Universidad San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ciencias y Sistemas
Introducción a la Programación y Computación 2

Inga. Claudia Liceth Rojas Morales
Ing. Marlon Antonio Pérez Türk
Ing. Byron Rodolfo Zepeta Arevalo
Ing. Jose Manuel Ruiz Juarez
Ing. Edwin Estuardo Zapeta Gómez

Tutores de curso:
Alexander Raymundo Ixvalan Pacheco
Jackeline Benitez
Javier Lima
Viany Paola Juárez
José Carlos Estrada

PROYECTO 2

OBJETIVO GENERAL

Se busca que el estudiante sea capaz de dar una solución al problema que se le plantea, mediante la lógica y conocimientos que se le han impartido durante la clase y que han sido aplicados en el laboratorio.

OBJETIVOS ESPECÍFICOS

- Que el estudiante sea capaz de aplicar abstracción a un problema dado.
- Implementar una solución utilizando el lenguaje de programación Python.
- Utilizar estructuras de programación secuenciales, cíclicas y condicionales
- Que el estudiante genere reportes con la herramienta Graphviz.
- Que el estudiante sea capaz de manipular archivos XML.
- Que el estudiante desarrolle programación orientada a objetos utilizando estructuras propias
- Que el estudiante utilice los conceptos de TDA y aplicarlos a memoria dinámica.

DESCRIPCIÓN DEL PROBLEMA

La empresa Chapín Warriors, S. A. ha desarrollado equipos automatizados para rescatar civiles y extraer recursos de las ciudades que se encuentran inmersas en conflictos bélicos. Con el fin de realizar las misiones de rescate y extracción, Chapín Warriors, S. A. ha construido drones autónomos e invisibles para los radares llamados ChapinEyes. Los ChapinEyes sobrevuelan las ciudades y construyen un mapa bidimensional de la misma, este mapa bidimensional consiste en una malla de celdas, donde cada celda es identificada como un camino, un punto de entrada, una unidad de defensa, una unidad civil, un recurso o una celda intransitable. La figura 1 muestra la malla de celdas que elabora ChapinEyes luego de sobrevolar una ciudad en conflicto.

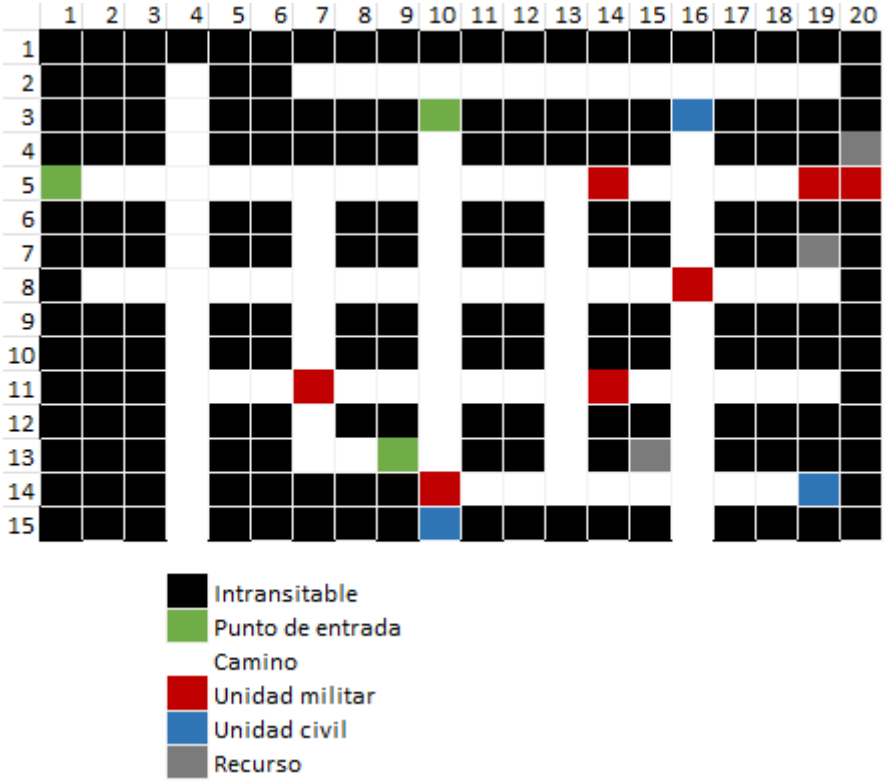


Fig. 1 – Malla de celdas creada por el dron ChapinEyes para una ciudad en conflicto

