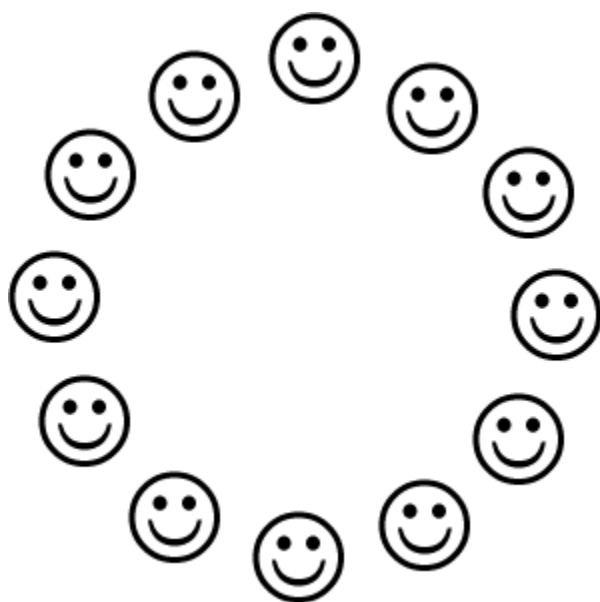
- El primer paso es conseguir que lista apunte al nodo anterior al que queremos eliminar. Esto se consigue haciendo que lista valga lista->siguiente mientras lista->siguiente sea distinto de nodo.
- Hacemos que lista->siguiente apunte a nodo->siguiente. Eliminamos el nodo.



Problema de Josephus.

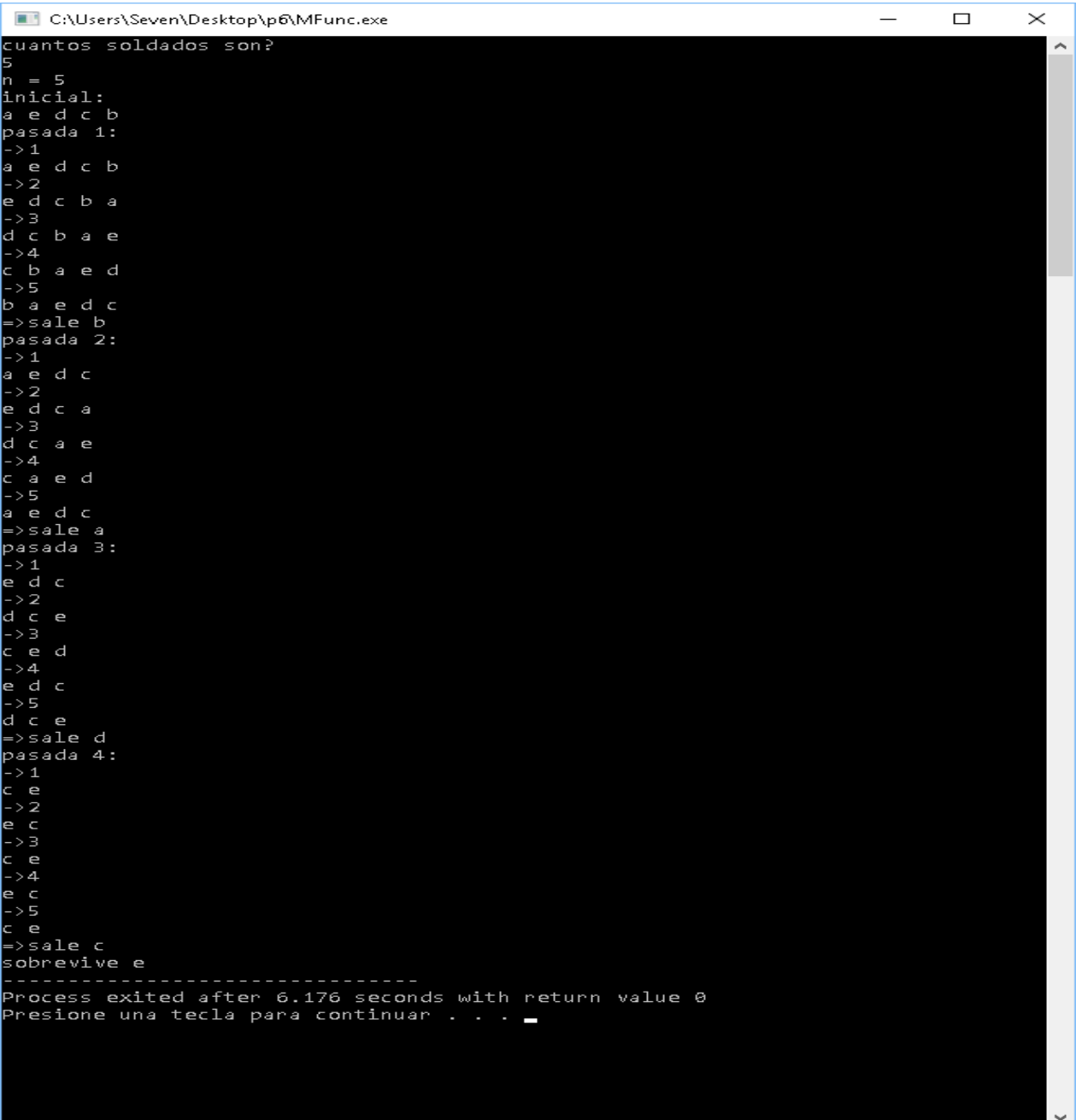
Josephus toma su nombre de un historiador judío del primer siglo llamado Flavius Josephus. Lo que busca el problema de Joseph es la única persona que sobreviviría de entre un grupo de n personas si cada m son ejecutados.

