PROYECTO 1 – ROBOT PARA COLOCAR PISOS

201807085 – Cristian Alexander Mejía Cahuec

Resumen

La empresa “Pisos Artesanales, S.A.” ha construido un azulejo especial con el que puede crear pisos con distintos patrones. Cada piso consiste en una matriz de R filas y C columnas de azulejos cuadrados. Cada azulejo es reversible, un lado es blanco y el otro es negro, para poder crear patrones diversos. Además, la empresa garantiza que, para los pisos ya instalados, podrá cambiar el patrón original por un nuevo patrón que el cliente desee sin necesidad de comprar nuevos azulejos.

Para esto se hará uso de un algoritmo que permitirá hacer ya sea intercambios entre pisos o volteos para poder llegar al patrón de destino con el menor costo posible.

Palabras clave

TDA: es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico. Un TDA es una abstracción que permite modelar las características de un elemento en particular.

XML: es un lenguaje de marcado similar a HTML.

Nodo: es un registro que contiene un dato de interés y al menos un puntero para referenciar (apuntar) a otro nodo.

POO: La Programación Orientada a Objetos es un paradigma de programación que parte del concepto de "objetos" como base.

Apuntador: es una variable que contiene la dirección de memoria de otra variable.

Abstract

The company "Pisos Artesanales, S.A." has built a special tile with which you can create floors with different patterns. Each floor consists of an array of R rows and C columns of square tiles. Each tile is reversible, one side is white and the other is black, to create different patterns. In addition, the company guarantees that, for already installed floors, it will be able to change the original pattern to a new pattern that the customer wants without the need to buy new tiles.

For this, an algorithm will be used that will allow either exchanges between floors or flips to be able to reach the destination pattern with the lowest possible cost.

Keywords

TDA: is a set of data or objects created in a personalized way by a programmer for a specific purpose. A TDA is an abstraction that allows modeling the characteristics of a particular element.

XML: It is a markup language similar to HTML.

Node: is a record that contains a data of interest and at least one pointer to reference (point) to another node.

OOP: Object Oriented Programming is a programming paradigm that starts from the concept of "objects" as a base.

Pointer: is a variable that contains the memory address of another variable.

Introducción

En este proyecto se trata de la elaboración de un algoritmo que permita cambiar de un patrón para un piso específico hacia otro que contenga las mismas dimensiones, en donde habrá 2 tipos de operaciones permitidas, intercambios entre pisos o volteos, donde cada operación tendrá un costo y esta podrá variar.

El algoritmo tomará esto en cuenta a la hora de hacer la transformación, y buscará realizar la trasformación con el menor costo posible.

TDA

Un Tipo de dato abstracto (en adelante TDA) es un conjunto de datos u objetos al cual se le asocian operaciones. El TDA provee de una interfaz con la cual es posible realizar las operaciones permitidas, abstrayéndose de la manera en como están implementadas dichas operaciones. Esto quiere decir que un mismo TDA puede ser implementado utilizando distintas estructuras de datos y proveer la misma funcionalidad.

El paradigma de orientación a objetos permite el encapsulamiento de los datos y las operaciones mediante la definición de clases e interfaces, lo cual permite ocultar la manera en cómo ha sido implementado el TDA y solo permite el acceso a los datos a través de las operaciones provistas por la interfaz.

Memoria dinámica

Se refiere a aquella memoria que no puede ser definida ya que no se conoce o no se tiene idea del número de la variable a considerarse, la solución a este problema es la memoria dinámica que permite solicitar memoria en tiempo de ejecución, por lo que cuanta más memoria se necesite, más se solicita al sistema operativo. El sistema operativo maneja la memoria gracias al uso de punteros, por la misma naturaleza del proceso nos impide conocer el tamaño de la memoria necesaria en el momento de compilar.

Un dato importante es que como tal este tipo de datos se crean y se destruyen mientras se ejecuta el programa y por lo tanto la estructura de datos se va dimensionando de forma precisa a los requerimientos del programa, evitándonos así perder datos o desperdiciar memoria si hubiéramos tratado de definir la cantidad de memoria a utilizar en el momento de compilar el programa.

XML

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

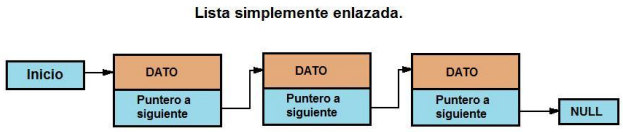
El lenguaje de marcado es un conjunto de códigos que se pueden aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. El lenguaje XML proporciona una plataforma para definir elementos para crear un formato y generar un lenguaje personalizado.

