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

JUEGO BLOCKS – PROYECTO 1 -

201313861 – Walther Andree Corado Paiz

Resumen

En el proyecto se desarrolló la implementación y utilización de de las Estructuras de Datos Abstracto conocidas por sus siglas TDA, específicamente el TDA Matriz Dispersa, desarrollada e implementada mediante la utilización de nodos y punteros.

La razón principal de la utilización de un TDA como solución de software, fue el aprendizaje, la utilización y el manejo de la memoria mediante los punteros y sus respectivas direcciones de memoria.

Para el entendimiento práctico, se implemento la utilización de graphiz, con el fin de generar de manera gráfica la implementación del TDA utilizado.

Se desarrollo una interfaz gráfica, sencilla y amigable con el usuario, con el fin de facilitar la dinámica de juego, e intuitiva para él mismo.

Las partidas se podrán guardar mediante un archivo xml y cargar nuevamente mediante la lectura del mismo.

La solución de software, cuenta con un reporte web en el que se puede aprecia la gráfica de nuestra matriz dispersa.

Palabras clave

Nodo

Matriz Dispersa

Tipo de Dato Abstracto (TDA)

Memoria

Puntero

Abstract

The project developed the implementation and use of Abstract Data Structures known by its acronym TDA, specifically the TDA Dispersed Matrix, developed and implemented through the use of nodes and pointers.

The main reason for using a TDA as a software solution was to learn, use and manage memory by means of pointers and their respective memory addresses.

For the practical understanding, the use of graphiz was implemented, in order to generate graphically the implementation of the TDA used.

A graphical interface was developed, simple and user-friendly, to facilitate the dynamics of the game, and intuitive for the user.

The games can be saved through an xml file and loaded again by reading it.

The software solution has a web report in which you can see the graph of our dispersed matrix.

Keywords

Node.

Dispersed Matrix

Data Type Abstract

Memory

Pointer

Introducción

En el proyecto se desarrollo la comprensión y ejecución el tipo de dato abstraco (TDA), especificanemente la Matriz Dispersa, mediante el manejo de memoria y la utilización de punteros.

Como estructura fundamental un nodo Raíz y listas doblemente enlazadas como cabeceras para el eje de las “X” y el eje de las “Y”, y los nodos interiores cuenta con 4 punteros que se conectan hacia arriba, derecha, abajo e izquierda.

Esta estructura de datos compleja simula las matrices estáticas con las que se cuentan en el ámbito de la programación, solucionando el problema de el tamaño estático y sin la posibilidad de contar con crecimiento, ya que cuenta con la ventaja de poder crecer tanto en “X” como en “Y” y optimizar el uso de memoria no utilizando los espacios internos no requeridos.

Desarrollo del tema

Iniciaremos definiendo la teoría necesaria para comprender los términos de la matriz dispersa y la implementación de esta, como solución de almacenamiento de memoria en nuestro software desarrollado.

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

Puntero: un puntero es un objeto del lenguaje de programación, cuyo valor se refiere a (o "apunta a") otro valor almacenado en otra parte de la memoria del ordenador utilizando su dirección. Un puntero referencia a una ubicación en memoria, y a la obtención del valor almacenado en esa ubicación se la conoce como desreferenciación del puntero.

Lista Doblemente enlazada: es una estructura de datos que consiste en un conjunto de nodos enlazados secuencialmente. Cada nodo contiene tres campos, dos para los llamados punteros, que son referencias al nodo siguiente y al anterior en la secuencia de nodos, y otro más para el almacenamiento de la información (la cual puede ser de cualquier tipo o contener varios tipos de informacion o variables)

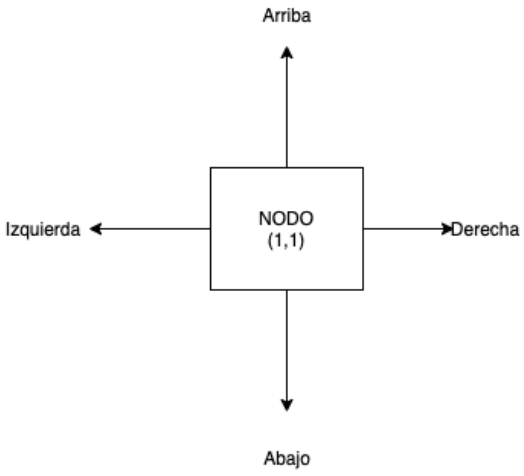
Matriz: una matriz se considera un conjunto de elementos ordenados en filas y columnas los cuales poseen un sistema coordenado de posiciones.

Habiendo definido esos términos procedermos a informar la lógica y mecánica detrás de nuestra matriz dispersa apoyandonos en algunos diagramas que nos ayuden a simplificar el entendimiento de este.

En primer lugar se procede a la creación de un Nodo raíz, este se crea al momento de instanciar nuestra matriz, ese nodo sirve de cabecera para la lista horizontal “X” y la lista vertical “Y”, a partir de este concepto procemos a explicar el mecanismo de la matriz.

