EJERCICIO 10

===main.cpp===

```
#include <iostream>
#include <vector>

using namespace std;

class Laberinto
{
private:
  vector<vector<int>> matriz;
  vector<pair<int, int>> camino;

public:
  Laberinto(const vector<vector<int>> &lab) : matriz(lab) {}

  bool resolver(int x, int y)
  {
    // Verificar límites
    if (x < 0 || y < 0 || x >= matriz.size() || y >= matriz[0].size())
      return false;

    // Verificar si es pared o ya visitado
    if (matriz[x][y] == 0 || matriz[x][y] == -1)
      return false;

    // Verificar si es salida
    if (matriz[x][y] == 2)
    {
      camino.push_back({x, y}); //CASO BASE: Si la celda actual es la salida (2),
se ha encontrado el camino
      return true;
    }

    // Marcar como visitado
    matriz[x][y] = -1;

    // Explorar en las 4 direcciones
    if (resolver(x + 1, y) || resolver(x - 1, y) || resolver(x, y + 1) ||
resolver(x, y - 1)) //CASO RECURSIVO
    {
      camino.push_back({x, y});
      return true;
    }

    return false;
  }

  void mostrarCamino()
```

```
  {
    if (camino.empty())
    {
      cout << "No hay salida desde la posición inicial." << endl;
    }
    else
    {
      cout << "Camino hacia la salida:" << endl;
      for (auto it = camino.rbegin(); it != camino.rend(); ++it)
      {
        cout << "(" << it->first << ", " << it->second << ")" << endl;
      }
    }
  }
};

int main()
{
  vector<vector<int>> laberinto = {
    {1, 0, 1, 1, 1},
    {1, 0, 0, 0, 1},
    {1, 1, 1, 0, 1},
    {0, 0, 1, 0, 1},
    {1, 1, 1, 1, 2}};

  Laberinto lab(laberinto);
  int inicioX = 0, inicioY = 0;

  if (lab.resolver(inicioX, inicioY))
  {
    lab.mostrarCamino();
  }
  else
  {
    cout << "No se encontró un camino hacia la salida." << endl;
  }

  return 0;
}

 /* ¿CUÁNDO ESTO SERÍA INFINITO?
    La recursión se volvería infinita si no se marca la celda como visitada,
    ya que el algoritmo podría volver a pasar por la misma posición una y otra vez.
    También ocurriría si no se valida correctamente que las coordenadas estén
dentro de los límites.
  ¿POR QUÉ ES UNA SOLUCIÓN NATURAL?
    Este problema se presta naturalmente a una solución recursiva porque se puede
dividir en subproblemas:
    "¿Puedo llegar a la salida desde alguna de las celdas vecinas?"
    La recursión permite explorar todas las rutas posibles de forma ordenada y
retroceder si una ruta no lleva a la solución.
```

```
    Sin embargo, hay que considerar que otros procesos pueden ser un poco más
eficientes e intuitivos.
  */
```