



UTPL

La Universidad Católica de Loja

Modalidad Abierta y a Distancia

Estructura de Datos

Guía didáctica

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



**Departamento de Ciencias de la Computación y
Electrónica**

Sección departamental de Inteligencia Artificial

Estructura de Datos

Guía didáctica

Autor:

Guamán Bastidas Franco Olivio



DSOF_2041

Asesoría virtual
www.utpl.edu.ec

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas


Recursos

Estructura de Datos

Guía didáctica

Guamán Bastidas Franco Olivio

Universidad Técnica Particular de Loja

 4.0, CC BY-NY-SA

Diagramación y diseño digital:

Ediloja Cía. Ltda.

Telefax: 593-7-2611418.

San Cayetano Alto s/n.

www.ediloja.com.ec

edilojainfo@ediloja.com.ec

Loja-Ecuador

ISBN digital - 978-9942-25-784-0



La versión digital ha sido acreditada bajo la licencia Creative Commons 4.0, CC BY-NY-SA: Reconocimiento-No comercial-Compartir igual; la cual permite: copiar, distribuir y comunicar públicamente la obra, mientras se reconozca la autoría original, no se utilice con fines comerciales y se permiten obras derivadas, siempre que mantenga la misma licencia al ser divulgada. <https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es>

30 de abril, 2020

Índice

1. Datos de información.....	8
1.1. Presentación-Orientaciones de la asignatura.....	8
1.2. Competencias genéricas de la UTPL.....	8
1.3. Competencias específicas de la carrera	8
1.4. Problemática que aborda la asignatura en el marco del proyecto.....	9
2. Metodología de aprendizaje.....	9
3. Orientaciones didácticas por resultados de aprendizaje.....	10
Primer bimestre.....	10
Resultado de aprendizaje 1 y 2	10
Contenidos, recursos y actividades de aprendizaje.....	10
Semana 1	10
Unidad 1. Estructuras lineales estáticas	11
1.1. Cadenas (strings).....	12
1.2. Arreglos unidimensionales.....	13
Actividades de aprendizaje recomendadas	14
Semana 2	15
1.3. Operaciones con arreglos.....	16
1.4. Arreglos multidimensionales	17
1.5. Registros	18
Actividades de aprendizaje recomendadas	18
Autoevaluación 1	20

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Semana 3	23
Unidad 2. Tipos de datos abstractos	23
2.1. Pilas y colas	23
Actividades de aprendizaje recomendadas	24
Semana 4	25
2.2. Conjuntos	25
2.3. Colas de prioridad	25
Actividades de aprendizaje recomendadas	26
Autoevaluación 2	27
Semana 5	29
Unidad 3. Estructuras lineales dinámicas.....	29
3.1. Apuntadores.....	29
3.2. Listas enlazadas	30
Actividades de aprendizaje recomendadas	31
Semana 6	32
3.3. Implementación de pilas y colas con listas enlazadas ..	32
Actividades de aprendizaje recomendadas	32
Autoevaluación 3	33
Actividades finales del bimestre	35
Semanas 7 y 8.....	35
Segundo bimestre	36
Resultado de aprendizaje 3	36
Contenidos, recursos y actividades de aprendizaje.....	36

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Semana 9	36
Unidad 4. Estructuras jerárquicas	36
4.1. Árboles.....	37
Actividades de aprendizaje recomendadas	37
Semana 10	38
4.2. Árboles binarios	38
4.3. Recorridos en árboles binarios	39
Actividades de aprendizaje recomendadas	39
Semana 11	40
4.4. Árboles de búsqueda binaria.....	40
Actividades de aprendizaje recomendadas	41
Semana 12	41
4.5. Árboles AVL.....	41
Actividades de aprendizaje recomendadas	42
Autoevaluación 4	43
Semana 13	46
Unidad 5. Grafos.....	46
5.1. Concepto y definiciones	46
5.2. Representación	47
5.3. Recorrido de grafos	48
Actividades de aprendizaje recomendadas	49
Autoevaluación 5	50

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Semana 14	52
Unidad 6. Archivos.....	52
6.1. Concepto de Archivos.....	52
6.2. Lectura de archivos	53
6.3. Escritura de archivos	54
6.4. Buffering	55
Autoevaluación 6	56
Actividades finales del bimestre	58
Semanas 15 y 16.....	58
4. Solucionario	59
5. Referencias bibliográficas	66
6. Recursos	67

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



1. Datos de información

1.1. Presentación-Orientaciones de la asignatura



1.2. Competencias genéricas de la UTPL

- Pensamiento crítico y reflexivo.
- Organización y planificación del tiempo.

1.3. Competencias específicas de la carrera

- Construir modelos específicos de ciencias de la computación mediante esquemas matemáticos y estadísticos, para propiciar el uso y explotación eficiente de datos e información.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

- Implementar aplicaciones de baja, mediana y alta complejidad, integrando diferentes herramientas y plataformas para dar solución a requerimientos de la organización.

1.4. Problemática que aborda la asignatura en el marco del proyecto

El estudio de las Estructuras de Datos es importante en el área de las Ciencias de la Computación, ya que mediante ellas podemos conocer los requerimientos y procesos necesarios para la creación, almacenamiento y administración de información, permitiendo así escoger los métodos óptimos que logren brindar un correcto orden e interrelación a la información según su naturaleza, optimizando así procesos de rendimiento, velocidad y redundancia de la misma.



2. Metodología de aprendizaje

La metodología por utilizar será aprendizaje basado en problemas (ABP), que puede definirse como un proceso de indagación en el cual se busca resolver preguntas, curiosidades, dudas e incertidumbres. Es un método docente basado en el estudiante como protagonista de su propio aprendizaje, donde la indagación por el alumno constituye parte importante del ABP que guiará el proceso del aprendizaje. Con este método, el aprendizaje de conocimientos tiene la misma importancia que la adquisición de habilidades y actitudes. Para ilustrar esta metodología, lo invito a que revise (Rodríguez, 2017), donde se realiza una apropiada explicación de esta.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



3. Orientaciones didácticas por resultados de aprendizaje



Primer bimestre

Resultado de aprendizaje 1y 2

- Utiliza tipos de datos primitivos para construir estructuras de datos.
- Escribe programas que usen cada una de las siguientes estructuras de datos: arreglos, registros, listas enlazadas, pilas y colas.

Contenidos, recursos y actividades de aprendizaje



Semana 1

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Unidad 1. Estructuras lineales estáticas

Estimado estudiante: bienvenido a la primera semana de estudios. Vamos a adentrarnos en el estudio de las Estructuras de Datos, y para ello comenzaremos por abordar las estructuras lineales estáticas, para las cuales es importante reconocer los tipos de datos.

Todo lenguaje de programación contiene elementos específicos que permiten realizar operaciones básicas de programación: tipos de datos, operadores e instrucciones o sentencias. Los tipos de datos se pueden clasificar de acuerdo con las siguientes categorías:

- De acuerdo con el tipo de información que representan.
 - Datos de tipo primitivo: representan un único dato simple que puede ser de tipo char, byte, short, int, long, float, double, boolean.
 - Variables referencia: implementadas mediante un nombre o referencia que contiene la dirección en memoria de un valor o conjunto de valores.
- Según cambie su valor o no durante la ejecución del programa.
 - Variables: el valor asociado puede cambiar las veces que sean necesarias durante la ejecución del programa.
 - Constantes: su valor no puede ser modificado.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

- Según su papel en el programa.
 - Variables miembro de una clase. Se definen dentro de una clase, fuera de los métodos. Pueden ser de tipos primitivos o referencias y también variables o constantes.
 - Variables locales. Se definen dentro de un método y desaparecen una vez finalizada la ejecución del método. También pueden ser de tipos primitivos o referencias.

Para ampliar la información sobre esta sección, lo invito a revisar la publicación realizada por [Sun Microsystems \(2008\)](#), en la cual se permite conocer también el tipo de información que podemos almacenar según cada tipo de dato, tal como se observa en la tabla de valores predeterminados.

A continuación, lo invito a revisar la introducción de la primera unidad de la *Guía Didáctica* (Guamán, 2017), en la cual se presenta una clasificación de las diferentes estructuras de datos según su funcionamiento.

1.1. Cadenas (strings)

Para abordar el tema “cadenas o *strings*”, es necesario que revise el apartado 1.1 de nuestra *Guía Didáctica* (Guamán, 2017). Preste especial atención a la definición dada sobre cadenas y a los apartados que desde ahí se solicita sean revisados.

Una buena información que complementa el tema de datos primitivos y strings, se puede observar en el siguiente video.

Taboada, A. (04 de febrero de 2017). [Archivo de video]. Recuperado de [6. Programación en Java || Introducción || Tipos de datos no primitivos y cadenas](#)

En este video, además de brindarnos una buena información acerca de los tipos de datos primitivos y no primitivos, preste especial atención a partir del cuarto minuto, donde se realiza una demostración de la utilización de cadenas o strings en el lenguaje de programación Java. Observe la forma sencilla de declaración de este tipo de variables y su diferencia con los datos de tipo carácter, especialmente con la utilización de comillas simples y dobles.

Es importante que desarrolle las actividades de aprendizaje recomendadas al final de la semana para reforzar los conocimientos adquiridos en cada uno de los puntos revisados.

1.2. Arreglos unidimensionales

Un *arreglo unidimensional* es un tipo de datos estructurado que está formado por una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales; por lo cual, los datos que se guarden en los arreglos deben ser todos del mismo tipo.

El tipo de acceso a los arreglos unidimensionales es directo, es decir, podemos acceder a cualquier elemento del arreglo sin tener que consultar elementos anteriores o posteriores, esto debido al uso de un índice para cada elemento del arreglo que proporciona su posición relativa dentro del arreglo. Los arreglos permiten entonces la realización de un conjunto de operaciones para manipular sus datos, a saber: ordenar, buscar, insertar, eliminar, modificar entre otras.

Es necesario que revise el apartado 1.2 de nuestra *Guía didáctica* (Guamán, 2017), en la cual se detalla cada una de las características que tienen los arreglos, así como algunos ejercicios que le permitirán involucrarse en el manejo de este tipo de datos en el lenguaje de programación Java.

