Primer Proyecto de Programación

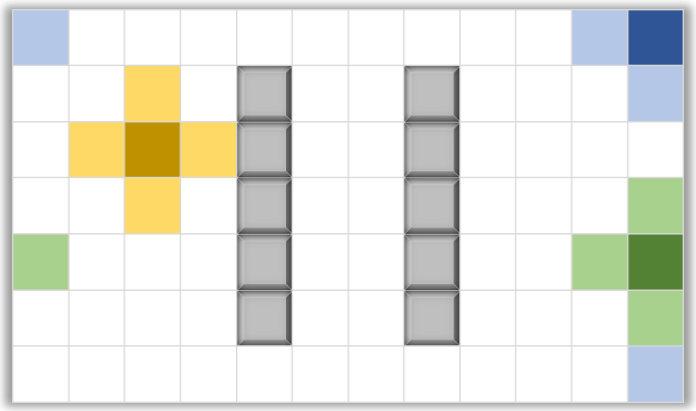
Colectivo de Programación

Grupo WebOO

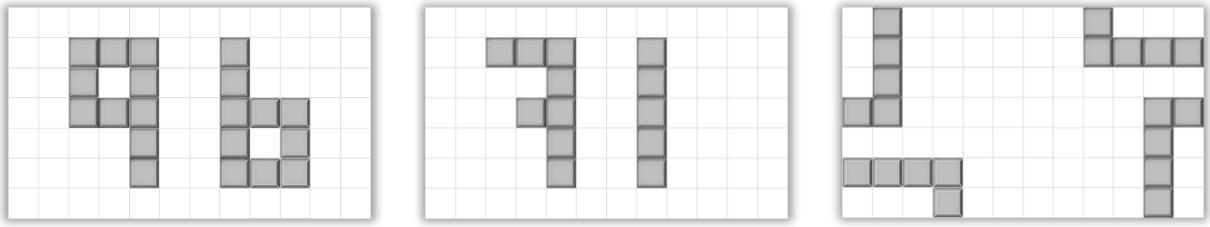
Curso 2019-2020

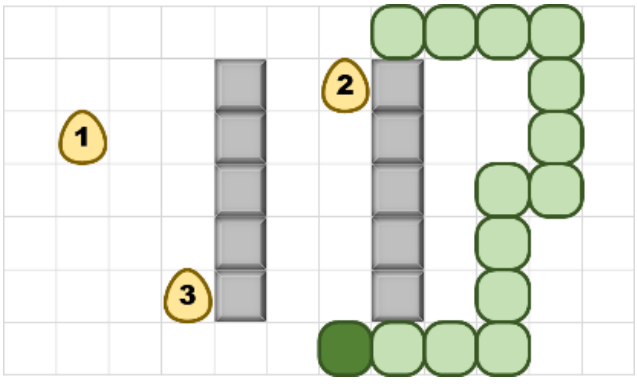
EL JUEGO

Sobre un mundo bidimensional de $N \times K$ casillas se han dispuesto un conjunto de obstáculos. Este mundo es circular por lo que la casilla adyacente a la derecha de una posición en el borde extremo derecho es la correspondiente en la misma fila pero en el extremo izquierdo, y de esta misma forma con cualquier borde del terreno. En la imagen siguiente se muestra un mundo válido y algunas casillas con sus adyacentes resaltadas.



Todas las casillas "libres" del mundo deben ser alcanzables unas desde otras. La siguiente imagen muestra un mundo inválido y otros dos válidos. Note que el mundo de la derecha puede llegar a las casillas de los "cuartos" por la propiedad circular del mundo.





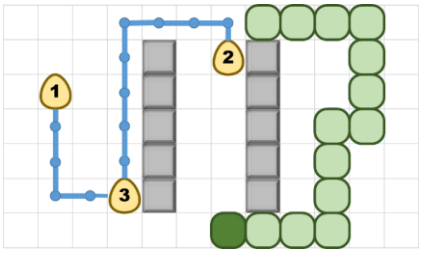
En este mundo habita una serpiente glotona que le gustan los huevos. La serpiente ocupa una secuencia de casillas adyacentes donde ninguna casilla se repite. La primera casilla representa la "cabeza" de la serpiente y la última la cola.

DESCRIPCIÓN

La serpiente comienza con una longitud de dos casillas. La serpiente se mueve avanzando la cabeza a alguna casilla adyacente (que no sea la casilla siguiente de su cuerpo) y moviendo todo su cuerpo hacia la casilla adyacente correspondiente de forma que se mantenga la integridad física. En el terreno hay depositados en un principio Q huevos. La serpiente se "come" un huevo avanzando hacia la casilla donde se encuentra. La serpiente siempre está moviéndose en determinada dirección a no ser que se le comande cambiar hacia otra dirección. No tendrá efecto comandarle que cambie hacia la dirección opuesta a la que tiene.

Reglas

- Los huevos están enumerados con valores de 1 a Q.
- Cada vez que la serpiente se come un huevo, esta crece dicho número de celdas. El valor del huevo más cercano se multiplica por 100 para incrementar el puntaje total del juego. Note que el último huevo solo aporta al crecimiento de la serpiente.
- El concepto de cercanía entre la serpiente y un huevo está dado por la menor secuencia de casillas no "obstáculos" que separan la cabeza de la serpiente del huevo. Si dos huevos estuvieran a la misma distancia, el valor del mayor se consideraría para el puntaje. Note que el cuerpo de la serpiente no se tiene en cuenta como "obstáculo" en el camino.



En esta imagen, si la serpiente se comiera el huevo "3" obtendría solo 100 puntos, puesto que el huevo "2" está más lejos.