La leyenda relata que durante una guerra contra los romanos, Joseph y 40 soldados fueron rodeados por el ejército romano. El grupo decidió morir antes que rendirse ante el enemigo pero Joseph y otro amigo no estaban de acuerdo con esta idea. El método que escogieron para suicidarse fue el formarse en un círculo e irse matando cada tercer persona. Joseph y su amigo escogieron ciertas posiciones para quedar últimos.



Experimentación y resultados.

La práctica consiste en llevar a cabo una aplicación que de resolución al problema de Josephus por medio de una lista circular simplemente enlazada.



```
C:\Users\Seven\Desktop\p6\MFunc.exe
cuantos soldados son?
5
n = 5
inicial:
a e d c b
pasada 1:
->1
a e d c b
->2
e d c b a
->3
d c b a e
->4
c b a e d
->5
b a e d c
=>sale b
pasada 2:
->1
a e d c
->2
e d c a
->3
d c a e
->4
c a e d
->5
a e d c
=>sale a
pasada 3:
->1
e d c
->2
d c e
->3
c e d
->4
e d c
->5
d c e
=>sale d
pasada 4:
->1
c e
->2
e c
->3
c e
->4
e c
->5
c e
=>sale c
sobrevive e
-----
Process exited after 6.176 seconds with return value 0
Presione una tecla para continuar . . . _
```

Ejecución del programa que da solución al problema de Josephus.

Pseudocódigo.

```
Funcion soldado <- josephus( NODO** cab, int ns )
    int n = generar no. aleatorio a partir de ns
    int i, j, stop=0
    NODO* aux=*cab, delAux
    mientras stop = 0 Hacer
        imprimir "pasada" + j
        Para i<-0 Hasta n-1 Con Paso 1 Hacer
            imprimir i+1
            imprimir lista de soldados empezando en aux
            si i <> n-1 entonces
                aux = aux->sig
            FinSi
        Fin Para
        si !(*cab==aux y *cab==aux->sig) entonces
            delAux = aux->sig
            Imprimir "sale" + aux->dato
            eliminar aux de lista
            aux = del aux
        FinSi
        si *cab==aux y *cab==aux->sig entonces
            stop = 1
        FinSi
        j++;
    FinMientras
Fin Funcion
```

Conclusiones.