¿Qué le han parecido los temas estudiados hasta el momento? Espero que no haya tenido ningún inconveniente con la comprensión de los mismos. Recuerde que puede comunicarse con el docente-tutor por cualquiera de los medios que se han socializado en el entorno virtual.

Vamos a continuación a realizar algunas actividades que le permitirán mejorar la comprensión de los temas, así como ponerse a prueba acerca de los temas tratados.



Actividades de aprendizaje recomendadas

Actividad de aprendizaje

1. Realice un cuadro comparativo entre Java y otros tres lenguajes de programación, con respecto al tratamiento de estructuras de tipo *string*.

Procedimiento

Realice una consulta vía internet de otros tres lenguajes de programación a su elección, y elabore un cuadro comparativo de las principales funcionalidades de los tipos de datos strings en cada uno de los lenguajes escogido y Java.

2. Si tuviera un arreglo de caracteres con las letras del texto "ESTRUCTURA", ¿cuál sería el índice de la letra "C"?

3. Un curso de computación tiene 50 alumnos matriculados y debe diseñar un arreglo para registrar las notas de los alumnos. Responda las preguntas y programe en Java las sentencias requeridas a continuación.
 - a. ¿De qué tipo de dato lo definiría?
 - b. ¿Qué tamaño tendría el arreglo?
 - c. ¿Cuáles son los índices inferior y superior del arreglo?
 - d. Realice la definición del arreglo en Java.
4. Realice un programa en Java que permita ingresar un arreglo de enteros de 5 elementos, y llevar a cabo las siguientes operaciones:
 - a. Mostrar o leer cada uno de los elementos ingresados.
 - b. Lea el elemento que se encuentra en la posición 3.

Procedimiento

Revise el material propuesto en la *Guía didáctica (Guamán, 2017)*; concentre su atención en los temas relacionados con la declaración de arreglos.



Semana 2

Inicialmente, los programas carecían de organización, estructuras de control y datos complejos, los cuales fueron introducidos a medida que evolucionaban los lenguajes; esto permitió crear datos compuestos a partir de la agrupación de datos simples por parte de los programadores. Muchos algoritmos requieren una presentación apropiada de los datos para lograr ser eficientes. Esta presentación junto con las operaciones permitidas se llama *estructura de datos*.

El establecimiento de técnicas de programación soportadas por los lenguajes, tienen como principal propósito el de optimizar costos (escritura, mantenimiento y uso de recursos), en la producción de programas. Las técnicas siempre han buscado reducir las instrucciones, al crear componentes reutilizables y organizar de mejor forma los datos, de tal manera que se puedan implementar algoritmos más eficientes, que disminuyan el tiempo y faciliten el procesamiento.

1.3. Operaciones con arreglos

El estudio de arreglos unidimensionales y multidimensionales implica, además de conocer los diferentes métodos de declaración y acceso –según el lenguaje de programación– conocer las diferentes operaciones que sobre ellos se pueden realizar.

Una explicación completa de este tema se puede encontrar a partir del ítem 1.2 hasta el ítem 1.3 de nuestra *Guía didáctica* (Guamán, 2017). Lo invito a revisar las secciones mencionadas y complementar la información con el libro-base, según las citas en la guía mencionadas.

A continuación pongo a su disposición una presentación, mediante la cual junto con gráficas y extractos de código fuente, explico paso a paso las principales operaciones que se realizan sobre arreglos, así como la lógica de los algoritmos utilizados para ello.

Estructuras de datos - Arreglos

[Ir a recursos](#)

Es importante que revise este material, ya que sobre él trataremos en la actividad síncrona de la siguiente semana, para la cual se ha

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

planificado una sesión de videocolaboración con el tema sobre tratamiento de arreglos.

¿Qué le pareció nuestro recurso de aprendizaje? Bastante didáctico, ¿no cree? A continuación avanzaremos hacia nuestro siguiente tema.

1.4. Arreglos multidimensionales

Los arreglos multidimensionales, como indica su nombre, son aquellos que tienen más de una dimensión. En la mayoría de lenguajes de programación, las direcciones de acceso a sus elementos son manejadas por medio de un par de corchetes, dentro de los cuales constarán los índices de fila y columna a los que corresponderá un elemento determinado.

Al igual que como ocurre con los arreglos unidimensionales, en los arreglos multidimensionales se pueden llevar a cabo una serie de operaciones como: declaración, creación, inicialización acceso, entre otras.

Una explicación completa de este tema lo puede usted encontrar en el ítem 1.4 de la *Guía didáctica* (Guamán, 2017), así como los diferentes ejemplos de operaciones que sobre ellos podemos realizar.

La presentación puesta a su disposición a continuación, al igual que la estudiada en el ítem anterior, permite revisar mediante imágenes y código fuente, los aspectos principales para considerar en cuanto a las operaciones con arreglos de más de una dimensión se refieren.

Estructuras de datos - Arreglos 2D

[Ir a recursos](#)

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Recuerde que por cualquier inquietud que surja del estudio de estas presentaciones, puede comunicarse con el docente a través de cualquiera de los medios por Universidad puestos a su disposición.

1.5. Registros

Un *registro* es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura. Estos elementos pueden ser, bien sea datos elementales (entero, real, caracter,...), u otras estructuras de datos. A cada uno de esos elementos se le denomina *campo*. Un registro asimismo se diferencia de un vector o arreglo en que este es una colección de datos iguales, es decir, todos del mismo tipo, mientras que en un registro los elementos que lo componen, aunque podrían serlo, no tiene por qué ser del mismo tipo.

Como podrá observar en la sección 1.5 de la *Guía didáctica (Guamán, 2017)*, se hace una explicación detallada de las estructuras “Registros”, así como una comparación de la forma de trabajo en los lenguajes C++ y Java.

Una vez abordados los temas solicitados, lo invito a realizar las siguientes actividades.



Actividades de aprendizaje recomendadas

Como estrategia de aprendizaje, le propongo el desarrollo de los siguientes ejercicios.

- Realice la programación del método “obtenerPosición” que se utiliza en el ejemplo para la inserción de un nuevo elemento.

- Realice la programación del método “desplazarDerecha” que se utiliza en el ejemplo para la inserción de un nuevo elemento.
- Realice la programación del método “desplazarIzquierda” que se utiliza en el ejemplo para la eliminación de un elemento.
- Realice un programa completo que permita llevar a cabo todas las operaciones con arreglos estudiadas, de acuerdo con una opción leída por teclado. Implemente el algoritmo QuickSort en el programa antes diseñado.

Procedimiento

Revise el material propuesto desde la *Guía didáctica (Guamán, 2017)*. Haga planificación previa de los tipos de movimientos que deberán realizar los elementos en los arreglos para visualizar de mejor forma el algoritmo por implementar; en caso de requerir ayuda, no dude en contactarse con el docente-tutor.

- Finalmente, para probar sus conocimientos, lo invito a desarrollar la siguiente autoevaluación.

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo previamente revisar los contenidos de la primera unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



Autoevaluación 1

Haga lectura atenta de cada una de las preguntas y seleccione las alternativas correctas.

1. Una de las características de los tipos de datos estáticos es:
 - a. Su tamaño en memoria puede variar dependiendo de la necesidad del programa.
 - b. Dependiendo del programa, pueden tener un tamaño fijo o variable.
 - c. Su tamaño en memoria debe ser definido antes de la ejecución del programa y no puede ser modificado durante su ejecución.
2. Entre las características de los datos de tipo string o cadenas tenemos que:
 - a. Permiten realizar operaciones con números enteros.
 - b. Pueden representar variables con los valores Verdadero o Falso
 - c. Permiten la manipulación de su contenido mediante la extracción de parte de ella.
3. Al decir que los arreglos deben ser homogéneos, nos referimos a que:
 - a. Todos los datos deben estar ordenados de acuerdo con un subíndice preestablecido.
 - b. Pueden ser unidimensionales o multidimensionales.
 - c. Todos los elementos deben ser del mismo tipo de datos.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

4. Los arreglos:
 - a. Permiten el almacenamiento solamente de datos simples.
 - b. No permiten el almacenamiento de datos de tipo compuesto (registros).
 - c. Pueden ser utilizados tanto para el almacenamiento de datos simples como compuestos.
5. El ingreso de datos en un arreglo:
 - a. Se debe hacer exclusivamente en el momento de su definición.
 - b. No se puede llevar a cabo por métodos fuera de la clase main.
 - c. Se puede llevar a cabo al momento de su definición o mediante una fuente externa.
6. Entre las principales operaciones que pueden aplicarse a arreglos tenemos:
 - a. Presentación, multiplicación, diferencia, eliminación.
 - b. Modificación, búsqueda, ordenación, eliminación.
 - c. Lectura, suma, resta, inserción,
7. La operación de búsqueda binaria en arreglos:
 - a. Puede ser ejecutada sobre cualquier tipo de arreglos (ordenados o desordenados).
 - b. Necesita que sus datos sean de tipo entero.
 - c. Puede ser implementada sobre arreglos previamente ordenados.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

8. ¿Cuál de las siguientes afirmaciones no corresponde a una inicialización de un arreglo bidimensional?
- a. `Int[][] a = new int[n][m];`
 - b. `Int a = new int[][];`
 - c. `Int A[][] = {{7,14,8,3},{6,19,7,2},{3,13,4,1}};`
9. Al referirse a arreglos bidimensionales, generalmente se nombran sus componentes en el siguiente orden:
- a. Filas, columnas.
 - b. Columnas, filas.
 - c. El orden es indiferente.
10. Para conocer el número de columnas de un arreglo en Java, debemos utilizar la siguiente expresión:
- a. `Columnas = a.Length;`
 - b. `Columnas = a.Length.Length;`
 - c. `Columnas = A[0].length.`

[Ir al solucionario](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)



Semana 3



Unidad 2. Tipos de datos abstractos

¡Bienvenido a la segunda unidad! En esta unidad revisaremos algunos de los principales tipos de datos abstractos, como son pilas, colas, conjuntos y las colas de prioridad, cuyo estudio no es complejo. Estoy seguro de que con un poco de dedicación, entenderá rápidamente los conceptos relacionados con cada uno de ellos, así como los algoritmos para su implementación.

