Universidad de San Carlos de Guatemala
Escuela de Ingeniería en Ciencias y Sistemas, Facultad de Ingeniería
Introducción a la programación y computación 2, 1er. Semestre 2021.

MANEJO DE DATOS Y SU VISUALIZACIÓN POR MEDIO DE MATRICES ORTOGONALES

201902781 – Rodrigo Antonio Porón De León

Resumen

El presente proyecto tiene como objetivo la utilización de estructuras de datos y tipos de datos abstractos (TDA) para el almacenamiento de datos en una matriz ortogonal, esto con el fin para que se puedan insertar datos en determinadas posiciones sin tener referencias o apuntadores a los nodos que derivan del nodo al que estamos apuntando. En otras palabras, en este tipo de matriz se pueden insertar datos en cualquier nodo de la matriz y no necesariamente en el primer nodo, siendo este el del índice (0,0).

Para la realización de esta matriz ortogonal como estructura de datos se realizó la creación de una lista de encabezados en la cual se guardaban tanto la coordenada y como la coordenada x del nodo. De igual forma se creó un objeto llamado Nodo Interno el cuál contaba con varios apuntadores, algunos son: arriba, abajo, izquierda, derecha. Esto como resultado daba un nodo que estaba enlazado a todos los de la matriz. Por lo que insertar datos en la matriz resulta de una manera más sencilla.

Palabras clave

Matriz ortogonal, Matriz traspuesta, Estructuras de programación, Memoria dinámica, HTML.

Abstract

The present project aims to use data structures and abstract data types (ADT) for storing data in an orthogonal matrix, this in order to insert data in certain positions without having references or pointers to the nodes that derive from the node we are targeting. In other words, in this type of matrix data can be inserted in any node of the matrix and not necessarily in the first node, this being the one with the index (0,0).

To carry out this orthogonal matrix as a data structure, a list of headers was created in which both the y coordinate and the x coordinate of the node were stored. In the same way, an object called Internal Node was created which had several pointers, some are: up, down, left, right. This resulted in a node that was linked to all of the ones in the array. So inserting data into the array is easier.

Keywords

Orthogonal matrix, Transposed matrix, Programming structures, Dynamic memory, HTML.

Introducción

Visualizar siempre es una manera óptima y adecuada de entender ciertos temas, especialmente temas como las estructuras de datos que sin una forma visual resulta extremadamente difícil entender el concepto o definición. Por lo que hacer visible lo que contiene una estructura de datos, en este caso una matriz ortogonal es muy beneficioso para el entendimiento del estudiante al momento de implementar este tipo de datos a un proyecto en desarrollo.

Crear una matriz ortogonal requiere de ciertos conocimientos específicos, sin embargo, es una forma muy práctica para guardar datos, ya que se localiza la información más fácil que con otras estructuras, por lo que su implementación en proyectos que hacen uso de información con grandes cantidades de datos.

Matriz ortogonal

Geométricamente, las matrices ortogonales representan transformaciones isométricas en espacios vectoriales reales1 (o más exactamente espacios de Hilbert reales) llamadas justamente, transformaciones ortogonales. Estas transformaciones son isomorfismos internos del espacio vectorial en cuestión. En el caso real, dichas transformaciones pueden ser rotaciones, reflexiones especulares o inversiones y son usadas extensivamente en computación gráfica. Por sus propiedades, también son usadas para el estudio de ciertos fibrados y en física se las usa en el estudio del movimiento de cuerpos rígidos y en la formulación de ciertas teorías de campos.

Se tiene una matriz ortogonal cuando dicha matriz multiplicada por su transpuesta da como resultado la matriz identidad. Si la inversa de una matriz es igual a la transpuesta entonces la matriz original es ortogonal.

Las matrices ortogonales tienen como característica que el número de filas es igual al número de columnas. Además, los vectores fila son vectores ortogonales unitarios y los vectores fila de la transpuesta también lo son.

Cuando una matriz ortogonal se multiplica por los vectores de un espacio vectorial produce una transformación isométrica, es decir, una transformación que no cambia las distancias y preserva los ángulos.