Martínez Islas Mauricio Joel.

El diseño volvió a ser un poco inconsistente gracias a querer ahorrar un poco de líneas, pero de ahí en fuera si se mantuvo limpio durante toda la elaboración. Ahora, lo que puede ser que haya resultado un poco mal es la manera en la que se hicieron los detalles (o sea cosas tan simples como cuando parar un ciclo, si usar banderas o no, si se deberían de elaborar tantos casos específicos, falta de generalización, cosas así), y francamente no salieron resultados muy agradables en ese sentido. Si sirve el código, pero simplemente la calidad es algo que falla mucho, a tal grado que se cuestiona si es usable o no.

También salió otro detalle muy notable. Es la manera en la que se generan los números aleatorios. Se sabe que el programa no está hecho para generar llaves, ni cosas así en donde sí se necesitaría de un numero aleatorio "seguro", pero aun así cabe destacar que la manera en la que se genera el numero aleatorio está mal y no se debería de hacer así nunca, nosotros, simplemente porque el problema no lidia con cosas que requieran de seguridad extrema, lo hicimos con srand(), pero pues simplemente es algo que no se hace.

Tenemos que mencionar también el hecho de que puede haber o no banderas que no sean necesarias, esto es un descuido de nuestra parte y si se puede llegar a ver como algo crítico, ya que no se está vigilando la simplicidad y limpieza de los algoritmos (aunque no tenga que ver mucho con la simplicidad, pero sí en que es una excusa fácil para salir de un ciclo). Lo último es que ciertas funciones no están muy bien nombradas, pero pues es lo mejor que se nos ocurrió.

Hernández Castellanos César Uriel.

Las listas circulares suponen una gran ventaja con respecto a las vistas en prácticas anteriores, ya que su ventaja radica en que se cuenta con un seguimiento del nodo que inicia y termina la lista, en cambio si se tratara de una lista doblemente enlazada, la lista circular es aún más sencillo pues no existe una dirección de manera única lo que nos brinda la posibilidad de implementar con mayor seguridad algoritmos de búsquedas.

En cuanto a la práctica, considero que resulto un ejercicio bastante interesante, ya que el problema tiene detrás una historia que realmente aconteció, lo que nos ayuda a salirnos un poco de contexto de la ingeniería, lo cual nos cita a aprender de más ramas del conocimiento.

En cuanto al desarrollo, considero fue bueno, como se pudo cumplir cada uno de los puntos que consideraba la práctica, además de que las listas circulares simplemente enlazadas nos simplificaron el problema de manera importante.

Bibliografia

- [1]A. Aho, J. Hopcroft and J. Ullman, *Estructuras de datos y algoritmos*, 1st ed. México: Addison Wesley Longman, 1998.
- [2]Y. Langsam, M. Augenstein and A. Tenenbaum, *Estructuras de datos con C y C++*, 1st ed. Pearson Prentice Hall, 2000.
- [3]"Data Structures - GeeksforGeeks", *GeeksforGeeks*, 2017. [Online]. Available: <http://www.geeksforgeeks.org/data-structures/>. [Accessed: 22- Apr- 2017].
- [4]"Data Structures - University of California, San Diego, Higher School of Economics | Coursera", *Coursera*, 2017. [Online]. Available: <https://www.coursera.org/learn/data-structures>. [Accessed: 22- Apr- 2017].
- [5]"Las listas circulares", *CCM*, 2017. [Online]. Available: <http://es.ccm.net/faq/2972-las-listas-circulares>. [Accessed: 22- Apr- 2017].
- [6]"Josephus Problem -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2017. [Online]. Available: <http://mathworld.wolfram.com/JosephusProblem.html>. [Accessed: 22- Apr- 2017].