2.1. Pilas y colas

Las *pilas* y *colas* representan estructuras lineales de datos en las que se pueden agregar o quitar elementos únicamente por uno de los extremos. Así, los elementos de una pila se eliminan en el orden inverso al que se insertaron. Debido a esta característica, se conoce como estructura LIFO (Last In, First Out). En cambio, los elementos de las colas se eliminan en el estricto orden de inserción, por lo que se conoce como estructura FIFO (First In, First Out).

Al implementar pilas o colas mediante arreglos, se tiene la limitación que se debe reservar el espacio en memoria con anticipación. Una vez dado un máximo de capacidad no es posible insertar un número

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

de elementos mayor que el máximo establecido. Si esto ocurre, se producirá un error conocido como desbordamiento (*overflow*).

Revise, por favor, la sección 2.1 de nuestra Guía didáctica para ampliar estos conocimientos. Asimismo, lo invito a revisar el documento ([Humeau, 2013](#)), tomado de *Slideshare*, en el cual se hace una explicación gráfica apropiada sobre los conceptos revisados de pilas y colas. Concentre su atención en las primeras trece diapositivas y pruebe en su computador los programas expuestos en ellas.



Actividades de aprendizaje recomendadas

Como estrategia de aprendizaje, le propongo el desarrollo de los siguientes ejercicios.

1. Elabore la implementación en Java de un programa que nos permita llevar a cabo las tareas de Push y Pop en una pila, mediante arreglos.
2. Elabore la implementación en Java de un programa que nos permita llevar a cabo las tareas de Push y Pop en una cola, mediante arreglos.

Procedimiento

Una vez claro el funcionamiento de los arreglos y sus subíndices, elabore un programa en el cual ambas estructuras (pilas y colas) compartan el mismo método para inserción de elementos (Push), pero difiera para cada caso el método Pop, ya que su funcionamiento es diferente.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



Semana 4

2.2. Conjuntos

Como podrá observar en la documentación de la asignatura, al hablar de conjuntos, describimos estructuras contenedoras de ciertos valores sin un orden concreto ni valores repetidos, sobre las cuales podemos aplicar ciertos tipos de operaciones.

Preste atención, por favor al ítem 2.2 de nuestra *Guía didáctica* (Guamán, 2017) y a las referencias que desde allí se realizan.

2.3. Colas de prioridad

Una cola de prioridades es un tipo de dato abstracto similar a una cola, en que los elementos adicionalmente tienen una prioridad asignada. En una cola de prioridades, un elemento con mayor prioridad será desencolado antes que un elemento de menor prioridad. Si dos elementos tienen la misma prioridad, se desencolarán siguiendo el orden de cola.

Remítase, por favor, al apartado 2.3 de nuestra *Guía didáctica* (Guamán, 2017) para ampliar la información dada.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Actividades de aprendizaje recomendadas

Como estrategia de aprendizaje, le propongo el desarrollo de los siguientes ejercicios.

- Plantee problemas en los cuales sea necesario el uso de los TAD “conjuntos” o “colas de prioridad”.

Finalmente, para probar sus conocimientos lo invito a desarrollar la siguiente autoevaluación.

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo previamente revisar los contenidos de la segunda unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Autoevaluación 2

Lea detenidamente cada una de las siguientes afirmaciones y determine si son verdaderas o falsas.

1. () Un tipo abstracto de datos es un grupo de datos que cumplen determinadas características, están asociados a ciertas operaciones y pueden ser implementados mediante distintas estructuras de datos.
2. () El TAD “pila”, también es conocido como una estructura FIFO.
3. () Una operación “Push”, sin importar el tipo de datos, refiere al procedimiento que permite la inserción de un nuevo elemento.
4. () Mediante la implementación de los métodos correctos, se pueden extraer datos desde una pila, sin importar si haya sido o no el último en ingresar.
5. () Una operación “Pop” es el nombre que se da a aquella que permite obtener el primer objeto ingresado en una cola.
6. () Las operaciones FIFO (First In First Out) son características del TAD Cola.
7. () Un conjunto se define como un grupo de elementos sin un orden concreto ni valores repetidos.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

8. () La operación de *unión de conjuntos* permite crear un nuevo conjunto con los elementos que son comunes en los conjuntos iniciales.
9. () La diferencia de conjuntos da como resultado un conjunto cuyos elementos pertenecen a alguno de los conjuntos iniciales, sin pertenecer a ambos a la vez.
10. () Las colas de prioridad, mediante la implementación de un nuevo parámetro, permiten tener un nuevo criterio de evaluación para la atención de los elementos de una cola.

[Ir al solucionario](#)

[Índice](#)

[Primer bimestre](#)

[Segundo bimestre](#)

[Solucionario](#)

[Referencias bibliográficas](#)

[Recursos](#)



Semana 5



Unidad 3. Estructuras lineales dinámicas

Al existir variedad de tipos de TDA, debemos tener la capacidad de discernir entre cuál de ellos debería ser la mejor opción para expresar o modelar un problema determinado. Cada estructura de datos presentará ciertas ventajas y desventajas frente a los demás, pero cada uno de ellos podrá ser el más adecuado para implementar de acuerdo con el problema que se pretende modelar.

Entre los diferentes tipos de datos, en esta sección abordaremos el tratamiento de punteros o apuntadores, los cuales serán de vital importancia, sobre todo si nos referimos a estructuras de datos dinámicas, como veremos en el presente tema.

3.1. Apuntadores

Un puntero es un tipo especial de variable, que tiene la capacidad de contener la dirección de memoria de otro objeto (apunta al espacio físico donde está el dato), tal como: char, int, float, array, una estructura, una función u otro puntero. Las variables de tipo puntero son indispensables para el tratamiento de estructuras de datos dinámicas, ya que permitirán la creación y seguimiento de nuevas

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

estructuras. Un ejemplo de la utilización de este tipo de elementos lo podemos observar en el siguiente video tomado de *YouTube*.

MasterHeHeGar. (2014). *Nodos y Punteros (EDDJava)*. [Archivo de video]. Recuperado de: [14 - Nodos y Punteros \(EDDJava\)](#).

Así como se describe en el video antes citado, debe diferenciar entre lo que representa un puntero y lo que es un nodo.

Remítase a la sección 3.1 de nuestra *Guía didáctica (Guamán, 2017)* para complementar esta información y realizar los ejercicios allí solicitados.

3.2. Listas enlazadas

Una lista enlazada es un TDA que permite almacenar datos de forma organizada, al igual que los arreglos o vectores, pero a diferencia de estos, su estructura es dinámica, por lo que no tenemos que saber el número de elementos que puede contener. Su funcionamiento podría ser comparado con el de un ferrocarril, en el cual se puede agregar o eliminar un vagón, de acuerdo con las necesidades. En el caso de las listas enlazadas, los nodos serían los vagones que la componen.

El video revisado la semana anterior mostró la estructura de los nodos que componen las listas enlazadas. A continuación, he publicado un video que nos muestra la creación de una lista enlazada con los métodos de insertar un nuevo nodo al inicio y el recorrido de la lista.

Guamán, F. (2019). *ListaEnlazada en Java, Ingreso y Recorrido*. [Archivo de video]. Recuperado de [ListaEnlazada en Java, Ingreso y Recorrido](#).

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

¿Qué le ha parecido el video? Si tiene alguna duda o desea aclarar algo de él, puede comunicarse con el tutor en horarios de oficina por los medios proporcionados para ello o en los horarios establecidos de tutoría.

Remítase a continuación al ítem 3.2 de nuestra *Guía didáctica* (Guamán, 2017) y revise las secciones del texto-base que ahí se han citado. Es recomendable igualmente el desarrollo de las actividades de aprendizaje adjuntas; aunque no son calificadas, pueden ser de gran ayuda para que se entienda completamente el funcionamiento de las listas enlazadas.



Actividades de aprendizaje recomendadas

Proponga ejemplos de listas enlazadas que pueda identificar en la vida diaria.

- Vagones de un tren.
- _____
- _____
- _____
- _____

Elabore la declaración en Java de un nodo que almacenaría los datos siguientes de una lista de clientes.

- Nombre.
- Dirección.
- Edad.
- Identificación.
- Género.

Realice la programación del código que para la semana 7 se ha puesto a su disposición.



Semana 6

3.3. Implementación de pilas y colas con listas enlazadas

Tal como se mencionó en la cuarta semana, los TDA pilas y colas también pueden ser implementados mediante listas enlazadas. Para este propósito, es necesario que revise la información mencionada en el apartado 3.3 de nuestra *Guía didáctica (Guamán, 2017)*, así como la actividad de aprendizaje proporcionada para esta semana.



Actividades de aprendizaje recomendadas

- Elabore la implementación en Java de un programa que nos permita llevar a cabo las tareas de Push y Pop en una pila, mediante arreglos.
- Elabore la implementación en Java de un programa que nos permita llevar a cabo las tareas de Push y Pop en una cola, mediante listas enlazadas.

Finalmente, para probar sus conocimientos, lo invito a desarrollar la siguiente autoevaluación.

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo revisar previamente los contenidos de la tercera unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Autoevaluación 3

Lea detenidamente cada una de las siguientes afirmaciones y determine si son verdaderas o falsas.

1. () Las estructuras dinámicas, requieren la creación de elementos llamados nodos, los cuales deben crearse o eliminarse de acuerdo con los requerimientos del programa.
2. () Las variables de tipo puntero o apuntador constituyen una alternativa a los índices para acceso a datos almacenados en arreglos unidimensionales.
3. () Durante la definición de los nodos que forman parte de las listas enlazadas, debemos incluir por lo menos un campo de tipo puntero que permita el acceso hacia un siguiente nodo.
4. () Para utilizar listas enlazadas, se debe previamente especificar la cantidad de elementos que la conformarán.
5. () Una lista doblemente enlazada, permite el almacenamiento de dos diferentes datos a la vez.
6. () Al realizar operaciones con listas enlazadas, siempre debemos tener presente el no perder la dirección del nodo raíz o cabeza, ya que es nuestra puerta de entrada o inicio de la lista.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

7. () La implementación de pilas mediante listas enlazadas permite optimizar el uso de la memoria, ya que no se cuenta con espacios no ocupados y puede crecer, según los requerimientos.
8. () En la implementación del TAD cola mediante listas enlazadas y de acuerdo con la programación, puede permitirse el ingreso de nuevos datos al inicio de la cola.
9. () Si durante la ejecución de un programa con listas enlazadas, perdemos la dirección de un siguiente nodo, será imposible su redireccionamiento y recuperar la continuidad del programa.
10. () Al trabajar con listas enlazadas no circulares, el campo *next* del nodo cola o último, siempre deberá estar apuntando hacia NULL.

