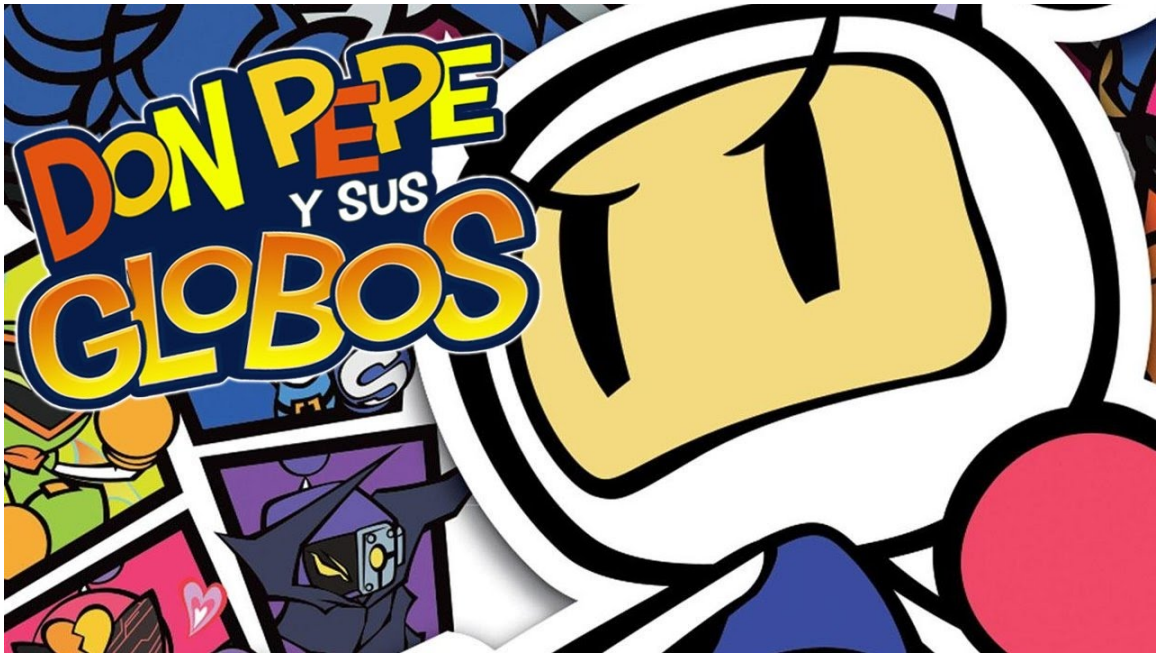
Proyecto #1 - Don Pepe y sus Globos

30 de mayo de 2019



1. La historia

Don Pepe, después de haber sufrido una depresión crónica, producto del mal gusto del equipo de traductores españoles de su gran juego: *Bomberman*, decide tomar venganza por mano propia, y recuperar el honor perdido de sus desarrolladores nipones. *¿Cómo lo hará, te preguntarás?* ¡Por medio de **explosivos**, obviamente!

Hay solo un *problema*. Ellos tienen *Minvos* guardianes, ¡y son muchos!

Don Pepe, no quiere gastar mucha energía y bombas, por ello decidió pedirle ayuda a los estudiantes de **Algoritmos y Estructuras de Datos**, para que buscaran por él la forma más eficiente de abrirse paso para iniciar su masacre.

Y, por supuesto, quiere que lo hagan gratis.

2. El enunciado

En un mapa de NxM caracteres, siendo:

@	Jugador
#	Pared destruible
=	Pared indestructible
.	Piso
x	Enemigo
Q	Bomba

El jugador, mediante una serie de eventos, deberá eliminar a todos los enemigos del mapa en la solución óptima menor a K pasos (dado por entrada).

2.1. Los eventos

Los eventos del jugador consisten, sin un orden particular, en:

- Esperar
- Moverse hacia arriba
- Moverse hacia la derecha
- Moverse hacia abajo
- Moverse hacia la izquierda
- Colocar bomba

2.1.1. El movimiento

El movimiento del jugador consiste en un solo paso en el sentido que dicte el evento, no existen pasos en diagonal ni saltos. El jugador *solo* podrá moverse a espacios que esten vacíos (.). No puede trepar sobre paredes ni atravesarlas. Tampoco puede moverse a una casilla con una bomba una vez haya salido de esta.