Mecanismo de matriz:

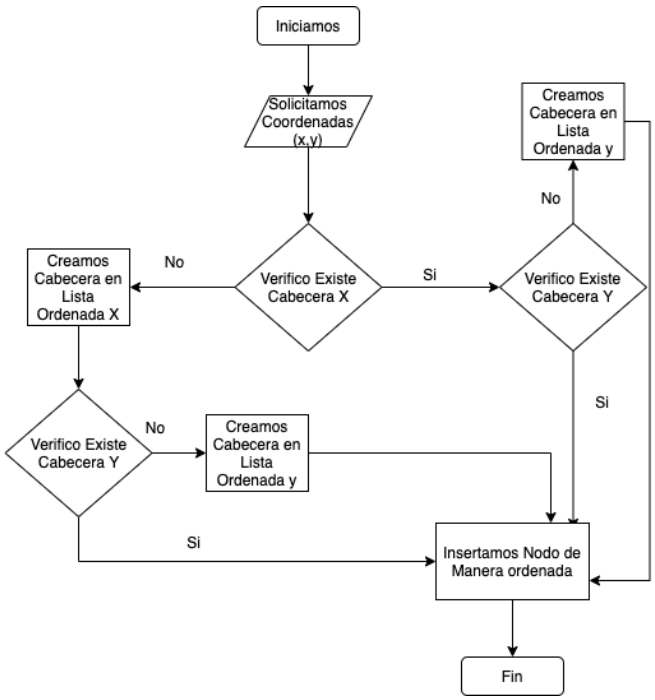
Se procede a crear un nodo tomaremos como ejemplo el nodo coordenadas (x,y) , como el nodo (1,1) este nodo contiene 4 apuntadores.



Al crear el nodo tenemos algunos casos que validar los cuales son los siguientes:

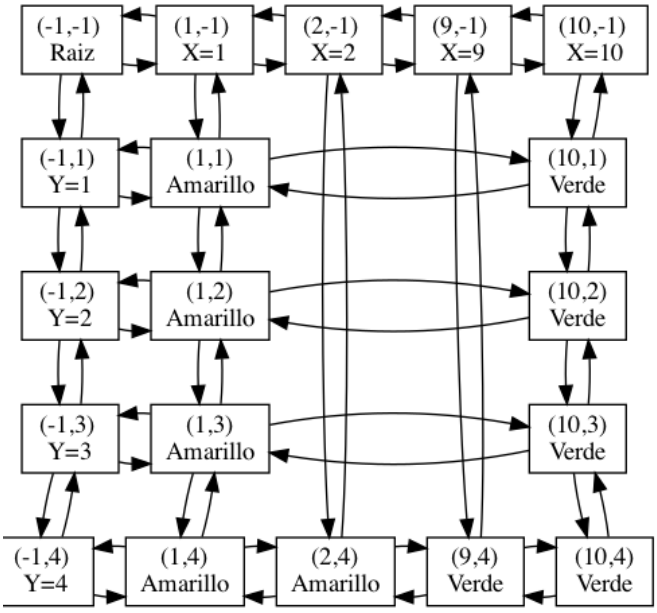
1. Que los nodo cabecera en (X,Y) no existan.
2. Que exista el nodo cabecera X y no Y.
3. Que exista el nodo cabecera Y y no X.
4. Que ambos nodos cabecera existan y solo se deba agregar el nodo.

Por lo cual definiremos el diagrama de flujo para cada caso anteriormente expuesto, tomando en cuenta que para todos los casos es necesario hacer las respectivas incersiones de los nodos de manera ordenada para poder llevar asi el control de la matriz dispersa.



De esta manera se encuentra representadas los 4 posibles casos que se encuentran descritos con anteriormente, para poder llevar a cabo se realizó un método de búsqueda de nodos los cuales nos devuelven un nodo y de no existir nos devuelve un valor false para poder crear las cabeceras según corresponde.

Con respecto a la optimización de la memoria en la utilización de este TDA, se encuentra en que si no es necesario crear todo el tamaño de la matriz no la crea sino es creado tanto cabezeras como nodos según sea necesario esto con el fin de no reservar espacios de memoria sin utilizar.



Justamente como se ve en la imagen anterior por ejemplo los nodos (2,1), (9,1) no han sido creados y tampoco las cabeceras entre 2 y 9 ejemplificando asi que solamente se utiliza la memoria referenciada, evitando el uso execivo del recuros.

Los nodos son importante tomar en cuenta que fuera de tener los 4 apuntadores definidos y explicados al inicio del tema, es importante indicar que cuentan con información / data en su interior, es decir al crear un nodo el espacio de memoria se crea de un tamaño específico que cumpla con lo necesario para resguardar valores de tipo int y string, para la solución de software propuesta utilizamos la siguiente definición para nuestro nodo.