[Ir al solucionario](#)

[Índice](#)

[Primer bimestre](#)

[Segundo bimestre](#)

[Solucionario](#)

[Referencias bibliográficas](#)

[Recursos](#)



Actividades finales del bimestre



Semanas 7 y 8

Actividad 1

- Revise todos los recursos educativos como preparación para la evaluación presencial.
- Resolución de inquietudes y participación en el chat semanal.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Segundo bimestre

Resultado de aprendizaje 3

Resuelve problemas usando algoritmos básicos de árboles y grafos.

Contenidos, recursos y actividades de aprendizaje



Semana 9



Unidad 4. Estructuras jerárquicas

Uno de los principales problemas por considerar al trabajar con datos es su organización y acceso; por esto, se buscan diferentes alternativas, que permitan lograr estos objetivos en el menor tiempo y con la menor utilización de recursos computacionales.

Los árboles constituyen una alternativa a las estructuras lineales, al evitar el acceso secuencial a la información, disminuyendo notablemente los tiempos de ejecución y el número de procesos por realizar.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

4.1. Árboles

Las estructuras jerárquicas constituyen una variación de las listas enlazadas, y al contrario de estas, permiten el acceso a sus elementos sin tener que estar apegados a un algoritmo secuencial. Su principal representante son los árboles, de los que tenemos algunas versiones, cada una de las cuales tiene ventajas y desventajas frente a otras. Asimismo, los árboles, en cualquiera de sus versiones, comparten ciertas características, propiedades y nomenclatura. Revise, por favor, el ítem 4.1 de nuestra Guía didáctica (Guamán, 2017) para conocer sus principales características.

Una vez que contamos con la información básica referente a árboles, lo invito a desarrollar la siguiente actividad de aprendizaje, la cual le permitirá poner en práctica los conocimientos adquiridos.



Actividades de aprendizaje recomendadas

1. Describa la definición de los conceptos que se indican a continuación:

- Nodo raíz: _____
- Nodo interno: _____
- Nodo no terminal: _____
- Nivel: _____
- Ramas: _____
- Camino: _____
- Altura: _____
- Subárbol: _____
- Árbol equilibrado: _____

2. Investigue cinco aplicaciones, además de las citadas previamente, que pudieran ser implementadas con árboles.



Semana 10

4.2. Árboles binarios

Un *árbol binario* es un tipo de árbol en que cada vértice máximo puede tener dos hijos; su nodo raíz está enlazado a dos subárboles binarios disjuntos denominados subárbol izquierdo y subárbol derecho. Los árboles binarios no son vacíos, ya que como mínimo tienen el nodo raíz.

Hay dos formas tradicionales de representar un árbol binario en memoria: por medio de datos tipo punteros también conocidos como variables dinámicas o listas, y por medio de arreglos. Sin embargo, la más utilizada es la primera, puesto que es la más natural para tratar este tipo de estructuras. Los nodos del árbol binario serán representados como registros que contendrán como mínimo tres campos. En un campo se almacenará la información del nodo y los dos restantes se utilizarán para apuntar al subárbol izquierdo y derecho del subárbol en cuestión.

Es momento de revisar el apartado 4.2 de nuestra *Guía didáctica* (Guamán, 2017), y el apartado “Árboles binarios” del texto básico (Allen, 2013).

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

4.3. Recorridos en árboles binarios

El recorrido de árboles se refiere al proceso de visitar de manera sistemática, exactamente una vez, cada nodo en una estructura de datos de árbol. Tales recorridos están clasificados por el orden en el cual se visitan los nodos.

Existen dos tipos de recorrido: por anchura y por profundidad. Concentraremos nuestro estudio en las variaciones de recorrido en profundidad: Preorden, Inorden y Posorden.

Remítase al ítem 4.3 de nuestra *Guía Didáctica (Guamán, 2017)*, revise las variaciones de la lógica encontrada entre cada recorrido, según la tabla 2, y elabore la actividad recomendada en ese capítulo.



Actividades de aprendizaje recomendadas

- Establezca las diferencias principales entre los tipos de recorridos en altura y profundidad establecidos para árboles binarios.
- Realice la programación de los principales tipos de recorridos en árboles binarios.
- Indique un posible caso donde se utilizaría cada uno de estos recorridos



Semana 11

4.4. Árboles de búsqueda binaria

Los *árboles de búsqueda binaria* son una variación de los árboles binarios, los cuales implementan una comparación de valores en el ingreso de datos, determinando así la posición del nuevo valor ingresado.

Revise, por favor, el ítem 4.4 de nuestra *Guía didáctica* (Guamán, 2017) y las citas del texto base desde allí hechas. Asimismo, a continuación, adjunto un archivo PDF, en el cual podrá observar las operaciones principales que se realizan sobre un árbol binario de búsqueda.

Guamán, F. (2019). *Árbol Binario de Búsqueda Java*. Recuperado de [enlace web](#)

Dependiendo del lenguaje utilizado, para su programación debe realizar ciertas variaciones en la programación; por ejemplo: en C++ se acepta el envío de parámetros por referencia, mientras que en Java no, lo que obliga a modificar la forma de codificar ciertos programas, aunque todos estén basados en el mismo algoritmo.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Actividades de aprendizaje recomendadas

1. Elabore gráficas de árboles de búsqueda binaria, con cada uno de los grupos de datos mostrados a continuación. Preste atención a las figuras resultantes.
 1. 65, 75, 30, 4, 41, 85.
 2. 65, 85, 41, 75, 30, 4.
 3. 4, 30, 41, 65, 75, 85.
2. Elabore un programa que permita llevar a cabo el almacenamiento de elementos en un árbol de búsqueda binaria.



Semana 12

4.5. Árboles AVL

Los árboles binarios de búsqueda balanceados permiten hacer búsquedas, inserciones y eliminaciones de elementos a muy altas velocidades, puesto que su función de complejidad tiene un tiempo promedio de $O(\log_2 N)$ donde N es el número de elementos en la lista.

En números concretos, buscar un elemento en una lista con un millón de elementos toma alrededor de 20 comparaciones y llamadas recursivas (ya que $2^{20} = 1048576$), rendimiento bastante sorprendente dada la gran cantidad de datos.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

Estudie el ítem 4.5 de nuestra *Guía didáctica* (Guamán, 2017), y las citas desde ahí realizadas hacia el texto-base de la asignatura.

Para un entendimiento completo del tema, le recomiendo leer detenidamente los ejemplos citados en el documento ([Sánchez, 2017](#)).

¿Qué le parecieron los ejemplos mostrados? Una explicación muy didáctica, ¿no le parece? Asimismo, espero no haya tenido problemas con la comprensión del tema, pero para cualquier inquietud, no dude en contactar al tutor por los medios ya conocidos.



Actividades de aprendizaje recomendadas

En su libreta de anotaciones grafique cinco ejemplos de árboles desbalanceados y aplique los métodos descritos para solucionar el problema.

Finalmente, para probar sus conocimientos lo invito a desarrollar la siguiente autoevaluación

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo revisar previamente los contenidos de la cuarta unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



Autoevaluación 4

Haga lectura atenta de cada una de las preguntas y seleccione las alternativas correctas.

1. En términos computacionales, un árbol:
 - a. Puede tener varios nodos raíz, siempre y cuando no exista un nodo hijo común.
 - b. Podrá generar tantos nodos raíz como sean necesarios en la aplicación.
 - c. Debe tener un solo nodo raíz, del cual se desprenderán todos los subárboles.
2. Aquellos nodos que no tienen descendientes se conocen como:
 - a. Nodo raíz.
 - b. Nodo hermano.
 - c. Nodo hoja.
3. Conocemos como “grado del árbol”:
 - a. El máximo grado de todos los nodos del árbol.
 - b. El máximo número de niveles encontrados en el árbol.
 - c. La sumatoria de todos los grados de los nodos del árbol.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

4. Se conoce como nivel de un nodo:
 - a. El número de arcos que deben ser recorridos para llegar a él.
 - b. El número de nodos hermanos que un nodo pueda tener.
 - c. El número de nodos por los cuales hay que pasar para llegar a él.
5. La altura del árbol es:
 - a. El máximo número de niveles de todos los nodos del árbol.
 - b. Igual al promedio de las diferentes alturas de sus subárboles.
 - c. El máximo número de nodos hijos que un nodo pueda tener.
6. Entre los recorridos de los árboles binarios, tenemos:
 - a. Anchura y profundidad.
 - b. Preorden y posorden.
 - c. Ascendente y descendente.
7. El recorrido que sigue el orden de visita "izquierda, derecha, raíz", se conoce con el nombre:
 - a. Anchura.
 - b. Profundidad.
 - c. Posorden.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

8. Un árbol de búsqueda binaria:
- a. Puede permitir a un nodo tener más de dos hijos, solo si se constituyen como nodos terminales u hojas.
 - b. No permite a ningún nodo tener más de dos hijos o subárboles.
 - c. Es aquel que solamente guarda información como “unos” o “ceros”.
9. La estructura final de un árbol de búsqueda binaria:
- a. Cambia de acuerdo con el orden de inserción de elementos.
 - b. Siempre será igual, sin importar el orden de inserción de elementos.
 - c. Siempre mantendrá la misma imagen, sin importar el ingreso de elementos.
10. La estructura final de un árbol AVL:
- a. Cambia de acuerdo con el orden de inserción de elementos,
 - b. Adapta su forma de acuerdo con el orden de inserción de elementos.
 - c. Siempre será igual, sin importar el orden de inserción de elementos.

