	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC, LCS	2009-2	12098	Algoritmos y Estructura de Datos

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
1	NOMBRE DE LA PRÁCTICA	Arreglos	2 horas

1. INTRODUCCIÓN

La ciencia informática enfatiza dos tópicos importantes: las estructuras de datos y los algoritmos. Estos tópicos son importantes porque las elecciones que usted haga para las estructuras de datos y los algoritmos de un programa afectarán al uso de la memoria (las estructuras de datos) y al tiempo del procesador (los algoritmos que interactúan con esas estructuras de datos). Cuando utiliza una estructura de datos o un algoritmo algunas veces descubre una relación inversa entre la utilización de memoria y el tiempo de CPU: cuanto menos memoria utiliza una estructura de datos, más tiempo de CPU necesitan los algoritmos asociados para procesar los *items de datos* de la estructura, que son valores de tipos primitivos u objetos, mediante referencias. De igual forma, cuanta más memoria utilice una estructura de datos, menor tiempo de CPU necesitan los algoritmos asociados y el procesamiento de los ítems de datos es mucho más rápido.

Una de las estructuras de datos de mayor importancia son los arreglos, los cuales nos permiten hacer un almacenamiento temporal en la memoria ram y de esta manera podemos aplicar una serie de métodos de ordenación y búsqueda en el grupo de datos que estemos manejando, por esa razón iniciaremos el curso con algunos ejercicios donde podamos aplicarlos y reafirmar el conocimiento que ya tenemos sobre ellos.

2. OBJETIVO (COMPETENCIA)

Desarrollar programas de aplicación dentro del paradigma de la programación orientada a objetos, utilizando, clases, arreglos de objetos y la clase vector, para elaborar programas más eficientes en lo que respecta a la utilización de la memoria y las estructuras de datos, con actitud analítica y ordenada.

3. FUNDAMENTO

Los algoritmos requieren ser una representación apropiada de los datos para lograr ser eficientes.
Esta representación junto con las operaciones permitidas se llaman --estructuras de datos--.
Típicamente todas la estructuras de datos permiten inserciones y borrados arbitrarias. Las estructuras de datos varían en cuanto al acceso de los datos.
Las clases y los objetos.
El corazón de la programación orientada objetos es el objeto. Un objeto es un tipo de datos con estructuras y estado. Cada objeto tiene definidas operaciones que pueden acceder a ese estado o manipularlo.
El agrupamiento de datos y las operaciones que se aplican sobre ellos, forman un agregado, mientras que el ocultamiento de los detalles del agregado, se conoce como encapsulación.
Las clases son construidas en un lenguaje de programación. Las clases son representaciones abstractas del mundo real y son encapsuladas.
Un objeto es una instancia de una clase. Un objeto o instancia es creada antes de ser utilizada en el programa. Usted puede crear una instancia y utilizarla con la palabra new .
La clase es básica y fundamental en la programación y en java es declarada siguiendo la siguiente sintaxis.

```
class Nombre_clase{
//sentencias de java
}
```

Para craer una instancia de tipo clase se realiza de la siguiente manera:
Nombre_de_la_clase nombre_de_la_variable **= new** nombre_de_la_clase()

Arreglos

El arreglo es una de las estructuras de datos más ampliamente utilizada por su flexibilidad para derivar en complejas estructuras de datos y su simplicidad. Empezaremos con una definición: *un **arreglos** es una secuencia de elementos, donde cada **elemento** (un grupo de bytes de memoria que almacenan un único ítem de datos) se asocia con al menos un **índice** (entero no-negativo).* Esta definición lanza cuatro puntos interesantes:

- Cada elemento ocupa el mismo número de bytes; el número exacto depende del tipo de datos del elemento.
- Todos los elementos son del mismo tipo.
- Tendemos a pensar que los elementos de un arreglo ocupan localizaciones de memoria consecutivas. Cuando veamos los arreglos bi-dimensionales descubrirá que no es siempre así.
- El número de índices asociados con cada elemento es la *dimensión* del arreglo.

Arreglos de Una Dimensión

El tipo de arreglo más simple tiene una dimensión: cada elemento se asocia con un único índice. Java proporciona tres técnicas para crear un arreglo de una dimensión: usar sólo un inicializador, usar sólo la palabra clave new, y utilizar la palabra clave new con un inicializador.

Nombre_de_la_clase[] nombre_de_la_variable = **new** nombre_de_la_clase[tamaño]

CLASE VECTOR

Algunas veces deseamos guardar objetos en un arreglo pero no sabemos cuántos objetos vamos a guardar.
Una solución es la de crear un arreglo cuya dimensión sea más grande que el número de elementos que necesitamos guardar.
Pero mientras un arreglo es de cierto tamaño dado, un objeto de tipo *Vector* puede dinámicamente crecer y decrecer conforme se vaya necesitando.
La clase **Vector** es parte del paquete **java.util** de la librería estándar de clases de Java.
Ofrece un servicio similar a un arreglo, ya que se pueden almacenar y accesar valores y referencias a través de un índice.
A diferencia de un arreglo, un **Vector** no está declarado para ser de un tipo particular.
Un elemento puede insertarse y eliminarse de una posición específica a través de la invocación de un sólo método.

Un objeto de tipo **Vector** maneja una lista de referencias a la clase **Object**, así no pueden almacenarse tipos de datos primitivos.

Para usar la clase Vector hemos de poner al principio del archivo del código fuente la siguiente sentencia import

•**import java.util.Vector;**

Cuando creamos un vector u objeto de la clase Vector

Vector vector=new Vector();

ALGUNOS METODOS DE LA CLASE VECTOR

Vector ( ) Constructor: crea un vector inicialmente vacío

void addElement (Objet obj) Inserta el objeto especificado al final del vector

void setElementAt (Object obj, int indíce) Inserta el objeto especificado en el vector en la posición específicada

void removeElementAt (int indíce) Elimina el objeto especificado en el índice del vector

void clear ( ) Elimina todos los objetos del vector

Object elementAt (int indíce) Regresa el componente en el índice especificado

boolean isEmpty ( ) Regresa verdadero si el vector no contiene elementos

int size ( ) Regresa el número de elementos en el vector

int size ( ) Regresa el número de elementos en el vector

EJEMPLO DE LA CLASE VECTOR

```
import java.util.Vector;

//  Demuestra el uso de un objeto de la clase Vector

public class Beatles {
   public static void main ()     {
      Vector band = new Vector ();
      band.addElement ("Paul");
      band.addElement ("Pete");
      band.addElement ("John");
      band.addElement ("George");
      System.out.println (band);
      band.removeElement ("Pete");
      System.out.println (band);
      System.out.println ("En la posición 1 está: " + band.elementAt (1));
      band.insertElementAt ("Ringo", 2);
      System.out.println ("Tamaño de la banda: " + band.size ());
      for (int i = 0; i < band.size (); i++)
         System.out.print ( band.elementAt (i) + " ");
   }
}
```

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	**MATERIAL DE APOYO**

Lunes

Programa 1.
El alumno realizará el siguiente programa utilizando arreglos de objetos.

Escriba un programa para lleva un registro de los maestros de una facultad, para cada maestro se deberán registrar los siguientes datos:
Numero de empleado
Nombre del maestro
Fecha en que ingreso a la facultad como docente
Grado académico
Categoría
El programa deberá presentar las siguientes opciones:

a) Registro de los datos (No deberá aceptar el que se repita el numero de empleado)
b) Consultas por numero de empleado
c) Modificaciones por numero de empleado
d) Bajas por numero de empleado
e) Ordenación por nombre del empleado de forma ascendente de la A a la Z.

Programa 2.
El alumno realizará el siguiente programa utilizando la clase Vector.

Escriba un programa que pida una serie de números enteros, obtenga el promedio e imprima de forma separada los números arriba del promedio y los números abajo del promedio.

Martes

Programa 1.
El alumno realizará el siguiente programa utilizando arreglos de objetos

Escriba un programa para llevar el registro de las alumnas de una escuela de arte, para cada alumna se pedirán los siguientes datos:

Numero de control
Nombre de la alumna
Edad
Disciplina, la cual puede ser:
Danza
Pintura
Canto
Piano
Teatro

El programa deberá presentar las siguientes opciones:

a) Registro del alumno
b) Consultas por :
    a. Numero de control
    b. Disciplina
c) Modificaciones
d) Bajas
e) Salir

Programa 2.

