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.

PROYECTO 1: ROBOT PISOS ARTESANALES

202001570– Nataly Saraí Guzmán Duarte

Resumen

En este ensayo se presenta la creación y desarrollo técnico del proyecto No. 1 del curso de Introducción a la Computación 2, del primer semestre del año 2022. Este proyecto se realizó con una interfaz en modo consola, haciendo uso de tipos de datos abstractos, memoria dinámica, objetos y módulos para leer archivos XML. El proyecto se llevó a cabo en el lenguaje de programación Python.

El programa desarrollado proporciona al usuario una gestión de datos de forma dinámica, para lo cual los datos se almacenan en tres tipos de listas, dos listas simples y una lista doblemente enlazada, lo cual hacen un uso adecuado de TDA, permitiendo así la posibilidad de realizar un conjunto de funciones indicadas en el documento de requerimiento.

El conjunto de datos obtenido en las listas se puede visualizar a través de los reportes que se crean en el momento de ejecución como son los PDF de las graficas de los pisos según el patrón que se requiere, y el archivo txt con las instrucciones para el movimiento adecuado para cambiar el patrón.

Palabras clave

Programación orientada a objetos, listas, TDA, memoria dinámica, clase.

Abstract

This essay presents the creation and technical development of project No. 1 of the Introduction to Computing 2 course, from the first semester of the year 20222. This project was carried out with an interface in console mode, using abstract data types, dynamic memory, objects and modules to read XML files. The project was carried out in the Python programming language.

The developed program provides the user with dynamic data management, for which the data is stored in three types of lists, two simple lists and a doubly linked list, which make adequate use of TDA, thus allowing the possibility of perform a set of functions indicated in the requirement document.

The set of data obtained in the lists can be viewed through the reports that are created at the time of execution, such as the PDF of the graphs of the floors according to the pattern that is required, and the txt file with the instructions for the proper movement to change the pattern.

Keywords

Object Oriented programming, lists, TDA, dynamic memory, class.

Introducción

La construcción del proyecto se fundamenta en el uso de la programación orientado a objetos, este paradigma nos permitió generar tipos de datos abstractos fundamentados en la información que el documento de requerimiento nos mostraba, así como todos los atributos que correspondían en los archivos a procesar. Para el procesamiento de los archivos se utilizó una librería cuyo propósito se basa en el procesamiento de archivo con extensión XML, esta librería permitió la lectura de datos así también como la construcción de archivos de salida con la extensión anteriormente mencionada.

El desarrollo del proyecto consiste en aplicar funcionalidades en los datos proporcionados en los archivos, la función principal es encontrar la forma más económica en la cual se pueda cambiar los pisos de color por medio de dos patrones. El proyecto proporciona al estudiante poner en práctica sus conocimientos y su lógica para encontrar la solución del problema propuesto en el documento de requerimiento.

Desarrollo del tema

El problema propuesto en el documento de especificaciones y requerimientos, especifica que se diseñó un nuevo robot, que tiene como habilidad principal buscar la manera de cambiar el color de los pisos de la manera más económica, este robot tiene un registro de los pisos, los patrones que estos pisos contienen, y las celdas de los patrones que los pisos contienen. De tal manera que puede buscar si puede mover un piso intercambiándolo con otro de forma horizontal y vertical, mas no se puede mover en forma inclinada, o realizar volteos para cambiar el color de dicho pis, el tomara la decisión de la forma en que le cueste menos cantidad de dinero realizar dicho cambio.

Para encontrar la solución al problema anteriormente mencionado, debemos saber que el programa a desarrollar debía contar con las siguientes especificaciones: los datos necesarios para los pisos seria proporcionados en archivos XML, para procesar la información de los pisos se tenía que poner en práctica la memoria dinámica haciendo uso de listas construidas por medio de tipos abstractos, y los reportes generados por el robot seria por medio de grafos, haciendo uso de Graphviz. A continuación, explicaremos a detalle el desarrollo del proyecto. Para proponer la solución del problema se aplicó un término muy conocido en el desarrollo de software y es el Ciclo de Vida del Software el cual implementa las siguientes etapas o fases:

a. Análisis
b. Diseño
c. Codificación
d. Prueba y corrección

A continuación, se explicará cada una de las etapas.

