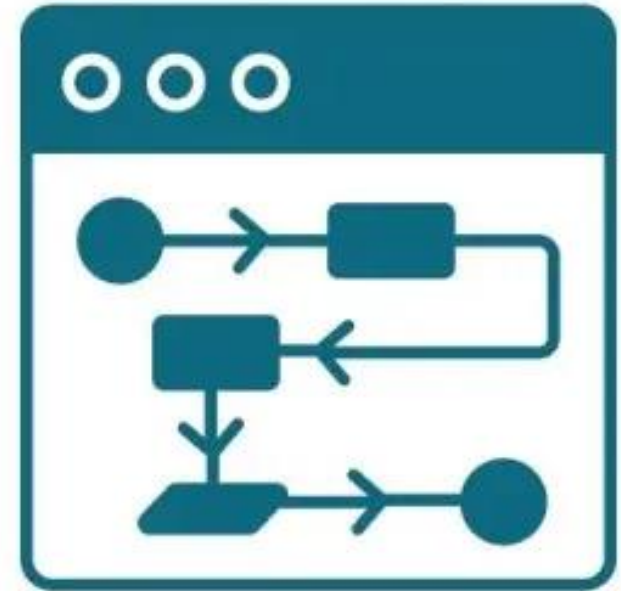




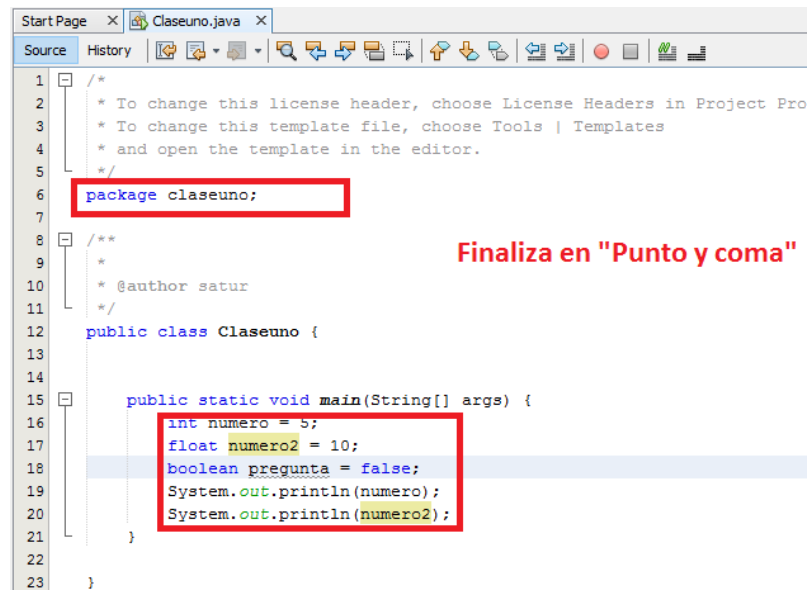
Algoritmos y Estructura de Datos



DOCENTE: MARX DANLY LEÓN TRUJILLO
FACULTAD DE ING. INDUSTRIAL Y SISTEMAS
E.P. INGENIERÍA DE SISTEMAS

ALGORITMOS Y ESTRUCTURA DE DATOS

- Introducción.
- Tipos Abstracto de Datos (TAD).
- Tipos de Datos.
- Estructura de Datos.
- Tipos de Datos definidos por el usuario.



```
1  /*
2  * To change this license header, choose License Headers in Project Pro
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package claseuno;
7
8  /**
9   *
10   * @author satur
11   */
12  public class Claseuno {
13
14
15      public static void main(String[] args) {
16          int numero = 5;
17          float numero2 = 10;
18          boolean pregunta = false;
19          System.out.println(numero);
20          System.out.println(numero2);
21      }
22
23  }
```

Finaliza en "Punto y coma"

ALGORITMOS Y ESTRUCTURA DE DATOS

INTRODUCCIÓN.

No existe una regla precisa para escribir un programa que resuelva un dado problema práctico. Al menos por ahora escribir programas es en gran medida un arte. Sin embargo con el tiempo se han desarrollado una variedad de conceptos que ayudan a desarrollar estrategias para resolver problemas y comparar a priori la eficiencia de las mismas.