El alumno realizará un programa utilizando la clase vector.

Escriba un programa en el cual inserte cadenas de caracteres y con ello presente el siguiente menú de opciones:

a) Mostrar Cadenas que inician con vocal
b) Mostrar Cadenas que inician con consonante
c) Eliminar las cadenas que no inicien ni con vocal ni con consonante
d) Mostrar todas las cadenas

Miércoles

Programa 1.
El alumno realizará el siguiente programa utilizando arreglos de objetos

Escriba un programa para llevar el registro de las películas de una videoteca particular, para cadea película deberá registrar los siguientes datos:
Nombre de la película
Nombre del director
Categoría
Duración en minutos

El programa deberá presentar las siguientes opciones:

a) Registro de la película
b) Consulta por nombre de la película
c) Bajas pro nombre de la película
d) Salir

Programa 2.

El alumno realizará un programa utilizando la clase vector.

Escriba un programa que funcione como un diccionario Ingles Español, en donde se presentes las siguientes opciones:

a) Agregar una nueva palabra
b) Modificar palabra
c) Consultar el significado de la palabra
d) Eliminar una palabra del diccionario.

Jueves

Programa 1.

El alumno realizará el siguiente programa utilizando arreglos de objetos

Escriba un programa para llevar el registro de los artículos en una papelería, para cada uno de los artículos se registraran los siguientes datos:

Numero de inventario del artículo
Descripción
Cantidad en existencia
Costo unitario

El programa deberá presentar las siguientes opciones:

a) Registro del articulo
b) Baja del artículo, la cual solo se podrá llevar a cabo si no hay ya artículos en existencia
c) Venta del articulo
d) Consulta por numero de inventario
e) Consulta general ordenada
f) Salir

Programa 2.

El alumno realizará un programa utilizando la clase vector.

Escriba un programa que capture una serie de números, obtenga el número mayor y el número menor e imprima cuantas veces se repite cada uno si es que estos ocurren.

Viernes

Programa 1.

El alumno realizará el siguiente programa utilizando arreglos de objetos

Escriba un programa para llevar el registro de los pacientes que acuden a un centro de nutrición, pare cada paciente se deberán registrar los siguientes datos:

Nombre del paciente
Sexo
Edad
Peso al ingresar al programa
Peso ideal

El programa deberá presentar las siguientes opciones:

a) Registro de datos del paciente
b) Consulta por nombre del paciente
c) Actualización, en esta opción se deberá actualizar el peso del paciente de acuerdo a los kilos que este aumento o disminuyo desde la cita anterior.
d) Baja, cuando termine el programa.
e) Salir.

Programa 2.

El alumno realizará un programa utilizando la clase vector.
Escribir un programa que presente las siguientes opciones:

a) Captura de cadenas de caracteres
b) Eliminar las cadenas repetidas
c) Mostrar las cadenas capturadas
d) Salir.

C)	CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC, LSC	2009-2	12098	Algoritmos y Estructuras de Datos

PRÁCTICA No.	LABORATORIO DE	Ingeniero en computación Licenciado en Sistemas Computacionales	DURACIÓN (HORA)
2	NOMBRE DE LA PRÁCTICA	Pilas	2

1. INTRODUCCIÓN

Las pilas son de estructuras de datos que son utilizadas por muchos programas de aplicación para almacenar temporalmente datos. Es importante comprender el funcionamiento de las pilas para comprender el funcionamiento interno de otras aplicaciones que hacen uso de ellas

2. OBJETIVO (COMPETENCIA)

Elaborar un programa que utilice pilas en la solución de un problema

3. FUNDAMENTO

La pila es una estructura lineal que recibe datos por un extremo y los retira por el mismo extremo, siguiendo la regla <<*ultimo elemento en entrar, primero en salir*>> La estructura se conoce como LIFO(Last-IN Firts-OUT).

El comportamiento de una Pila en estructura de datos es similar al comportamiento de una pila en el mundo cotidiano. Esto es, si se tiene una pila de diez libros, y se desea remover el quinto libro, se deben remover primero aquellos que se encuentran sobre el libro deseado. De lo contrario, la pila de libros corre el riesgo de derrumbarse. De manera similar, si se desea agregar un libro, esto se hace colocando el libro nuevo sobre el último libro que se agregó.

Representación de una pila

	MAX
CCC	TOPE
BBB	
AAA	

PILA

AAA	**BBB**	**CCC**			
		TOPE			MAX

En esta practica, usted implementará una clase Stack en la que se podrán almacenar hasta 100 elementos. Con esta clase, usted resolverá un problema de conversión de formatos de ecuaciones. Posteriormente, usted reescribirá su programa para que ahora utilice la clase Stack que es parte del API de Java. De esta manera, tendrá la oportunidad de reutilizar código que fue escrito por otra persona a la vez que usted comprende que se requiere para implementar su propia pila. Ambas habilidades son importantes debido a que se debe tener la habilidad para utilizar código escrito por terceros así como generar su propio código.

Esta es la clase Stack ya creada.

```
import java.util.EmptyStackException;

public interface Stack
{
   public boolean isEmpty();
   public void push( Object o );
   public Object pop() throws EmptyStackException;
    public void clear();
}
```

CONDICIONES QUE SE TIENEN QUE APLICAR EN LAS PILAS

DESBORDAMIENTO(overflow)

Si la pila estuviera llena y se intenta insertar un nuevo valor se produce un error llamado DESBORDAMIENTO(overflow). Debido a que TOP=MAX .

SUBDESBORAMIENTO (underflow)

Si se trata de eliminar un elemento y la pila esta vacía produce un error llamado SUBDESBORAMIENTO (underflow).

Operaciones de código que debe tener toda pila:

IsEmpty es el método llama en el método pop para comprobar si esta vació
Entonces regresa true
Sino regresa false

Pop es un método utilizado en pilas para eliminar elementos.
Este método primero debe checar si la pila esta vacía
Si esta vacía manda un mensaje y sale de la subrutina
Si no esta vacía saca el elemento, decremento y retorna el elemento.

Push. Es el método utilizado en las pilas para insertar elementos
Este método primero comprueba si la pila está llena
Si está llena manda un mensaje y sale de la subrutina
Si no está llena, incrementa y agregar el elemento.

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	**MATERIAL DE APOYO**
Computadora con sistema operativo Linux y el entorno de desarrollo integrado Eclipse	Práctica de laboratorio impresa.

B) DESARROLLO DE LA PRÁCTICA

Ejercicio para realizar todos:

Realizar un ejercicio donde emplee una pila con un arreglo donde trabaje 10 valores. Implemente los métodos push, pop y IsEmpty. Para entender claramente el funcionamiento de una pila.
Al final están ejercicio que el maestro decide cuales deberá realizar el alumno.
Todos los ejercicios deberán realizarse con interfaz grafica.

Lunes
En el estacionamiento de la UABC tienen un solo carril para aceptar hasta 15 automóviles. Únicamente hay una entrada/salida al estacionamiento en un extremo del carril. Si llega un estudiante para remover un automóvil que no está cerca de la de la salida, todos los carros que bloquean su ruta se quitan, se remueve el auto del alumno y los autos vuelven a ingresar en el mismo orden que estaban. Esto lo controlamos por medio del número de placas de cada auto. Se debe suponer que cada auto llega y sale en el mismo orden especificado de entrada. Cuando llega un automóvil debe mostrar un mensaje para saber si hay lugar para estaciona el auto. Si no hay espacio el automóvil se va sin enterar. Cuando sale un automóvil el mensaje debe incluir también cuantas veces se movió el automóvil.

Martes
Genere un arreglo de 15 valores, los valores a insertar, generarlos por medio de la función random, al generar un valor lo debe presentar en pantalla y después, por medio de una opción tendrá que aceptarlo o rechazarlo al momento de insertarlo en la pila debe emplear los métodos requeridos por las estructura de pila. Una vez creada la pila genera un procedimiento que permita copiar estos elementos a una pila origen a otra pila destino, manteniendo el orden de la pila original en la destino (se recomienda utilizar una pila auxiliar)
Miercoles
Por medio la estructura de pilas crear un programa para calcular el factorial de un valor dado.
Por ejemplo el factorial de 5 tendría que crear varias pilas.

