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 2022.

CHAPÍN WARRIORS, S.A

201504459 – SANDRA DAYANIRA LOPEZ GOMEZ

Resumen

Se desarrollo una solución para el proyecto 2 de instrucción a la programación 2, se describe el proceso de solución y conceptos clave, que permiten una mejor compresión del proceso de solución, utilizando programación orientada a objetos y estructura de datos, se desarrolla un programa que cumple con los requerimientos solicitados. Utilizando las herramientas requeridas para la graficar la matriz dispersa, siendo capaz de manipular XML, utilizando los conceptos de TDA y aplicarlos a memoria dinámica.
Presentamos los diagramas de clases, y funciones, el manejo correcto de nodos, utilizando listas enlazadas para para almacenar la información del archivo XML, para la graficar los azulejos.

Palabras clave

TDA: En modelo matemático compuesto por una colección de operaciones definidas.
Nodo: Cada uno de los elementos de una lista enlazada, un árbol o grafo.
XML: Lenguaje Marcado Extendible, metalenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web.
Matriz Dispersa: Son matrices en las cuales existen gran cantidad de valores nulos o cero.

Abstract

A solution was developed for project 2 of instruction to programming 2, the solution process and key concepts are described, which allow a better understanding of the solution process, using object-oriented programming and data structure, a program is developed that meets the requested requirements. Using the tools required to graph the octagonal matrix, being able to manipulate XML, using the concepts of TDA and apply them to dynamic memory. We present the diagrams of classes, and functions, the correct handling of nodes, using linked lists to store the information of the XML file, to graph the tiles.

Keywords

TDA: In a mathematical model composed of a collection of defined operations.
Node: Each of the items in a linked list, a tree, or graph.
XML: Extendable Markup Language, metalanguage that allows you to define markup languages developed by the World Wide Web.
Sparse Matrix: These are matrices in which there are a large number of null or zero values

Introducción

El proceso de el desarrollo de un software requiere la aplicación de conceptos y tener una metodología ordenada para la resolución del problema. Desde el análisis del requerimiento, definir las funciones principales y los requisitos que el cliente o en este caso el catedrático solito para la realización de este proyecto. Se tiene en cuenta las restricciones y la solicitud de aplicar los conceptos de programación, como memoria dinámica, estructura de datos, manejo de datos abstractos, aplicados en el paradigma orientado a objetos.

Definimos las variables a utilizar en este programa y aplicamos las herramientas necesarias para que el usuario realice todas las acciones solicitadas, representando de forma gráfica como el diseño de nuestras clases, funciones y objetos principales.

Desarrollo del tema

Aplicando los conceptos adquiridos en la clase teórica de introducción a la programación II y en el laboratorio de dicha clase se desarrolló una solución óptima para el sistema solicitado en la práctica. Iniciando con un análisis preliminar del problema para llegar a una solución.

Siguiendo las fases de desarrollo de software se describe el proceso para desarrollar el software.

1. **Análisis de Sistemas y requisitos:**

   Damos lectura al documento, para analizar los requerimientos del sistema y las especificaciones del cliente para el desarrollo del software en este caso del catedrático. Las especificaciones principales son:

   - Utilizar el paradigma de programación orientado a objetos (POO).
   - Utilizar lectura de archivos XML para el ingreso de los datos al sistema.
   - Almacenar los datos en estructuras de memoria dinámica (Listas Enlazadas).
   - Graficar la estructura de datos utilizada por medio de la extensión de Graphviz, creando un archivo .dot.
   - La interfaz debe ser visualizada en cmd o la terminal del sistema y debe ser intuitivo para el usuario.

Definimos los requerimientos mínimos del sistema para la ejecución del programa.

- MS Windows XP o superior.
- Python 2.6 (opcionalmente Python 2.7, para Plone 4.2 y superior).
- Descargar los códigos fuentes de los paquetes de Graphviz, para visualizar la grafica de los datos almacenados.