Por ejemplo:

Supongamos que queremos resolver el “Problema del Agente Viajero” (TSP, por “Traveling Sales man Problem”) el cual consiste en encontrar el orden en que se debe recorrer un cierto número de ciudades (esto es, una serie de puntos en el plano) en forma de tener un recorrido mínimo.

ALGORITMOS Y ESTRUCTURA DE DATOS

Este problema surge en una variedad de aplicaciones prácticas, por ejemplo: encontrar caminos mínimos para recorridos de distribución de productos o resolver el problema de “la vuelta del caballo en el tablero de ajedrez”, es decir, encontrar un camino para el caballo que recorra toda las casillas del tablero pasando una sola vez por cada casilla.

Una forma abstracta de plantear una estrategia es en la forma de un “algoritmo”, es decir una secuencia de instrucciones cada una de las cuales representa una tarea bien definida y puede ser llevada a cabo en una cantidad finita de tiempo y con un número finito de recursos computacionales. Un requerimiento fundamental es que el algoritmo debe terminar en un número finito de pasos, de esta manera él mismo puede ser usado como una instrucción en otro algoritmo más complejo.

ALGORITMOS Y ESTRUCTURA DE DATOS

Entonces, comparando diferentes algoritmos para el TSP entre sí, podemos plantear las siguientes preguntas:

- ¿Da el algoritmo la solución óptima?
- Si el algoritmo es iterativo, ¿converge?
- ¿Cómo crece el esfuerzo computacional a medida que el número de ciudades crece?

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS ABSTRACTO DE DATOS

Una vez que se ha elegido el algoritmo, la implementación puede hacerse usando las estructuras más simples, comunes en casi todos los lenguajes de programación: escalares, arreglos y matrices. Sin embargo algunos problemas se pueden plantear en forma más simple o eficiente en términos de estructuras informáticas más complejas, como listas, pilas, colas, árboles, grafos, conjuntos.

Por ejemplo: el TSP se plantea naturalmente en términos de un grafo donde los vértices son las ciudades y las aristas los caminos que van de una ciudad a otra. Estas estructuras están incorporadas en muchos lenguajes de programación o bien pueden obtenerse de librerías. El uso de estas estructuras tiene una serie de ventajas.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS ABSTRACTO DE DATOS

- ❖ Se ahorra tiempo de programación ya que no es necesario codificar.
- ❖ Estas implementaciones suelen ser eficientes y robustas.
- ❖ Se separan dos capas de código bien diferentes, por una parte el algoritmo que escribe el programador y por otro las rutinas de acceso a las diferentes estructuras.
- ❖ Existen estimaciones bastante uniformes de los tiempos de ejecución de las diferentes operaciones.
- ❖ Las funciones asociadas a cada estructura son relativamente independientes del lenguaje o la implementación en particular. Así, una vez que se plantea un algoritmo en términos de operaciones sobre una tal estructura es fácil implementarlo en una variedad de lenguajes con una performance similar.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS ABSTRACTO DE DATOS

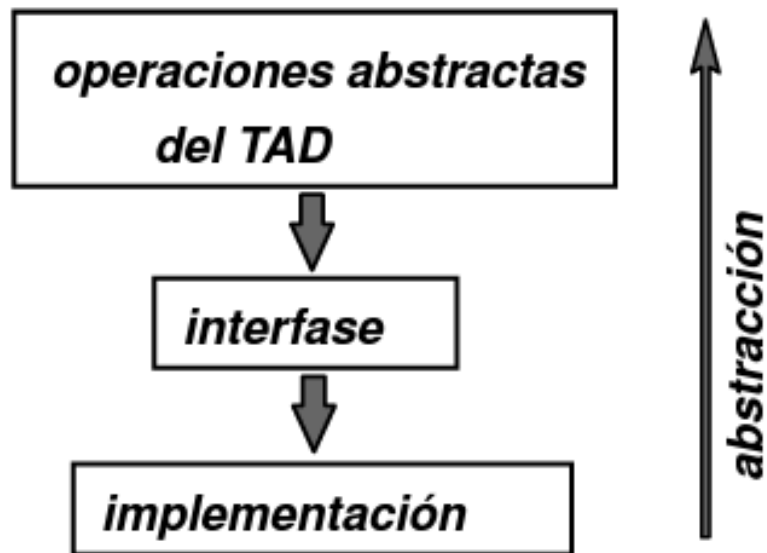
Un “**Tipo Abstracto de Datos**” (TAD) es la descripción matemática de un objeto abstracto, definido por las operaciones que actúan sobre el mismo.