5
4
3
2
1

20
3
2
1

60
2
1

120
1

120

Jueves
Desarrollar un programa que extraiga N elementos de una pila si esta contiene N elementos, o hasta que la pila se vacié. Los N elementos debe devolverse en una pila y colocarlos en el orden inverso en el que estaban en la pila original.

Viernes
Desarrollar un programa donde después de haber capturado 10 valores enteros indique si ambas pilas son iguales. Se entiende por dos pilas iguales cuando todos los elementos son idénticos y aparecen en el mismo orden.

A consideración del docente.
1. Crear una clase de nombre *Stack* que permita almacenar hasta 100 elementos en ella. Implementar los siguientes métodos: *push(elem), pop(), isEmpty(), getTop().* Escribir un pequeño programa de prueba para comprobar su funcionamiento. Usted debe implementar la clase completamente, no podrá heredar de la clase *Stack* que es parte del API de Java.
2. Utilizar la clase *Stack* que desarrolló en el paso anterior para implementar el programa que le fue asignado y que está descrito abajo. El programa deberá tener una interfaz de usuario como la de la Figura 1.
3. Utilizar la clase *Stack* que es parte del API de Java para implementar el programa que le fue asignado y que está descrito abajo. El programa deberá tener una interfaz de usuario como la de la Figura 1.

Figura 1: Interfaz de usuario.

A.-Elaborar un programa que utilice una pila para convertir una expresión aritmética de formato postfijo a prefijo.

B.- Elaborar un programa que utilice una pila para convertir una expresión aritmética de formato infijo a prefijo.
Ejemplos:

Interfija	Prefija
A+B	+AB
A+B-C	-+ABC
(A+B)*(C-D)	*+ AB-CD

C.- Elaborar un programa que utilice una pila para convertir una expresión aritmética de formato interfija a postfijo.
Ejemplos:

interfija	postfija
A+B	AB+
A+B-C	AB+C-
(A+B)*(C-D)	AB+CD-*

D.- Elaborar un programa que utilice una pila para convertir una expresión aritmética de formato prefijo a postfija

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC, LSC	2009-2	12098	Algortimos y Estructuras de Datos

PRÁCTICA No.	LABORATORIO DE	Algoritmos y Estructuras de Datos	DURACIÓN (HORA)
3	NOMBRE DE LA PRÁCTICA	Recursividad	1 hora

1 INTRODUCCIÓN

La recursividad es un tema que es muy común que se utilice en el área de la programación, en donde se utilizan ciclos en lugar de realizar auto llamadas al mismo método, sin embargo dentro de algoritmos y estructuras de datos se utiliza frecuentemente para la implementación de las diferentes estructuras.

2 OBJETIVO (COMPETENCIA)

El alumno utilizara la recursividad en programas para resolver problemas y analizar los efectos de la recursividad en los programas para poder determinar los casos en los que es apropiado el uso de la misma.

Formuló	Revisó	Aprobó	Autorizó
M.C. Gloria Etelbina Chavez Valenzuela M.I María Luisa González Ramírez	M.C. Gloria Etelbina Chavez Valenzuela		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

¿Que es la recursión?

Un método recursivo es un método que, directa o indirectamente, se hace una llamada a sí mismo. Esto puede parecer un círculo vicioso, ya que el fin lo determina la condición que se le asigne.

La recursión es una herramienta muy potente de resolución de problemas. Muchos algoritmos se pueden expresar más fácilmente usando una formulación recursiva. Y lo que es más, hay muchos problemas cuyas soluciones más eficientes usan esta formulación recursiva natural. Pero debemos ser cuidadosos para no crear una lógica circular que produzca bloques infinitos.

Para realizar un método recursivo se deben tomar en cuenta la siguiente:

1ro. Debe haber una condición que se debe cumplir en algún momento para terminar.

2do. Cualquier llamada a un método recursivo debe progresar a obtener la condición del punto anterior.

Ejemplos métodos con recursividad.

```
// Evalúa la suma de los n primeros enteros.

public static long suma(int n)
{
     if (n == 1)
          Return 1;
     else
          return suma( n -1 ) +n;
}
```

Suponiendo que n =3
Primero revisará si n es igual a 1 y esto resulta falso
Como fue falso se autollama de nuevo al método suma y le manda (3-1) .

Ahora n=2
Primero revisará si n es igual a 1 y esto resulta falso
Como fue falso se autollama de nuevo al método suma y le manda (2-1) .

Ahora n=1
Primero revisará si n es igual a 1 como es verdadero, termina y manda un 6.

3 FUNDAMENTO

Otros ejemplos de uso de recursividad es para cuando calculamos Fibonacci.

```
//calculo del n ésimo numero de Fibonacci

public static long fib(int i)
{
      if ( n < = 1)
            return n;
      else
      return fib(n-1)+ fib(n-2);
}
```

Secuencia de Fibonacci.
0,1,1,2,3,5,8,13,21,34.

Implementación recursiva de la función de factorial

```
// Evalúa n!
public static long factorial (int n)
{
      if (n <=1)
            return 1;
      else
            return n*factorial(n-1);
}
```

Secuencia de factorial
4!=4*3*2*1
Para que utilizar la recursividad.

Formato para prácticas de laboratorio

3 FUNDAMENTO

Más adelante usted podrá aplicar recursividad en la búsqueda binaria, en el método de ordenación de Quick sort y en árboles binarios.

También tiene bastante aplicación en comunicaciones, por ejemplo en la codificación y decodificación de mensajes. Tambien en la criptografía de la clave pública, cada participante, publica el código que otros pueden usar para enviarles mensajes codificados, pero se guarda su código secreto para decodificar los mensajes recibidos.

Hay varios factores a considerar en la decisión sobre si usar o no una solución recursiva en un problema. En general la solución no recursiva es más eficiente en términos de tiempo y espacio de computadora. La solución recursiva puede necesitar un considerable gasto de memoria para las múltiples llamadas al método, puesto que deben almacenarse las direcciones de vueltas y copias de las variables locales temporales. Si un programa se ha de ejecutar frecuentemente (como un compilador y/o debe ser eficiente, el uso de la programación recursiva puede ser una buena elección.

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con Linux Instalado y Eclipse.	Practica impresa y apuntes de clase.

B DESARROLLO DE LA PRÁCTICA

4 PROCEDIMIENTO (DESCRIPCIÓN)

Ejercicios.

Todos los laboratorios realizaran el ejercicio de Fibonacci y factorial
Realizarlo hasta un valor dado por el usuario, este valor siempre será establecido por el docente.
Siga las indicaciones de su maestro y realice el programa.

Para todos los programas, utilice recursividad para realizar cada una de las siguientes funciones, integrelas todas en un programa.

Problema 1.

Llene de forma aleatoria un arreglo de 20 elementos y realice las siguiente acciones.
1.- Obtener el elemento máximo de un arreglo.
2.- Sumar los numeros pares de un arreglo
3.- Mostrar el arreglo.
4.- Mostrar los numeros pares del arreglo.
5.- Realice la busqueda lineal.

Problema 2.

Llene con ceros y unos al azar un arreglo de 20 elementos y realice las siguiente acciones.
1.- Cuente cuantos ceros y cuantos unos tiene el arreglo
2.- Cambie los ceros por 2 y los unos por 7
3.- Mostrar el arreglo.
4.- Cree una cadena con los valores de los indices donde se encuentran el valor indicado. Por ejemplo, los indices donde se encuentra el 7 son: 2,11,15,17 y 20.
5.- Ordene los valores de tal forma que en los indices pares esten los unos y en los indices nones los ceros.

Problema 3.

Llene de forma aleatoria un arreglo de 20 elementos y realice las siguiente acciones.
1.- Obtenega el elemento mínimo del arreglo.
2.- Obtenga el promedio de los elementos del arreglo
3.- Calcule el producto de los elementos del arreglo, si hay un cero ese valor no multiplicarlo.
4.- Mostrar el arreglo completo.
5.- Cambie los valores pares a unos.

Problema 4.

Lea una cadena y realice las siguientes acciones.
1.- Cuente cuantas vocales y consonantes tiene la cadena.
2.- Realizar la busqueda lineal de un elemento de la cadena.

