Instituto Tecnológico de Costa Rica **I Semestre 2021**

Centro Académico de Alajuela **II Proyecto Programado**

IC-4700 Lenguajes de Programación **MSc. Samanta Ramijan Carmiol**

Proyecto Programado

Programación Funcional en Racket.

1. Motivación

Como actividad para la puesta en práctica de programación funcional, mediante el lenguaje Racket, se propone el desarrollo de un solucionador de laberintos.

2. Objetivos Formativos.

La presente asignación pretende servir como herramienta para que las y los estudiantes logren el objetivo de aprender Racket como lenguaje ejemplo para el paradigma de programación funcional. Además de tener la oportunidad de trabajar con imágenes desde el entorno de desarrollo Dr.Racket.

3. Metodología.

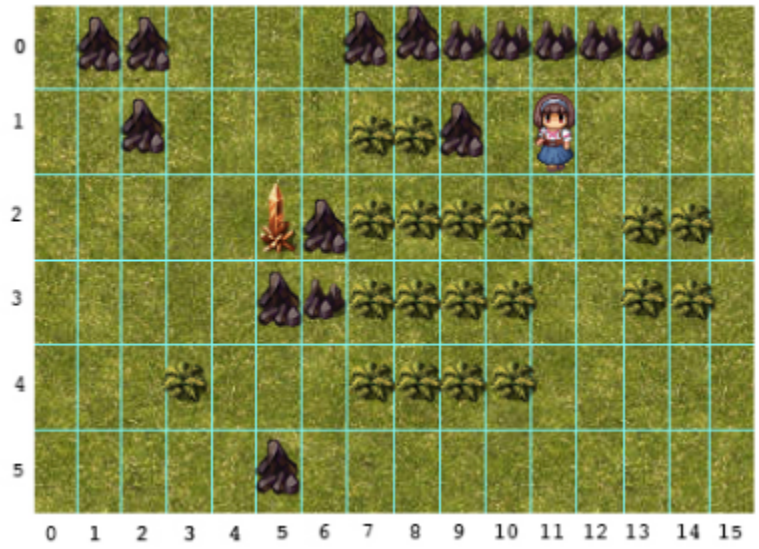
De manera similar al primer proyecto, las y los estudiantes deberán implementar un programa inteligente, en este caso, el programa deberá ser capaz de generar laberintos, resolverlos y brindar una solución visual a la persona usuaria.

En grupos de trabajo de tres estudiantes, las y los estudiantes deberán modelar, programar y documentar un programa diseñado siguiendo las pautas del paradigma funcional e implementado en Racket.

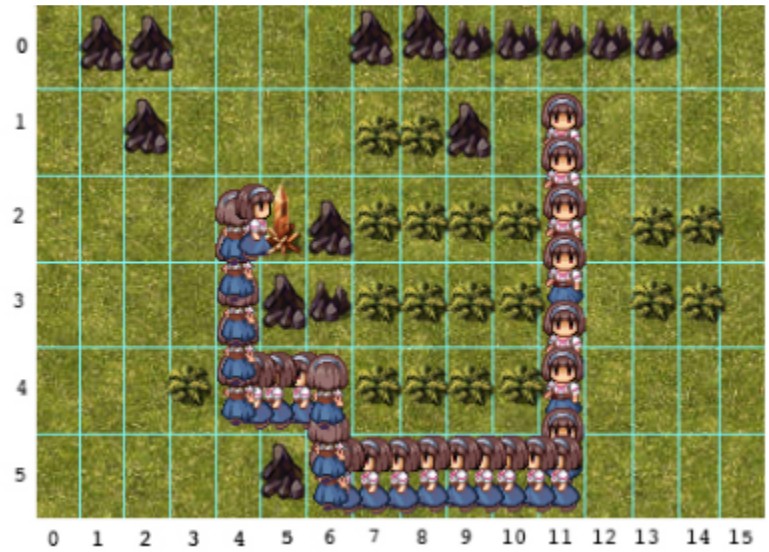
Un personaje deberá poder navegar un espacio dividido en *casillas*, obteniendo como parámetros un punto de partida y una meta.

El espacio deberá ser generado de forma aleatoria cada vez, lo llamaremos laberinto, porque no por medio de todos los caminos será posible llegar a la meta, ya que habrán dos tipos de obstáculos evitando el paso del personaje. El personaje podrá moverse hacia las casillas adyacentes (arriba, abajo, izquierda o derecha) en caso de que no haya un obstáculo.

En la siguiente figura, tenemos una representación del laberinto, con la niña como personaje, las rocas y la vegetación como obstáculos y el fuego como meta.



Cada grupo de trabajo, deberá elegir las imágenes que representen a su personaje, dos obstáculos y la meta final. Se deberá implementar un algoritmo recursivo capaz de obtener la *ruta más corta* posible, entre el punto de partida y la meta. La siguiente figura, busca ejemplificar el comportamiento del solucionador, mostrando todas las *casillas* que visita el personaje para llegar a la meta. Idealmente, el programa deberá, limpiar el escenario antes de cada movimiento, es decir que solo deberá aparecer el personaje en una *casilla* a la vez.