Un archivo XML se divide en dos partes: prolog y body. La parte prolog consiste en metadatos administrativos, como declaración XML, instrucción de procesamiento opcional, declaración de tipo de documento y comentarios. La parte del body se compone de dos partes: estructural y de contenido (presente en los textos simples).

El diseño XML se centra en la simplicidad, la generalidad y la facilidad de uso y, por lo tanto, se utiliza para varios servicios web. Tanto es así que hay sistemas destinados a ayudar en la definición de lenguajes basados en XML, así como APIs que ayudan en el procesamiento de datos XML - que no deben confundirse con HTML.

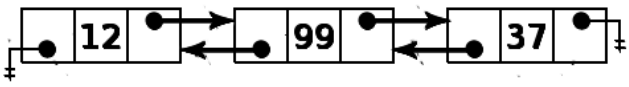
Listas Enlazadas

Una lista enlazada es una de las estructuras de datos fundamentales, y puede ser usada para implementar otras estructuras de datos. Consiste en una secuencia de nodos, en los que se guardan campos de datos arbitrarios y una o dos referencias, enlaces o punteros al nodo anterior o posterior. El principal beneficio de las listas enlazadas respecto a los vectores convencionales es que el orden de los elementos enlazados puede ser diferente al orden de almacenamiento en la memoria o el disco, permitiendo que el orden de recorrido de la lista sea diferente al de almacenamiento.



Listas Doblemente Enlazadas

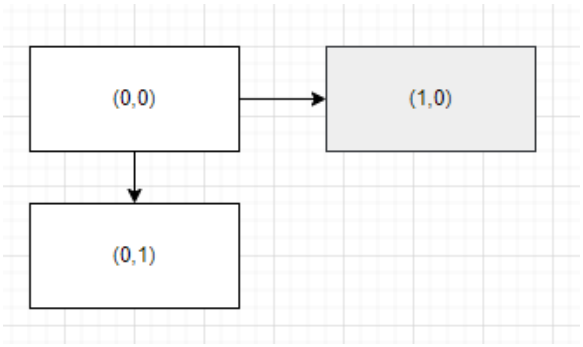
Es una estructura de datos que consiste en un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados enlaces, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información (en este caso un entero). El enlace al nodo anterior del primer nodo y el enlace al nodo siguiente del último nodo, apuntan a un tipo de nodo que marca el final de la lista, normalmente un nodo centinela o puntero null, para facilitar el recorrido de la lista. Si existe un único nodo centinela, entonces la lista es circular a través del nodo centinela.



El doble enlace de los nodos permite recorrer la lista en cualquier dirección. Mientras que agregar o eliminar un nodo en una lista doblemente enlazada requiere cambiar más enlaces que en estas mismas operaciones en una lista enlazada simple, las operaciones son más simples porque no hay necesidad de mantener guardado el nodo anterior durante el recorrido, ni necesidad de recorrer la lista para hallar el nodo anterior, la referencia al nodo que se quiere eliminar o insertar es lo único necesario.

Algoritmo A

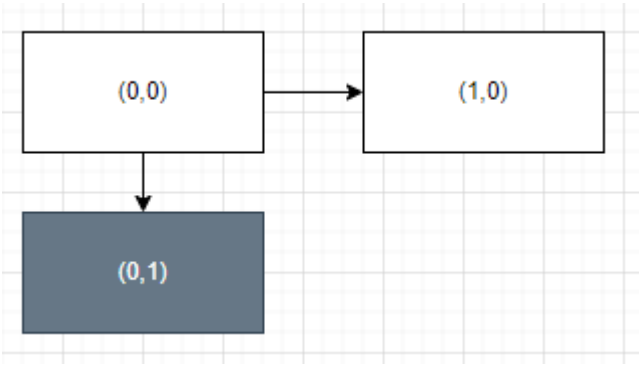
El algoritmo consiste en recorrer primero la lista enlazada que contiene el patrón de destino posición por posición y buscar el mismo nodo dentro de la lista que contiene el patrón de origen, una vez obtenido el nodo compara si es distinto o igual al nodo de la lista destino. De ser igual continuara recorriendo la lista destino, si es distinto entonces **primero revisará cuál es el nodo que está a la derecha y cuál es el nodo que está debajo para buscar si en esas posiciones está el color que nosotros estamos buscando.** De no ser así si no se encuentra en esas posiciones entonces hará un volteo, y continuará así hasta llegar al final de la lista habiendo ya hecho todos los cambios necesarios.