- Cada vez que se eliminen todos los huevos del terreno, otros nuevos Q huevos aparecen en casillas no ocupadas del terreno.
- La serpiente no puede "comerse" a sí misma ni comerse un obstáculo. Si esto sucede termina el nivel con el puntaje que se haya obtenido.
- Si no se pueden ubicar más huevos el nivel termina.

APLICACIÓN

Usted debe desarrollar una aplicación de *Windows Forms*, *WPF* o *Unity* que permita la creación de niveles, describiendo el mundo (sus dimensiones), obstáculos, el parámetro Q y la velocidad (casillas por segundo).

Esta aplicación debe ser capaz de "guardar" el nivel creado en un fichero y leer un nivel para modificarlo. En cualquier momento se puede ir a "modo juego" para probar el nivel.

Con respecto a la interacción con el usuario se le aconseja que se utilice las teclas de flechas para indicar los cambios de dirección de la serpiente.

SOBRE LEGIBILIDAD DEL CÓDIGO

Identificadores

Todos los identificadores (nombres de variables, métodos, clases, etc) deben ser establecidos cuidadosamente, con el objetivo de que una persona distinta del programador original pueda comprender fácilmente para qué se emplea cada uno.

Los nombres de las variables deben indicar con la mayor exactitud posible la información que se almacena en ellas. Por ejemplo, si en una variable se almacena "la cantidad de obstáculos que se ha encontrado hasta el momento", su nombre debería ser `cantidadObstaculosEncontrados` o `cantObstaculos` si el primero le parece demasiado largo, pero nunca ~~`c0b`~~, ~~`aux`~~, ~~`temp`~~, ~~`miVariable`~~, ~~`juan`~~, ~~`contador`~~, ~~`contando`~~ o ~~`paraQueNoChoque`~~. Note que el último identificador incorrecto es perfectamente legible, pero no indica "qué información se guarda en la variable", sino quizás "para qué utilizo la información que almaceno ahí", lo cual tampoco es lo deseado.

- Entre un nombre de variable un poco largo y descriptivo y uno que no pueda ser fácilmente comprensible por cualquiera, es preferible el largo.
- Como regla general, los nombres de variables **no** deben ser palabras o frases que indiquen acciones, como ~~eliminando~~, ~~saltar~~ o ~~parar.~~

Existen algunos (muy pocos casos) en que se pueden emplear identificadores no tan descriptivos para las variables. Se trata generalmente de pequeños fragmentos de código muy comunes que "todo el mundo sabe para qué son". Por ejemplo:

```
int temp = a;
a = b;
b = temp;
```

"Todo el mundo" sabe que el código anterior constituye un intercambio o *swap* entre los valores de las variables a y b, así como que la variable `temp` se emplea para almacenar por un instante uno de los dos valores. En casi cualquier otro contexto, utilizar `temp` como nombre de variable resulta incorrecto, ya que solo indica que se empleará para almacenar "temporalmente" un valor y en definitiva todas las variables se utilizan para eso.

Como segundo ejemplo, si se quiere ejecutar algo diez veces, se puede hacer

```
for (int i = 0; i < 10; i++)
  ...
```

en lugar de

```
for (int iteracionActual = 0; iteracionActual < 10; iteracionActual++)
  ...
```

Para los nombres de las propiedades (*properties* en inglés) se aplica el mismo principio que para las variables, o sea, expresar “qué devuelven” o “qué representan”, solo que los identificadores deben comenzar por mayúsculas. No deben ser frases que denoten acciones, abreviaturas incomprensibles, etc.

Los nombres de los métodos deben reflejar “qué hace el método” y generalmente es una buena idea utilizar para ello un verbo en infinitivo o imperativo: Agregar, Eliminar, ConcatenarArrays, ContarPalabras, Arranca, Para, etc.

En el caso de las clases, obviamente, también se espera que sus identificadores dejen claro qué representa la clase: Serpiente, Celda, Ambiente, Programa.

Comentarios

Los comentarios también son un elemento esencial en la comprensión del código por una persona que lo necesite adaptar o arreglar y que no necesariamente fue quien lo programó o no lo hizo recientemente. Al incluir comentarios en su código, tome en cuenta que no van dirigidos solo a Ud., sino a cualquier programador. Por ejemplo, a lo mejor a Ud. le basta con el siguiente comentario para entender qué hace determinado fragmento de código o para qué se emplea una variable:

```
// lo que se me ocurrió aquel día
```

Evidentemente, a otra persona no le resultarán muy útiles esos comentarios.

Algunas recomendaciones sobre dónde incluir comentarios

- Al declarar una variable, si incluso empleando un buen nombre para ella pueden quedar dudas sobre la información que almacena o la forma en que se utiliza
- Prácticamente en la definición de todos los métodos para indicar qué hacen, las características de los parámetros que reciben y el resultado que devuelven
- En el interior de los métodos que no sean demasiado breves, para indicar qué hace cada parte del método

Es cierto que siempre resulta difícil determinar dentro del código qué es lo obvio y qué es lo que requiere ser comentado, especialmente para Ud. que probablemente no tiene mucha experiencia programando y trabajando con código hecho por otras personas. Es preferible entonces que “por si acaso” comente su código lo más posible.

Otro aspecto a tener en cuenta es que los comentarios son fragmentos de texto en lenguaje natural, en los cuales deberá expresarse lo más claramente posible, cuidando la ortografía, gramática, coherencia, y demás elementos indispensables para escribir correctamente.

Todos estos elementos son importantes para la calidad de todo el código que produzca a lo largo de su carrera, pero además **tendrán un peso considerable en la evaluación de su proyecto de programación**.