Cuando usamos una estructura compleja como un conjunto, lista o pila podemos separar tres niveles de abstracción diferente, saber las “operaciones abstractas” sobre el TAD, la “interfaz” concreta de una implementación y finalmente la “implementación” de esa interfaz.

En ciencias de la computación un tipo de dato abstracto o tipo abstracto de datos es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS ABSTRACTO DE DATOS



Descripción de los diferentes niveles de abstracción en la definición de un TAD

ALGORITMOS Y ESTRUCTURA DE DATOS

OPERACIONES ABSTRACTAS Y CARACTERÍSTICAS DEL TAD

- Contiene elementos, los cuales deben ser diferentes entre sí.
- No existe un orden particular entre los elementos del conjunto.
- Se pueden insertar o eliminar elementos del mismo.
- Dado un elemento se puede preguntar si está dentro del conjunto o no.
- Se pueden hacer las operaciones binarias bien conocidas entre conjuntos a saber, unión, intersección y diferencia.

ALGORITMOS Y ESTRUCTURA DE DATOS

INTERFAZ DEL TAD

La “interfaz” es el conjunto de operaciones (con una sintaxis definida) que producen las operaciones del TAD. Por supuesto depende del lenguaje a utilizar, si bien algunas veces es también común que una librería pueda ser usada desde diferentes lenguajes y trate de mantener la interfaz entre esos diferentes lenguajes.

Por ejemplo la implementación del TAD en la librería STL es (en forma muy simplificada) la siguiente:

STL es una serie de librerías de C++ que ofrecen una colección de plantillas y algoritmos para trabajar con esas estructuras genéricas.

ALGORITMOS Y ESTRUCTURA DE DATOS

INTERFAZ DEL TAD

```
1. template<class T>
2. class set {
3. public:
4.     class iterator { /* ... */ };
5.     void insert(T x);
6.     void erase(iterator p);
7.     void erase(T x);
8.     iterator find(T x);
9.     iterator begin();
10.    iterator end();
11. };
```

Código 1.7: *Interfaz de la clase set<> [Archivo: stl-set.cpp]*

ALGORITMOS Y ESTRUCTURA DE DATOS

IMPLEMENTACIÓN DEL TAD

Finalmente la “implementación” de estas funciones, es decir el código específico que implementa cada una de las funciones declaradas en la interfaz. Como regla general podemos decir que un programador que quiere usar una interfaz abstracta como el TAD, debería tratar de elaborar primero un algoritmo abstracto basándose en las operaciones abstractas sobre el mismo. Luego, al momento de escribir su código debe usar la interfaz específica para traducir su algoritmo abstracto en un código compilable. En el caso del TAD veremos más adelante que internamente éste puede estar implementado de varias formas, a saber con listas o árboles, por ejemplo. En general, el código que escribe no debería depender nunca de los detalles de la implementación particular que esta usando.

ALGORITMOS Y ESTRUCTURA DE DATOS

ABSTRACCIÓN

Es el mecanismo que permite seleccionar partes de un todo complejo para su consideración, ignorando el resto.

Permite filtrar aquellos aspectos relevantes y obtener soluciones más generales.

ALGORITMOS Y ESTRUCTURA DE DATOS

Un TDA está dado por un grupo de datos que cumplen cierta condición especificada para el TDA, más un conjunto de operaciones que representan el comportamiento del TDA

ALGORITMOS Y ESTRUCTURA DE DATOS

ELEMENTOS DE UN TDA

- ✓ Tipo de dato.
- ✓ Invariantes o axiomas.
- ✓ Operaciones.
- ✓ Precondiciones.
- ✓ Postcondiciones

ALGORITMOS Y ESTRUCTURA DE DATOS

Ejemplo de un TDA

Supongamos que deseamos crear un entero más grande del que se nos proporciona el lenguaje que estamos usando:

Podemos para ello crear el TDA GranEntero, especificando:

- ✓ Tipo de dato.
- ✓ Invariantes (condición que se sigue cumpliendo después de la ejecución de determinadas instrucciones.)
- ✓ Operaciones.
- ✓ Precondiciones.
- ✓ Postcondiciones

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DE DATOS

Un computador procesa información leyendo y moviendo datos codificados binariamente de una dirección de memoria a otra. Sin entrar en detalles demasiado técnicos, es importante destacar que el procesador necesita saber por anticipado la cantidad de bits (binary digit) que van a ocupar en memoria esos datos. Con la emergencia de los lenguajes de programación de alto nivel, se logró que los programadores se liberasen del tedioso proceso de tener que ubicar y desplazar los datos en memoria manualmente. El que esto se haya podido lograr depende, entre otras cosas, del desarrollo de la idea de los tipos de datos.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DE DATOS

La idea básicamente consiste en codificar la información, tal y como esta se presenta en nuestro mundo social y que nos es inmediatamente comprensible, en una serie de bits que pueda procesar el computador. No es lo mismo trabajar con números enteros que trabajar con números racionales; tampoco es lo mismo trabajar con caracteres como los del alfabeto occidental que trabajar con el alfabeto cirílico -sin mencionar los ideogramas chinos. Por otra parte, a diferencia de los números, las palabras no se puede sumar, restar, multiplicar o dividir. En fin, el computador no sólo necesita saber con qué tipo de dato está trabajando, sino que también necesita saber qué tipo de operaciones se pueden realizar con ellos.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DE DATOS

Prácticamente todo lenguaje de programación (con ligeras diferencias) cuenta con los siguientes tipos de datos:

- ✓ Integer
- ✓ Short
- ✓ Long
- ✓ Float
- ✓ Double
- ✓ String
- ✓ Char
- ✓ Boolean

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DE DATOS

Por lo general, a los tipos descritos se les llama tipos primitivos, ya que son los más básicos. Sin embargo, los lenguajes de alto nivel cuentan en sus librerías estándares (core libraries) con una buena cantidad de tipos de datos que son de utilidad para gran variedad de tareas. Y, desde luego, el programador puede crear los tipos de datos que necesite para su problema de dominio específico.

ALGORITMOS Y ESTRUCTURA DE DATOS

Tipo	Representación / Valor	Tamaño (en bits)	Valor mínimo	Valor máximo	Valor por defecto
boolean	true o false	1	N.A.	N.A.	false
char	Carácter Unicode	16	\u0000	\uFFFF	\u0000
byte	Entero con signo	8	-128	128	0
short	Entero con signo	16	-32768	32767	0
int	Entero con signo	32	-2147483648	2147483647	0
long	Entero con signo	64	-9223372036854775808	9223372036854775807	0
float	Coma flotante de precisión simple Norma IEEE 754	32	$\pm 3.40282347E+38$	$\pm 1.40239846E-45$	0.0
double	Coma flotante de precisión doble Norma IEEE 754	64	$\pm 1.79769313486231570E+308$	$\pm 4.94065645841246544E-324$	0.0

ALGORITMOS Y ESTRUCTURA DE DATOS

ESTRUCTURA DE DATOS

Las estructuras de datos en programación son diferentes formas de organizar información para manipular, buscar e insertar estos datos de manera eficiente.

Las estructuras de datos es una rama de las ciencias de la computación que estudia y aplica diferentes formas de organizar información dentro de una aplicación, para manipular, buscar e insertar estos datos de manera eficiente.

Entre las diferentes estructuras de datos podemos encontrar las siguientes:

ALGORITMOS Y ESTRUCTURA DE DATOS

ESTRUCTURA DE DATOS

Las estructuras de datos en programación son diferentes formas de organizar información para manipular, buscar e insertar estos datos de manera eficiente.

Las estructuras de datos es una rama de las ciencias de la computación que estudia y aplica diferentes formas de organizar información dentro de una aplicación, para manipular, buscar e insertar estos datos de manera eficiente.