Int x

Int y

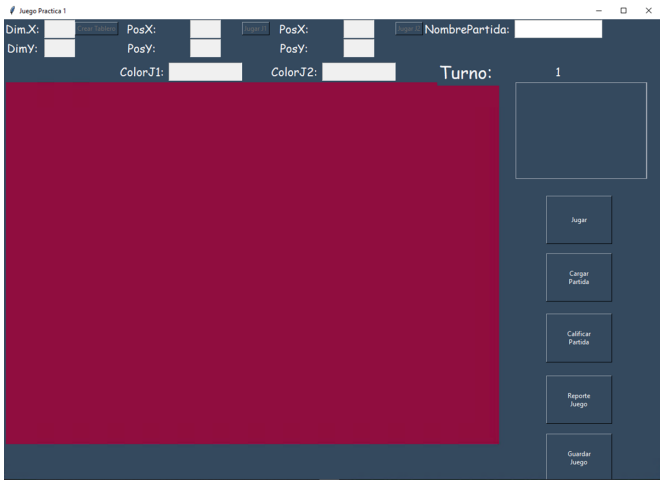
String Color

Esas tres variables son de suma importancia para poder llevar a cabo la lógica del juego.

Lógica de Juego:

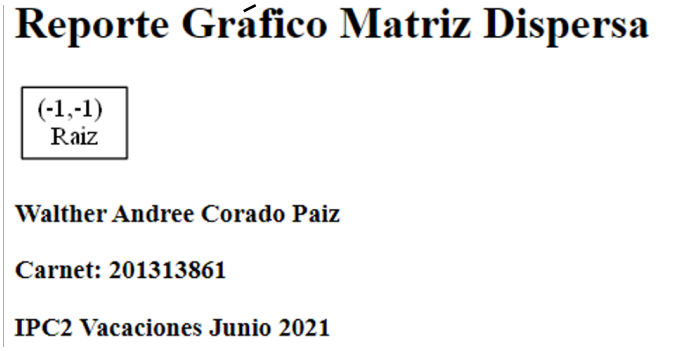
Se procedio a utilizar un tablero dinámico de botones el cual representa nuestro tablero de juego, es importante resaltar que el mismo no genera ninguna inserción dentro de nuestra matriz dispersa.

Mediante una interfaz sencilla e intituiva se solicita al usuario que nombre una partida, el tamaño del tablero y las coordenadas iniciales de las diferentes piezas con las que cuenta el juego.



Cuando alguno de los 2 jugadores procede a ingresar la coordenada de inicio y presionar Jugar, se realizan las validaciones correspondientes y si el movimiento es válido se permite el ingreso de dicha información a la matriz, tomando en cuenta el flujo y los casos anteriormente descritos en este documento.

Con esto se confirma que el generar un tablero de mxn dimenciones no altera nuestra matriz que al ser instanciada solamente crea el nodo raíz.



Nuestra propuesta de software utiliza de manera responsable y correcta el recurso de memoria de los equipos, dejando en el módulo de interfaz gráfica la validación de movimientos, para no acceder de manera recurrente nuestra matriz mas que para el proceso de inserción o graficación de la misma.

Conclusiones

La matriz dispersa optimiza el uso de recurso en memoria y tiene la ventaja de poder crecer indefinidamente o bien hasta que el recurso de memoria sea utilizado por completo.

El uso y manejo de punteros en un nodo es de total cuidado, ya que se puede perder la referencia muy fácilmente y así perder toda la secuencia de un TDA.

Los TDA tiene una gran ventaja sobre las estructuras estáticas que ya cuenta el ámbito de la programación, ya que nos permiten almacenar gran cantidad de datos.

Referencias bibliográficas

ANON., 2021. XML Tree. *W3schools.com* [online] [accessed. 2021-06-22]. Available at: https://www.w3schools.com/xml/xml_tree.asp

ADITYA SHARMA, 2019. Tkinter Tutorial: Python GUI Programming & How Use Tkinter to Make GUI Windows. DataCamp Community [online] [accessed. 2021-06-22]. Available at: https://www.datacamp.com/community/tutorials/gui-tkinter-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=278443377092&utm_targetid=aud-299261629574:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=9077356&gclid=CjwKCAjw8cCGBhB6EiwAgORey1MYorNb2LJoRf8QbOS4Gy4UE8ROpOEsatnFkSh-K4MW0QUhyCuILxoCJp8QAvD_BwE

GRAPHVIZ, 2021. Graphviz. Graphviz [online] [accessed. 2021-06-22]. Available at: https://graphviz.org/

Problemas resueltos de listas / Mónica Adriana Carreño León…[et al.] – México : UABCS, 2010. 322 P. : il. ; 28 cm. ISBN: 978-607-7777-15-1 Aispuro.

RAMIRO, 2020. ▷ Lista doblemente enlazada con ejemplos de Python | Pharos. Pharos [online] [accessed. 2021-06-22]. Available at: https://pharos.sh/lista-doblemente-enlazada-con-ejemplos-de-python/