Un representante típico de las matrices ortogonales son las matrices de rotación. Las transformaciones de las matrices ortogonales sobre un espacio vectorial se denominan transformaciones ortogonales.

Matriz traspuesta

Una matriz traspuesta no es más que el resultado de intercambiar sus filas por columnas (o viceversa).

Estructura de Datos en Programación

Lo primero que debemos tener claro es la definición de “estructura de datos” y que a partir de esta definición parten otros conceptos y tipos. Que las estructuras de datos se estudian como conceptos sobre programación, y no sobre un lenguaje de programación en específico, por lo que cada lenguaje puede tener diferentes implementaciones de estructuras de datos.

Una “estructura de datos” es una colección de valores, la relación que existe entre estos valores y las operaciones que podemos hacer sobre ellos; en pocas palabras se refiere a cómo los datos están organizados y cómo se pueden administrar. Una estructura de datos describe el formato en que los valores van a ser almacenados, cómo van a ser accedidos y modificados, pudiendo así existir una gran cantidad de estructuras de datos.

Lo que hace específica a una estructura de datos es el tipo de problema que resuelve. Algunas veces necesitaremos una estructura muy simple que sólo permita almacenar 10 números enteros consecutivamente sin importar que se repitan y que podamos acceder a estos números por medio de un índice, porque nuestro problema a resolver está basado en 10 números enteros solamente. O tal vez nos interese almacenar N cantidad de números enteros y que se puedan ordenar al momento de insertar uno nuevo, por lo que necesitaremos una estructura más flexible.

¿Por qué es útil una estructura de datos?

Las estructuras de datos son útiles porque siempre manipularemos datos, y si los datos están organizados, esta tarea será mucho más fácil.

Consideremos el siguiente problema:

¿De qué forma puedo saber si una palabra tiene exactamente las mismas letras que otra palabra? O dicho de otra forma, ¿cómo saber si una palabra es una permutación de otra palabra?

Si queremos resolver este problema con papel y lápiz, basta con escribir las palabras y visualmente identificar si ambas palabras tienen las mismas letras, una forma rápida es contar mentalmente el número de cada tipo de letra en ambas palabras y si tienen el mismo número entonces si es una permutación.

Sin embargo, si el problema tuviese que resolverse en con programación se debe implementar otra forma.

Qué tal si pensamos en las palabras como una estructura de casillas de letras secuenciales (array o arreglo) donde cada letra representa una casilla en esta estructura. A cada casilla le damos un índice con el cual podemos acceder a ella para saber qué letra tiene contenida, esto le indicará a la computadora cómo reconocer una palabra.

Ahora, para resolver el problema debemos saber cómo comparar las palabras y sus letras, lo primero que podemos evaluar es el length o longitud de la palabra, o sea, cuántos caracteres tiene; si las palabras tienen diferente número de caracteres entonces no tiene sentido comparar más, ya sabemos que NO tendrán exactamente las mismas letras. Lo siguiente será contar las letras por tipo de letra. En el ejemplo anterior de las palabras “casa” y “saca” contamos las letras mentalmente, pero las escribimos en el papel con una estructura muy específica, cada letra que encontramos se asocia al número que contamos; entonces, ¿habrá alguna estructura de datos que me ayude a asociar las letras con el número que contamos?, la respuesta es sí. Podemos pensar en esta asociación como una estructura de tipo key-value (llave-valor) llamada dictionary ó diccionario, donde la letra es la llave y su valor asociado es el número contado; además, en esta estructura no debemos repetir la llave porque de lo contrario tendremos letras repetidas y no podremos contar, es por eso que la estructura dictionary es distinta al array, ya que en el array si podemos repetir letras, pero en esta otra estructura no deberíamos permitirlo ya que eso sería lo que nos permita resolver el problema de contar.