2.1.2. La bomba

Al ejecutarse el evento *colocar bomba*, esta se colocará en la casilla donde se encuentre el jugador. Dicha bomba explotará luego de 3 pasos (3 ambientes recursivos), afectando la fila y columna desde la posición de la bomba hasta que alcanze una pared (sea destruible o indestructible) o el límite del mapa.

```
===============
=......=......=
=...@..*.......
=.#=***Q*******
=......*.......
=......#......=
===============

===============
=......=......=
=...@..........
=.#=...........
=..............
=.............=
===============
```

Los enemigos *no* frenan una explosión. Si son alcanzados por ellas, estos son eliminados.

```
===============
=......*......=
=...@..*......=
=..=***Q***xxx=
=......=......=
=.............=
===============
```

```
===============
=.............=
=...@.........=
=..=..........=
=......=......=
=.............=
===============
```

Las paredes destruibles que sean alcanzadas por la explosión seran destruidas y, como consecuencia, reemplazadas por piso caminable (.). La explosión no afecta paredes destruibles o enemigos detrás de la primera pared destruida.

```
===============
=......#......=
=...@..*......=
=.#***Q***=x.=
=......#......=
=......x......=
===============
```

```
===============
=.............=
=...@.........=
=.#.........=x.=
=.............=
=......x......=
===============
```

La bomba no debe alcanzar al jugador en una solución válida, es decir éste debe salir ileso de todas las explosiones, incluyendo la última.

2.2. La entrada

Se leerá el mapa desde un archivo llamado “Entrada.txt”, con el límite de pasos K, las dimensiones N y M del mapa, y el mapa en cuestión a partir de la próxima línea. Ejemplo:

```
32 10 11
==========
=#=@===x==
=x#.....#=
=.=#==.#.=
=..x====.=
===.#===.=
=...==.#x=
=.==x===.=
=..#....x=
==#=x.====
==========
```

2.3. La salida

De conseguir eliminar a todos los enemigos, la salida del programa deberá ser:

`Logra eliminar a todos los enemigos en` *`cantidad`* `pasos.`
`SERIE_DE_PASOS`

Donde *SERIE_DE_PASOS* es la concatenación de las acciones, con la representación indicada en la próxima subsección.

De no conseguir eliminar a todos los enemigos, será:

`No logra eliminar a todos los enemigos.`

2.3.1. Representación de las acciones en la salida

Las acciones tomadas en el mejor camino (el más corto) encontrado serán expresadas en la salida como una cadena de caracteres, cada uno representando una acción:

_	Esperar
N	Moverse hacia arriba
E	Moverse hacia la derecha
S	Moverse hacia abajo
O	Moverse hacia la izquierda
Q	Soltar la bomba

Puede ver ejemplos del formato en la salida de los casos de prueba dados.

2.3.2. Casos especiales en la salida

Si el mapa de entrada no posee enemigos, no indica la posición del jugador (o indica a más de uno) o posee un caracter no válido, el programa deberá terminar, después de imprimir el mensaje de error correspondiente de la próxima lista:

- `No existen enemigos.`
- `No existe jugador.`
- `Hay más de un jugador.`
- `Caracter inválido:` *`caracter_en_cuestion`*

Puede asumir que todos los mapas dados tienen dimensiónes constantes (rectangular, sin caracteres de más o de menos)

3. Casos de Prueba

Entrada	Salida
25 10 4 ========== =x=.=@==== =.#x....x= ==========	Logra eliminar a todos los enemigos en 15 pasos. SEQON_SOOOOQEEN
36 10 8 ========== =x.@.===x= ====.....= =x==#===.= =x#.....x= =..===.=== ==#....#x= ==========	Logra eliminar a todos los enemigos en 34 pasos. EQSE_OSSEQESSQNNEEQOOSSOOOQONOQNE_

4. Entrega del proyecto

Deben enviar el correo con la solución al correo de sus respectivos preparadores:

Alexanyer Naranjo alexanyersb1@gmail.com
Vittorio Adesso vittorio.adesso@gmail.com

El asunto del correo a enviar con la solución del proyecto debe tener la siguiente estructura:

[AyED] Proyecto #1 Sección1_Sección2 Nombre1 Nombre2.

Por ejemplo: [AyED] Proyecto #1 C1_C2 11111111_22222222

Se debe adjuntar un archivo comprimido (.zip, .rar, entre otros) con el **nombre del asunto**.

El archivo comprimido debe contener el archivo de código fuente en el lenguaje de programación C++, junto a un **archivo PDF** que describa el análisis que los llevó a la solución del problema.

Si no se entrega el proyecto con este formato, acarreará una penalización en la nota.

La fecha de entrega queda pautada a más tardar para el día **21 de Junio de 2019** hasta las 11:59 PM (GMT-4).

5. Nota adicional

Aparte de este enunciado obtendrán otro programa: `PepeSimulator`

Este debe ser ejecutado con la cadena de caracteres del recorrido (obtenido por su solución propuesta) como primer argumento (usando la opción -s. Pueden ver otras opciones usando –help), y tiene la funcionalidad de simular gráficamente el camino que sigue el personaje. Ejemplo: `./PepeSimulator -s NOQOS_E`

En el caso de que el camino no sea posible (erróneo), el programa lo indicará. Es importante que `PepeSimulator` se encuentre en la misma carpeta de "Entrada.txt" puesto a que obtendrá el mapa de él.

`PepeSimulator` es una forma de validar los resultados que se obtienen con su solución propuesta (sin embargo no dirá si es la mejor solucion, solo si el camino dado es posible o no).