Etapa de Análisis:

En esta etapa se consideró cada una de las especificaciones propuestas en el documento de especificaciones y requerimientos y se logró determinar que lo más eficiente para solución sería utilizar el paradigma de programación orientada a objetos, ya que este nos permite el manejo de clases y objetos. Las clases son una plantilla abstracta de los individuos que interactúan en el problema. Tomando como base los individuos que interactúan en el robot, se logró determinar que se necesita una clase llamada Piso, la cual tendría los atributos como nombre, filas, columnas, costo de intercambio y costo de volteo, listado de patrones y listado de celdas; también se

necesita una clase llamada patrón, la cual tendría los atributos como código y secuencia, y finalmente una clase llamada celda que tiene los atributos fila, columna y color. Estas tres clases se convirtieron en los tipos de datos utilizados para generar las listas dinámicas donde almacenaran los datos.

Las clases lista simples y listas dobles eran indispensables ya que estas nos proporcionarían el manejo de datos, la lista simple fue la encargada de almacenar los datos de pisos y una de patrones, dentro de cada nodo patrones se necesitó una lista doble donde se almaceno las celdas.

Tabla I. *Lista de clases.*

CATEGORÍA	CATEGORÍA
PISO	TDA
PATRON	TDA
CELDA	TDA
LISTA PISOS	LISTA DATOS
LISTA PATRON	LISTA DATOS
LISTA CELDAS	LISTA DATOS

Fuente: elaboración propia.

Luego de tener los datos se procedería aplicar el algoritmo para encontrar el costo más económico encontrado por el robot.

Etapa de Diseño:

Obteniendo la información de la etapa de análisis, se procedió a crear el diseño de las clases propuestas en la Tabla I y la relación entre ellas. Para esta etapa se utilizó el diagrama de clases que nos proporciona UML. Para el diagrama utilizamos relaciones de asociación para relacionar las clases TDA con las listas de datos, el diagrama también cuenta con la multiplicidad en las relaciones entre las clases.

El diagrama de clases ya finalizado es el siguiente:

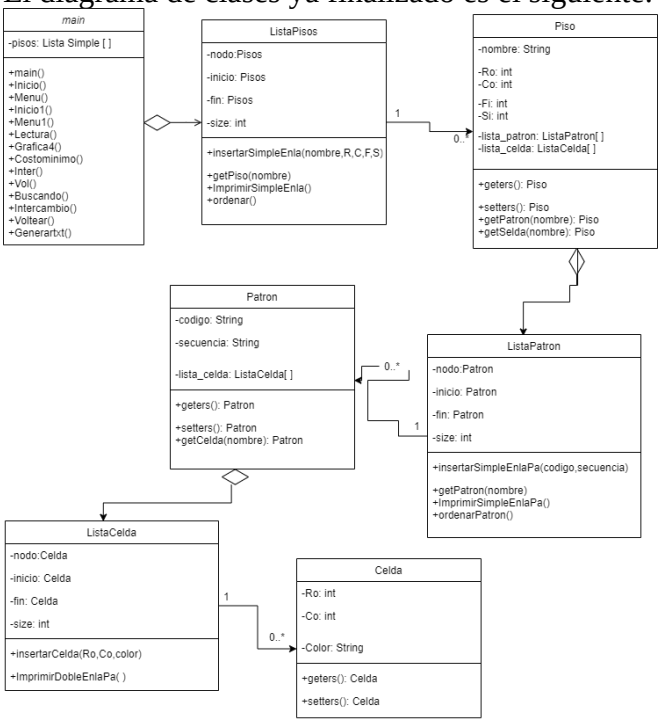

Figura 1. Diagrama de clases UML.
Fuente: elaboración propia.