Las celdas están clasificadas de la siguiente forma:

- Punto de entrada: Celda donde puede iniciar una misión de rescate o una misión de extracción. Estas celdas son transitables.
- Intransitable: Celda donde no es posible transitar.
- Camino: Celda donde si es posible transitar.
- Unidad militar: Celda donde existe una unidad militar, toda unidad militar tiene una valoración que consiste en un número entero que representa su capacidad de combate, mientras mayor es el número entero, mayor es su capacidad de combate. Una celda de tipo “Unidad militar” es transitable, siempre y cuando se pueda superar su capacidad de combate.
- Unidad civil: Celda donde existe una unidad civil. Una celda de tipo “Unidad civil” siempre es transitable.

- Recurso: Celda que contiene algún recurso físico dentro de la ciudad. Una celda de tipo “Recurso” no puede ser transitada.

Para poder completar las misiones de rescate o extracción de recursos de ciudades en conflicto, Chapín Warriors, S. A., ha creado unidades robóticas que pueden realizar dichas misiones. Estas unidades robóticas están clasificadas de la siguiente forma.

- ChapinRescue: Unidad robótica capaz de ingresar a la ciudad en uno de los puntos de entrada, seguir un camino seguro (celdas transitables) hasta la “unidad civil” que se desea rescatar y rescatarla. Las unidades ChapinRescue no pueden enfrentar “Unidades Militares” de la ciudad en conflicto.
- ChapinFighter: Unidad robótica capaz de ingresar a la ciudad en uno de los puntos de entrada, seguir un camino hasta un “recurso” y extraerlo. Las unidades ChapinFighter tienen una valoración que consiste en un número entero que representa su capacidad de combate, mientras mayor es el número entero mayor es su capacidad de combate. Las unidades ChapinFighter pueden derrotar “Unidades Militares” siempre y cuando su capacidad de combate sea mayor a la capacidad de combate de la “Unidad militar”, en este caso, la capacidad de combate de ChapinFighter disminuye su capacidad de combate restando la capacidad de combate de la “Unidad Militar” vencida.

Para poder completar el sistema de rescate y extracción, la empresa Chapín Warriors, S.A., lo ha contratado para crear un **sistema de control** que recibirá la malla de celdas del dron ChapinEye, a continuación, el sistema de control será capaz de definir una serie de misiones e indicará los robots que pueden completar la misión, así como los caminos que deben seguir en la malla de celdas para ejecutar las misiones.

Archivo de configuración

El **sistema de control** recibirá un archivo XML de configuración con la siguiente estructura:

```
<?xml version="1.0"?>
<configuracion>
  <listaCiudades>
    <ciudad>
      <nombre filas=”valorEntero” columnas=”valorEntero”> valorAlfanumerico </nombre>
      <fila[1] numero=”valorEntero”> “”””””””””””””””””””[2] </fila>
      <fila numero=”valorEntero”> “**** **                  *” </fila>
      <fila numero=”valorEntero”> “**** *****E*****C****” </fila>
      <fila numero=”valorEntero”> “**** ***** ***** ***R” </fila>
      …
      <unidadMilitar fila=”valorEntero” columna=”valorEntero> valorEntero[3] <unidadMilitar/>
      …
    </ciudad>
    …
  </listaCiudades>
  <robots>
    <robot>
      <nombre tipo=”ChapinFighter”[4] capacidad=”valorEntero”[5]> valorAlfanumerico </nombre>
    </robot>
  </robots>
</configuracion>
```

1. Dentro de la etiqueta “ciudad” hay n etiquetas “fila”, dependiendo del valor indicado en el atributo “filas” de la etiqueta “nombre”.
2. El valor de la etiqueta “fila” es un valor alfanumérico delimietado por comillas (“) que contiene m caracteres, donde m es igual al valor entero definido en el atributo “columnas”. Los caracteres válidos para este valor son los siguientes: asterisco (* - representa una celda intransitable), espacio (‘ ‘ – representa una celda transitable), letra E (E – representa un punto de entrada), letra C (C – representa una unidad civil), letra R (R – representa un recurso).
3. Representa la capacidad de combate
4. Los tipos posibles son ChapinRescue o ChapinFighter.
5. Dentro de la etiqueta robot, la etiqueta nombre tiene el atributo “capacidad” solamente si el atributo tipo es “ChapinFighter” y representa la capacidad de combate del robot.