Entre las diferentes estructuras de datos podemos encontrar las siguientes:

ALGORITMOS Y ESTRUCTURA DE DATOS

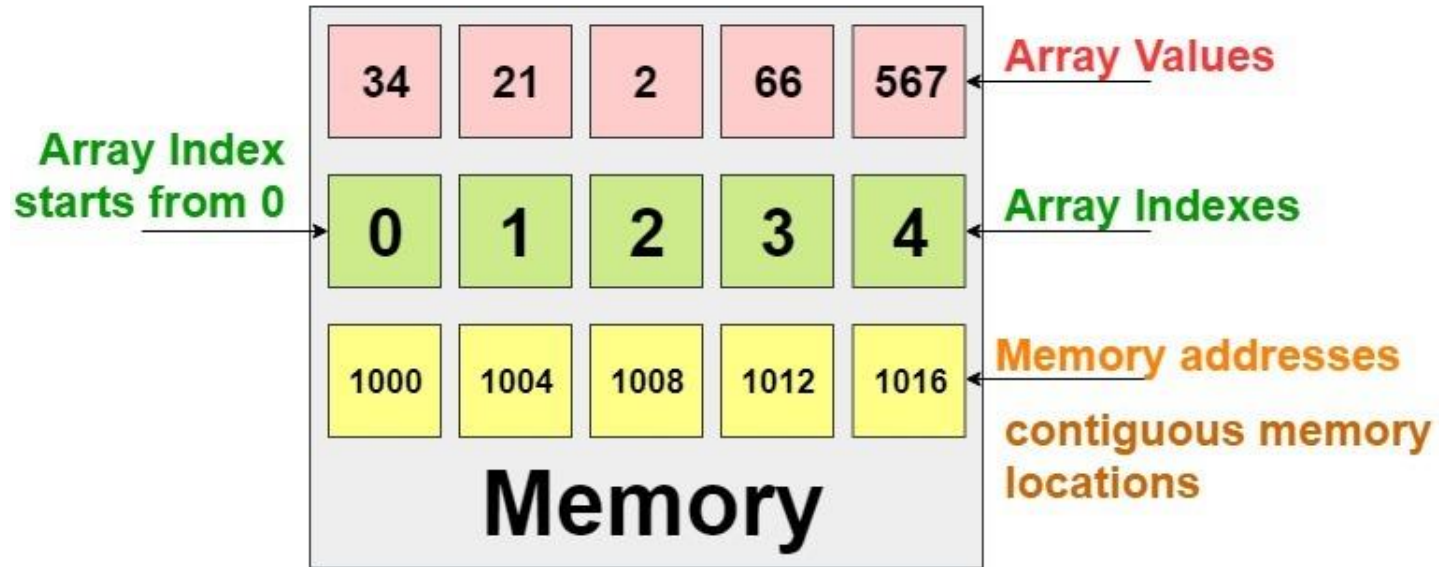
ARRAYS

Los arrays son una estructura que almacena los datos un elemento al lado del otro. En la mayoría de lenguajes de programación esta estructura de datos es de tamaño fijo y no puede guardar datos de diferentes tipos, es decir no puedo guardar valores numéricos y booleanos al mismo tiempo, aunque claramente hay excepciones, por ejemplo: Javascript.

Es recomendable usar arrays cuando el acceso a estos datos se realizan de manera aleatoria, en caso contrario es recomendable usar las listas.

ALGORITMOS Y ESTRUCTURA DE DATOS

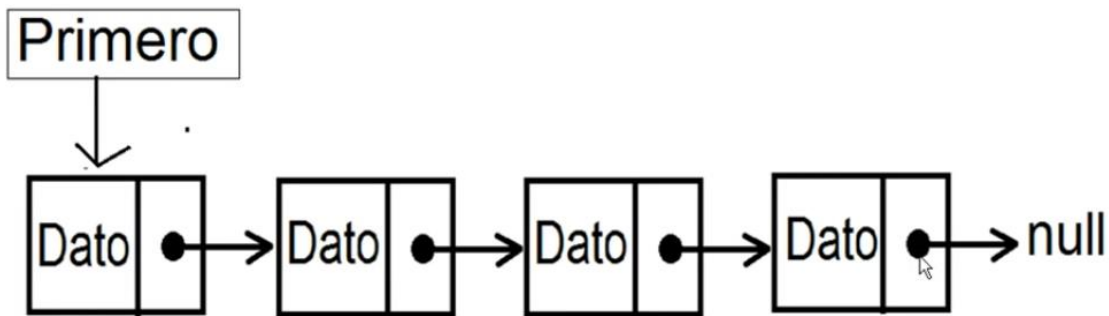
```
int x[ ] = new int[ ] {34, 21, 2, 66, 567};
```