B DESARROLLO DE LA PRÁCTICA

3.- Mostrar cada carácter de la cadena de forma recursiva.
4.- Modifique los caracteres que no son letras por un *.
5.- Cambie la cadena para que todas las vocales esten primero que todas las consonantes y al final coloque los *.

Problema 5.
Lea una cadena y realice las siguientes acciones.
1.- Implemente una función que nos diga si la cadena es un palíndromo(cadena simétrica). Por ejemplo "Anita lava la tina"
2.- Invierta la cadena.
3.- Mostrar cada carácter de la cadena de forma recursiva.
4.-Mostrar los caracteres dentro de la cadena que no son ni letras ni numeros.
5.-Lea un numero e identifique si es un número primo o no.

C CÁLCULOS Y REPORTE

5 RESULTADOS Y CONCLUSIONES

Formato para prácticas de laboratorio

6 ANEXOS

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC,LSC	2009-2	12098	Algortimos y Estructuras de Datos

PRÁCTICA No.	LABORATORIO DE	Ingeniero en Computación y Licenciado en Sistemas Computacionales	DURACIÓN (HORA)
4	NOMBRE DE LA PRÁCTICA	Colas	2

1. INTRODUCCIÓN

La estructura de datos conocida como cola, es una de las estructuras de datos más utilizadas.
En esta práctica el alumno, con los conocimientos ya recibidos en clase elaborara programas en donde se aplique el comportamiento de la estructura de datos cola, elaborando la clase y los métodos que esta requiere.

2. OBJETIVO (COMPETENCIA)

El alumno estructurara programas en los que se utilicen colas para guardar datos.

3. FUNDAMENTO

Una cola es una lista lineal en la que los datos se insertan por un extremo (final) y se extraen por el otro extremo (frente). Por lo mismo la cola es una estructura FIFO (First in First out).
El servicio de atención a clientes en un almacén, es un ejemplo típico de una cola.

Desde el punto de vista de estructura de datos, la cola es similar a una pila, en donde los datos se almacenan de modo lineal y el acceso a los datos solo esta permitido por los extremos de la cola.

COLAS IMPLEMENTADAS CON ARRAYS.

Al igual que las pilas, las colas se pueden implementar utilizando arreglos o listas enlazadas. En este caso consideraremos la implementación con arreglos.
La definición de una cola ha de contener un array para almacenar los elementos de esta, y marcadores o punteros para indicar cual es el elemento que sigue en salir y cual es el último que ingreso, el siguiente código marca cada uno de los métodos que se implementaran para el manejo de colas.

Colas
frente : int final : int dato : object[] Max : int
+ Colas() + colaVacia( ) : boolean + colaLlena( ) : boolean + insertar(object dato) + eliminar( )

Algoritmos para la cola lineal son:

- *Creación de una cola vacía*
  Frente y final = -1

- *Verificación de que una cola esta vacía*
  Si final es igual a -1

- *Verificación de que una cola esta llena*
  Si final = Max -1

- *Añadir un dato al final de la cola.*
  Si la cola esta llena
  Mostrar un mensaje "no hay espacio"
  Si no
  Si la cola esta vacía
  Se incrementa frente
  Se incrementa final
  Se almacena el dato en el vector en la posición final

- *Eliminación de los datos de la cabeza de la cola*
  Si la cola esta vacía
  Mostrar un mensaje "no hay elementos para eliminar"
  Si no
  Se incrementa frente

  // para no desperdiciar espacio al eliminar se recorren los elementos
  Si la cola esta vacía
  Mostrar mensaje "no hay elementos para eliminar"
  Si no
  Repetir con x desde 0 hasta final-1
  Asignar en vector posición x el dato del vector en posición x+1
  Decrementar final

4. PROCEDIMIENTO (DESCRIPCIÓN) A) EQUIPO NECESARIO	MATERIAL DE APOYO

Se desarrollara una clase que contenga los métodos básicos para manipular datos en la estructura cola, el atributo dato será de tipo object.
Cada grupo elaborara el problema que indique su maestro, para ello creara una clase con los métodos necesarios donde se implementaran los métodos creados en el punto anterior para el manejo de las colas; así como una aplicación principal para su funcionamiento.

Problema 1.
Una sucursal bancaria cuenta con 3 cajas de atención a clientes. Una para clientes con tarjeta preferencial, otra para clientes con cuenta en ese banco y otra para clientes que no tienen cuenta.
Elabore un programa que simule la llegada de clientes y les pregunte si tienen cuenta o tarjeta preferencial, asignarlo en la fila correspondiente registrando su nombre. Generar un número aleatorio para determinar de cual fila se atenderá el cliente, por cada cliente atendido se consideran 2 minutos. Mostrar la salida de cada cliente con el tiempo que espero. Por ejemplo si se atendieron 2 clientes de la fila 2, después 1 de la fila 1, luego el primero de la fila 3, al salir este cliente se mostrara “Ignacio”, espero 6 minutos.

Problema 2.
Un pequeño mercado cuenta con 2 cajas de pago y con 10 carritos de compra. El cliente al llegar debe esperar a que haya un carrito disponible, al finalizar su compra se coloca en la caja con menos fila, al momento de ser atendido el carrito vuelve a estar disponible para su uso. Representar mediante el uso de colas lineales la fila de carritos y de las cajas. Simular el funcionamiento en el mercado, mediante opciones para a) llegada de cliente (debe tomar un carrito) b) colocarse en una fila para pago c) realizar pago en caja 1 o 2. A Cada cliente se le asignara un número consecutivo conforme a su llegada iniciando en 1000. En cualquier momento se puede consultar el estado de las filas de pago, así como la fila de carritos.

Problema 3.
Elabora un programa que permita eliminar y modificar palabras en un archivo de texto. Cada palabra leída del archivo será almacenada en una posición de la estructura cola y cuando el usuario desee eliminar o modificar una palabra, se guardara en una estructura auxiliar el nuevo archivo. Mostrar como quedo el archivo después de eliminar o modificar e indicar cuantas veces se encontró la palabra indicada. También se deberá guardar una copia de la estructura original.

Problema 4.
Un pequeño centro medico de un poblado cuenta con 50 vacunas diarias para la neumonía, los niños menores de 6 años y adultos mayores de 65 tienen preferencia por lo que al llegar las personas se hacen 3 filas distintas. Para la aplicación de la vacuna se pasan por cada 3 niños a 1 adulto, si al finalizar con estas dos filas, todavía hay vacunas disponibles se aplican a la fila de aquellas personas que están entre los 7 y 64 años de edad.

Elabora un programa que simule la llegada de 60 personas, generando mediante un número aleatorio su edad, debes asignarlos en la fila correspondiente e indica el tamaño de cada fila; después se simulara la atención como se indico anteriormente. Al final mostrar cuantas personas fueron atendidas de cada fila.

Problema 5.

Se realizara un juego para adivinar números, y se determinara mediante porcentaje los puntos obtenidos por el usuario.

En cada posición de la estructura se almacenará un número (el que adivinara), y un porcentaje, ambos valores serán generados aleatoriamente. Los números para adivinar podrán ser entre 1000 y 1500 y el porcentaje entre 5 y 30, el llenado de la estructura terminara cuando los porcentajes sumen mas de 90 y no mas de 100, asignando en caso necesario un ultimo registro con el porcentaje faltante.

El juego consiste en intentar adivinar todos los números generados, si la diferencia entre el número registrado y el valor dado por el usuario es menor a 50 se considera aceptado y se acumula el porcentaje de ese número, tiene dos oportunidades por valor, cada valor se va extrayendo hasta vaciar la lista, al mismo tiempo se va generando otra cola con los datos de la estructura inicial mas el número que da el usuario; finalmente se muestra el porcentaje obtenido y un listado de la segunda estructura generada.

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC y LSC	2009-2	12098	Algoritmos y estructuras de datos

PRÁCTICA No.	LABORATORIO DE	Algoritmos y estructuras de datos	DURACIÓN (HORA)
5	NOMBRE DE LA PRÁCTICA	Uso de listas enlazadas	2

1 INTRODUCCIÓN

Las listas son una forma de representar datos o de guardarlos. La forma más común de listas son las enlazadas. Las listas estan formadas por nodos, la computadora asigna espacio a los nodos dentro de la memoria de manera no secuencial, por el contrario de los arreglos, en donde la asignacion es secuencial. En todas las listas se debe de tener una referencia para controlar la lista, esta referencia generalmente se le llama *inicio*.