Etapa de Codificación:

Para esta etapa del proyecto utilizamos los siguientes componentes:

a. Sistema operativo: Windows 10
b. Editor de código: Visual Studio Code
c. Lenguaje de Programación: Python 3.
d. Librería para XML: Element Tree
e. Creación Reportes: Graphviz

Debemos saber que la codificación va de la mano con el diagrama mostrado en la figura 1, se explicara la funcionalidad de cada una de las clases y los métodos utilizados.

Se comenzó codificando los dos TDA que serán la base del proyecto, la clase piso cuenta con distintos métodos empezando por un constructor quien se encarga de inicializar los atributos de la clase, también cuenta con los métodos getPiso y getPatron ambos métodos retorna un piso y un listado de

patrones respectivamente. La clase patrón al igual que la clase anterior cuenta con un constructor y con sus respectivos getters y setteres, así como también la clase celda que cuenta con lo mismo.

Las siguientes clases a codificar fueron las listas simples y dobles ya que estas utilizan como nodos los tipos abstractos. Estas clases fueron implementadas haciendo uso de algoritmos de listas dinámicas cuentan ambas con sus métodos insertar e imprimir y cuenta también con un método que retorne un nodo del tipo TDA del cual la lista se basa, también cuenta con un método de ordenar alfabéticamente.

Ya teniendo la base fundamental del proyecto se procedió a codificar la parte visual, construyendo en la clase main, el método menú y proceso menú que ambos procesos generan en consola las opciones que el usuario podrá realizar como ejemplo está la figura 3 de la sección de Anexos.

La lectura de archivos XML se implementó haciendo uso de la librería Element Tree la cual nos proporciona parsear el contenido del archivo a un objeto el cual podemos recorrer para obtener los datos y estos fueron almacenados en las listas dinámicas creadas, un ejemplo de los archivos XML de entrada esta la figura 4 de la sección de Anexos.

Obteniendo los datos ya almacenados se procedió a codificar y aplicar el algoritmo sobre la información de los pisos y sus posiciones de las celdas en sus patrones para poder encontrar el costo mínimo de cambio.

La salida de datos XML se utilizó al igual que la lectura el implementar la librería Element Tree la cual a base de recorrer nuestra lista se obtuvo los archivos XML como ejemplo esta la figura 5 de la sección de Anexos.

Para generar los reportes haciendo uso de Graphviz se implementó un método que generaba un archivo con extensión dot (ejemplo en la figura 6 de la sección de Anexos) y que al procesarlo produce el reporte en PDF para que sea útil para el usuario.

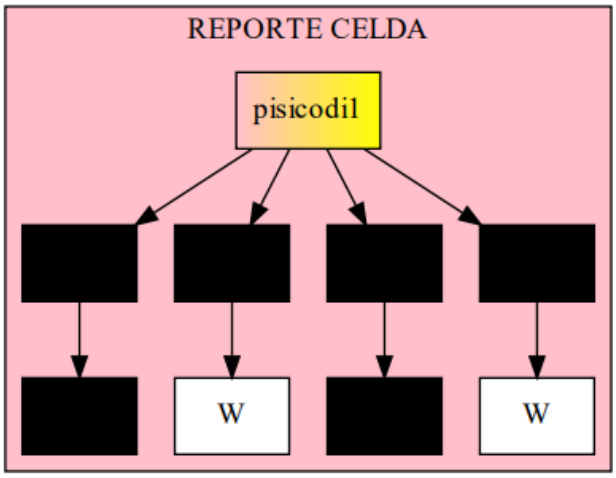

Figura 2. Reporte de Ejemplo.
Fuente: elaboración propia.

La codificación de este proyecto fue muy laborioso por el manejo de listas dinámicas construidas por TDA, se puso en práctica los conceptos de programación orientada a objetos, TDA y uso de lenguaje Python para aumentar los conocimientos en este lenguaje.

Etapa de Prueba y Corrección:

En esta etapa se realizaron pruebas de cada una de las funciones en especial las del manejo de datos, la creación de reportes con Graphviz y la salida de los archivos txt para ello se utilizó un archivo de entrada, cuya imagen está en la figura 4 de la sección de Anexos, al finalizar las pruebas se realizaron las correcciones necesarias para cumplir con las especificaciones.

