```
entero encontrar_entrada(caracter A[][], entero filas, columnas)
{
   entero pos[2], i, j;

   for i = 1 to filas
   {
      for j = 1 to columnas
      {
         if A[i][j] == 'e' then
         {
            pos[0] = i;
            pos[1] = j;
         }
      }
   }

   return pos;
}

caracter busqueda_salida(caracter A[][], entero x, entero y)
{
   entero cord_x;
   entero cord_y;
   caracter salida;

   if A[x][y + 1] == 'o' then
   {
      cord_x = x;
      cord_y = y + 1;
```

```
    matriz[x][y + 1] = '*';
    salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x + 1][y] == 'o' then
{
    cord_x = x + 1;
    cord_y = y;
    matriz[x + 1][y] = '*';
    salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x][y - 1] == 'o' then
{
    cord_x = x;
    cord_y = y - 1;
    matriz[x][y - 1] = '*';
    salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x - 1][y] == 'o' then
{
    cord_x = x - 1;
    cord_y = y;
    matriz[x - 1][y] = '*';
    salida = busqueda_salida(matriz, cord_x, cord_y);
}
/* ------------------------------------------------- */
else if A[x][y + 1] == 'e' then
{
    cord_x = x;
    cord_y = y + 1;
```

```
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x + 1][y] == 'e' then
{
   cord_x = x + 1;
   cord_y = y;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x][y - 1] == 'e' then
{
   cord_x = x;
   cord_y = y - 1;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x - 1][y] == 'e' then
{
   cord_x = x - 1;
   cord_y = y;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
/* ------------------------------------------------- */
else if A[x][y + 1] == 's' then
{
   return s;
}
else if A[x + 1][y] == 's' then
{
   return s;
}
```

```
else if A[x][y - 1] == 's' then
{
   return s;
}
else if A[x - 1][y] == 's' then
{
   return s;
}
/* ---------------------------------------------------- */
else if A[x][y + 1] == '*' then
{
   cord_x = x;
   cord_y = y + 1;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x + 1][y] == '*' then
{
   cord_x = x + 1;
   cord_y = y;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x][y - 1] == '*' then
{
   cord_x = x;
   cord_y = y - 1;
   salida = busqueda_salida(matriz, cord_x, cord_y);
}
else if A[x - 1][y] == '*' then
{
```

```
        cord_x = x - 1;
        cord_y = y;
        salida = busqueda_salida(matriz, cord_x, cord_y);
    }
    /* ------------------------------------------------- */
    else
    {
        return n;
    }
}

entero main(A[][], entero filas, entero columnas)
{
    entero pos[2], i, j;
    caracter salida;

    pos = encontrar_entrada(A, filas, columnas);
    salida = busqueda_salida(matriz, pos[0], pos[1]);

    if salida == 's' then
    {
        print("Hay salida y el recorrido es:\n");
        for i = 1 to filas
        {
            for j = 1 to columnas
            {
                print("%c", matriz[i][j]);
            }
            print("\n");
```

```
        }
    }
    else
    {
        print("Laberinto sin solucion");
    }

    return 0;
}
```

/* -- */

ORDEN DE EFICIENCIA:

el orden de eficiencia en la lectura de la matriz es de n^2 ya que si se asume n = m tardara n * n veces en cargar la matriz

por otra parte el recorrido del laberinto es de orden n ya que recorrera el laberinto tantas veces como caminos tenga