Existen diferentes tipos de listas, como listas lineales, listas lineales circulares, listas dobles y listas dobles circulares. El tipo de lista depende de la cantidad de enlaces que contenga y del estado de la ultima referencia de la lista. En esta practica trataremos de las listas lineales

Formato para prácticas de laboratorio

2 OBJETIVO (COMPETENCIA)

El alumno creará programas que resuelvan problemas utilizando como estructura de datos principal las listas doblemente ligadas lineales.

Formuló	Revisó	Aprobó	Autorizó
M.I María Luisa González Ramírez	M. C. Gloria E. Chavez		M.C. Miguel Angel Martínez
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

3 FUNDAMENTO

Las listas son conjuntos de registros conceptualmente contiguos pero que físicamente no tienen porqué estarlo (aunque lo normal es que si lo estén); su orden lógico no tiene nada que ver con el orden físico de los elementos en el conjunto. Esto se consigue porque cada elemento de la lista dispone de una referencia que señala al próximo elemento. De esta forma la lista se mantiene enlazada,y es fácil recorrerla en cualquier sentido .

Una lista lineal sencilla consiste en varios nodos enlazados, cada nodo esta formado por un campo que hace referencia a otro nodo y ademas de un campo de información. Ver figura 1.

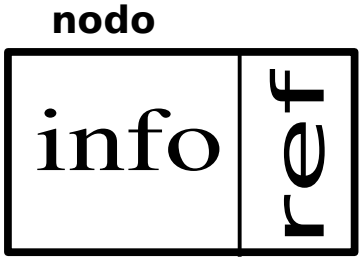


Figura 1

Las listas son controladas por una referencia generalmente llamada *inicio.* Esta referencia debe estar inicializada a null. Con la ejecucion del programa la referencia inicio puede cambiar de valor.

Para añadir un nuevo elemento al final de la lista solo hay que recorrer la lista hasta el último nodo, para que despues insertemos el nuevo nodo.

Formato para prácticas de laboratorio

3 FUNDAMENTO

En java existe la clase LinkedList. Esta clase cuenta con una gran variedad de métodos. A continuación se mencionan algunos de ellos.

1) void `add(int index, Object element)` Inserta el objeto element en la posicion especificada por index.
2) boolean `add(Object o)` Agrega el objeto o al final de la lista.
3) void `addFirst(Object o)` Inserta el objeto al principio de la lista.
4) void `addLast(Object o)` Inserta el objeto al final de la lista.
5) Object `get(int index)` Regresa el elemento que se encuentra en la posicion index de la lista
6) Object `getFirst()` Regresa el primer elemento de la lista.
7) Object `getLast()` Regresa el último elemento de la lista.

Para más referencia consulte
http://java.sun.com/j2se/1.4.2/docs/api/java/util/LinkedList.html

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Equipo de computo con sistema operativo linux	Practica impresa

B DESARROLLO DE LA PRÁCTICA

4 PROCEDIMIENTO (DESCRIPCIÓN)

Utilice la clase LinkedList de java para realizar los programas.
Siga las indicaciones de su maestro para hacer el o los programas que le indiquen

1.- Hacer un programa que lea un archivo y almacene todas las palabras en una lista sin que haya palabras repetidas. Tener las siguientes opciones en un menú.

- Agregar palabras. Se pedirá la palabra nueva al usuario
- Mostrar toda la lista ordenada.
- Eliminar palabra.
- Eliminar las palabras que comiencen en x carácter. Se pedirá el carácter al usuario.
- Almacenar en un archivo. Deberá guardar todos los elementos de la lista en un archivo, el nombre del archivo se pedirá al usuario.

2.- Hacer un programa que lleve el control de la renta de casas. Debe de contar con el siguiente menú.

- Altas de casa. Se pedirá, la dirección, la cantidad de habitaciones, cantidad de baños, capacidad de la cochera , el estado general de la casa que tendrá las siguientes opciones: buena, regular, mala y además contará con un campo de EnRenta que será booleano, este campo indicara si está o no en renta, al dar de alta una casa por defecto estará en falso.
- Baja de una casa.Se pedirá al cliente que indique los motivos para darla de baja del banco de rentas y se almacenará en un archivo.
- Mostrar todas las casas rentadas.
- Mostrar todas las casas que no estan rentadas.
- Modificar el estado de EnRenta.

3.-Hacer un programa que almacene en una lista los nombres de los archivos de un directorio indicado por el usuario. Con las siguientes opciones de menú.

- Mostrar longitud del archivo.
- Mostrar los archivos que comiencen con la letra introducida por el usuario.
- Mostrar extension del archivo.
- Borrar un elemento de la lista(OJO.. de la lista, no del sistema)
- Modificar el nombre de un archivo de la lista.

4 PROCEDIMIENTO

4.- Escriba un programa que lea un texto de longitud indeterminada y que almacene las palabras ademas de la frecuencia de aparicion. El programa contará con el siguiente menú.

- Mostrar x palabra con su frecuencia.
- Mostrar las palabras que tengan x frecuencia.
- Eliminar x palabra.
- Eliminar las palabras con x frecuencia.
- Modificar la frecuencia de x palabra.
- Modificar palabra. Si ya se encuentra en la lista deberá incrementar la frecuencia de la palabra.

5.- Utilizando la lista como pila y utilizando sus operaciones basicas, hacer un programa con el siguiente menú.

- Introducir un número.
- Mostrar en binario. Mostrar el número con todo el procedimiento necesario para convertirlo a binario.
- Mostrar en octal. Mostrar el número con todo el procedimiento necesario para convertirlo a octal.
- Mostrar en hexadecimal. Mostrar el número con todo el procedimiento necesario para convertirlo a hexadecimal.

C CÁLCULOS Y REPORTE

5 RESULTADOS Y CONCLUSIONES

6 ANEXOS

Formuló	Revisó	Aprobó	Autorizó
M.I María Luisa González Ramírez	M. C. Gloria E. Chavez		M.C. Miguel Angel Martínez
Maestro	Coordinador de la Carrera	Gestión de la Calidad	Director de la Facultad

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC, LSC	2009-2	12098	Algoritmos y Estructura de Datos

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
6	NOMBRE DE LA PRÁCTICA	Listas Doblemente Ligadas Lineales	2

1. INTRODUCCIÓN

Las estructuras de datos que se manejaran en esta práctica son las listas doblemente ligadas lineales, estas estruc-
turas son muy flexibles y con numerosas aplicaciones en el mundo de la programación.
En esta práctica el alumno implementará una clase que represente una lista doblemente ligada lineal y la utilizará en un programa de aplicación.

2. OBJETIVO (COMPETENCIA)

Realizar programas de cómputo, utilizando listas doblemente ligadas lineales, para optimizar el uso de la memoria en aplicaciones prácticas, con actitud crítica y responsable.

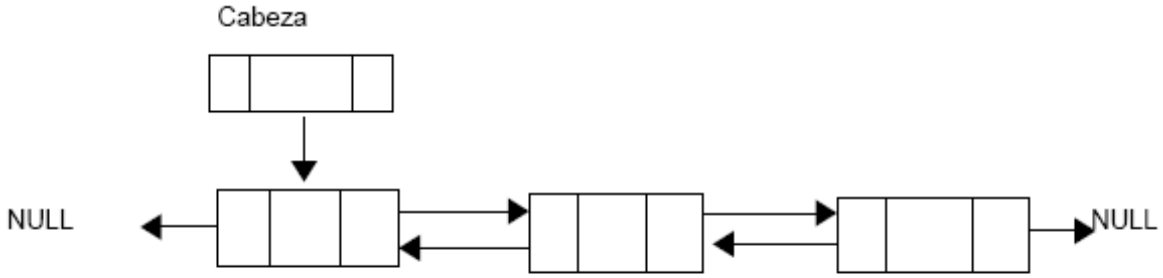
3. FUNDAMENTO

LISTAS DOBLEMENTE LIGADAS LINEALES.

Una lista doblemente ligada es una colección de nodos, en la cual cada nodo contiene dos enlaces, uno a su nodo predecesor y el otro a su nodo sucesor.

La lista es eficiente, ya que a diferencia de las simplemente ligadas tiene un recorrido directo (hacia adelante) y un recorrido inverso (hacia atrás).