[Ir al solucionario](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)



Semana 13



Unidad 5. Grafos

Los *grafos* son una estructura de datos que sirve para modelar muchos problemas, los cuales se pueden expresar de manera computacional. A diferencia de los árboles, no son una estructura rígida, por ello permite su utilización en aplicaciones que cuenten con objetos con relaciones arbitrarias entre sí.

5.1. Concepto y definiciones

Para las ciencias de la computación y la matemática, un grafo es una representación gráfica de diversos puntos que se conocen como nodos o vértices, los cuales se encuentran unidos mediante líneas que reciben el nombre de aristas. Al analizar los grafos, los expertos logran conocer cómo se desarrollan las relaciones recíprocas entre aquellas unidades que mantienen algún tipo de interacción.

Los grafos se pueden clasificar de diversas maneras según sus características, por ejemplo:

- Los grafos simples, son aquellos que surgen cuando una única arista logra unir dos vértices.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

- Los grafos complejos, en cambio, presentan más de una arista en unión con los vértices.
- Un grafo es conexo si dispone de dos vértices conectados mediante un camino. ¿Qué quiere decir esto? Que para el par de vértices (p, r) , tiene que existir algún camino que permita llegar desde p hasta r .
- Un grafo está fuertemente conexo si el par de vértices tiene conexión mediante, como mínimo, dos caminos diferentes.

Existen algunos términos que serán utilizados para nombrar los elementos involucrados en los grafos. Remítase a la sección 5.1 de nuestra *Guía didáctica* (Guamán, 2017). Revise los términos y las descripciones que para cada uno de ellos se hace en ese apartado.

5.2. Representación

Existen diferentes formas de representar un grafo, y algunos métodos para almacenarlos en computador. La estructura de datos utilizada dependerá de las características del grafo, y el algoritmo usado para manipularlo. Entre las más comunes están las listas y matrices; con frecuencia se usa una combinación de ambas, así tenemos:

Lista de incidencia: el grafo está representado por una matriz de A (aristas) por V (vértices), donde [arista, vértice] contiene la información de la arista (conectado o no conectado).

Lista de adyacencia: el grafo está representado por un arreglo de listas de adyacencia. Para un vértice i , la lista de adyacencia está formada por todos los vértices adyacentes a i .

Matriz de adyacencia: el grafo está representado por una matriz cuadrada M de tamaño n^2 , donde n es el número de vértices. Si hay

una arista entre un vértice x y un vértice y , entonces el elemento $m_{\{x,y\}}$ es 1; de lo contrario, es 0.

Matriz de incidencia: el grafo está representado por una matriz de A (aristas) por V (vértices), donde [vértice, arista] contiene la información de la arista (1 - conectado, 0 - no conectado).

Cada una de las formas de representación antes citadas, han sido descritas en nuestra *Guía didáctica* (Guamán, 2017), en las secciones 5.2.1, 5.2.2 y 5.2.3.

Revise, por favor, los ejemplos gráficos, que para cada caso se han descrito en las secciones mencionadas

5.3. Recorrido de grafos

El recorrido de un grafo implica tener que pasar por cada vértice que lo compone, empezando desde un vértice determinado. Para ello, se cuenta con recorridos tanto en anchura como en profundidad.

Es recomendable que implemente los programas descritos en las secciones 5.3.2 y 5.3.3 de nuestra *Guía didáctica* (Guamán, 2017); elabore una corrida de escritorio de cada uno de ellos.

Todos los temas tratados, se encuentran en la guía didáctica citada, y recuerde que, si necesita aclarar algún tema, debe remitirse al tutor en los horarios y medios indicados.

Muy bien estimado estudiante: hemos llegado a la finalización del estudio de la asignatura en este segundo bimestre y, por ende, del presente ciclo. Esperamos que los conocimientos adquiridos sean los básicos y necesarios para iniciar el desarrollo de aplicaciones que tengan eficiencia y eficacia. Por ello, le invitamos a desarrollar las siguientes actividades de aprendizaje, esperando que todo le vaya mejor. ¡Adelante y éxitos!.

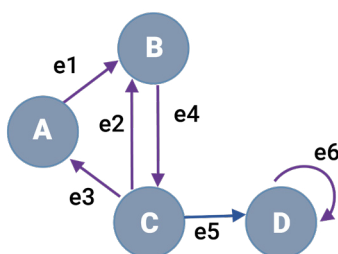
[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)



Actividades de aprendizaje recomendadas

Antes de desarrollar las siguientes actividades, haga una revisión de la unidad referente a grafos descrita en la *Guía Didáctica (Guamán, 2017)*.

1. Describa ejemplos en los cuales se pueda utilizar grafos como estructuras de datos.
 - a. _____
 - b. _____
 - c. _____
 - d. _____
2. Elabore un grafo y obtenga los valores correspondientes a los conceptos mencionados en el literal 5.1 de la *Guía Didáctica*.
3. Dado el siguiente grafo, elabore su correspondiente matriz de adyacencia.



Finalmente, para probar sus conocimientos lo invito a desarrollar la siguiente autoevaluación.

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo revisar previamente los contenidos de la quinta unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial.



Autoevaluación 5

Lea detenidamente cada una de las siguientes afirmaciones y determine si son verdaderas o falsas.

1. () En el ámbito computacional, un grafo está compuesto por un conjunto de nodos y arcos relacionados entre sí.
2. () Formalmente, un grafo se define como $G = (n, c)$, siendo **n** un nodo dado del grafo, y **c** los caminos asociados al nodo.
3. () Los grafos dirigidos, gráficamente se representan con aristas carentes de una dirección específica.
4. () Se conoce al grado de entrada como al número de arcos que inciden en un nodo.
5. () Como orden del grafo se conoce la secuencia de arcos y vértices necesarios para alcanzar un nodo destino.
6. () Las matrices de adyacencia requieren la utilización de n^2 espacios de memoria para la información de los grafos.
7. () La implementación de listas de adyacencia, al igual que con las matrices de adyacencia, pueden requerir que haya una utilización innecesaria de memoria para el almacenamiento de información.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

8. () Los recorridos en listas de adyacencia trabajan solamente sobre los vértices existentes en el grafo.
9. () Las matrices dispersas, pueden contener tanta información como las matrices de adyacencia, pero no ocupan tanta memoria como ellas.
10. () Las matrices dispersas se apoyan en arreglos alternativos para poder representar solamente los enlaces existentes en el grafo.

[Ir al solucionario](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)



Semana 14



Unidad 6. Archivos

Los datos creados en los programas existen solamente durante su ejecución y se pierden al cerrar el programa, pero la mayoría de las veces se requiere de la persistencia de los datos, es decir que, al salir del programa, estos queden almacenados y permitan el acceso a ellos, ya sea desde el mismo u otros programas.

6.1. Concepto de Archivos

Es un conjunto de unidades de información (bits) almacenados en un dispositivo; poseen una identificación única y una extensión, la cual determina qué tipo de archivo es y sus características. Dentro de estos encontramos pequeños paquetes de datos expresados en bits, los mismos que se encuentran ordenados en registros o líneas, dependiendo su modo de agrupación, de quien haga el archivo.

Entre las operaciones que podemos realizar sobre archivos tenemos: creación, eliminación, modificación, reubicación, compresión, renombrado y dependiendo del tipo de archivo, pueden ser ejecución o activación.

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

Para ampliar esta información, diríjase al documento Archivos en Java.pdf, compartido en el Entorno Virtual de Aprendizaje.

6.2. Lectura de archivos

Un programa no puede manipular los datos de un archivo directamente. Para usar un archivo, un programa siempre debe abrir el archivo y asignarlo a una variable, que llamaremos el archivo lógico. Todas las operaciones sobre un archivo se realizan a través del archivo lógico.

La lectura de datos de un archivo, independientemente del lenguaje de programación utilizado para ello, debe seguir la siguiente lógica:

```
archivo = abrir(<nombre>)
```

- *Nombre* es un string que contiene el nombre del archivo.
- *Archivo* es el archivo lógico a través del que se manipulará el archivo.
- *Abrir* es la palabra reservada (según el lenguaje utilizado), que permite cargar el archivo <nombre> en el archivo lógico para poder ser manipulado.

Si el archivo no existe, ocurrirá un error de entrada y salida (IOException).

La manera más simple de leer el contenido de un archivo es hacerlo línea por línea. Para esto, basta con poner el archivo lógico en un ciclo for, ej:

For linea in archivo:

```
# hacer algo
```

Una vez que los datos han sido leídos del archivo, hay que cerrarlo:

```
archivo.close()
```

Estas operaciones, podrá verlas implementadas en lenguaje Java en el documento *Archivos en Java.pdf*, compartido en el Entorno Virtual de Aprendizaje.

6.3. Escritura de archivos

El ejemplo anteriormente mencionado, supone que el archivo por leer existe previamente, y está listo para ser abierto y leído. Ahora veremos cómo crear los archivos y cómo escribir datos en ellos, para que otro programa después pueda así mismo abrirlos y leerlos.

Dependiendo del lenguaje de programación, se utilizarán ciertos parámetros que indican el tipo de archivo y cómo se va a trabajar sobre el mismo.

Por ejemplo, en el lenguaje C se utilizan los parámetros “w” o “a” para indicar que un archivo va a ser abierto para escribir en él desde el comienzo o se va a continuar escribiendo desde el final respectivamente.

El lenguaje Java en este caso utiliza los objetos “*FileWriter*” y “*PrintWriter*”, cuyo trabajo es:

- **FileWriter:** genera un archivo en el cual podamos escribir con el objeto **PrintWriter**.
- **PrintWriter:** este objeto posee los métodos necesarios para poder imprimir líneas completas en el archivo creado con la clase **FileWriter**.

Un ejemplo completo de código en Java, para la lectura y escritura de archivos, lo podrá encontrar en el documento *Archivos en Java.pdf*, compartido en el Entorno Virtual de Aprendizaje.

6.4. Buffering

El proceso de copiar archivos de texto sería muy lento si tuviéramos un texto extenso, dado que el programa tiene que realizar una conexión al archivo por cada letra. Para solucionar este inconveniente se ha creado el buffer, el cual es un archivo de transición entre java y el archivo donde copia o pega. Se genera un espacio en la memoria interna entre el programa y su copia. La información del archivo se volcará en el buffer y java accederá al buffer para cargar o descargar la información poco a poco.