De modo que para operar la estructura array tenemos que hacerlo recorriendo una a una todas las casillas para obtener las letras mediante su índice; la casilla 0 contiene a la letra “c”, la casilla 1 contiene la letra “a” y así sucesivamente. Para el caso de la estructura dictionary podemos decir que vamos a operar en ella a través de su key (llave); la llave “c” tiene asociado el valor 1, la llave “a” tiene asociado el valor 2, y así sucesivamente. Cada vez que encontremos una letra la pondremos en la estructura dictionary, si no existe se agrega y se asocia con el valor 1, sí ya existe entonces “contamos” y a su valor le sumamos +1.

Hasta este punto hemos definido concretamente el problema por lo que resulta más fácil resolverlo de esta forma usando programación.

Tipos de estructuras de datos

Al hablar de estructuras de datos debemos pensar en

primera instancia en cómo los datos se representan en la memoria, ¿se trata de estructuras contiguas o enlazadas?, al partir de esta pregunta podemos darnos la idea correcta sobre la base de nuestra estructura y cómo es que los datos se van a almacenar.

Habiendo aclarado eso, se pueden mencionar estructuras tales como las contiguamente asignadas que están compuestas de bloques de memoria únicos, los cuales incluyen a los arrays, matrices (las cuales hemos utilizado para la solución de este proyecto), heaps, y hash tables.

Existen también las estructuras enlazadas que estás compuestas de distintos fragmentos de memoria unidos por pointers (punteros), estas estructuras incluyen a las lists, tres y graphs.

Tipo de dato abstracto

El concepto de tipo de dato abstracto (TDA, Abstract Data Type), fue propuesto por primera vez hacia 1974 por John Guttag y otros, pero no fue hasta 1975 que por primera vez Barbara Liskov lo propuso para el lenguaje CLU.

Los Lenguajes de Programación Orientados a Objetos son lenguajes formados por diferentes métodos o funciones y que son llamados en el orden en que el programa lo requiere, o el usuario lo desea. La abstracción de datos consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa. Esto es similar a una situación de la vida cotidiana. Cuando yo digo la palabra “perro”, usted no necesita que yo le diga lo que hace el perro. Usted ya sabe la forma que tiene un perro y también sabe que los perros ladran. De manera que yo abstraigo todas las características de todos los perros en un solo término, al cual llamó “perro”. A esto se le llama ‘abstracción’ y es un concepto muy útil en la programación, ya que un usuario no necesita mencionar todas las características y funciones de un objeto cada vez que este se utiliza, sino que son declaradas por separado en el programa y simplemente se utiliza el término abstracto (“perro”) para mencionarlo.

En el ejemplo anterior, “perro” es un Tipo de Dato Abstracto y todo el proceso de definirlo, implementarlo y mencionarlo es a lo que llamamos Abstracción de Datos.

Vamos a poner un ejemplo real de la programación. Supongamos que en algún Lenguaje de Programación Orientado a Objetos un pequeño programa saca el área de un rectángulo de las dimensiones que un usuario decida. Pensemos también que el usuario probablemente quiera saber el área de varios

rectángulos. Sería muy tedioso para el programador definir la multiplicación de 'base' por 'altura' varias veces en el programa, además que limitaría al usuario a sacar un número determinado de áreas. Por ello, el programador puede crear una función denominada 'Área', la cual va a ser llamada, con los parámetros correspondientes, el número de veces que sean necesitadas por el usuario y así el programador se evita mucho trabajo, el programa resulta más rápido, más eficiente y de menor longitud.

Para lograr esto, se crea el método Área de una manera separada de la interfaz gráfica presentada al usuario y se estipula ahí la operación a realizar, devolviendo el valor de la multiplicación. En el método principal solamente se llama a la función Área y el programa hace el resto.

Al hecho de guardar todas las características y habilidades de un objeto por separado se le llama Encapsulamiento y es también un concepto importante para entender la estructuración de datos. Es frecuente que el Encapsulamiento sea usado como un sinónimo del Ocultación de información.

Estructura de programación secuencial

Toda industria que requiere de tener programas más grandes se debe enfrentar a un reto mayor en donde se presentan dificultades para controlar las máquinas.