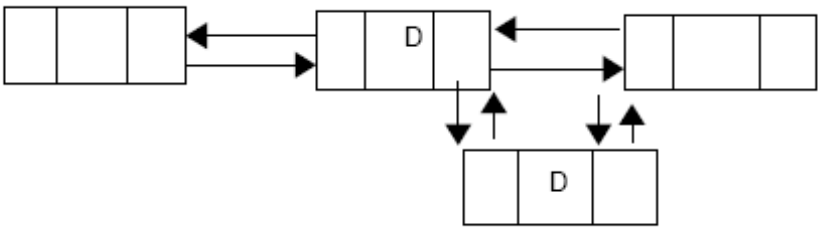
Sin embargo, existen numerosas aplicaciones en las que es conveniente poder acceder a los elementos o nodos de una lista en cualquier orden. En este caso se recomienda de este tipo de lista. En estas cada elementos de ella contiene dos referencias, aparte de el valor almacenado en el elemento. Una referencia apunta al siguiente elemento en la lista y la otra referencia apunta al elemento anterior. Como se muestra en la siguiente figura:



Lista doblemente ligada de tres elementos

Nodo

Existen operaciones de insertar y eliminar (borrar) en cada dirección, la siguiente figura muestra el problema de insertar un nodo p a la derecha del nodo actual. Deben asignarse 4 nuevos enlaces.



El diagrama de clases es de las operaciones básicas con listas doblemente ligadas lineales.

Nodo
Int dato Nodo adelante Nodo atras
Boolean vacia() Void insertarInicio(int) Void insertarFinal(int) Void insertarAntesRef(int int) Void insertDespuesRef(int int) Int borrarInicio() Int borrarFinal() Int borrarReferencia(int) Int borrarAntesReferencia(int) Int borraDespuesReferencia(int) Void mostarLista() OrdenarLista()

4. PROCEDIMIENTO (DESCRIPCIÓN)		
A)	EQUIPO NECESARIO	MATERIAL DE APOYO

Todos los alumnos realizarán las operaciones básicas de listas, además del programa que le corresponda según el día de la semana en el cual tenga la clase de laboratorio.

Lunes
Programa 1.
Escriba un programa utilizando listas doblemente ligadas lineales que realice lo siguiente:

a) Insertar cadenas de caracteres en una lista de manera ordenada
b) Mostrar la cantidad de caracteres que tiene cada una de las cadenas insertadas
c) Eliminar las cadenas de más de 5 caracteres de la lista e insertarlas en una lista adicional.
d) Mostrar el contenido de ambas listas.
e) Eliminar la lista que se indique.
f) Salir.

Martes
Programa 2.
Escriba un programa utilizando listas doblemente ligadas lineales que realice lo siguiente:

a) Insertar estados de la república mexicana con su respectiva capital
b) Hacer un búsqueda por estado y que muestre la capital que le corresponde
c) Mostrar la lista de estados y capitales, ordenadas por nombre del estado
d) Eliminar un estado especifico
e) Terminar.

Miércoles
Programa 3.
Escriba un programa utilizando listas doblemente ligadas lineales que realice lo siguiente:

a) Insertar valores enteros en una lista principal (positivos y negativos), no se podrán insertar 2 valores del mismo signo en forma consecutiva.
b) Separar los valores positivos en una lista y los negativos en otra
c) Mostrar las 3 listas ordenadas
d) Eliminar de la lista de números positivos los valores mayores o iguales a una referencia dada.
e) Eliminar de la lista de los números negativos los valores menores o iguales a una referencia dada
f) Salir.

Jueves
Programa 4.
Escriba un programa utilizando listas doblemente ligadas lineales que realice lo siguiente:

a) Insertar palabras en la lista
b) Mostrar cuantas veces se repite cada una de las palabras que se insertaron en la lista
c) Cuál es la palabra que más veces se repite
d) Eliminar las palabras que no se repiten
e) Mostrar la lista
f) Salir

Viernes
Programa 5.

Un conjunto es una secuencia de elementos iguales(del mismo tipo) sin duplicidades. Escribir un programa que permita crear dos conjuntos, en dos listas diferentes y desplegar los siguientes resultados:

a) Escribir los elementos de cada conjunto
b) La unión de los conjuntos
c) Intersección de los conjuntos
d) Diferencia entre los conjuntos
e) Inclusión de un conjunto en otro.
f) Salir.

C)	CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

	UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA **FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)** **DOCUMENTO DEL SISTEMA DE CALIDAD**

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC,LSC	2009-2	12098	Algoritmos y Estructuras de Datos

PRÁCTICA No.	LABORATORIO DE	Ingeniero en Computación y Licenciado en Sistemas Computacionales	DURACIÓN (HORA)
7	NOMBRE DE LA PRÁCTICA	Listas ligadas circulares	2

1. INTRODUCCIÓN

Las listas ligadas circulares, al igual que las listas lineales, son una colección de nodos, sin embargo, en una lista ligada circular el primer y el último nodo están unidos. Esto se puede hacer tanto para listas enlazadas simples como para las doblemente enlazadas.

.

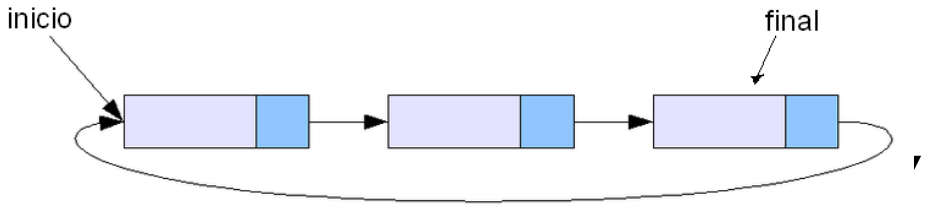
2. OBJETIVO (COMPETENCIA)

El alumno creará programas que resuelvan problemas utilizando como estructura de datos principal las listas ligadas circulares.

3. FUNDAMENTO

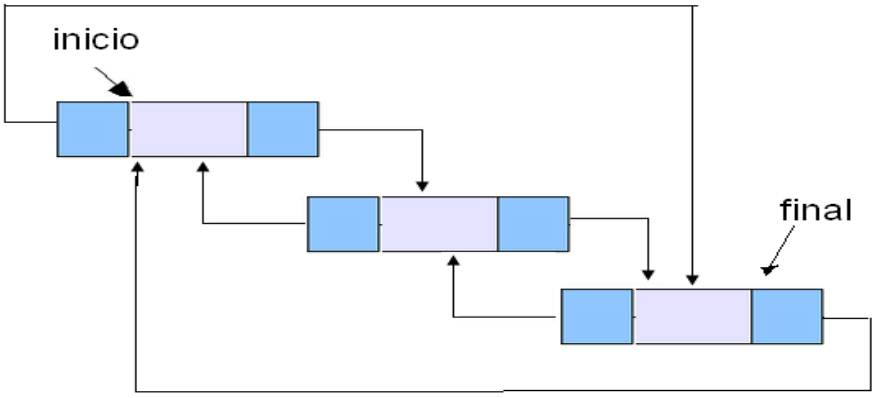
Para la manipulación de las listas se utiliza nodo de referencia que indica la dirección del primer nodo en la lista, también es posible tener un nodo que indique la dirección del ultimo elemento en la lista.
Las listas circulares se consideran eficientes, es importante tener especial cuidado en los ciclos de recorrido ya que puede resultar en un ciclo infinito

Listas simples circulares. Cada nodo tiene un enlace, similar al de las listas enlazadas simples, excepto que el puntero siguiente del último nodo apunta al primero



Formuló	Revisó	Aprobó	Autorizó
L.S.C. Lourdes E. Ramírez Fernández	M.C. Gloria Etelbina Chavez Valenzuela L.S.C. Mónica Lam Mora		M.C. Miguel Ángel Martínez Romero
Maestro	Coordinador de Programa Educativo	Gestión de Calidad	Director de la Facultad

Listas dobles circulares. En este tipo de lista doble, el puntero izquierdo del primer nodo apunta al último nodo de la lista, y el puntero derecho del último nodo apunta al primer nodo de la lista.