Misiones

Una vez el **sistema de control** tenga configuradas ciudades y robots, se le podrá solicitar al **sistema de control** que genere una estrategia para realizar una misión. Los tipos de misión posibles son los siguientes:

1. Misión de rescate
    - Estas misiones son realizadas por robots de tipo ChapinRescue. Si se tiene disponibles más de un robot ChapinRescue, el usuario deberá seleccionar uno. Si no hay robots ChapinRescue estas misiones no pueden ser llevadas a cabo.
    - Se debe identificar la ciudad donde se realizará la misión de rescate, solamente las ciudades con “unidades civiles” pueden tener misiones de rescate.
    - Toda misión de rescate debe iniciar en un punto de entrada.
    - Si la ciudad tiene varias “unidades civiles” se deberá elegir la unidad civil a rescatar, si solamente hay una “unidad civil” no se solicitará esta elección.
    - Las misiones de rescate deben identificar un camino seguro, es decir, deben evitar cualquier camino que contenga “unidades militares”.
2. Misión de extracción de recursos
    - Estas misiones son realizadas por robots de tipo ChapinFighter. Si se tiene disponibles más de un robot ChapinFighter, el usuario deberá seleccionar uno

(debe poder ver las capacidades de combate de cada robot). Si no hay robots ChapinFighter estas misiones no pueden ser llevadas a cabo.

- Se debe identificar la ciudad donde se realizará la misión de extracción de recursos, solamente las ciudades con “Recursos” pueden tener misiones de extracción.
- Toda misión de extracción debe iniciar en un punto de entrada.
- Si la ciudad tiene varias celdas con “recursos” deberá elegir el recurso que desea extraer, si solamente hay un “recurso” no se solicitará esta elección.
- Las misiones de extracción de recursos pueden transitar por celdas con “unidades militares” siempre y cuando su capacidad de combate sea mayor a la capacidad de combate de la “unidad militar”, en estos casos, el robot utilizado disminuirá su capacidad de combate restando la capacidad de combate de la “unidad militar” derrotada.

Requerimientos del Sistema de control

El sistema a desarrollar debe cumplir los siguientes requisitos

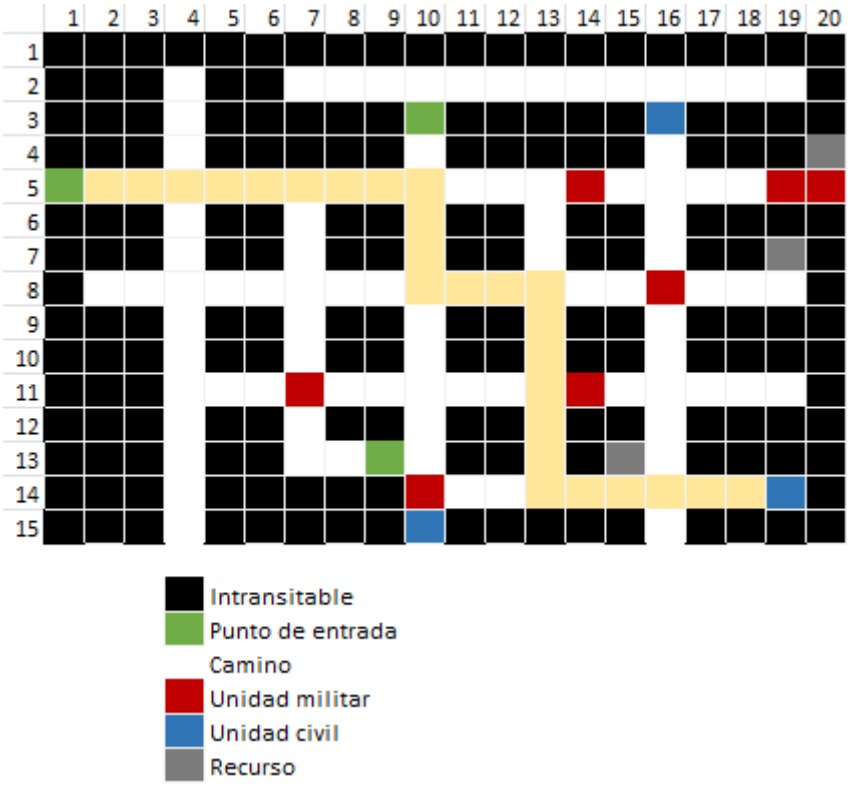
- Ser amigable y fácil de utilizar
- Cargar archivos de configuración. Si se cargan varios archivos de configuración y los nombres de las ciudades o robots se repiten, entonces, se actualizarán los datos dentro del sistema de control.
- Ejecutar misiones y mostrar gráficamente el camino a seguir y los datos resultantes de la misión. Si la misión no es posible, la salida será “Misión Imposible”. Es posible que una misión pueda ser ejecutada siguiendo diferentes caminos, para este proyecto solamente se requiere identificar un camino que permita completar exitosamente la misión.
- Utilizar Graphviz para la presentación gráfica de las misiones.