Muy bien estimado estudiante: hemos llegado a la finalización del estudio de la asignatura en este segundo bimestre y, por ende, del presente ciclo. Esperamos que los conocimientos adquiridos sean los básicos y necesarios para iniciar el desarrollo de aplicaciones que tengan eficiencia y eficacia. Por ello, lo invito a desarrollar las siguientes actividades de aprendizaje, esperando que todo le vaya de lo mejor. ¡Adelante y éxitos!

Finalmente, para probar sus conocimientos lo invito a desarrollar la siguiente autoevaluación.

Estimado estudiante: para proceder a dar respuesta a las preguntas de esta autoevaluación, le recomiendo revisar previamente los contenidos de la sexta unidad. Recuerde que esta actividad tiene como propósito medir los conocimientos de la temática y prepararlo para su examen presencial.



Autoevaluación 6

Lea detenidamente cada una de las siguientes afirmaciones y determine si son verdaderas o falsas.

1. () Los datos creados en tiempo de ejecución, se conservan en un archivo llamado buffer.
2. () La persistencia de datos, se refiere a la característica de permanecer almacenados en algún tipo de medios.
3. () Un programa, diferente al que crea un grupo de datos persistentes, ¿podrá tener acceso a esos datos?
4. () La extensión de un archivo, indica el tamaño del mismo.
5. () El Archivo lógico, es una variable utilizada por los lenguajes de programación para la manipulación de archivos.
6. () Las operaciones de apertura y cierre de archivos son necesarias siempre, independientemente del lenguaje de programación.
7. () El objeto FileWriter genera un archivo en el cual podremos escribir a través del objeto PrintWriter.
8. () El objeto PrintWriter, contiene métodos que permiten la impresión de líneas completas de un archivo.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

9. () El buffer, es un archivo de transición, entre el lenguaje de programación y el archivo donde se almacenan los datos.
10. () Las operaciones de conexión del buffer con el archivo destino, se realizan por cada nueva letra que se digita.

[Ir al solucionario](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)



Actividades finales del bimestre



Semanas 15 y 16

Actividad 1

- Revise todos los recursos educativos como preparación para la evaluación presencial.
- Resolución de inquietudes y participación en el chat semanal.
- Revise todos los recursos educativos como preparación para la evaluación presencial.
- Resolución de inquietudes y participación en el chat semanal.

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



4. Solucionario

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
1	c	La característica principal de los tipos de datos estáticos, es que el usuario previamente define su tamaño y este no podrá ser cambiado en la ejecución del programa.
2	c	Los tipos de datos string sirven para manipular cadenas de texto, y entre ese tipo de manipulación se encuentra la de extracción de partes de esa cadena.
3	c	Homogeneidad es un sinónimo de igualdad, por lo cual todos sus elementos deben ser del mismo tipo.
4	c	Aunque con fines didácticos se utilicen datos de tipo entero en los ejemplos para el estudio, los datos por almacenar pueden ser registros que brinden información más completa.
5	c	Entre las formas de ingreso de datos a un arreglo se encuentran: durante la definición, por medio del teclado, desde una fuente externa, etc.
6	b	Aunque existen muchas operaciones que se pueden aplicar sobre arreglos, las principales son: lectura, modificación, búsqueda, ordenación y eliminación.
7	c	Un requerimiento de este tipo de búsqueda es que sus datos estén previamente ordenados; la ordenación también puede darse alfabéticamente.
8	b	Para declarar un arreglo, los corchetes deben ir, bien sea en el tipo de datos o en el nombre del arreglo, previamente al signo "=".
9	a	En todos los lenguajes de programación se utiliza la nomenclatura en ese orden: filas y columnas.

Autoevaluación 1		
Pregunta	Respuesta	Retroalimentación
10	a	La expresión <nombre del arreglo>.length permite conocer el número de elementos de un arreglo.

Ir a la
autoevaluación

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Autoevaluación 2		
Pregunta	Respuesta	Retroalimentación
1	V	Un TAD tiene un conjunto de valores y operaciones; cumple con los principios de abstracción y ocultación de la información.
2	F	La pila es una estructura LIFO “Last In First Out”.
3	V	Se conoce con el nombre de Push a la operación de inserción de elementos, tanto en pilas como en colas.
4	F	Los procesos de inserción y extracción de elementos de pilas y colas son de forma ordenada.
5	V	Se conoce con el nombre de Pop a la operación de extracción de elementos, tanto en pilas como en colas; en este caso, es correcto ya que en las colas se extrae siempre el primer objeto que ha ingresado.
6	V	Las colas son estructuras FIFO “First In First Out”.
7	V	El concepto se apega a la definición de “conjunto”.
8	F	Esa es la definición de intersección de conjuntos.
9	F	Esa es la definición de diferencia simétrica
10	V	Es imprescindible la utilización de un nuevo parámetro que permita obtener un nuevo criterio de evaluación.

Ir a la
autoevaluación

Autoevaluación 3		
Pregunta	Respuesta	Retroalimentación
1	V	Las estructuras dinámicas, por su naturaleza, requieren que sus elementos sean creados o eliminados de acuerdo con sus necesidades.
2	F	Este tipo de variables son utilizadas para estructuras dinámicas.
3	V	Las variables de tipo puntero son imprescindibles en la estructura de los nodos, ya que permiten referir al siguiente elemento.
4	F	Las listas enlazadas son estructuras dinámicas, por cuanto no es necesario especificar previamente su tamaño.
5	F	Lo doblemente enlazada se refiere a que posee dos variables de tipo puntero para direccionar hacia otros nodos.
6	V	El nodo raíz o cabeza es la única forma de llegar hacia cualquiera de los nodos.
7	V	Las listas enlazadas no cuentan con elementos que no posean información.
8	F	Si cambiamos el orden de entrada de elementos a la ya establecida en los TAD cola, entonces se pierde el concepto y simplemente no sería una cola.
9	V	La pérdida de la dirección de un nodo significa que no hay forma de acceso a él.
10	V	Las variables de tipo puntero siempre deben apuntar hacia algún lugar, en el caso último nodo al no existir más datos debe ser apuntado hacia null.

Ir a la
autoevaluación

Autoevaluación 4		
Pregunta	Respuesta	Retroalimentación
1	c	El concepto de árbol se basa en un único nodo raíz.
2	c	Es el nombre proporcionado en la nomenclatura de elementos de los árboles.
3	a	Es el máximo número de descendientes directos que puede tener un nodo.
4	b	La raíz siempre tiene nivel 0, sus hijos nivel 1, y así sucesivamente.
5	a	Es el nivel del nodo hoja, más alejado de la raíz.
6	b	Los tres tipos de recorridos son: preorden, inorden y posorden.
7	c	Esa es su definición.
8	b	La definición de "binario" quiere decir que tiene dos variables de enlace.
9	a	Esta puede tender a desequilibrarse de acuerdo con el orden de ingreso de elementos.
10	b	Los árboles AVL adaptan su forma para mantener el equilibrio de las ramas.

Ir a la
autoevaluación

Autoevaluación 5		
Pregunta	Respuesta	Retroalimentación
1	V	Es parte de la definición de Grafos.
2	F	La definición es $G(V, E)$, donde V representa los vértices del grafo y E el conjunto de aristas.
3	F	Los grafos dirigidos contienen un direccionamiento o flechas que indican la relación entre sus nodos.
4	V	Es el número de arcos que tienen direccionamiento hacia un determinado nodo.
5	F	Orden del grafo es el número de nodos vértices del grafo.
6	V	Hay que representar la relación de cada nodo con los demás nodos del grafo, por lo cual se requieren n^2 espacios de memoria.
7	F	En las listas de adyacencia se intenta evitar justamente reservar espacio para aquellos arcos que no contienen ningún tipo de información.
8	V	Por cuanto no existen espacios no utilizados, se trabaja únicamente sobre vértices existentes.
9	V	Ocupan menos espacio de memoria debido a que se optimiza la organización de la información.
10	V	Utilizan arreglos para almacenar: valores numéricos de la relación, índices de las columnas y punteros a inicios de fila.

Ir a la
autoevaluación

Autoevaluación 6		
Pregunta	Respuesta	Retroalimentación
1	F	Se conservan en la memoria Ram, y se pierden al cerrar el programa.
2	V	Es la característica de permanecer grabados en algún medio.
3	V	Si, mediante las sentencias de código adecuadas para ello.
4	F	Indica el tipo de archivo al que pertenece.
5	V	Es declarado por el lenguaje de programación para comunicarse con el archivo original.
6	V	Son operaciones básicas en el tratamiento de archivos.
7	V	Es la función principal de este objeto.
8	V	Son funciones básicas de este tipo de objeto.
9	V	Permite optimizar con ello el tiempo de ejecución del programa, evitando la conexión por cada carácter.
10	F	Es una operación que el buffer evita que suceda.

[Ir a la
autoevaluación](#)



5. Referencias bibliográficas

Allen, M. (2013). *Estructuras de datos en Java. (Cuarta edición)*. Madrid: Pearson Education.

Guamán, F. (2017). *Guía didáctica: Estructuras de datos*. Loja, Ecuador: Editorial Universidad Técnica Particular de Loja.