Para reducir la complejidad de que el programador realice manualmente las instrucciones con miles de líneas, se utiliza la programación secuencial.

Primero se debe entender, de forma general, que la programación es un proceso que realiza una máquina, después de recibir instrucciones, para conseguir un resultado.

Entonces, ¿qué es la programación secuencial? También conocido como estructura secuencial, es aquella en la que una instrucción o acción sigue a otra en secuencia. En este tipo de programación se presentan operaciones de inicio a fin, inicialización de variables, operaciones de asignación, cálculo, sumarización, entre otras.

La programación secuencial es más simple y fácil de usar. Como las instrucciones están relacionadas, será más sencillo entender lo que hace cada función en una instrucción. Las tareas se llevan a cabo de tal manera que la salida de una es la entrada de la siguiente y así sucesivamente hasta finalizar un proceso; por esta razón se le conoce como secuencial.

Estructura de programación cíclica

Se llaman problemas repetitivos o cíclicos a aquellos en cuya solución es necesario utilizar un mismo conjunto de acciones que se puedan ejecutar una cantidad específica de veces. Esta cantidad puede ser fija (previamente determinada por el programador) o puede ser variable (estar en función de algún dato dentro del programa).

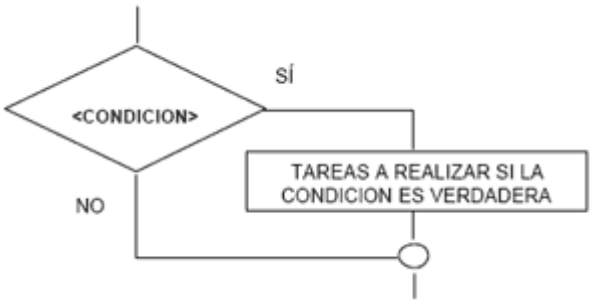
Estructura de programación condicional

Las estructuras condicionales comparan una variable contra otro(s)valor (es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen tres tipos básicos, las simples, las dobles y las múltiples.

Simples:

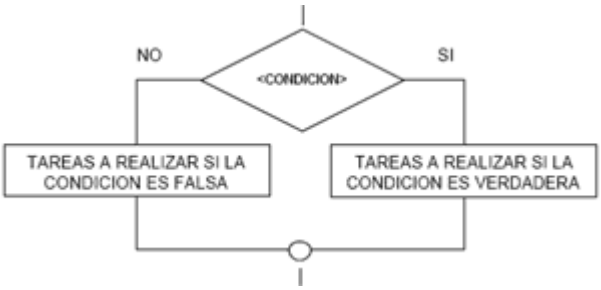
Las estructuras condicionales simples se les conoce como tomas de decisión, ya que funciona como un if

si hablamos de algún lenguaje de programación. Estas tomas de decisión tienen la siguiente forma:



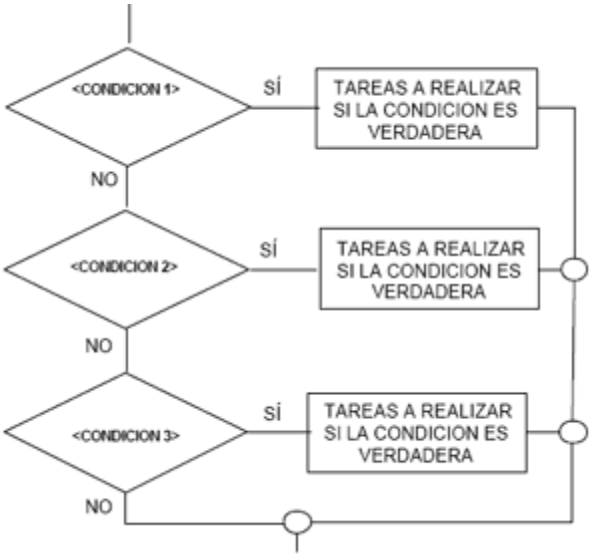
Dobles:

Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representa de la siguiente forma:



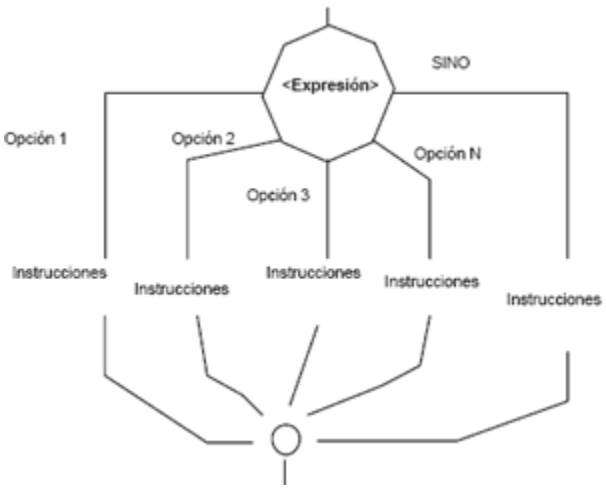
Múltiples:

Las estructuras de comparación múltiples son tomas de decisión especializadas que permiten comparar una variable contra distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma común es la siguiente:

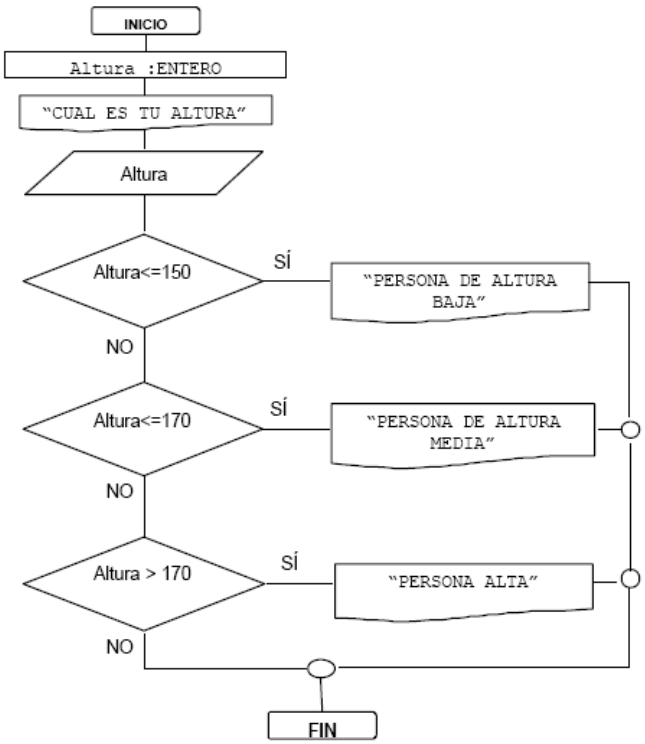


Múltiples (en caso de):

Las estructuras de comparación múltiples, es una toma de decisión especializada que permiten evaluar una variable con distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas. La forma es la siguiente:



Según lo expuesto en la estructura de programación condicional se muestra a continuación un ejemplo, utilizando los métodos expuestos anteriormente.



Conclusiones

Existen muchos tipos de estructuras de datos, las cuales tienen su razón de ser, por lo que para sacarle un mayor provecho se debe estudiar qué estructura es la conveniente a utilizar según las necesidades del proyecto.

Conocer los tipos de estructuras de programación que existen puede resultar en la simplificación de un problema grande a uno pequeño con tan solo implementar la estructura de manera correcta, por lo que su estudio es importante para el desarrollo de un programador.

Referencias bibliográficas

Autycom, (2021). Programación secuencial: características y ventajas. Programación secuencial: características y ventajas | AUTYCOM

Desarrolloweb, (2005). Estructuras condicionales. Estructuras condicionales (desarrolloweb.com)

Wikipedia, (2021). *Tipo de dato abstracto.* https://es.wikipedia.org/wiki/Tipo_de_dato_abstracto

Wikipedia, (2021). Matriz ortogonal. https://es.wikipedia.org/wiki/Matriz_ortogonal