**UNIVERSIDAD CENTROAMERICANA**
**JOSÉ SIMEÓN CAÑAS**

**FUNDAMENTOS DE PROGRAMACIÓN**
**PROYECTO FINAL**

# “DIDE’S PALACE”

## Propuesta de juego


**Integrantes**:

Castillo Oliva, Susan Abigail, 00221125

Espinal Echegoyen, Daniel Alexander, 00100725

Guadrón Vásquez, Francisco Javier, 00077225

Guerra Sagastizado, Eduardo Josué 00043625

**Docente:**

Ing. Guillermo Cortés


**Sección:**

01

**Fecha de entrega:**

Martes 03 de junio de 2025

# DICE'S PALACE

## Descripción del juego

Dice's palace es un juego de laberinto, donde el jugador se verá atrapado en un castillo y su objetivo será buscar el camino para poder salir, para ello entrará a cuatro salas diferentes, la cuales serán laberintos y para pasar tendrá que encontrar el camino correcto que lo llevará al jefe y derrotarlo.

## Temática y ambientación

La historia se centra en la travesía de un ladrón moderno que roba un casino, sin embargo, al lograr su objetivo es teletransportado a un castillo, remontándose a la época medieval, sin pistas de donde esta y como llego ahí, en lo profundo del castillo él se dispone a encontrar la salida, cueste lo que cueste donde se encontrará con un sin fin de decisiones y posibles caminos que deberá tomar para poder salir con vida del castillo y regresar a su mundo actual.

## Mecánica principal

**1)** Elementos gráficos que se verán en consola:

   a. **Laberinto:** El Jugador verá su posición y podrá moverse entre los diferentes caminos del laberinto, además podrá ver la posición del jefe en el laberinto, que será donde tendrá que llegar.
   b. **Barra de vida del personaje**
   c. **Barra de vida del jefe:** Se mostrará cuando el personaje este luchando con el jefe.

**2)** El jugador empezará con estadísticas predefinidas (5 vidas), y tendrá que encontrar el camino correcto que lo llevará al jefe para derrotarlo.

**3)** **BONUS:** El jugador en algunas partes del laberinto se podrá encontrar con un camino con puertas (Simbolizados con * dentro del laberinto), será decisión del jugador que puerta querrá abrir (Puerta de la derecha o puerta de la izquierda), su decisión puede traerle:

a. Gane del nivel automáticamente (En el mejor de las suertes)
b. Aumentar vidas.
c. Reducir vidas.
d. No ocurrir nada y que siga su camino como iba.

## Niveles - ¿Como está diseñado el juego?

**1)** El juego contará con 4 niveles, cada uno tendrá un jefe distinto a los otros, entre más se avanza de nivel, más difícil resultará derrotar al jefe.

**2)** Dificultad en cada nivel:
   a. En cada nivel habrá un jefe que el personaje tendrá que derrotar
   b. Tiempo limite para pasar cada nivel (Ejem. 5 minutos *(Sujeto a cambios)*)

**¿Cómo derrotar al jefe?**

1. **NIVEL 1: Acertijos**
   Para derrotar el jefe el jugador tendrá que resolver 4 acertijos que el jefe le irá dando, entre más responda correctamente la vida del jefe se irá bajando.
2. **NIVEL 2: Busca la diferencia**
   Al jugador se le mostrarán dos strings casi iguales y tendrá que encontrar el carácter distinto en 30 segundos para lograr derrotar al jefe.
3. **NIVEL 3: Ordenar números**
   El jefe le presentará una serie de 4 números que estarán desordenados, el jugador tendrá que ordenar estos números en 15 segundos.
4. **NIVEL 4: Mini juego de habilidad**
   El jefe le dirá al jugador que apriete una serie de teclas rápidamente.
   El jugador tendrá que ser hábil y teclear correctamente las teclas que se le presenten para poder derrotar al jefe.

## Aplicación de temas vistos en clase

1) **cout y cin:** Para ingresar o mostrar los diferentes datos al jugador.

2) **Variables y tipos de datos:** Para almacenar información ingresada por el usuario, tales como: Nombre del jugador, ingreso de respuestas cuando este derrotando al jefe (Datos numéricos, cadenas de texto), etc.

3) **rand () y srand ():** Genera los efectos aleatorios y srand () inicializa la partida para que la posición del jefe sea distinta cada vez que se ejecute el juego y se ocupará al momento de realizar los minijuegos cuando este derrotando al jefe.

4) **if, else, switch, while, for:** Se utilizarán para repetir bloques de código de forma controlada, especialmente al recorrer datos almacenados en archivos, como listas de acertijos, personajes, diálogos o niveles.

5) **Arrays:** Se utilizarán para almacenar y manejar colecciones de datos relacionados, como conjuntos de acertijos, nombres de personajes o secuencias de diálogo, además para dibujar el laberinto se hará uso de arreglos bidimensionales (Matrices).

    a. Para el movimiento del personaje dentro del laberinto se hará uso de la función _getch(); que capturará la tecla ingresada por el jugador, estas teclas se evaluarán, ya que el jugador solo se moverá con la teclas S, W, A, D, para hacer posible el movimiento se hará uso de las coordenadas almacenadas en la matriz y por cada movimiento del jugador se redibujará el laberinto, actualizando la nueva posición del jugador.

6) **Funciones:** Se utilizarán para organizar y reutilizar fragmentos de código que se repiten o que realizan tareas específicas dentro del juego, como ejemplo, la función que dibujará el mapa, función que moverá al personaje, funciones para los diferentes minijuegos, cargar datos desde archivos, etc.

7) **Procesamiento de archivos:** Se utilizará para gestionar información importante del juego, como diálogos, personajes, niveles y eventos. También permitirá almacenar y reutilizar datos que se repiten, como funciones para mostrar vidas obtenidas, o el estado actual del juego.

8) **Manejo de errores:** Se utilizará para asegurar que el juego no se bloquee o se comporte de forma inesperada cuando el jugador ingrese datos incorrectos o se produzca un fallo al leer archivos o realizar acciones aleatorias.

9) **Estructura de datos:** Se utilizará para almacenar información relacionada con el jugador, estructurar los niveles, acceder a estadísticas actuales, etc. para así mantener el código más organizado, evitando tener variables sueltas.