ALGORITMOS Y ESTRUCTURA DE DATOS

LISTAS ENLAZADAS

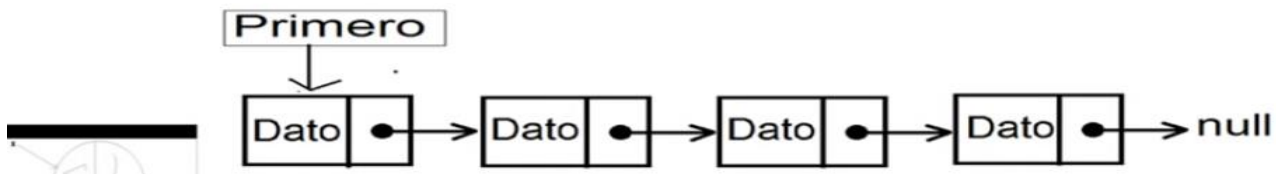
Las listas enlazadas son un tipo de estructura de datos similar a los arrays con la diferencia de que por defecto no tenemos por qué saber la cantidad de elementos que va a contener. Estas listas se componen de nodos los cuales tienen dos atributos: el primero es el item o elemento que va a contener este nodo y el segundo atributo es una referencia al siguiente elemento de la lista.



ALGORITMOS Y ESTRUCTURA DE DATOS

Codificación en Java

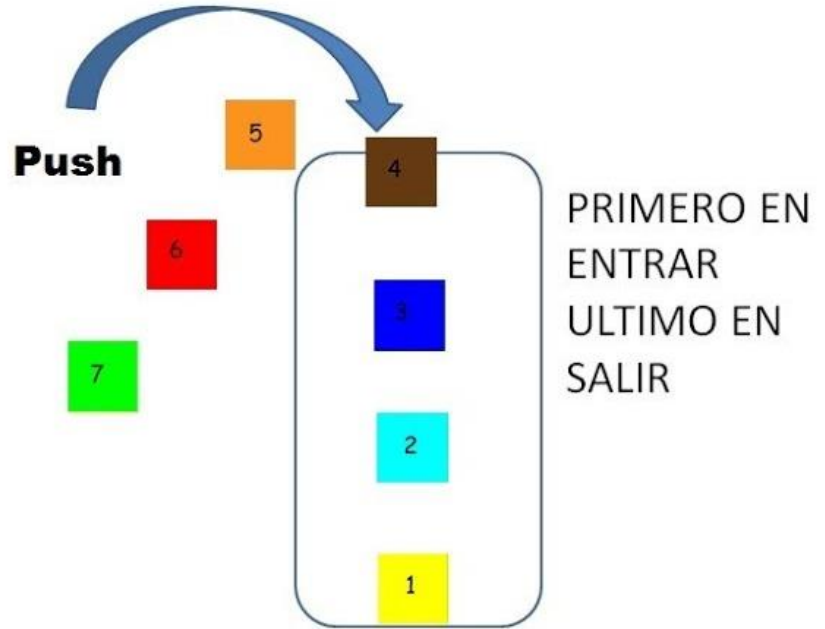
```
public boolean vacia() {  
    if (Primero == null) {  
        return true ;  
    } else {  
        return false ;  
    }  
}
```