Humeau, L. (2013). *Pilas, colas y listas estructura de datos*. Recuperado de <https://es.slideshare.net/diwal10/pilas-colas-y-listas-estructura-de-datos>

MasterHeHeGar. (2014). *Nodos y Punteros (EDDJava)*. [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=lwEay3h89zc>

Programación ATS. (04 de febrero de 2017). *Programación en Java || Introducción || Tipos de datos no primitivos y cadenas*. [Archivo de video]. Recuperado de <https://youtu.be/URPsISB-5nk>

Rodríguez, A. (2017). *Aprendizaje basado en problemas*. [Archivo de video]. Recuperado de <https://www.youtube.com/watch?v=CuMbD6Tak1c>

Sánchez, J. (2017). *Ejemplo gráfico de la construcción de un árbol AVL*. Recuperado de <https://sites.google.com/site/jlscestructuras/ejemplo-grafico-de-la-construccion-de-un-arbol-avl>

Sun Microsystems. (2008). *The Java Tutorials. Tipos de datos primitivos*. Recuperado de <http://www.codexion.com/tutorialesjava/java/nutsandbolts/datatypes.html>

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos



6. Recursos

Documento 1. Estructuras de datos - Arreglos

[Índice](#)[Primer bimestre](#)[Segundo bimestre](#)[Solucionario](#)[Referencias bibliográficas](#)[Recursos](#)

Contenidos

- Arreglos
 - Definición
 - Recorrido de arreglos
 - Ejercicios
- Trabajando con arreglos



2

soy+ utpl

Arreglos

- Definición: colección finita, homogénea y ordenada de elementos.
 - Finita, porque todo arreglo tiene un límite.
 - Homogénea, porque todos los elementos son del mismo tipo.
 - Ordenada, porque se puede determinar cuál es el enésimo elemento.
- Es una estructura de datos que puede almacenar N elementos de un mismo tipo de datos.
- Normalmente, el arreglo se utiliza para almacenar tipos de datos, tales como: **char, int o float**.



3

Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Arreglos

- Un arreglo tiene dos partes: componentes e índices.
- Componentes: hacen referencia a los elementos que forman el arreglo.
- Índices: permiten referirse a los componentes del arreglo en forma individual.

Representación gráfica de un arreglo

Componentes

8	10	39	29	31	25	32	40	22	38
0	1	2	3	4	5	6	7	8	9

Índices



Ejemplos

- Si tuviéramos un arreglo de caracteres con el texto "ESTRUCTURA", ¿cuál sería el índice de la primera letra "T"?
- Un curso de computación tiene 40 alumnos matriculados. Si tuviese que diseñar un arreglo para registrar las notas de los alumnos, responda lo siguiente:
 1. ¿De qué tipo de dato se definiría?
 2. ¿Qué tamaño tendría el arreglo?
 3. ¿Cuál es el índice inferior del arreglo?
 4. ¿Cuál es el índice superior del arreglo?



Trabajando con arreglos

Existen algunas operaciones que se pueden desarrollar con los arreglos, entre las principales tenemos:

- Inicialización
- Ingresar
- Eliminar
- Insertar
- Modificar
- Presentar
- Ordenar
- Buscar
- Otras...



Trabajando con arreglos

Inicializar un arreglo

Existen varias maneras de inicializar un arreglo; una manera muy sencilla es inicializar en la **declaración de variables**.

A continuación, podemos observar tres formas diferentes de inicializar un arreglo en Java.

```
1  public static void main(String[] args) {  
2  
3      int[] myArray1;  
4      myArray1 = new int[10];  
5  
6      int[] myArray2 = new int[10];  
7  
8      int[] myArray3 = {30, 27, 5, 24, 39};  
9  }
```



Trabajando con arreglos

El acceso a los elementos de un arreglo se realiza por medio de un subíndice; por ejemplo, si se quisiera acceder al primer elemento del siguiente arreglo,

myArray2

8	10	39	29	31	25	32	40	22	38
0	1	2	3	4	5	6			

habrá que hacerlo de la siguiente manera:

```
* nro = myArray2[0];
```

En la variable **nro** se almacenará el valor de 8, para acceder al segundo valor:

```
* nro = myArray2[1];
```

Se almacenará el valor de 10, y así sucesivamente con los demás elementos.



Trabajando con arreglos

Presentar elementos de un arreglo

La presentación de elementos también se realiza por medio de sus subíndices; para ello, también podemos valernos de la sentencia **for**, tal como se muestra en el siguiente método.

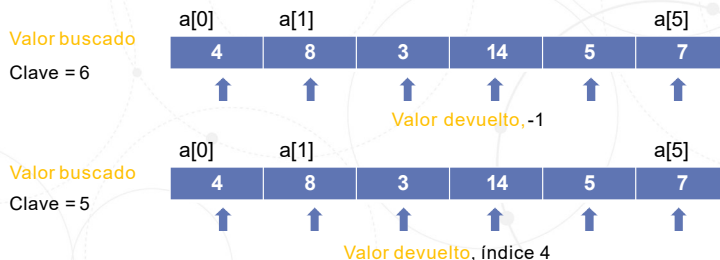
```
15 |
16 | public void presentarArreglo(int arr[]){
17 |     for(int i = 0; i<arr.length; i++){
18 |         System.out.print(arr[i]+" ");
19 |     }
20 | }
21 |
22 | }
```



Trabajando con arreglos

Búsqueda secuencial

- No es necesario que el arreglo esté ordenado.
- Comparar el elemento buscado con cada uno de los elementos, hasta encontrarlo o llegar al fin del mismo.



12 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Trabajando con arreglos

Búsqueda secuencial

- En este caso, el método regresa el índice del elemento encontrado, o el valor -1, si no existe.

```

51 public int Secuencial(int[] X, int elemento){
52     for(int i=0; i<X.length; i++){
53         if(X[i]==elemento)
54             return i;
55     }
56     return -1;
57 }
  
```



13 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Trabajando con arreglos

Eliminar elementos de un arreglo

- Implica la búsqueda del elemento en el arreglo; determinar su existencia y realizar su eliminación lógica.

```

18 public int eliminar(int arre[], int elemento, int n){
19     int pos=Secuencial(arre, elemento);
20     if(pos!=-1){
21         desplazarIzquierda(arre, pos, n);
22         n=n-1;
23     }
24     return n;
25 }
  
```

- Recorrer los elementos de su derecha una casilla hacia la izquierda y reducir en una unidad el número de elementos existentes en el arreglo.



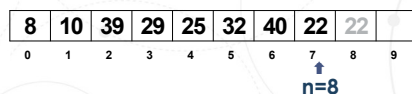
Trabajando con arreglos

Eliminar elementos de un arreglo

- En este ejemplo utilizamos la variable **n** para saber el número de elementos existentes en el arreglo



- El recorrido de elementos hacia la izquierda sería:



Diseña el método necesario para realizar el desplazamiento de los elementos una posición a la izquierda.

Trabajando con arreglos

Insertar elementos en un arreglo

Consiste en agregar nuevos elementos a un arreglo existente en caso de que cuente con espacios disponibles; entre los principales casos tenemos:

- Insertar al inicio del arreglo.
- Insertar en el primer espacio vacío del arreglo.
- Insertar cuidando que no se repitan los elementos.
- Insertar en una posición previamente determinada.
- En caso de que el arreglo esté ordenado, insertar en el lugar que le corresponde.

Cada uno de los casos mencionados tendrá una manera diferente de ser tratado.



Trabajando con arreglos

Insertar elementos en un arreglo

En el caso, inserción de un elemento nuevo en un arreglo desordenado, sin importar si se repite, se implementaría, tal como lo podemos observar en el siguiente método.

```
88 public int insertaCasol(int arre[],int elemento, int n){
89     if(n < arre.length){
90         arre[n]=elemento;
91         n=n+1;
92     }
93     return n;
94 }
```



Trabajando con arreglos

Insertar elementos en un arreglo

Si deseamos en cambio, controlar que no se repitan los elementos; podríamos utilizar el método de búsqueda previamente estudiado, así:

```
8 public int insertaCaso2(int arre[],int elemento, int n){
9     if(n < arre.length){
10         if(Secuencial(arre, elemento)==-1){
11             arre[n]=elemento;
12             n=n+1;
13         }
14     }
15     return n;
16 }
```



Realice los métodos necesario para los demás casos de inserción.

Trabajando con arreglos

Ordenar un arreglo

Existen algunos algoritmos que difieren tanto en su complejidad como en rendimiento; estos algoritmos pueden ser clasificados de la siguiente manera:

- Métodos iterativos. Entre los principales tenemos:
 - Burbuja, Inserción, Selección y Shellsort
- Métodos recursivos. Los principales son:
 - MergeSort y QuickSort



Revise el archivo "REA_Algoritmos de Ordenamiento.pdf" que podrá descargar desde el Entorno Virtual de Aprendizaje.

Trabajando con arreglos

Ordenar un arreglo

Método burbuja:

```

23 public void ordenaBurbuja(int x[]) {
24     int aux;
25     for(int i=1; i<x.length; i++)
26         for(int j=0; j<x.length - i; j++)
27             if (x[j] > x[j+1]) {
28                 aux = x[j];
29                 x[j] = x[j+1];
30                 x[j+1] = aux;
31             }
32 }
  
```



Trabajando con arreglos

Ordenar un arreglo

Método QuickSort:

```

34 public void ordenaQuick(int x[], int ini, int fin) {
35     int aux;
36     int a = ini;
37     int b = fin;
38     int pivot = x[(ini+fin)/2];
39     do {
40         while(x[a] < pivot)
41             a++;
42         while(x[b] > pivot)
43             b--;
44         if(a < b) {
45             aux = x[a];
46             x[a] = x[b];
47             x[b] = aux;
48         }
49         a++;
50         b--;
51     } while(a <= b);
52     if(ini < b)
53         ordenaQuick(x, ini, b);
54     if(a < fin)
55         ordenaQuick(x, a, fin);
56 }
  
```



Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Trabajando con arreglos

Búsqueda binaria

- Necesariamente el arreglo debe estar ordenado.
- El algoritmo realiza comparaciones para determinar si el valor buscado está ubicado en el punto medio del arreglo o estaría en su mitad superior o inferior.
- De no encontrarse en el primer paso y una vez determinada su posible ubicación, se subdivide el arreglo, y así sucesivamente hasta encontrar el valor o determinar su inexistencia.
- Al igual que en el caso de la búsqueda secuencial, este algoritmo devolverá la posición del elemento buscado, o un valor de -1, en caso de que tal elemento no se encuentre en el arreglo.