2. **Diseño y Arquitectura del Software:** Teniendo definidas las funciones se diseña el diagrama de clases para representar como se trabajará el software, con los objetos, métodos y funciones.
   El diseño de diagramas de clases que represente el manejo de clases, objetos y funciones, en este caso se crearon 2 diagramas:
   - *"Diagrama 1*" para el manejo de datos del archivo de texto, almacenamiento y consulta.
   - *"Diagrama 2"* representa el manejo de los datos necesarios para la representación gráfica de los Mapas

Definición de las variables del Diagrama I

Definimos las variables principales, que serán dadas por el archivo de texto y la función que tiene cada una, las cuales crean un objeto llamado NodoAzulejo.

Se utilizan dos listas enlazadas para el almacenamiento de los datos:

- *Lista Ciudad:* Almacena el objeto NodoCiudad.
- *Lista Robot:* Almacenara los Robots que pueden utilizarse en la misión.
- *Lista Celda:* Almacenara las diferentes celdas que se visualizan en el mapa.
- *Nodo Celda:* Almacena los datos proporcionados por el archivo XML.
- *Nodo Robot:* Almacena los datos de los patrones proporcionados por el archivo XML.
- *Nodo Celda:* Almacena los datos de los patrones proporcionados por el archivo XML.

Tabla I:

Variables del Objeto "NodoCiudad".

Nombre	Representación en el Documento XML	Función
Nombre	<nombre= " ">	Nombre del mapa
Filas	<fila="">	Cantidad de Filas
Columna	<columna="">	Cantidad de Columnas
fila	<fila numero="">	Numero de fila al guardar.
Unidad Militar	<Unidad Militar>	Unidades militares dentro del mapa

Fuente: elaboración propia.

Se define otro objeto llamado Nodo Código, en cual se almacena el patrón y el nombre del código.

Tabla II:

Variables del Objeto "NodoPatron".

Nombre	Representación	Función
Codigo	<patron codigo = "">	Nombre de Piso
Robot	<robot>	Nombre del robot
tipo	<tipo="">	Tipo de robot
Siguiente	Nulo	Apuntador para enlazar el nodo que continua

Fuente: elaboración propia.

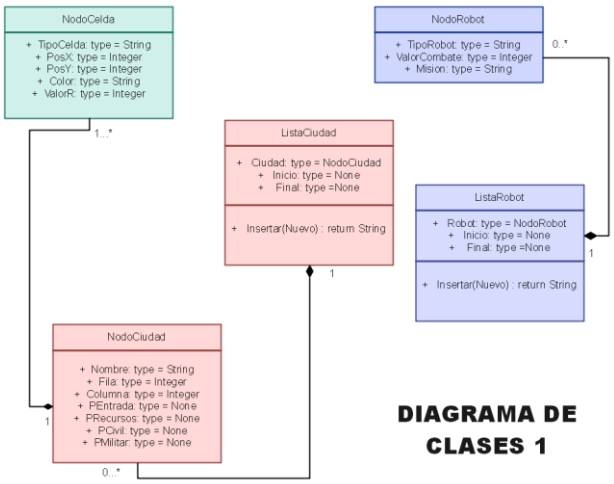

Diagrama 1. Diagrama de Clases, para el almacenamiento de datos del archivo XML.

Fuente: Elaboración propia.

Definición de las variables del Diagrama II

Definimos las variables principales, para el almacenar el patrón del azulejo y graficarlo, se

declaran 3 objetos y 5 listas para el manejo de estos datos, para poder formar la matriz octogonal.

- *Nodo Dato:* Almacena las coordenadas del nuevo dato en la matriz octogonal, junto con la letra correspondiente al patrón, del azulejo.

- *Nodo CabeceraX :* Alacena la posición en la que se insertara la nueva lista, este nodo será la cabecera de las listas horizontales.

- *Nodo CabeceraY :* Alacena la posición en la que se insertara la nueva lista, este nodo será la cabecera de las listas Verticales.

- *Lista Fila:* Almacena el objeto Dato, esta lista se almacenará en la CabeceraY para hacer la referencia de memoria de su cabecera.