Algoritmo B

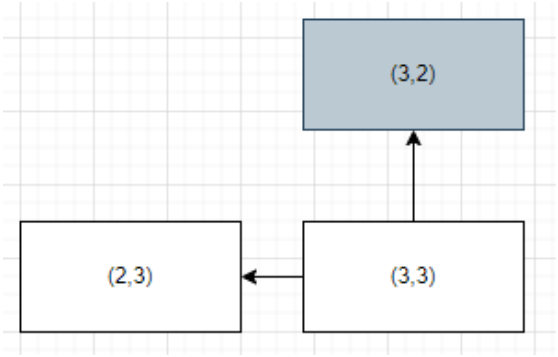
El algoritmo consiste en recorrer primero la lista enlazada que contiene el patrón de destino posición por posición y buscar el mismo nodo dentro de la lista que contiene el patrón de origen, una vez obtenido el nodo compara si es distinto o igual al nodo de la lista destino. De ser igual continuara recorriendo la lista destino, si es distinto entonces **primero revisará cuál es el nodo que está**

abajo y luego revisará el nodo de la derecha para buscar si en esas posiciones está el color que nosotros estamos buscando. De no ser así si no se encuentra en esas posiciones entonces hará un volteo, y continuará así hasta llegar al final de la lista habiendo ya hecho todos los cambios necesarios.



Algoritmo D

El algoritmo consiste en recorrer primero la lista enlazada que contiene el patrón de destino posición por posición y buscar el mismo nodo dentro de la lista que contiene el patrón de origen, una vez obtenido el nodo compara si es distinto o igual al nodo de la lista destino. De ser igual continuara recorriendo la lista destino, si es distinto entonces **primero revisará cuál es el nodo que está a la derecha y cuál es el nodo que está debajo para buscar si en esas posiciones está el color que nosotros estamos buscando.** De no ser así si no se encuentra en esas posiciones entonces hará un volteo, y luego recorrerá la lista de forma inversa en donde ahora **buscará cuál es el nodo que está a la izquierda y luego el nodo que está arriba, se agrega una condición en la que los nodos por los que ya pasó y verificó que son iguales entonces estos no se van a tocar,** por lo que si no encuentra el color buscado en la posición de la izquierda ni la derecha, hará un volteo.