Conclusiones

Los TDA tiene una gran facilidad para ser implementados en distintas estructuras de datos, y aun así proveer la misma funcionalidad, esto hace que los tipos abstractos sea muy versátiles.

La memoria dinámica tiene una gran ventaja, la cual beneficia tanto al programador como al equipo de cómputo, ya que gracias a que esta se crea y se destruye en tiempo de ejecución, la manipulación de datos es más sencilla y la computadora no desperdicia fragmentos de memoria.

La implementación de listas enlazadas tiene un gran beneficio respeto a las listas convencionales es que el orden de los elementos puede ser diferente al orden de almacenamiento en la memoria ya que estos en si son fragmentos de memoria que usan apuntadores para saber cuál es el fragmento siguiente.

Referencias bibliográficas

A. V. Aho, J. E. Hopcroft, (1998). *Estructura de datos y algoritmos.* Addison-Wesley Publishing Company, Inc.

Anexos

El menú del proyecto proporción al usuario seis opciones la primera permite el ingreso de la ruta de archivo que será procesado, la segunda permite mostrar los pisos cargados alfabéticamente, la tercera permite mostrar los patrones cargados alfabéticamente, la cuarta genera un archivo pdf con la gráfica de un patrón en específico, la quinta puede cambiar un patrón y la sexta termina la ejecución del programa.

```
========= ROBOT PISOS ARTESANALES =========
|1. Cargar Archivo                         |
|2. Mostrar los pisos cargados             |
|3. Mostrar los patrones cargados          |
|4. Mostrar graficamente el patron         |
|5. Seleccionar un nuevo código de patron  |
|6. Salir                                  |
============================================
Ingrese Opcion:
>.
```

Figura 3. Menú del proyecto.
Fuente: elaboración propia

Los archivos de entrada son proporcionados por el usuario, tienen extensión XML y tienen la siguiente estructura.

```xml
<?xml version="1.0"?>
<pisosArtesanales>
    <piso nombre="pisito">
        <R> 2 </R>
        <C> 4 </C>
        <F> 1 </F>
        <S> 1 </S>
        <patrones>
            <patron codigo="pisicodi1">
                WBWBWWWB
            </patron>
            <patron codigo="aisicodi2">
                BWBWWWWW
            </patron>
        </patrones>
    </piso>
    <piso nombre="coloreada">
        <R> 3 </R>
        <C> 3 </C>
        <F> 1 </F>
        <S> 1 </S>
```

Figura 4. Ejemplo de archivos de entrada.
Fuente: elaboración propia

Los archivos de salida son generados por el programa, tienen extensión txt y tienen la siguiente estructura.

```
pisicodi1.txt: Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
Buscando cambios que se deben hacer
Patron correcto en la posicion: 0,0
Patron correcto en la posicion: 0,1
Patron correcto en la posicion: 0,2
Patron correcto en la posicion: 0,3
Patron correcto en la posicion: 1,0
Patron correcto en la posicion: 1,1
Cambiar en la posicion: 1,2 El patron: B por: W
Volteo del piso en la posicion: 1,2 al color: W
Patron correcto en la posicion: 1,3
```

Figura 5. Ejemplo de archivos de salida.
Fuente: elaboración propia

La opción cinco, cuya función es generar el reporte gráfico, nos crea un archivo tipo dot que al procesarlo nos deja archivos PDF.

```
digraph G {
  node [shape=box fillcolor="pink" style=filled]
  subgraph cluster_p {
      label="Reporte Patrones"
      bgcolor="pink:yellow" nodoP[label="piso1" shape="box"];
  name0[label="W" shape="box"];
  name1[label="B" shape="box"];
  name2[label="W" shape="box"];
  name3[label="W" shape="box"];
  name4[label="B" shape="box"];
  name5[label="W" shape="box"];
  nodoP ->name0;
  nodoP ->name1;
  name0 ->name2;
  name1 ->name3;
  name2 ->name4;
  name3 ->name5;
}
}
```

Figura 6. Ejemplo de archivos dot de grafos.
Fuente: elaboración propia.