Salida esperada

El sistema de control debe mostrar gráficamente la ciudad seleccionada y los datos de la misión elegida. Si se hubiese elegido la ciudad de la figura No. 1 y se solicitara la misión de rescate de la “unidad civil” ubicada en la fila 14 columna 19, entonces, la figura No. 2 muestra una posible salida para esta misión.

La figura No. 3 muestra la salida si en la ciudad de la figura No. 1 se solicitara una misión de extracción de recursos para el recurso ubicado en la fila 7 columna 19. Para este ejemplo se asume que todas las unidades militares poseen una capacidad de combate de 20 y que el robot02 de tipo ChapinFighter posee una capacidad de combate de 100.

Ruta de rescate:



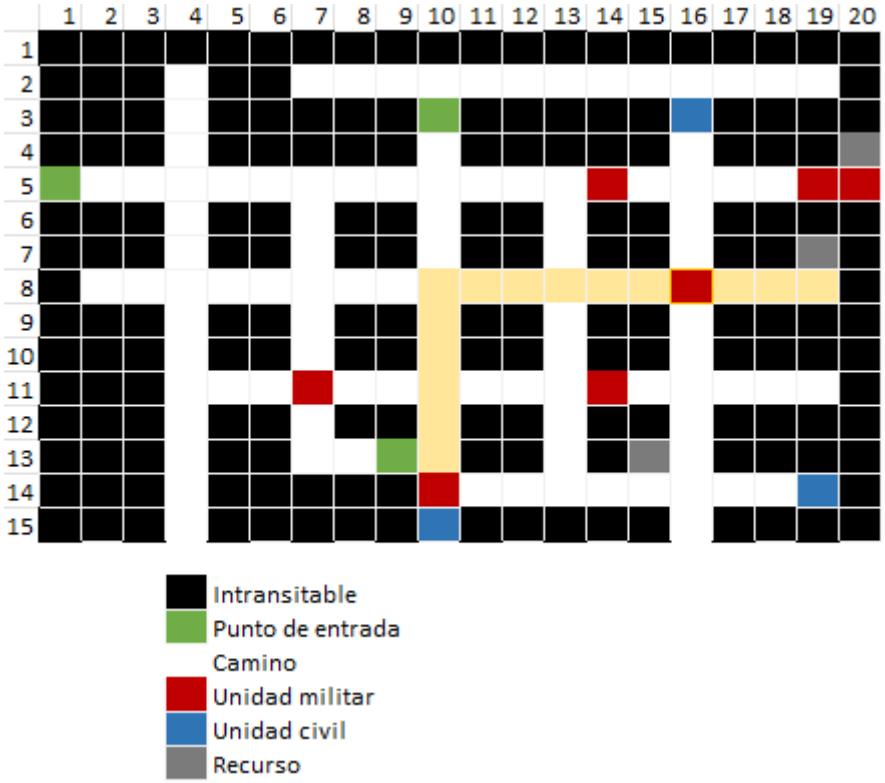
Tipo de misión: rescate

Unidad civil rescatada: 14,19

Robot utilizado: robot01 (ChapinRescue)