- *Lista Columna:* Almacena el objeto Dato, esta lista se almacenará en la CabeceraX para hacer la referencia de memoria de su cabecera.

- *Lista X:* Almacena el objeto Dato, esta lista se tendrá como dato la cabecera la CabeceraX para hacer la referencia de memoria de su cabecera.

- *Lista Y:* Almacena el objeto Dato, esta lista se tendrá como dato la cabecera la CabeceraY para hacer la referencia de memoria de su cabecera.

Tabla III.

Variables del Objeto Nodo Dato

Nombre	Función
X	Posición en el Eje X
Y	Posición en el Eje Y
Letra	Carácter W o B
Siguiente	Apuntador para enlazar el nodo que continua
Anterior	Apuntador para enlazar el nodo que antecede
Arriba	Apuntador para enlazar el nodo que esta arriba
Abajo	Apuntador para enlazar el nodo que esta abajo

Fuente: elaboración propia.

Tabla IV.

Variables del Objeto Nodo CabeceraX

Nombre	Función
PosX	Posición en el Eje X de la cabecera
ListaH	Lista Columna, donde se almacena el Objeto Nodo Dato
Letra	Carácter W o B
Siguiente	Apuntador para enlazar el nodo que continua
Anterior	Apuntador para enlazar el nodo que antecede

Fuente: elaboración propia.

Tabla IV.

Variables del Objeto Nodo CabeceraY

Nombre	Función
PosY	Posición en el Eje Y de la cabecera
ListaV	Lista Columna, donde se almacena el Objeto Nodo Dato
Letra	Carácter W o B
Arriba	Apuntador para enlazar el nodo anterior
Abajo	Apuntador para enlazar el nodo Siguiente

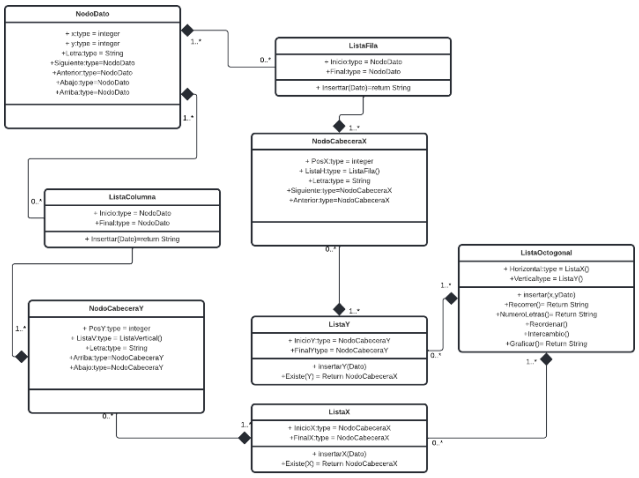

Diagrama 2. Diagrama de Clases, de la Matriz Octogonal.

Fuente: Elaboración propia

1. **Ordenamiento de Matriz Octagonal**

Para el ordenamiento de la matriz octogonal y graficar el nuevo patrón , se procede a almacenar el nuevo patrón en una Matriz de memoria estática, obteniendo la las posiciones exactas de las letras las cuales deben cambiarse o voltear. Posteriormente se mandan esos datos a la matriz Octogonal donde se realiza la función de ordenar o Voltear. Se muestra la bitácora de los cambios y se procede a calcular el precio final, de los cambios realizados.

3. Interfaz Grafica

Se le presenta al usuario una interfaz entendible y fácil de manejar, donde podrá iniciar el programa y directamente cargar el archivo y guiarse por las diferentes opciones.

1. Cargar Archivo: Se carga el archivo XML para obtener la información
2. Graficar Mapa: Graficara por medio de Graphviz, generando un archivo .dot, debe elegir el azulejo a graficar.
3. Iniciar Misión: Se abre una ventana donde se le pide al usuario describir la misión a realizar
4. Seleccionar Robot: Se selecciona robot para poder realizar la misión
5. Realizar Misión: Se realiza la misión

6. Mostrar Códigos Cargados: Mostrara una lista de los códigos cargados en consola.
7. Salir: Saldrá del programa.

Conclusiones

Al desarrollar un software, se debe tomar en cuenta el proceso o metodología correcta, ya que nos da una idea clara del camino que llevaremos a lo largo del proyecto solicitado.