22 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Índice

Primer bimestre

Segundo bimestre

Solucionario

Referencias bibliográficas

Recursos

Trabajando con arreglos

Búsqueda Binaria

- Índices de la lista son **bajo = 0** y **alto = n-1**

Calcular el índice del punto central del arreglo

$$\text{central} = (\text{bajo} + \text{alto}) / 2$$

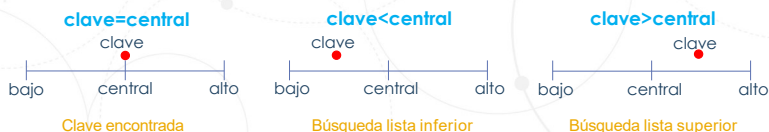
$$\text{central} = (0 + 7) / 2 = 3$$

Valor buscado

Clave = 225

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]
13	44	75	100	120	275	325	510

Comparar el valor de este elemento central con la clave



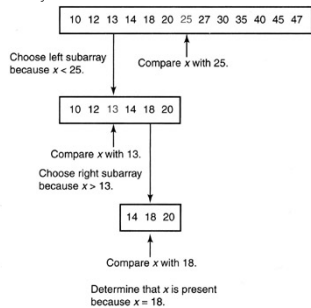
23 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Trabajando con arreglos

Búsqueda binaria

Binary Search.



```

public int BusquedaBinaria(int[] X, int elemento){
    int _inicio, _fin, _medio;
    _inicio=0;
    _fin=X.length;
    while(_inicio<=_fin){
        _medio=( _inicio+_fin)/2;
        if(X[_medio]==elemento){
            return _medio;
        }else{
            if(X[_medio]>elemento){
                _fin=_medio-1;
            }else{
                _inicio=_medio+1;
            }
        }
    }
    return -1;
}
    
```

Trabajando con arreglos

Comparación entre búsquedas secuencial y binaria

Números de elementos examinados

Tamaño de la lista	Búsqueda binaria	Búsqueda secuencial
1	1	1
10	4	10
1.000	11	1.000
5.000	14	5.000
100.000	18	100.000
1.000.000	21	1.000.000



Preguntas



Gracias

[Ir al contenido](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)

Documento 2. Estructuras de datos - Arreglos 2D



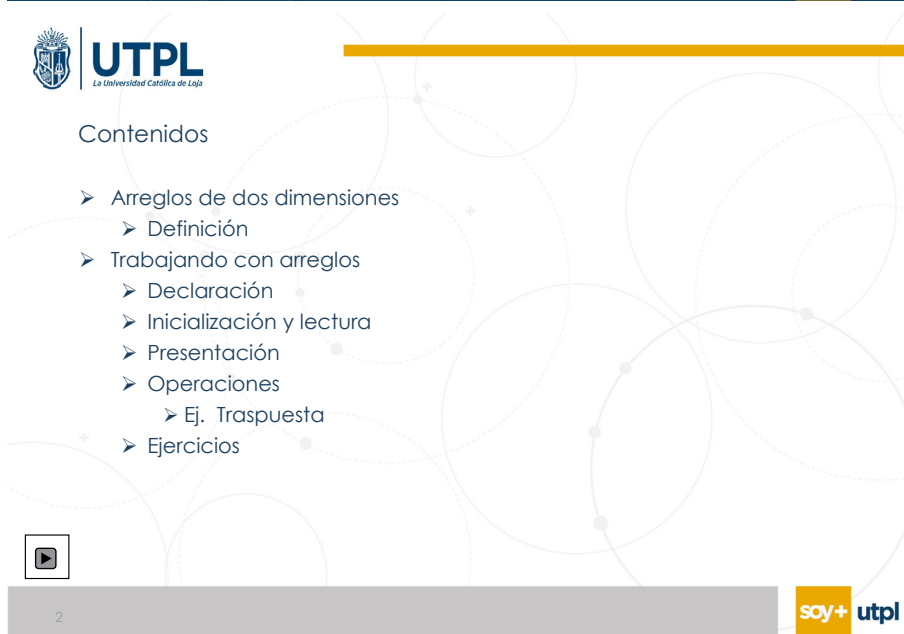
UTPL
La Universidad Católica de Loja

ESTRUCTURAS DE DATOS Y ALGORITMOS

ARREGLOS DE MÁS DE UNA DIMENSIÓN
M.Sc. Franco Guamán Bastidas

Universidad Técnica Particular de Loja
Departamento de Ciencias de la Computación y Electrónica
Sección Departamental de Inteligencia Artificial


 **soy+ utpl**



UTPL
La Universidad Católica de Loja

Contenidos

- Arreglos de dos dimensiones
 - Definición
- Trabajando con arreglos
 - Declaración
 - Inicialización y lectura
 - Presentación
 - Operaciones
 - Ej. Traspuesta
 - Ejercicios

 **soy+ utpl**

2

Arreglos de dos dimensiones

Su lógica no difiere en gran medida de las ya revisadas; entre los factores para tener en cuenta, se encuentran:

- Se aumentará un índice más por cada nueva dimensión del arreglo.
- Para su recorrido se utilizará un proceso o sentencia de control repetitiva más por cada nueva dimensión.
- En Java un arreglo bidimensional es tratado como un arreglo de arreglos; por lo cual, debemos tener un especial cuidado cuando se haga referencia a determinada columna, por ejemplo para conocer su tamaño.



3 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Arreglos de dos dimensiones

- Gráficamente, un arreglo bidimensional se representa de la siguiente forma:

• Representación gráfica de un arreglo

		COLUMNAS			
		0	1	2	3
FILAS	0	[0][0]	[0][1]	[0][2]	[0][3]
	1	[1][0]	[1][1]	[1][2]	[1][3]
	2	[2][0]	[2][1]	[2][2]	[2][3]
	3	[3][0]	[3][1]	[3][2]	[3][3]

Posición de un elemento dentro de una matriz.



4 Estructuras de Datos y Algoritmos - Franco Guamán

soy+ utpl

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Trabajando con arreglos

Inicializar un arreglo

Su declaración e inicialización es similar a la de los arreglos unidimensionales.

Tipo

Número de elementos

int[][] myMatriz = new int[x][y];

nombreArray



Trabajando con arreglos

Lectura de un arreglo

Para su lectura se debe incrementar los procesos repetitivos para control de la nueva dimensión.

```

4 public void leerArreglo(int mat[][]) {
5     for(int i=0; i<mat.length; i++){
6         for(int j=0; j<mat[0].length; j++){
7             System.out.print("\nDato ["+i+"] ["+j+"]: ");
8             mat[i][j]=leer.nextInt();
9         }
10    }
11 }
  
```



Trabajando con arreglos

EL acceso a los elementos de un arreglo se realiza por medio de sus correspondientes subíndices; por ejemplo, para acceder al número 7, sería de la siguiente manera:

```
myMatriz
  0  10 39 29 31
  1   8 23  7 12
  2  56 22 17 37
    0  1  2  3

nro = myMatriz[1][2];
```

Debemos tener en cuenta, que siempre mencionaremos en primer lugar el índice que se refiere a las filas y, luego, a las columnas.



Trabajando con arreglos

Presentar elementos de un arreglo

La presentación de elementos igualmente incrementa la utilización de procesos repetitivos.

```
23 public void presentarArreglo(int mat[][]){
24     for(int i=0; i<mat.length; i++){
25         for(int j=0; j<mat[0].length; j++){
26             System.out.print(mat[i][j]+" ");
27         }
28         System.out.println("\n");
29     }
30 }
```



Trabajando con arreglos

Operaciones en un arreglo bidimensional

Aunque podemos realizar las operaciones citadas con los arreglos unidimensionales, los arreglos multidimensionales son comúnmente utilizadas para realizar operaciones de tratamiento de matrices como:

- Multiplicación
- Determinante
- Traspuesta
- Multiplicación por escalar
- Etc.



Trabajando con arreglos

Matriz traspuesta. Este ejemplo cambia de posición sus elementos; así, lo que en un inicio eran filas, se convierten en columnas.

	0,0	0,1	0,2	0,3
A	1,0	1,1	1,2	1,3
	2,0	2,1	2,2	2,3
	3,0	3,1	3,2	3,3

	0,0	1,0	2,0	3,0
A^T	0,1	1,1	2,1	3,1
	0,2	1,2	2,2	3,2
	0,3	1,3	2,3	3,3

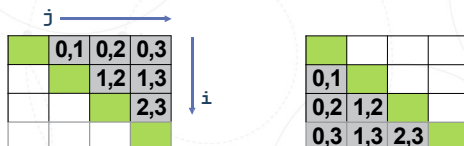
Este ejemplo sencillo muestra las bases para el desarrollo de otros problemas relacionados con matrices.

Debemos primeramente conocer la zona sobre la cual se va a trabajar, y cuál será el resultado esperado.



Trabajando con arreglos

Traspuesta: los índices iniciales y los finales son:



- Así, el primer proceso repetitivo (*i*) debe generar números desde el 0 hasta números de filas -1
- El segundo proceso repetitivo (*j*), en cambio, no tomará en cuenta la diagonal principal, por lo cual debe empezar en el siguiente número después de *i* (*i*+1) y terminará en el número de columnas (*A*[0].length)

Trabajando con arreglos

Los índices resultantes son una simple inversión de orden de los ya generados; por lo cual, solamente se hará un intercambio de posiciones, así:

```

32 public void traspuesta(int mat[][]) {
33     int aux;
34     for(int i=0; i<mat.length-1; i++)
35         for(int j=i+1; j<mat[0].length; j++){
36             aux = mat[i][j];
37             mat[i][j] = mat[j][i];
38             mat[j][i] = aux;
39         }
40     }
  
```



Trabajando con arreglos

Dependiendo de la complejidad del ejercicio solicitado, se debe incrementar el número de procesos repetitivos, o reutilizarlos generados para alguna otra serie.



Desarrolle los ejercicios planteados en el REA "Ejercicios con arreglos", subido al EVA.
Cualquier consulta, diríjase a su docente a cargo.



13 Estructuras de Datos y Algoritmos - FrancoGuamán

soy+ utpl

Preguntas



14 Estructuras de Datos y Algoritmos - FrancoGuamán

soy+ utpl

Índice

Primer
bimestre

Segundo
bimestre

Solucionario

Referencias
bibliográficas

Recursos

Gracias

[Ir al contenido](#)

[Índice](#)

[Primer
bimestre](#)

[Segundo
bimestre](#)

[Solucionario](#)

[Referencias
bibliográficas](#)

[Recursos](#)