ListasCirculares
inicio : Nodo final : Nodo // opcional
+ ListasCirculares () + listaVacia( ) : boolean + insertarInicio(object dato) + insertarFinal (object dato) + insertarAntesRef (object dato, int ref) + insertarDespuesRef (object dato, int ref) + eliminarInicio( ) :object + eliminarFinal( ) :object + eliminarAntesRef (int ref) :object + eliminarDespuesRef (int ref) :object + eliminarRef(int ref) :object + mostrarLista( )

Los punteros inicio y final apuntan a null cuando la lista esta vacía.

4. PROCEDIMIENTO (DESCRIPCIÓN)

A) EQUIPO NECESARIO	MATERIAL DE APOYO

Problema 1.

Elaborar un programa que lea un archivo de texto y almacene cada palabra en un nodo de una lista doblemente ligada circular y realice los siguientes procedimientos:

- insertar al final de la lista.
- indicar cuantas veces se repite ´x´ palabra.
- eliminar el nodo que se encuentra antes de una referencia dada.
- mostrar los datos en orden inverso.
- sustituir una palabra

Problema 2.

Para una lista doblemente ligada circular que almacene la lista final de un grupo, utilizar la clase Alumnos, con los atributos: matricula, nombre y calificación final, realice un procedimiento para cada punto:

- insertar al inicio de la lista.
- organizar los datos en la lista en orden alfabético.
- eliminar un registro, buscar por matricula
- calcular el promedio del grupo.
- Mostrar la lista en columnas

Problema 3.

Se desea emular el comportamiento de un mostrador de una cocina, para ello, implementaremos una lista circular simplemente enlazada, que tiene el funcionamiento de las estructuras FIFO, se utilizara la clase Órdenes con los siguientes atributos: número de mesa y descripción del pedido. Realice un procedimiento para cada punto:

- añadir la orden,
- modificar la orden, buscar por numero de mesa
- eliminar una orden.
- mostrar ordenado por número de mesa. (la lista original no debe ser modificada)

Problema 4.

Se desea llevar un control de las temperaturas diarias en la ciudad, los atributos de la clase Temperaturas son: fecha (dd,mm,yy) y temperatura máxima; utilizar una lista simplemente ligada circular que realice los siguientes procedimientos:

- insertar en forma ordenada en base a la fecha.
- mostrar la temperatura mas alta y la fecha.
- eliminar un registro ´x´.
- mostrar los datos.
- modificar una temperatura.

Problema 5.

Utilizando la clase Llamadas con los atributos: hora, nombre, numero telefónico y tipo (recibida – realizada), realizar un programa que permita llevar el registro de todas las llamadas; implementar en una lista circular los siguientes procedimientos:

- insertar al final.
- mostrar quien llama más veces.
- eliminar después de una referencia.
- separar los registros según el tipo de llamada en dos listas y mostrar cada una de ellas.
- eliminar todos los registros de una persona ´x´.
- mostrar para ´x´ persona, todas las llamadas, especificando el total de recibidas y realizadas

C) CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formato para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC,LSC	2009-2	12098	Algortimos y Estructuras de Datos

PRÁCTICA No.	LABORATORIO DE	Ingeniero en Computación	DURACIÓN (HORA)
8	NOMBRE DE LA PRÁCTICA	Algoritmos de Ordenamiento	2

1 INTRODUCCIÓN

Ordenar es clasificar la información de acuerdo a un criterio. Por ejemplo en un directorio telefónico. Tener los datos ordenados hace mas fácil la búsqueda.
Si la búsqueda es rápida, la eficiencia de cualquier programa mejora.

2 OBJETIVO (COMPETENCIA)

El alumno conocerá los métodos de ordenación y búsqueda mas comunes para poder tener un amplio criterio de selección de los diferentes métodos para su aplicación en problemas de computación.

3 FUNDAMENTO

La clase Arrays de java contiene un método para ordenar. Puede ordenar arreglos de tipos primitivos de datos y así como arreglos de objetos . Por ejemplo:
Si creamos arreglos de tipos primitivos de datos.

```
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String[] nombres = {"Susana", "Anita", "David"};
        Arrays.sort(nombres);
        System.out.println(Arrays.toString(nombres));


        double[] valores = {120.0, 0.5, 0.0, 999.0, 77.3};
        Arrays.sort(valores);
        System.out.println(Arrays.toString(valores));
}
}
```

Al ejecutar este programa se mostraran los vectores de nombre y de valores ordenados de menor a mayor.

Otra forma de ordenar es utilizando la clase ***Collections*** que contiene también un método ***sort*** . El método **sort** se utiliza de dos formas.
En la primera forma la sintaxis del método es la siguiente:

```
static void sort(List<T> list)
```

se envía la lista de objetos que deseamos ordenar. Los objetos que contenga la lista deberán de implementar la interface ***Comparable.*** Por ejemplo, tenemos la clase Alumno

```
public class Alumno implements Comparable {

   private String nombre;
   private int edad;

   public int getEdad() {
            return edad;    }

ublic void setEdad(int edad)
        { this.edad = edad;     }
```

3 FUNDAMENTO

```
    public String getNombre() {
      return nombre;    }
   public void setNombre(String nombre) {
 this.nombre = nombre;    }
   public Alumno(String nombre, int edad) {
      this.nombre = nombre;
      this.edad = edad;
   }

public String toString() {
      return "\nNombre: " + nombre + " Edad: " + edad;
   }
//Este es el metodo que debemos sobrescribir
   public int compareTo(Object o) {

      if (!(o instanceof Alumno)) {
         throw new ClassCastException("Se espera un objeto de ALMNOS.");
      }
      int anotherPersonAge = ((Alumno) o).getEdad();
      return this.edad - anotherPersonAge;
   // throw new UnsupportedOperationException("Not supported yet.");
   }
}
```

Como se puede observar se sobrescribe el método **`compareTo`** de acuerdo al tipo de ordenamiento que se desea realizar, en este caso se ordeno por la edad del alumno. El programa para probar lo anterior se muestra a continuación.

```
public class Main {

   public static void main(String[] args) {

Vector misalumnos=new Vector();

Alumno o=new Alumno("Jose",34);
Alumno o2=new Alumno("maria",32);

misalumnos.add(o);
misalumnos.add(o2);
Collections.sort(misalumnos);
System.out.println(misalumnos);}
}
```

Formato para prácticas de laboratorio

3 FUNDAMENTO

Otra forma de utilizar el método sort es utilizando un comparador. A continuación se muestra esta otra forma.
La sintaxis del método ***sort*** de la clase ***Collections*** es la siguiente:

```
static <T> void sort(List<T> list, Comparator<? super T> c)
```

Primero debemos hacer un comparador que implemente la interface ***Comparator*** . Debemos sobrescribir los métodos necesarios, en nuestro caso solo sobrescribiremos el método ***compare().***
Los dos métodos de la interface ***Comparator*** son:

```
int compare(Object o1, Object o2)
boolean equals(Object obj)
```

Primero debemos hacer una clase que almacene información como por ejemplo la clase Datos que se muestra a continuación.

```
public class Datos {
private String nombre;
private int edad;

    public int getEdad() {
        return edad;  }
    public void setEdad(int edad) {
        this.edad = edad;    }
    public String getNombre() {
        return nombre;    }
    public void setNombre(String nombre) {
        this.nombre = nombre;    }

    public Datos(String nombre, int edad) {
        this.nombre = nombre;
        this.edad = edad;
    }
public String toString(){
    return "\nNombre: "+nombre+" Edad: "+edad;
}
}
```

3 FUNDAMENTO

Después se debe hacer una clase que implemente la interface ***Comparator*** y sobrescribir el método **compare**, como se muestra a continuación.

```
public class EdadComparar implements Comparator{

    public int compare(Object o1, Object o2) {
         Datos ob1=(Datos)o1;
        Datos ob2=(Datos)o2;
        return ob1.getEdad()-ob2.getEdad();

    }
}
```

