REPÚBLICA BOLIVARIANA DE VENEZUELA
UNIVERSIDAD CENTRAL DE VENEZUELA
ESCUELA DE COMPUTACIÓN

INFORME DEL PROYECTO DEL LABERINTO

Autores: Daniel Moros C.I.30.035.269
Jean Piñango C.I. 27.580.472

Caracas, enero del 2025.

Introducción

El código presentado es un programa en C++ que simula un laberinto en el cual un jugador debe moverse para alcanzar una salida, evitando trampas y recolectando tesoros. El programa utiliza estructuras básicas de programación como ciclos, condicionales y funciones para gestionar el estado del juego y las interacciones del jugador con el entorno del laberinto.

Descripción General

El programa se divide en varias secciones clave:

1. Declaración de Variables Globales: Se declaran variables globales para almacenar el estado del juego, como la posición del jugador, la cantidad de vidas, y los contadores de elementos del laberinto.
2. Funciones de Utilidad: Se definen funciones para validar dimensiones, contar elementos, y gestionar movimientos del jugador.
3. Función Principal: La función main gestiona la entrada de datos, la configuración inicial del laberinto y el procesamiento de los movimientos del jugador.

Declaración de Variables Globales

Las variables globales incluyen:

- entrada, tipo: Para la entrada de datos y el tipo de elemento.
- elementos: Para almacenar la lista de elementos en el laberinto.
- Contadores como TotalPortales, TotalEspacios, TotalTrampas, TotalMuros, TotalTesoros, tesoros_encontrados, trampas_activadas.
- Coordenadas de la salida del laberinto SalidaX, SalidaY.
- Coordenadas del jugador jugadorX, jugadorY.

Estas variables permiten mantener el estado del juego y realizar operaciones necesarias durante la ejecución del programa.

Funciones de Utilidad

El programa define varias funciones para modularizar y organizar el código:

1. validarDimensiones(int N, int M):
   - Valida que las dimensiones del laberinto estén dentro de un rango permitido (0 a 9).
   - Retorna 1 si las dimensiones son inválidas, de lo contrario retorna 0.
2. contarElementos(char T):
   - Incrementa los contadores de elementos del laberinto según el tipo de elemento (P para portales, X para trampas, . para espacios vacíos, # para muros, T para tesoros).
   - Retorna 1 si el elemento es válido, de lo contrario retorna 0.

3. contarMovimiento(char movimiento):
    - Actualiza las coordenadas del destino del jugador según el movimiento (w, a, s, d).
    - Retorna 1 si el movimiento es válido, de lo contrario retorna 0.

4. Funciones de Estado del Juego:
    - tesorosEncontrados(): Imprime la cantidad de tesoros encontrados.
    - trampasActivadas(): Imprime la cantidad de trampas activadas.
    - vidaRestante(int L): Imprime la vida restante del jugador.
    - estadoFinal(): Imprime el estado final del juego (MUERTO, LOGRADO, SORPRENDENTE, ATRAPADO).
5. validarMovimientos():
    - Valida y procesa los movimientos del jugador.
    - Actualiza la posición del jugador y la vida según los elementos encontrados en el destino

Función Principal

La función main es el punto de entrada del programa y realiza las siguientes tareas:

1. Entrada de Datos Inicial:
    - Lee la cantidad de vidas iniciales del jugador (L).
    - Lee las dimensiones del laberinto (N, M).
    - Valida las dimensiones del laberinto.
2. Configuración del Laberinto:
    - Lee la cantidad de elementos en el laberinto.
    - Para cada elemento, lee su tipo y coordenadas, y actualiza los contadores y la posición inicial del jugador si es necesario.
3. Procesamiento de Movimientos:
    - Lee la cantidad de movimientos a ingresar.
    - Para cada movimiento, actualiza la posición del jugador y verifica si cae en una trampa o encuentra un tesoro.
    - Imprime el estado final del juego.

Conclusión

El código del proyecto de laberinto está estructurado y utiliza funciones para modularizar las tareas principales del juego. Las variables globales permiten mantener el estado del juego y las funciones de utilidad facilitan la validación y el procesamiento de los elementos y movimientos del jugador. El uso de ciclos y condicionales asegura que el programa pueda manejar diferentes escenarios y estados del juego de manera eficiente.