ALGORITMOS Y ESTRUCTURA DE DATOS

PILAS

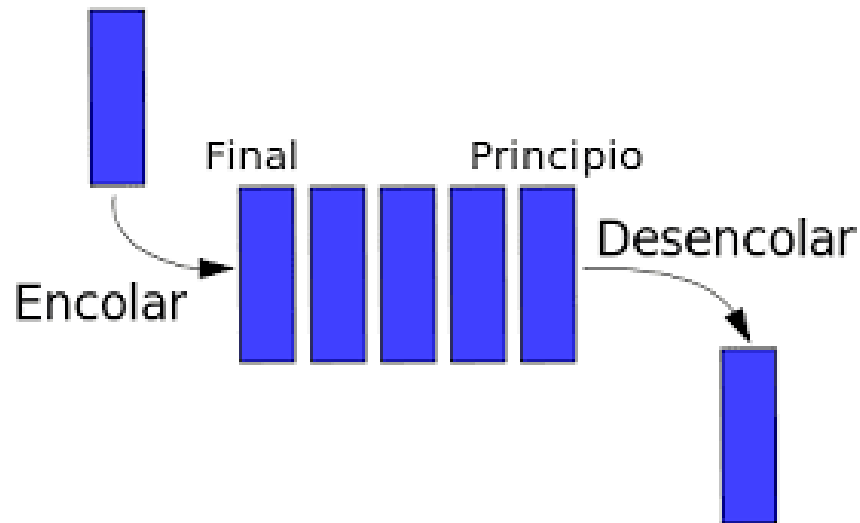
Las pilas son un tipo de listas que tienen la particularidad de sólo poder eliminar o insertar en la cima de la lista. A estas acciones se le conocen como apilar y desapilar y conlleva a que el último elemento que ingresa a la pila sea el primero en salir a lo cual se le conoce como LIFO (Last in First out).



ALGORITMOS Y ESTRUCTURA DE DATOS

COLAS

Esta estructura es otro tipo de lista que nos permite emular el comportamiento de una fila o cola de la vida real donde el primer elemento en ingresar a la fila es el primero en salir, lo que quiere decir que las inserciones (Encolar) se realizan al final y las extracciones (Desencolar) se realizan al frente de la cola, lo cual se conoce como FIFO (First in First out).

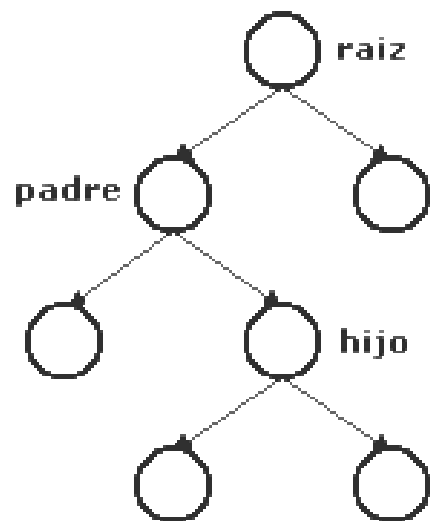


ALGORITMOS Y ESTRUCTURA DE DATOS

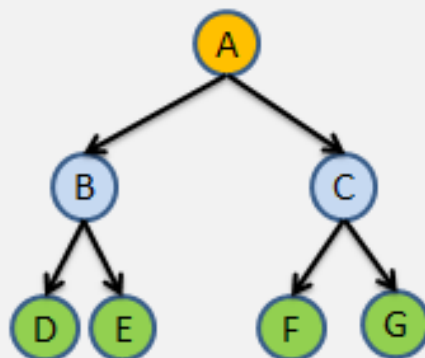
ÁRBOLES BINARIOS

Los arboles binarios son estructuras de datos que se componen de una nueva clase de nodo donde cada uno contiene un item o un valor, una referencia a un nodo que será el hijo izquierdo y otra referencia para el nodo derecho, una característica esencial de los arboles binarios es que la inserción de sus elementos se realizan siguiendo un criterio, si el item del nodo a insertar es menor a su nodo padre, la inserción se realiza por la izquierda y en caso contrario la inserción se realiza por el lado derecho y como consecuencia de esto nuestro árbol siempre estará organizado de tal manera que los hijos izquierdos de cada nodo serán menores a el y los derechos serán mayores, lo cual nos permite realizar búsquedas muy eficientes debido a la organización.