Tener claro los conceptos solicitados para el desarrollo del mismo, en este caso manejar bien los pilares del paradigma de programación orientado a objetos, de igual manera el manejo de memoria dinámica y como funciona, los nodos en la estructura de datos, si en caso no se hace bien las referencias de memoria no se perderá la dirección del dato almacenado y por lo tanto se pierde el dato. Saber manejar estos conceptos garantizan una representación adecuada de los datos requeridos en Graphviz.

Independientemente que lenguaje se maneje realizar y entender los algoritmos para la resolución de los problemas expuestos en el proyecto hacen un desarrollo optimo del mismo.

Referencias bibliográficas

Máximo 5 referencias en orden alfabético.

Diagrama de clases. Teoria y ejemplos. (2020, November 22). Retrieved March 8, 2022, from DiagramasUML.com website: https://diagramasuml.com/diagrama-de-clases/

Diagrama de clases. Teoria y ejemplos. (2020, November 22). Retrieved March 8, 2022, from DiagramasUML.com website: https://diagramasuml.com/diagrama-de-clases/

Paradigma de la programación orientada a objetos. (2017). Retrieved March 8, 2022, from Github.io website: https://ferestrepoca.github.io/paradigmas-de-programacion/poo/poo_teoria/index.html

Anexos 1: Interfaz Gráfica de Usuario

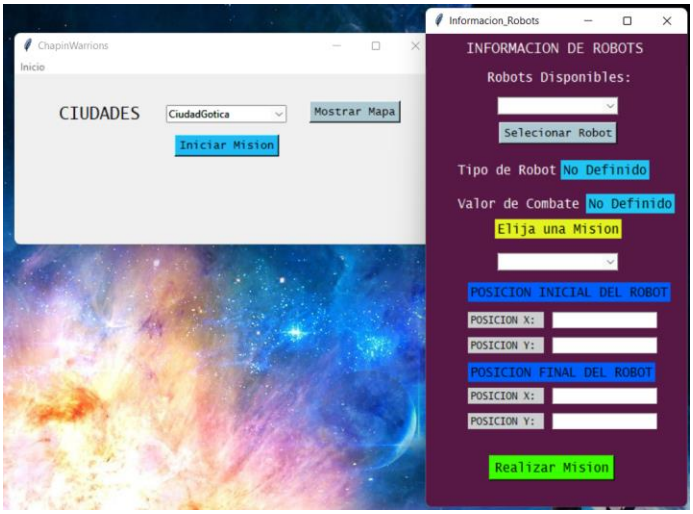

Figura 1: Menú de Usuario

Anexos 2: Grafica de NaPAS

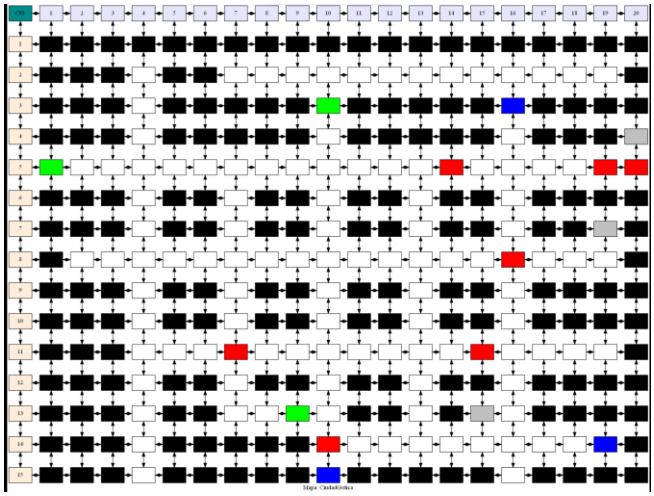

Figura 2: Matriz Octogonal