Fundamentos de la programación

Grado en Desarrollo de Videojuegos

Examen parcial Febrero 2016

Indicaciones generales:

- **Lee atentamente** el enunciado e implementa el programa tal como se pide, con los métodos y requisitos que se especifican.
- El programa, además de ser correcto, debe estar bien estructurado y comentado. Se valorarán la claridad, la concisión y la eficiencia.
- Se entregará un único archivo `Program.cs` (generado con MonoDevelop) con el programa completo.

En este ejercicio vamos a implementar una **variante del juego Mastermind**[1], para jugar contra el ordenador. El juego consiste en lo siguiente: el ordenador elige una combinación secreta de `N` dígitos (`N`$\leq$ 10) entre 0 y 9 **sin** repeticiones que se almacenará en un array `combSecr` de tamaño `N`. El jugador tratará de adivinar dicha combinación mediante intentos sucesivos. En cada intento el jugador introduce otra combinación que se almacenará en otro array `combJug` (también de tamaño `N`), que el ordenador valora diciendo:

- el número de *muertos*: número de dígitos que aparecen en ambas combinaciones en las posiciones correctas;
- el número de *heridos*: número de dígitos que aparecen en ambas combinaciones, pero en posiciones incorrectas.

Por ejemplo, para `N=4` podemos tener la combinación secreta: | 1 | 6 | 4 | 0 |. Si el jugador prueba la combinación | 6 | 9 | 4 | 1 |, el ordenador informará de que hay 1 muerto (el 4, que coincide en ambos arrays en la misma posición) y dos heridos (el 1 y el 6 que aparecen en ambas combinaciones pero en posiciones incorrectas). Nótese que el 4 contabiliza como muerto, pero no como herido, i.e., los muertos no contabilizan como heridos.

El tamaño `N` de las combinaciones debe declararse como constante en el programa. A continuación se muestra una posible partida completa para `N=4`, con una combinación secreta generada por el ordenador (que no se mostraría en pantalla) y los sucesivos tanteos del jugador:

2	6	4	5	`combSecr`

3	2	1	6	muertos: 0	heridos: 2
2	1	3	5	muertos: 2	heridos: 0
2	3	6	5	muertos: 2	heridos: 1
2	4	6	5	muertos: 2	heridos: 2
2	6	4	5	muertos: 4	heridos: 0

Se pide implementar los siguientes métodos (la puntuación se indica en cada [illegible] y se simplifican las reglas de juego.

Por ejemplo,

- si `i=0` tendremos `combSecr` = | 5 | 3 | 1 | 9 |
- si `i=3` tendremos `combSecr` = | 9 | 6 | 0 | 2 |
- si `i=6` tendremos `combSecr` = | 2 | 8 | 4 | 7 |

Recordemos: para generar números aleatorios se declara (una única vez) `Random rnd = new Random();`. Después se invoca a `rnd.Next (a,b);` para obtener un número aleatorio del intervalo $[a, b-1]$.

- [**1pt**] `bool muerto(combSecr, combJug, pos)`: devuelve `true` si los arrays `combSecr` y `combJug` contienen el **mismo dígito en la posición** `pos` (es decir, `pos` corresponde a una posición de muerto); `false` en caso contrario.

- [**2pt**] `bool herido(combSecr, combJug, pos)`: devuelve `true` si el dígito de la posición `pos` en el array `combSecr` aparece en el array `combJug` en una posición distinta de `pos`; *false* en caso contrario.

- [**1pt**] `void evalua(combSecr, combJug, m, h)`: utilizando las funciones anteriores, devolverá el **número total de muertos y heridos** en los parámetros `m` y `h` respectivamente, correspondientes a las combinaciones almacenadas en los arrays `combSecr` y `combJug`.

  Por ejemplo, si `combSecr`=| 2 | 6 | 4 | 5 | y `combJug`=| 2 | 3 | 6 | 5 |, devolvería `m=2` y `h=1`.

- [**2pt**] `void leeCombinacion(combJug)`: lee de teclado un entero (de `N` dígitos sin repetición), lo descompone dígito a dígito y almacena dichos dígitos en el array `combJug`.

  Por ejemplo, si el usuario introduce el número 3425, tendremos `combJug` = | 3 | 4 | 2 | 5 |. Recordemos que un número puede descomponerse en dígitos utilizando reiteradamente la división entera y el resto de la división entera. En concreto, tenemos $3425/10 = 342$ y $3425\,\%10 = 5$.

- [**2pt**] `Main()`: utilizando los métodos anteriores genera una combinación secreta y a continuación pide reiteradamente combinaciones al usuario hasta que el usuario acierte la combinación secreta. Para cada combinación intentada, el ordenador informará del número de muertos y heridos de la misma. Cuando el jugador acierta la combinación secreta, el juego termina y se informa al jugador del número de intentos que ha realizado.

A continuación se muestra una posible partida completa:

```
Tu combinacion (4 digitos): 1256
Muertos: 0  Heridos: 1

Tu combinacion (4 digitos): 6235
Muertos: 0  Heridos: 1

Tu combinacion (4 digitos): 2687
Muertos: 2  Heridos: 1
```