Datos Estructurados

Emmanuel Cruz Hernández
emmanuel_cruzh@ciencias.unam.mx

31 de marzo de 2020

Contenido

Datos Estructurados

Una **estructura de datos** es una colección de datos, organizada de cierta manera y que permite el acceso a sus elementos siguiendo una determinada disciplina. [1]

Características de los datos estructurados

- **Dinámicas o estáticas**: pueden o no cambiar el número de sus elementos durante la ejecución, respectivamente.
- **Homogéneas o heterogéneas**: donde todos sus elementos son del mismo tipo o de varios tipos distintos, respectivamente.

Características de los datos estructurados

- **Lineales o no lineales**: donde se puede formar a los elementos en una sola fila (enumerarlos) uno después del otro o no hay manera única de hacer esto, respectivamente.
- De acceso **directo o secuencial**: si es que se puede obtener a un elemento en particular extrayéndolo directamente o se tiene que recorrer a toda la estructura desde el principio para encontrar al elemento que se desea.

Algunas Estructuras de Datos

Algunas de las estructuras de datos más conocidas son las listas, pilas, colas, árboles, tablas de dispersión, entre otras.

Listas

Una lista ℓ es:

1. La lista vacía, aquella que no tiene ningún elemento. Representada formalmente como [ ] o $\emptyset$.
2. El primer elemento de una lista, seguido de una lista. Representado formalmente como (a , ℓ), donde ℓ es a su vez una lista.

Comportamiento de las listas I

- **Agregar**: agregar a la lista el objeto que pasa como parámetro. El tamaño de la lista crece en una unidad.
- **¿Contiene?**: regresa true si el elemento está en la lista y false en otro caso. Este método no cambia el estado de la lista.
- **Eliminar**: Si el elemento existe en la lista lo elimina, en caso contrario no se realizan operaciones. Si la operación es exitosa se reduce el tamaño de la lista en una unidad.

Comportamiento de las listas II

- **¿Es vacía?**: devuelve true si la lista está vacía y false en otro caso.
- **Obtener elemento**: si la lista no es vacía, devuelve el valor almacenado en la posición especificada por el parámetro.
- **Vaciar**: elimina todos los elementos de la lista. El tamaño de la lista después de esta operación es cero.
- **Tamaño**: devuelve la cantidad de elementos que contiene la lista. [2]

Arreglos

Los arreglos son contenedores de objetos y pueden almacenar un número fijo de elementos (o valores) que pueden ser accesados mediante indexado del propio arreglo. Los corchetes se utilizan para denotar arreglos. [3]

Inicialización

- <*Tipo*> **[ ]** <*Nombre*> **;**
- <*Tipo*> **[ ]** <*Nombre*> = <*Inicialización de Arreglo*> **;**
- <*Tipo*> <*Nombre*> **[ ]** **;**
- <*Tipo*> <*Nombre*> **[ ]** = <*Inicialización de Arreglo*> **;**

Ejemplos de inicialización de Arreglos

- **int [ ]** arr1 **;**
- **int [ ]** arr2 = **new int[**10**];**
- **int** arr3 **[ ] ;**
- **int** arr4 **[ ]** = **new int[**10**];**

Inicialización directa de Arreglos

De la misma manera en que las variables de tipos primitivos pueden ser inicializadas directamente, también lo son los elementos de un arreglo.

<Tipo> *<Nombre>* **[ ]** = { *<Lista de elementos>* };

<Tipo> **[ ]** *<Nombre>* = { *<Lista de elementos>* };

Ejemplo de inicialización directa de Arreglos

int [ ] arr5 = {1, 2, 3, 4,5};

int arr6 **[ ]** = {1, 2, 3, 4, 5};

Acceso a los elementos de un Arreglo

Los Arreglos tienen acceso directo a sus elementos, es decir, no es necesario recorrer la estructura.

<Nombre>[*<Posición del elemento>*]

Por ejemplo, arr6[0] obtiene el elemento en la posición 0 del arreglo llamado *arr6*. Entonces el valor de la operación anterior es 1.

Longitud de un arreglo

Para obtener la longitud de un arreglo se hace de la siguiente manera:

<*Nombre del Arreglo*>**.length**

Por ejemplo, arr6.*length* daría como resultado 5. Que es el tamaño del arreglo.
NOTA: El tamaño del arreglo no es necesariamente la cantidad de elementos que este contiene.

Arreglos Multidimensionales

Los Arreglos multidimensionales pueden declararse mediante añadir conjuntos de corchetes. Por ejemplo, considérense las siguientes declaraciones e inicializaciones para arreglos de dos, tres y cuatro dimensiones:

int [ ][ ] dosD = **new int** [10][10];
float [ ][ ][ ] tresD = **new float** [8][10][12];
boolean [ ][ ][ ][ ] cuatroD = **new boolean** [4][8][12][16];

Otra declaración de Arreglos multidimensionales

Los arreglos multidimensionales pueden ser inicializados mediante listas anidadas de inicialización:

int [ ][ ] m = { {1, 2, 3, 4}, { 4, 5, 6, 7, 8} };

Acceso a Arreglos multidimensionales

Para poder acceder a los espacios de un arreglo multidimensional, se debe especificar la posición del elemento dentro de los n corchetes del Arreglo de dimensión n.

Por ejemplo: m[1][2] accede al espacio con valor 5.

Longitud de los arreglos multidimensionales

Un arreglo multidimensional es efectivamente una arreglo de arreglos (y de arreglos, de arreglos, etc.), ası que cada dimensión se representa por otro objeto arreglo.

Esto quiere decir que cada entrada de un arreglo es otro arreglo. Este último arreglo se comporta como uno simple.

Por ejemplo: m[0].length será igual a 4.

Bibliografía


Elisa Viso G. Canek Peláez V.
Introducción a las Ciencias de la Computación con Java.
Segunda edition, 2012.


Amparo López Gaona.
Estructuras de Datos con Java: un enfoque práctico.
2011.


Jorge L. Ortega Arjona.
Notas de Introducción al Lenguaje de Programación Java.
2004.