La interacción de la persona usuaria se limita a elegir el punto de partida, el programa debe valorar además si es posible llegar del punto de partida a la meta y en caso de no poderse, indicarlo a la usuaria.

4. Especificación.

Cada grupo deberá presentar los siguientes entregables, según las fechas establecidas en la siguiente sección.

1. **Repositorio remoto:** Se deberá configurar un repositorio grupal en GitHub, de modo que todos los integrantes del grupo puedan clonar el proyecto y hacer sus aportes desde su máquina local. Se deberá incluir además a la profesora bajo el nombre de usuaria `tec-ramijan`. Dicho repositorio contendrá:
    - El código fuente, archivos escritos en el lenguaje de programación Racket con extensión .rkt.
    - Una carpeta `img` que contiene las imágenes necesarias para ejecutar el programa.
    - Un archivo `README.md` en el que se incluya una breve descripción del proyecto, pasos necesarios para la ejecución del programa desde el entorno de desarrollo Dr.Racket y los enlaces a los videos explicativos de todos los integrantes del grupo.

    Debido a que el repositorio muestra las contribuciones de cada estudiante, así como la hora y fecha de los cambios realizados, es obligatorio actualizar de forma periódica y no solamente al finalizar el proyecto.

2. **Video explicativo:** Cada miembro del grupo por separado deberá realizar un video, capturando la pantalla de su computadora. En el video debe explicar el diseño de la solución, debe mostrar y explicar sus aportes de código al proyecto y además hacer una demostración de una ejecución exitosa del código. El video deberá ser de entre 10 y 15 minutos. Cada estudiante deberá realizar su propio video, publicarlo a través de YouTube e incluir la URL del video en el archivo `README.md`
3. **Documentación:** Cada grupo de trabajo presentará un documento que deberá completarse según el calendario presente en la siguiente sección.
    a. Portada: Nombre de la institución, la sede, nombre del curso y la profesora, semestre actual, integrantes del grupo (con sus respectivos

nombres y carnets).

b. Diseño de la solución: En esta sección se deben describir en prosa las partes indicadas a continuación, puede incluirse diagramas de flujo y/o representaciones gráficas que ayuden a explicar la solución.
    i. Datos: Definición de variables, constantes y estructuras de datos necesarias para la implementación de la solución.
    ii. Descripción de los algoritmos seleccionados para implementar el proyecto, deben explicarse tanto los algoritmos propios del grupo de trabajo, como aquellos
c. Lecciones aprendidas: En esta sección se presentan las lecciones organizacionales, técnicas y teóricas que se hayan aprendido durante la elaboración del proyecto.

5. Calendario.

Entregable	**Fecha límite de Entrega**
Acceso al repositorio remoto para les estudiantes que participan y la profesora.	Miércoles 14 de abril (7 pm)
Diseño de la Solución (documentado en el repositorio)	Lunes 19 de abril (7 pm)
Código completo en el repositorio	Lunes 26 de abril (7 pm)
Video Explicativo	Martes 27 de abril (1 pm)

6. Rúbrica.

Criterio	Indicadores de rendimiento		
	Muy bien	**Regular**	**Deficiente**
Repositorio	Actualizaciones periódicas del repositorio (5). Colaboración equitativa, todos los miembros del grupo realizan aportes equivalentes. (10). 15		
Documentación	Las explicaciones de la solución son claras y fáciles de entender (5) la documentación y la solución del código se basan una en la otra (5). 10		
Ruta más corta	El programa encuentra la ruta más corta, en todas las ocasiones. El algoritmo implementado es recursivo. 40	El programa encuentra la ruta más corta, en todas las ocasiones. El algoritmo implementado es iterativo. 30	El programa no es capaz de encontrar la ruta más corta, en todas las ocasiones. 20
Representación Visual	Se logra representar el laberinto y su solución de forma automática, clara y amigable. 15	Hay una representación visual con errores menores. 10	Se logra representar el solucionador desde la consola interactiva y sin utilizar imágenes. 5
Programación Funcional	La solución es planteada utilizando programación funcional. Se evita definir variables dentro de las funciones. La recursividad es predominante frente a la iteración. 10	La solución es planteada utilizando principalmente programación funcional. El uso de recursividad es considerablemente mayor a la iteración, pero hay código iterativo. 6	La solución es mayoritariamente iterativa. Se utilizan frecuentemente variables dentro de las funciones y hay muchas funciones iterativas. 2
Video Explicativo (calificación individual)	El video cumple con todos los lineamientos solicitados en este documento. 10	El video no cumple con algunos de los lineamientos solicitados en este documento. 6	El video es poco informativo o evidencia falta de claridad por parte del miembro. 4
Puntaje total	**100 puntos**	**Valor porcentual**	**25%**