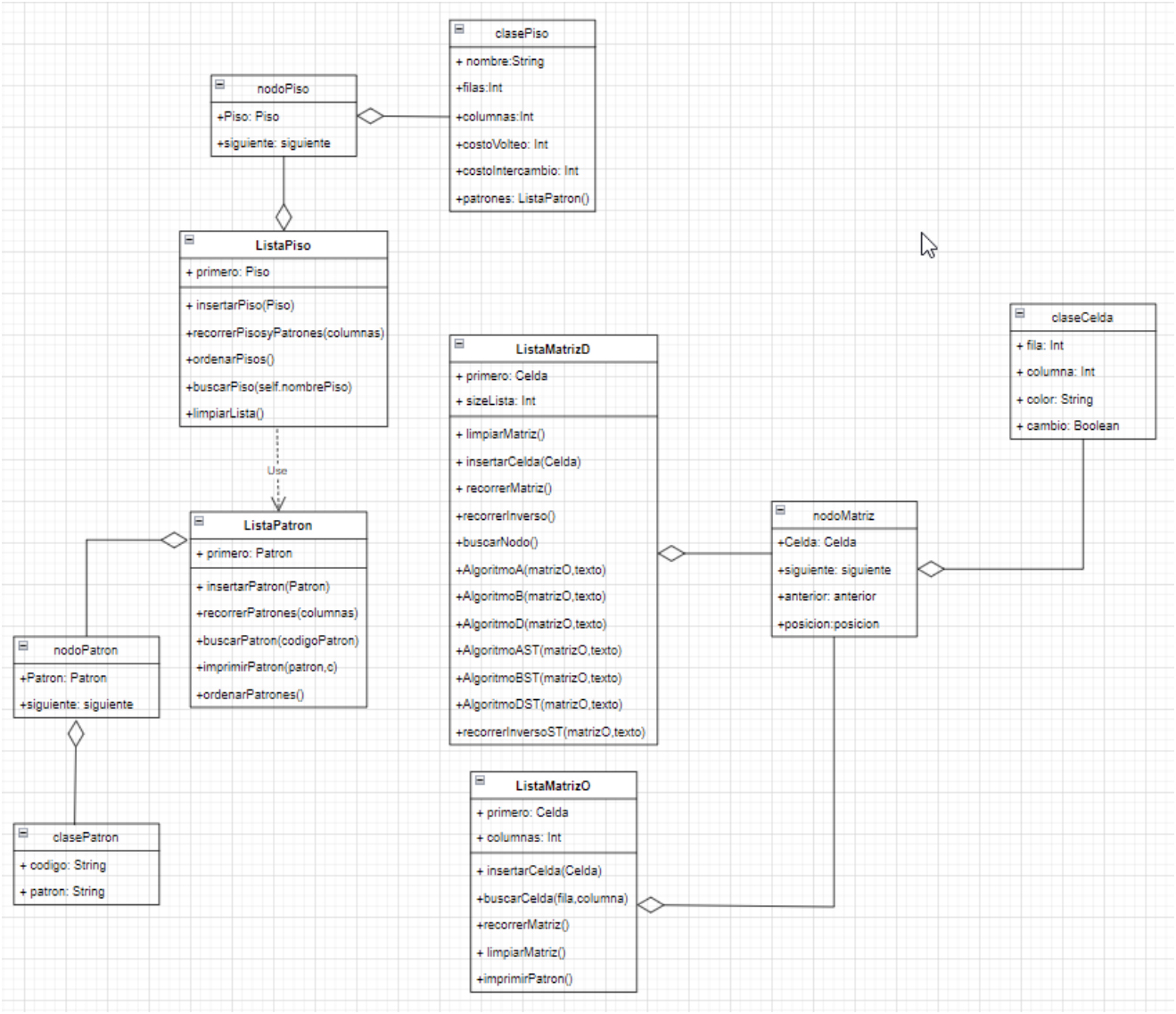
Conclusiones

- Para este proyecto se hicieron uso de estructuras de datos abstractas (TDA) utilizando el paradigma de la programación orientada a objetos, que permite hacer estructuras personalizadas que facilitan la resolución de este problema, estructurando las distintas partes del programa como módulos que se unen entre sí y son fáciles de modificar.
- Con el uso de los distintos algoritmos elaborados se puede concluir que son óptimos para la resolución de este problema y se obtendrá el menor costo posible para realizar el cambio entre patrones.
- La utilización de TDA's permite hacer un uso óptimo de la memoria ya que son estructuras personalizadas.

Referencias bibliográficas

- Problemas resueltos de listas / Mónica Adriana Carreño León 2010.
- CAIRÓ O., GUARDATI S., Estructura de Datos, México, Mc Graw-Hill, 2001.
- JOYANES A. LUIS, Algoritmos y Estructuras de Datos: Una Perspectiva en C., España, Mc Graw-Hill, 2005.

Anexos


clasePiso
+ nombre:String
+filas:Int
+columnas:Int
+costoVolteo: Int
+costoIntercambio: Int
+patrones: ListaPatron()
nodoPiso
+Piso: Piso
+siguiente: siguiente
ListaPiso
+ primero: Piso
+ insertarPiso(Piso)
+recorrerPisosyPatrones(columnas)
+ordenarPisos()
+buscarPiso(self.nombrePiso)
+limpiarLista()
Use
ListaPatron
+ primero: Patron
+ insertarPatron(Patron)
+recorrerPatrones(columnas)
+buscarPatron(codigoPatron)
+imprimirPatron(patron,c)
+ordenarPatrones()
nodoPatron
+Patron: Patron
+siguiente: siguiente
clasePatron
+ codigo: String
+ patron: String
ListaMatrizD
+ primero: Celda
+ sizeLista: Int
+ limpiarMatriz()
+ insertarCelda(Celda)
+ recorrerMatriz()
+recorrerInverso()
+buscarNodo()
+AlgoritmoA(matrizO,texto)
+AlgoritmoB(matrizO,texto)
+AlgoritmoD(matrizO,texto)
+AlgoritmoAST(matrizO,texto)
+AlgoritmoBST(matrizO,texto)
+AlgoritmoDST(matrizO,texto)
+recorrerInversoST(matrizO,texto)
claseCelda
+ fila: Int
+ columna: Int
+ color: String
+ cambio: Boolean
nodoMatriz
+Celda: Celda
+siguiente: siguiente
+anterior: anterior
+posicion:posicion
ListaMatrizO
+ primero: Celda
+ columnas: Int
+ insertarCelda(Celda)
+buscarCelda(fila,columna)
+recorrerMatriz()
+ limpiarMatriz()
+imprimirPatron()