DISEÑO DE SISTEMA DE CONTROL PARA MISIONES DE RESCATE Y EXTRACCIÓN DE LA EMPRESA "CHAPIN WARRIOS S.A."

201403745 – Linda Madelin Fabiola Quelex Sep

Resumen

El proyecto pertenece al "Laboratorio de Introducción a la Programación y Computación 2" de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, donde se requiere desarrollar un sistema de control para realizar misiones de rescate o extracción en distintas ciudades.

Para el diseño de la solución se utilizó el lenguaje de programación Python, tipos de datos abstractos, paradigma de programación orientada a objetos, manejo de archivos XML, entre otros aspectos técnicos que se detallan en el ensayo.

El sistema de control cuenta con un menú a través de la consola para interactuar con el usuario e ir completando las misiones que se requieran, se utilizaron listas enlazadas y el concepto de matriz dispersa para la funcionalidad del sistema de control.

Palabras clave
Matriz dispersa, Python, XML, listas enlazadas.

Abstract

The project belongs to the "Laboratory of Introduction to Programming and Computing 2" of the Faculty of Engineering of the University of San Carlos of Guatemala, where it is required to develop a control system to carry out rescue or extraction missions in different cities.

For the design of the solution, the Python programming language, abstract data types, object-oriented programming paradigm, handling of XML files, among other technical aspects that are detailed in the essay, were obtained.

The control system has a menu through the console to interact with the user and complete the required missions, linked lists and the sparse matrix concept were used for the functionality of the control system.

Keywords
Sparse matrix, Python, XML ,*linked list.*

Introducción

El presente ensayo es sobre el proyecto 2 del Laboratorio de Introducción a la Programación y Computación 2, de la Facultad de Ingeniería, de la Universidad de San Carlos de Guatemala.

El proyecto fue nombrado como "Diseño de un sistema de control para misiones de rescate y extracción."

Para la realización del proyecto se utilizó el lenguaje de programación Python y el paradigma de programación orientada a objetos donde se implementó una matriz dispersa para cumplir con los requerimientos del proyecto.

Se solicita que el sistema de control debe ser capaz de generar mapas bidimensionales de ciudades para los robots ChapinEyes que sobrevuelan las ciudades para completar misiones.

Desarrollo del tema

Para el desarrollo del proyecto 2 del Laboratorio de Introducción a la Programación 2 (LABIPC2) se desarrolló e implementó los siguiente:

1. **Clase: NodoCabecera**
   Contiene dentro de sus atributos id, siguiente, anterior, que serán utilizados en las clases ListaCabeceras y MatrizDispersa.

2. **Clase: ListaCabeceras**
   Contiene los atributos primero, ultimo, tipo y size, y los métodos de "insertarnodocabecera" y "mostrarCabecera", con los cuales se implementa una lista doblemente enlazada que simula los ejes X y Y de la matriz dispersa.
   El método "insertarnodocabecera" crea los nodos que son requeridos optimizando el espacio en memoria, por ejemplo, si se requiere una matriz de 4*4, crea solo los nodos cabecera que son requeridos en este caso solo 1 y 3 para el eje x, y 2 y 4 para el eje y.

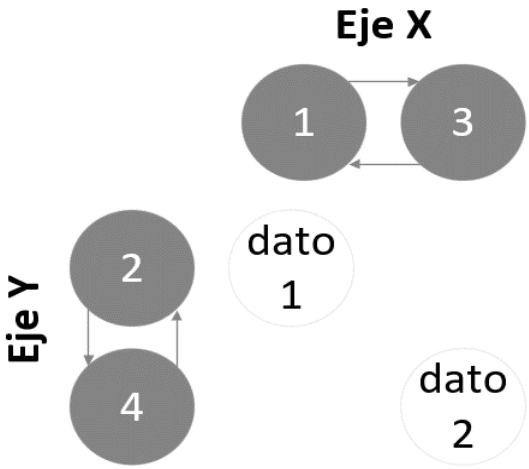


Figura 1. Lista Cabeceras.

Fuente: elaboración propia, marzo 2022.