Con esta clase el ordenamiento sera por edades de menor a mayor. El programa principal para comprobar el funcionamiento de las clases se muestra enseguida.

```
public class Main {

    public static void main(String[] args) throws IOException {

        ArrayList lista = new ArrayList();
        int c = 0;
        int op = 0;
        BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in));
        do {
            System.out.println("1.-altas\n2.-Mostrar ordenado nombre\n3.-Mostrar
ordenado edad" +"\n4.-Salir");
            op = Integer.parseInt(teclado.readLine());
            switch (op) {
                case 1:
                    System.out.println("Da el nombre: ");
                    String nom = teclado.readLine();
                    System.out.println("Da la edad: ");
                    int edad = Integer.parseInt(teclado.readLine());
                    lista.add(new Datos(nom, edad));
                    break;
                case 2:
                    Collections.sort(lista, new NombreComparar());
                    for (int i = 0; i < lista.size(); i++) {
                        Datos object = (Datos) lista.get(i);
                        System.out.println(object);}
break;
```

Formato para prácticas de laboratorio

3 FUNDAMENTO

```
 case 3:
                    Collections.sort(lista, new EdadComparar());
                    for (int i = 0; i < lista.size(); i++) {
                        Datos object = (Datos) lista.get(i);
                        System.out.println(object);

                    }

                    break;
            }
        } while (op != 4);
    }
}
```

4 PROCEDIMIENTO (DESCRIPCIÓN)

A EQUIPO NECESARIO	MATERIAL DE APOYO
Computadora con el sistema operativo Linux	Practica impresa

B DESARROLLO DE LA PRÁCTICA

Formato para prácticas de laboratorio

4 PROCEDIMIENTO (DESCRIPCIÓN)

Realice el ejemplo de la clase Datos con el comparador de Edad y ademas agregue una clase que compare el nombre, como se puede ver en la opción 2 del programa principal.

1.- Realice un programa que capture Números. En la clase Numero se almacenaran números reales. El programa deberá contar con las opciones de:

- Altas.- llenar de forma aleatoria un arreglo de objetos tipo Numero,
- Mostrar ordenado con la clase Collections,
- Mostrar ordenado con QuickSort y
- Mostrar una tabla que muestre el tiempo que tardo en ordenar con uno y con otro.

2.- Realice el programa anterior pero utilice el método Shell.

3.- Realice el programa anterior pero utilice los ordenamientos Quicksort, shell, inserción directa, selección directa y ordenado con la clase Collections. Compare los tiempos.

4.- Realice una comparación de tiempo entre los métodos quicksort, shell, insercion y seleccion directa que se vio en clase. Almacene números enteros generados de manera aleatoria.

5 RESULTADOS Y CONCLUSIONES

El alumno debe obtener los resultados presentados en la práctica para los programas de ejemplo, así como explicar claramente el funcionamiento de todos los programas de la práctica.

6 ANEXOS

Para más información consulte la pagina http://java.sun.com

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA (UNIDAD MEXICALI)
DOCUMENTO DEL SISTEMA DE CALIDAD

Formatos para prácticas de laboratorio

CARRERA	PLAN DE ESTUDIO	CLAVE ASIGNATURA	NOMBRE DE LA ASIGNATURA
IC, LSC	2009-2	12098	Algoritmos y Estructura de Datos

PRÁCTICA No.	LABORATORIO DE		DURACIÓN (HORA)
9	NOMBRE DE LA PRÁCTICA	Búsqueda Binaria	2

1. INTRODUCCIÓN

La búsqueda es una de las operaciones más importantes en el procesamiento de la información, esta permite recuperar datos previamente almacenados. El resultado puede ser un éxito, si el elemento que se busca se encuentra o fracaso en otro caso.

2. OBJETIVO (COMPETENCIA)

Crear programas de cómputo, implementado algoritmos de búsqueda binaria, para comprobar su eficiencia rapidez, con actitud perseverante y creativa.

3. FUNDAMENTO

Búsqueda Binaria

La búsqueda binaria consiste en dividir el intervalo de búsqueda en dos partes, comparando el elemento buscado con el central. En caso de no ser iguales se redefinen los extremos del intervalo(según el elemento central sea mayor o menor que el buscado) disminuyendo el espacio de búsqueda. El proceso concluye cuando el elemento es encontrado, o bien cuando el intervalo de búsqueda se anula.
Este método funciona únicamente con arreglos ordenados. Con cada iteración del método el espacio de búsqueda se reduce a la mitad, por lo tanto el número de comparaciones a realizar disminuye notablemente . Esta disminución resulta significativa cuanto más grande sea el tamaño del arreglo.

En java contamos con el método *binarySearch*, de la clase Arrays la cual se implementa en el paquete java.util.
binary Search, permite buscar un valor en un arreglo ordenado ascendentemente utilizando para ello el algoritmo de búsqueda binaria. Se ha resaltado que el arreglo a de estar previamente ordenado para que funcione apropiadamente el método, caso contrario puede obtenerse resultados inesperados. Como se indicó antes se trata de un método sobrecargado para los diferentes tipos de fatos primitivos y para Object.
Por ejemplo veamos el método específico de doublé(los otros son prácticamente iguales)

Public static int *binarySearch*(double[] a, double key)

El método recibe el arreglo y el valor que quiere buscar; este valor obviamente siempre es del mismo tipo. Devuelve un entero que corresponde al índice del elemento en el arreglo que coincide con el valor buscado

Ejemplo:

```
double[] c = {10, 20, 30, 40, 50};
int i;

i = Arrays.binarySearch(c, 30);
//devuelve i = 2
i = Arrays.binarySearch(c, 10);
// i = 0, recordando que el índice siempre empieza en 0
```

¿Qué pasa si no encuentra el valor?
Devuelve el valor -(punto de inserción) – 1. Punto de inserción es el punto donde, de acuerdo al orden, se esperaría que estuviera ese valor. Utilizando el mismo arreglo c que hemos creado, veamos algunas búsquedas:

```
i = Arrays.binarySearch(c, 25); // i = -3
i = Arrays.binarySearch(c, 45); //i = -5
```

4. PROCEDIMIENTO (DESCRIPCIÓN)	
A) EQUIPO NECESARIO	**MATERIAL DE APOYO**

Todos los alumnos escribirán un programa utilizando el método binarySearch de Java, en el cual generaran una serie de números aleatorios y realizaran la búsqueda de un número en particular.
Adicional a esto realizarán el programa que les corresponda según el día de la semana en el cual tenga su clase de laboratorio.

Lunes

Escriba un programa para llevar un control de los juegos de video, para cada uno de los juegos se deberá tener un registro de lo siguiente:
Numero de control (El cual no puede ser repetido)
Nombre del juego

El programa presentará las siguientes opciones:

a) Altas
b) Consulta por numero de control

c) Modificaciones por numero de control
d) Bajas por numero de control
e) Salir

Martes

Escribirá un programa para llevar un control de los libros de una librería, para cada uno de los libros se debran registrar los siguientes datos:
ISBN (El cual no puede repetirse)
Nombre del libro
Nombre del autor
Edición
Cantidad en existencia
Costo unitario
El programa deberá presentar las siguientes opciones:

a) Registro de los libros
b) Consulta por numero de ISBN
c) Ventas de libros
d) Bajas (La cual solo se podrá realizar si ya no hay libros en existencia)
e) Salir

Miércoles

Realizar un programa para llevar el control de los artículos en una ferretería, para cada uno de ellos se deberán registrar los siguientes datos:
Numero de control del artículo
Descripción del artículo
Cantidad en existencia
Precio unitario

El programa deberá presentar las siguientes opciones:

a) Altas
b) Consulta por numero de control del articulo
c) Ventas
d) Bajas por numero de control del articulo (La cual solo se podrá realizar si el articulo ya no está en existencia)
e) Salir

Jueves

Escriba un programa para llevar el registro de los alumnos de una escuela de idiomas, para cada alumno se deberán registrar los siguientes datos:
Numero de control
Nombre del alumno
Idioma que estudia, los cuales pueden ser (Aleman, Ingles, Frances y Japones)
Grado que cursa.
El programa presentara las siguientes opciones:

a) Registro de alumnos

b) Consulta por numero de control
c) Modificaciones por numero de control
d) Bajas por numero de control
e) Salir

Viernes

Escriba un programa para llevar el registro de los niños de una guardería, para cada niño se pedirán los siguientes datos:
Numero de control
Nombre del niño
Fecha de nacimiento
Grupo
El programa presentara las siguientes opciones:

a) Registro
b) Consulta por numero de control
c) Modificaciones por numero de control
d) Bajas por numero de control
e) Salir

C)	CÁLCULOS Y REPORTE

5. RESULTADOS Y CONCLUSIONES

6. ANEXOS

7. REFERENCIAS