Actividad 2

Instrucciones

1. Lea y comprenda cuidadosamente lo que se le solicite.
2. La Actividad puede ser desarrollada en grupos de 2 o 3 humanoides.
3. La Actividad debe ser desarrollada con el lenguaje de C++ y bajo el Paradigma Orientado a Objetos.
4. Aplicar las Buenas Prácticas para resolver el ejercicio.
5. Procure que los métodos desarrollados sean altamente cohesivos y con bajo acoplamiento.
6. Si los métodos desarrollados no están aplicando el Principio de Responsabilidad Única, considerar en refactorizar el código.
7. Crear un repositorio en GIT y programar con su compañero el código solicitado.
8. Al finalizar esta actividad, agregar el modelo UML y el código final al repositorio, hacer un push a la rama main. Proceda a compartirlo con el profesor. Recuerde como buena práctica, realizar un commit por cada funcionalidad completada.
9. Finalmente, se discutirán de forma rápida en una mesa redonda los principales hallazgos y pensamientos finales sobre el trabajo.

Descripción

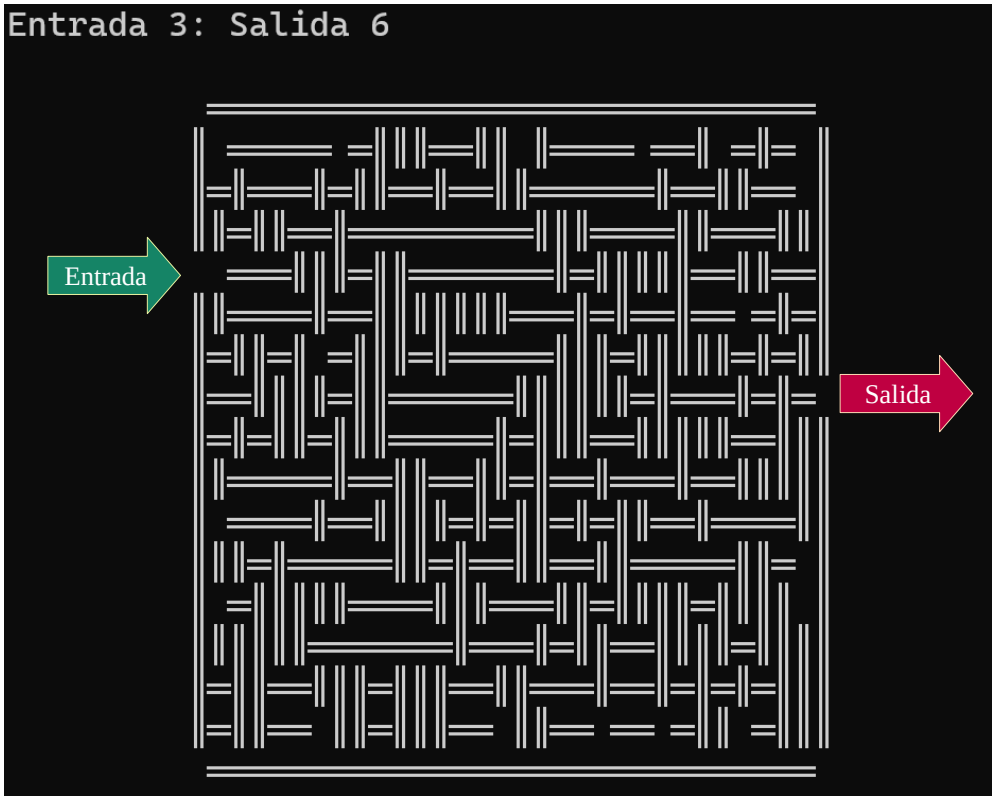
Partiendo del programa suministrado, disponible en el Aula Virtual, en donde se tiene la estructura necesaria para crear un programa generador de laberintos.

Realice lo siguiente:

1. Analizar el código suministrado, comprender la estructura y generar el UML de la solución suministrada.
2. Seguidamente, determinar los elementos (clases y métodos) necesarios que le permitan a un usuario poder realizar las siguientes funcionalidades:
   1. Se deberá crear un método que permita generar aleatoriamente un laberinto. Realizar el proceso de implementación por medio del siguiente algoritmo iterativo:

```
1. Choose the initial cell, mark it as visited and push it to the stack
2. While the stack is not empty
   1. Pop a cell from the stack and make it a current cell
   2. If the current cell has any neighbours which have not been visited
      1. Push the current cell to the stack
      2. Choose one of the unvisited neighbours
      3. Remove the wall between the current cell and the chosen cell
      4. Mark the chosen cell as visited and push it to the stack
```

   2. Crear un método que permita pintar el laberinto similar a como se observa en la siguiente figura. Tener en cuenta que las paredes del laberinto son los caracteres ASCII 186 y 205.



   3. Crear un método que resuelva el laberinto. Realizar el proceso de implementación por medio de un algoritmo recursivo. El resultado esperado debería ser como: Inicio -> Norte -> Este -> … -> Fin.
   4. Crear una variación del método anterior que permita observar el recorrido desde la entrada hasta la salida por medio de un rastro (utilizar el carácter ASCII 42).
3. Funcionalidad adicional:
   1. Crear un método que genere el laberinto, pero por medio de la implementación de un algoritmo recursivo.

Nota:
Recordar la funcionalidad para posicionar el cursor en cualquier parte de la pantalla.

```
 void goToXY(int coordX, int coordY) {
     HANDLE hcon;
     hcon = GetStdHandle(STD_OUTPUT_HANDLE);
     COORD dwPos;
     dwPos.X = coordX;
     dwPos.Y = coordY;
     SetConsoleCursorPosition(hcon, dwPos);
 }
```

Bibliotecas necesarias.

```
#include <random>
#include <stack>
using namespace std;
```

Funciones de la biblioteca stack.

```
std::stack<int> indexRow;
indexRow.push(value);
indexRow.top();
indexRow.pop();
```