3. **Clase: NodoCeldaMalla**
   Contiene los atributos: coordenadaX, coordenadaY, clasificacióncelda, valoración, siguiente, anterior, up, down, right, left y size y los métodos get y set para cada atributo, los cuales serán utilizados en la Matriz Dispersa.

4. **Clase: Matriz Dispersa**
   Esta clase requiere de las clases (1) ListaCabeceras y (2) NodoCeldaMalla, donde se definen los atributos fila y columna con la clase Lista Cabeceras, cuenta con el método de insertar para agregar cada nodo de tipo NodoCeldaMalla para crear el mapa bidimensional de las ciudades, para ello se comienza con la creación de filas y columnas a los cuales se les da el acceso a los nodos de la matriz, se realizan las validaciones para

verificar si ya existe información las coordenadas de la matriz.

Esta clase de Matriz Dispersa también cuenta con el método de graficar para generar un PDF con la representación gráfica del mapa bidimensional haciendo uso de Graphviz, se realizaron las validaciones de acuerdo a los requerimientos para la configuración de colores, la salida de este método se muestra en la siguiente figura

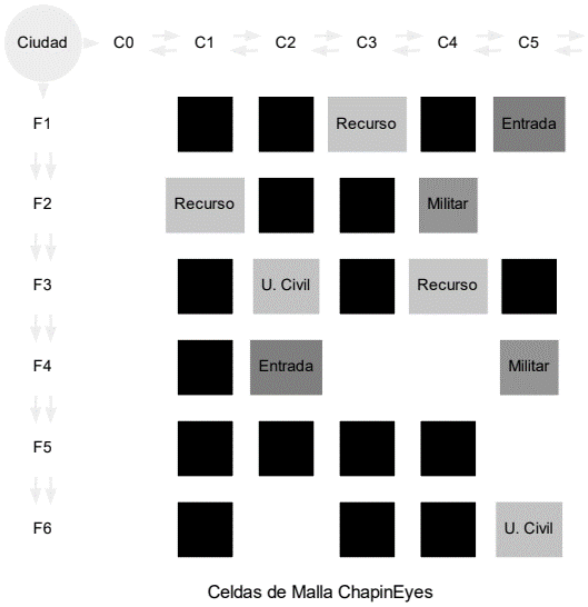


Figura 2. Ejemplo Mapa Bidimensional.

Fuente: elaboración propia, marzo 2022.

También se desarrolló el método de actualizarMilitar para agregar a cada mapa bidimensional las unidades militares, el método recibe como parámetros la coordenadaX, coordenadaY y la valoraciónmilitar, recorre la matriz para verificar si ya existe la coordenada x y y, según los parámetros de entrada, si ya existe actualiza el atributo valoración del nodo de la matriz, si no existe devuelve “no existe”.

```
def actualizarMilitar(self,fila,columna, valoraciónmilitar):
    try:
        tmp:NodoCeldaUnitariaMalla=self.filas.getCabecera(fila).getAcceso()
        while tmp!=None:
            if tmp.coordenadaX== fila and tmp.coordenadaY == columna:
                tmp.clasificacioncelda="M"
                tmp.valoracion=valoraciónmilitar
                print(tmp.clasificacioncelda)
                return tmp

            tmp=tmp.getright()
        return None
    except:
        print('No existe')
        return None
```

Figura 3. Método actualizarMilitar.

Fuente: elaboración propia, marzo 2022.

5. **Clase: NodoTipoCelda**
   Contiene los atributos de cantiadacelda, siguiente, anterior y id y los métodos get y set de cada atributo.

6. **Clase: ListasTipoCelda**
   Contiene los atributos size, primero y ultimo para implementar una lista doblemente enlazada para ir almacenando la cantidad de celdas de tipo civil en cada ciudad, con esta clase se realizan las validaciones iniciales para realizar una misión de rescate

7. **Clase: NodoTipoCeldaRecursos**
   Contiene los atributos de cantiadacelda, siguiente, anterior y id y los métodos get y set de cada atributo.

