

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

**Ing. Claudia Liceth Rojas Morales**  
**Ing. Marlon Antonio Pérez Türk**  
**Ing. José Manuel Ruiz Juárez**  
**Ing. Edwin Estuardo Zapeta Gómez**  
**Ing. Fernando José Paz González**

**Tutores de curso:**  
**Andrea María Cabrera Rosito**  
**Josué Alfredo González Caal**  
**Paula Gabriela García Reynoso**  
**Mario César Morán Porras**  
**Denilson Florentín De León Aguilar**



## **PROYECTO No. 2**

### **OBJETIVO GENERAL**

Modelar, documentar e implementar una solución al problema que se plantea utilizando las herramientas de desarrollo presentadas en clase y laboratorio.

### **OBJETIVOS ESPECÍFICOS**

- Implementar una solución utilizando el lenguaje de programación Python.
- Utilizar estructuras de programación secuenciales, cíclicas y condicionales.
- Generar reportes con la herramienta Graphviz.
- Manipular archivos XML.
- Utilizar los conceptos de **TDA** y aplicarlos a memoria dinámica.
- Utilizar estructuras de programación propias.
- Utilizar el paradigma de programación orientada a objetos.

## DESCRIPCIÓN DEL PROBLEMA

El Ministerio de Educación de Guatemala ha invitado a todas las universidades del país a participar en un concurso para desarrollar soluciones mecatrónicas innovadoras. La mecatrónica es una rama multidisciplinaria de la Ingeniería que desarrolla dispositivos y tecnologías que involucran varios campos del conocimiento en los que se unen sistemas, electrónica, mecánica y control.

El concurso consiste en crear un dispositivo que sea capaz de recolectar objetivos que estarán ubicados en una maqueta rectangular especial donde existirán caminos y paredes. Los caminos estarán **definidos de forma horizontal o vertical, pero nunca en diagonal**. El dispositivo por desarrollar debe ser colocado en la entrada de la maqueta y debe determinar un camino que le permita recolectar los objetivos determinados **sin regresar nunca por un camino ya recorrido**.

La figura 1 muestra un ejemplo de la configuración inicial de una maqueta con caminos y paredes.

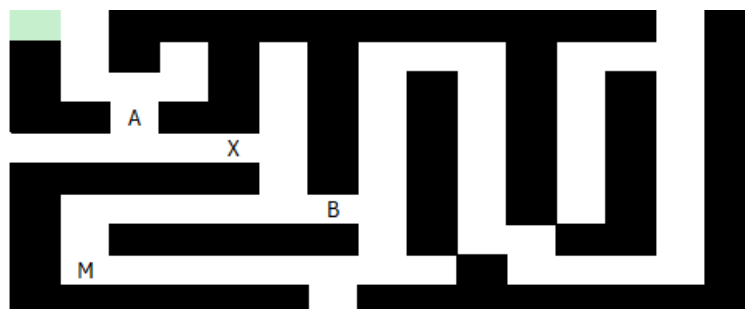




Figura No. 1 – Configuración inicial de una maqueta con caminos y paredes

La imagen de la figura 1 tiene la siguiente notación:

-  Representa la entrada a la maqueta
-  Representa una pared

Los espacios en blanco representan caminos y las **letras A, B, M y X representan objetivos**. El problema consiste en visitar los objetivos en un orden preestablecido, por ejemplo, A, X, B y M creando un camino lineal **que no reutilice segmentos de camino recorridos previamente**. La figura 2 muestra el ejemplo de la forma de recolectar los objetivos en el orden indicado.

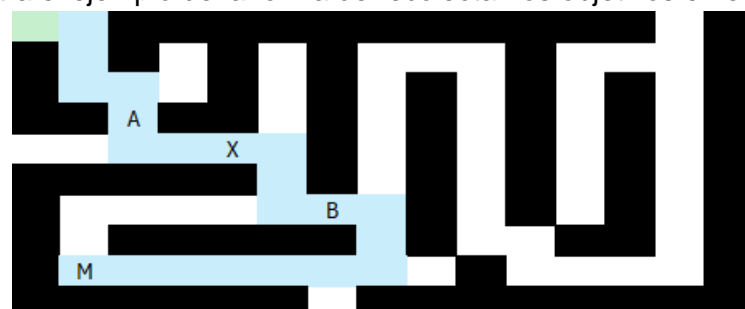


Figura No. 2 – Camino lineal sin reutilizar segmentos anteriores para recolectar los objetivos en el orden A, X, B y M

En la figura 2, siguiendo el camino marcado en azul, es posible recolectar los objetivos siguiendo el orden A, X, B y M sin reutilizar segmentos de camino.

La figura 3 muestra una maqueta que posee una variación:

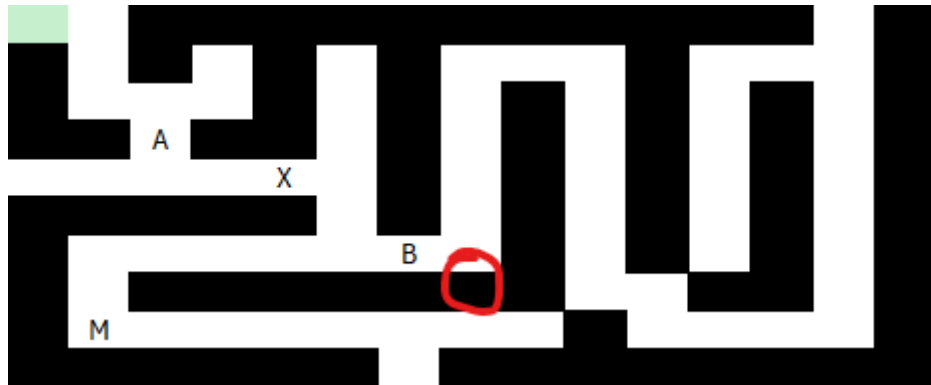


Figura No 3 – Maqueta con variación marcada con círculo rojo

Al modificar la maqueta con la variación de la figura 3 e intentar recolectar los objetivos en el orden A, X, B y M **no se podrá encontrar un camino lineal sin reutilizar segmentos de camino como se muestra en la figura 4.**