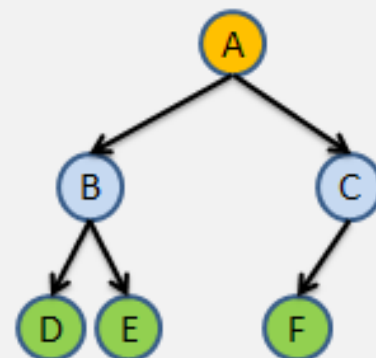
ALGORITMOS Y ESTRUCTURA DE DATOS



Árbol Binario **lleno**



Árbol Binario **NO** lleno



ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DE DATOS ESTRUCTURADOS

Los tipos de datos primitivos que acabamos de ver se caracterizan por poder almacenar un único valor. Salvo este reducido conjunto de tipos de datos primitivos, que facilitan el trabajo con números, caracteres y valores booleanos, todos los demás tipos de Java son objetos, también llamados tipos estructurados o "Clases".

Los tipos de datos estructurados se denominan así porque en su mayor parte están destinados a contener múltiples valores de tipos más simples, primitivos. También se les llama muchas veces "tipos objeto" porque se usan para representar objetos. Puede que te suene más ese nombre.

ALGORITMOS Y ESTRUCTURA DE DATOS

CADENA DE CARACTERES

Aunque las cadenas de caracteres no son un tipo simple en Java, sino una instancia de la clase String, el lenguaje otorga un tratamiento bastante especial a este tipo de dato, lo cual provoca que, en ocasiones, nos parezca estar trabajando con un tipo primitivo.

Aunque cuando declaramos una cadena estamos creando un objeto, su declaración no se diferencia de la de una variable de tipo primitivo de las que acabamos de ver:

```
1 | String nombreCurso = "Iniciación a Java";
```

ALGORITMOS Y ESTRUCTURA DE DATOS

CADENA DE CARACTERES

Y esto puede confundir al principio. Recuerda: Las cadenas en Java son un objeto de la clase String, aunque se declaren de este modo.

Las cadenas de caracteres se delimitan entre comillas dobles, en lugar de simples como los caracteres individuales. En la declaración, sin embargo, no se indica explícitamente que se quiere crear un nuevo objeto de tipo String, esto es algo que infiere automáticamente el compilador.

Las cadenas, por tanto, son objetos que disponen de métodos que permiten operar sobre la información almacenada en dicha cadena. Así, encontraremos métodos para buscar una subcadena dentro de la cadena, sustituirla por otra, dividirla en varias cadenas atendiendo a un cierto separador, convertir a mayúsculas o minúsculas, etc.

ALGORITMOS Y ESTRUCTURA DE DATOS

VECTORES O ARRAYS

Los vectores son colecciones de datos de un mismo tipo. También son conocidos popularmente como arrays e incluso como "arreglos" (aunque se desaconseja esta última denominación por ser una mala adaptación del inglés).

Un vector es una estructura de datos en la que a cada elemento le corresponde una posición identificada por uno o más índices numéricos enteros.

Los elementos de un vector o array se empiezan a numerar en el 0, y permiten gestionar desde una sola variable múltiples datos del mismo tipo.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS DEFINIDOS POR USUARIOS

Además de los tipos estructurados básicos que acabamos de ver (cadenas y vectores) en Java existen infinidad de clases en la plataforma, y de terceros, para realizar casi cualquier operación o tarea que se pueda ocurrir: leer y escribir archivos, enviar correos electrónicos, ejecutar otras aplicaciones o crear cadenas de texto más especializadas, entre un millón de cosas más.

Todas esas clases son tipos estructurados también.

Y por supuesto puedes crear tus propias clases para hacer todo tipo de tareas o almacenar información. Serían tipos estructurados definidos por el usuario.

ALGORITMOS Y ESTRUCTURA DE DATOS

TIPOS ENVOLTORIOS O WRAPPER

Java cuenta con tipos de datos estructurados equivalentes a cada uno de los tipos primitivos que hemos visto.

Así, por ejemplo, para representar un entero de 32 bits (int) de los que hemos visto al principio, Java define una clase llamada Integer que representa y "envuelve" al mismo dato pero le añade ciertos métodos y propiedades útiles por encima.

Además, otra de las finalidades de estos tipos "envoltorio" es facilitar el uso de esta clase de valores allí donde se espera un dato por referencia (un objeto) en lugar de un dato por valor (para entender la diferencia entre tipos por valor y tipos por referencia lee este artículo. Aunque está escrito para C#, todo lo explicado es igualmente válido para Java).

ALGORITMOS Y ESTRUCTURA DE DATOS