Fig. No. 2 – Salida de una misión de rescate (El robot con nombre robot01 debe haber sido creado en el archivo de configuración como un robot de tipo ChapinRescue)

Ruta de extracción de recurso:



Tipo de misión: extracción de recursos

Recurso extraído: 7,19

Robot utilizado: robot02 (ChapinFighter – Capacidad de combate inicial 100, Capacidad de combate final 80)

Fig. No. 3 – Salida de una misión de extracción de recursos (El robot con nombre robot02 debe haber sido creado en el archivo de configuración como un robot de tipo ChapinFighter con capacidad de combate 100)

CONSIDERACIONES

Se deberá realizar la implementación utilizando programación orientada a objetos, algoritmos desarrollados por el estudiante e implementación de estructuras a través de Tipos de Dato Abstracto (TDA) desarrollados por el estudiante; que permitan almacenar la información de los archivos de entrada y poder interactuar con dicha información. El estudiante deberá abstraer la información y definir qué estructuras implementar que le faciliten la solución. Por lo que puede implementar pilas, colas, listas simples, dobles, circulares o listas de listas para poder solventar el proyecto. **No está permitido el uso de estructuras implementadas de Python (list, dict, tuple, set).**

El cómo se utilicen las estructuras anteriormente descritas para guardar los datos del archivo de entrada queda a discreción del alumno. Tomar en cuenta que al ingresar el archivo de entrada todos los datos dentro del archivo XML deben de ser cargados a las listas para luego tener la habilidad de ejecutar las misiones.

Debe utilizarse versionamiento para el desarrollo del proyecto. Se utilizará la plataforma **Github** en la cual se debe crear un repositorio en el que se gestionará el proyecto. Se deben realizar 4 releases o versiones del proyecto (se recomienda realizar una por semana del tiempo disponible). Se deberá agregar a sus respectivos auxiliares como colaboradores del repositorio. El último release será el release final y se deberá de realizar antes de entregar el proyecto en la fecha estipulada.

DOCUMENTACIÓN

Para que el proyecto sea calificado, el estudiante deberá entregar la documentación utilizando el formato de ensayo definido para el curso. En el caso del proyecto, el ensayo puede tener un mínimo de 4 y un máximo de 7 páginas de contenido, este máximo no incluye los apéndices o anexos donde se pueden mostrar modelos y diseños utilizados para construir la solución. Es obligatorio incluir el diagrama de clases que modela la solución de software presentada por el estudiante.

RESTRICCIONES

- Solo se permitirá la utilización de los IDEs discutidos en el laboratorio.
- Uso de TDA implementados por el estudiante obligatorio; el uso de estructuras de Python (list, dict, tuple, set) resultará en penalización del 100% de la nota.
- Uso obligatorio de programación orientada a objetos (POO) desarrollada por completo por el estudiante. De no cumplir con la restricción, no se tendrá derecho a calificación.
- El nombre del repositorio debe de ser **IPC2_Proyecto2_#Carnet.**
- El estudiante debe entregar la documentación solicitada para poder optar a la calificación.
- Los archivos de entrada no podrán modificarse.
- Deben existir 4 releases mínimo, uno por cada semana, de esta manera se corrobora el avance continuo del proyecto.
- Se calificará de los cambios realizados en el cuarto release. Los cambios realizados después de ese release no se tomarán en cuenta.
- Cualquier caso de copia parcial o total tendrá una nota de 0 y será reportada a escuela.
- Para dudas concernientes al proyecto se utilizarán los foros en UEDI de manera que todos los estudiantes puedan ver las preguntas y las posteriores respuestas.
- **NO HABRÁ PRÓRROGA.**

ENTREGA

- La entrega será el **01 de abril** a más tardar a las 11:59 pm.
- La entrega será por medio de la UEDI.
- La documentación debe estar subida en el repositorio en una carpeta separada.
- Para entregar el proyecto en UEDI se deberá subir un archivo de texto con el link del repositorio.