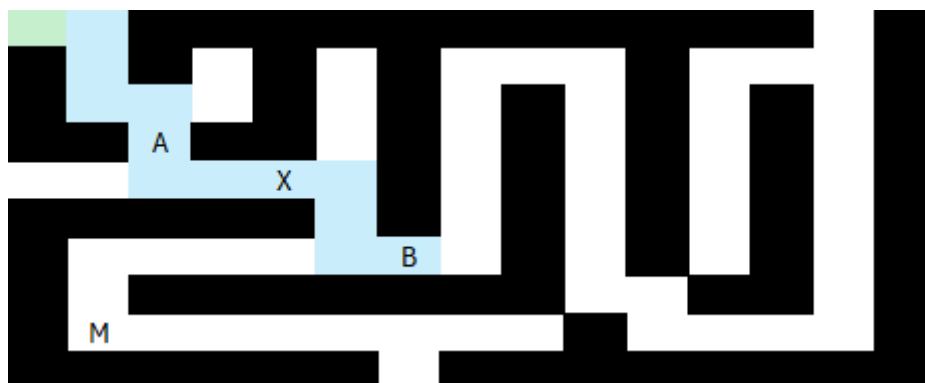


Figura No. 4 – Camino lineal sin reutilizar segmentos anteriores

En el caso de la maqueta configurada en las figuras 3 y 4, al intentar crear un camino lineal sin reutilizar segmentos anteriores, solamente podremos recolectar los objetivos A, X y B. Debido a que para recolectar el objetivo M se necesita regresar por un segmento de camino ya reutilizado, **entonces no es posible crear una solución que incluya los 4 objetivos.**

La Escuela de Sistemas le ha seleccionado para crear un algoritmo capaz de identificar el camino lineal sin reutilizar segmentos de caminos anteriores para **recolectar "N"** objetivos en un orden predeterminado utilizando programación orientada a objetos y TDAs lineales cuyas estructuras de programación deben ser propias.

## ENTRADA

Se utilizará un archivo XML para determinar “N” maquetas determinando las dimensiones de la maqueta, su entrada, “M” objetivos representados por strings distintos a “\*” y “-”, y la estructura de paredes y caminos, representando las paredes con el carácter “\*” y los caminos con el carácter “-”.

### entrada.xml

```
<?xml version="1.0"?>
<config>
  <maquetas>
    <maqueta>
      <nombre> [valorAlfanumerico] </nombre>
      <filaS> [valorNumerico] </filas>
      <columnas> [valorNumerico] </columnas>
      <entrada>
        <fila> [valorNumerico] </fila>
        <columna> [valorNumerico] </columna>
      </entrada>
      <objetivos>
        <objetivo>
          <nombre> [valorAlfanumerico] </nombre>
          <fila> [valorNumerico] </fila>
          <columna> [valorNumerico] </columna>
        </objetivo>
        ...
      </objetivos>
      <estructura> [ filas X columnas caracteres * para pared y - para camino ] </estructura>
    </maqueta>
    ...
  </maquetas>
</config>
```

Tomar en cuenta que este archivo es incremental, es decir, se pueden ingresar varios archivos de entrada, de tal manera que se pueden crear nuevas maquetas.

La cantidad de maquetas a cargar y sus dimensiones únicamente estarán limitadas por la capacidad de memoria de la computadora que ejecute el programa.

## INTERFAZ DE USUARIO

Se debe crear una interfaz de usuario fácil de utilizar e intuitiva que permita al usuario realizar las siguientes funciones:

- a. **Inicialización** – Para que el sistema pueda inicializarse sin ninguna información previa.
- b. Cargar un archivo XML de entrada
- c. Gestión de maquetas
  - a. **Ver listado de maquetas ordenado alfabéticamente**
  - b. Ver configuración de maqueta (utilizando Graphiz): Presentar la maqueta identificando claramente paredes, caminos, entrada, ubicación de objetivos y el orden en que los objetivos deben ser recolectados.
- d. Resolución de maquetas
  - a. **Ver gráficamente el camino para recolectar objetivos de una maqueta seleccionada (utilizando Graphiz)**
- e. Ayuda – Mostrar la información del estudiante y un link hacia la documentación del proyecto.

## 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 (TDAs) propios del estudiante, es decir, no deben utilizar estructuras de datos propias de Python (list, dict, tuple, set). **El estudiante deberá abstraer la información y definir qué estructuras implementar para construir la solución.**

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 sugiere desarrollar un Release por semana). 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 del diseño del software desarrollado** y se recomienda incluir el modelo conceptual y los diagramas de actividades que modelan 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.
- Los archivos de salida deben llevar la estructura mostrada en el enunciado obligatoriamente.
- **Deben existir 4 releases mínimo**, podría ser uno por cada semana, de esta manera se corrobora el avance continuo del proyecto.
- Se calificará de los cambios realizados en el último release hasta la fecha de entrega. 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 domingo **02 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.