8. **Clase: ListaTipoCeldaRecursos**
   Contiene los atributos size, primero y ultimo para implementar una lista doblemente enlazada para ir almacenando la cantidad de celdas de tipo recurso en cada ciudad, con esta clase se realizan las validaciones iniciales para realizar una misión de extracción.

9. **Clase: Nodo Ciudad**
   Contiene los atributos de nombreciudad, siguiente, anterior, id, MatrizDispersa, ListaTipoCelda y ListaTipoCeldaRecursos, cuenta con los métodos get y set de cada atributo.

10. **Clase: ListaCiudades**
    Esta clase cuenta con los atributos size, primero y ultimo para crear una lista doblemente enlazada y cargar la información de las ciudades que vienen del archivo XML. Cuenta con los métodos de insertar, mostrar y retornar_nodo. Esta clase integra todas las clases anteriormente definidas.

11. **Clase: NodoRobot**
    Contiene los atributos nombrerobot, tipomisión, valoración, siguiente, anterior, size y id, cuenta con los métodos get y set de cada atributo.

12. **Clase: ListaRobots**
    Contiene como atributos: size, primero y ultimo; para implementar una lista doblemente enlazada que sirve para almacenar la información de los robots que vienen en los archivos XML, cuenta con los métodos de insertar_robot, mostrar_robot, retornar_nodo y retornar_NombreRobotRescate los cuales sirven para realizar las validaciones iniciales para realizar una misión de rescate o de extracción y para mostrar al usuario la información del robot que esta eligiendo o que realizará la misión.

13. **Main:**
    Contiene el método elementTree el cual realiza la lectura de los archivos de entrada de tipo XML, realizando recorridos y almacenando en los listas y matriz dispersa anteriormente mencionadas.

En esta clase se desarrolló el menú para el sistema de control utilizando la consola.

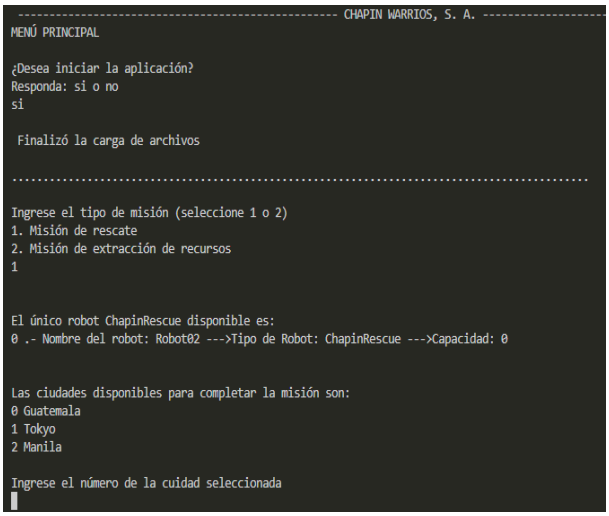


Figura 4. Menú

Fuente: elaboración propia, marzo 2022.

A través del menú el usuario puede seleccionar el tipo de misión a realizar; el sistema retornará la información de los robots disponibles y el usuario seleccionara el robot, luego el sistema retorna las ciudades disponibles según al información del archivos XML; el usuario debe ingresar el número de la ciudad seleccionada donde el sistema de control hará la verificación de las unidades civiles y recursos según el tipo de misión y mostrará al usuario si es posible realizar la misión, si es posible, entonces mostrará el mapa bidimensional de la ciudad para que el usuario ingrese las coordenadas de la unidad civil a rescatar o recurso a extraer, posteriormente el sistema de control mostrará un mensaje con la información de la misión realizada.

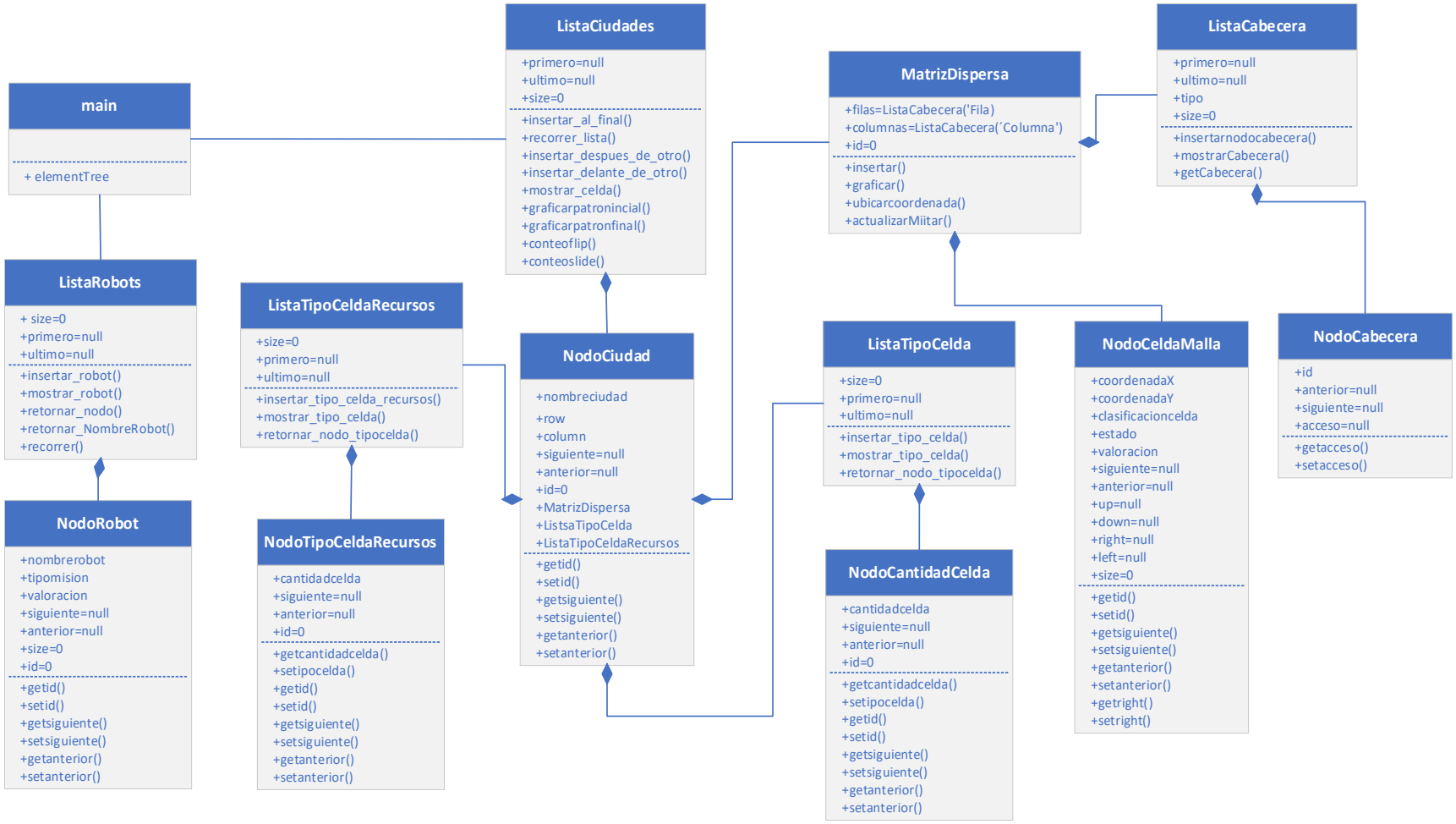
Finalmente mostrará al usuario si desea realizar otra misión o desea salir de la aplicación.

Referencias bibliográficas

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*, Third Edition. The MIT Press, 3rd edition, 2009.
2. Robert Kruse, CL Tondo, et al. *Data structures and program design in C*. Pearson Education India, 2007.
3. Mark Allen Weiss. *Data Structures and Algorithm Analysis in C.* Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 199

APÉNDICE

1. Diagrama de Clases del Proyecto



FUENTE: diagrama de clases proyecto 2, elaboración propia, marzo 2022.