Fundamentos de la programación I

Práctica 1. Conducción de coches

Indicaciones generales:

- La línea 1 del programa (y siguientes) deben contener los nombres de los alumnos de la forma:

  `// Nombre Apellido1 Apellido2`

- **Lee atentamente** el enunciado e implementa el programa tal como se pide, con la representación y esquema propuestos, y con los requisitos que se especifican.
- El programa, además de correcto, debe estar bien estructurado y comentado. Se valorarán la claridad, la concisión y la eficiencia.
- La entrega se realizará a través del campus virtual, subiendo únicamente el archivo `Program.cs`, con el programa completo.
- El **plazo de entrega** finaliza el 25 de octubre.

En esta práctica vamos a implementar un juego elemental de conducción de coches en consola. Tenemos una pista de un ancho dado y un coche situado en la misma. Representamos el coche con `<o>`, las posiciones libres de la pista con `'·'` y los bordes con `'|'`. Por ejemplo, para un pista de ancho 9 con el coche situado en la posición central tendremos:

```
|···<o>···|
```

En cada frame del juego solo se muestra el fragmento de pista ya recorrida, hasta la posición actual (no se ve la pista por delante). La pista puede describir curvas a la izquierda o la derecha y el jugador deberá mover el coche lateralmente para evitar la colisión. Tras 4 frames de juego podríamos ver en pantalla:

```
  |···<o>···|
 |····<o>··|
 |···<o>···|
|····<o>··|
```

La primera fila corresponde al estado inicial, con el coche centrado en la pista. Tras un *frame* del juego, en la segunda fila la pista ha ido a la izquierda y el coche recto. En el tercer frame la pista ha continuado recta y el coche ha ido a la izquierda. Y en el cuarto la pista ha ido hacia la izquierda y el coche ha seguido recto. Este último frame corresponde al estado actual del juego, a partir del cual continuará la pista y podrá moverse el coche.

En cada frame el usuario puede mover el coche a la izquierda pulsando `'a'` (como ocurre en el segundo frame en el ejemplo de arriba), a la derecha pulsando `'d'`, o no moverlo, si no pulsa nada. El ancho de la pista viene dado por una constante `ANCHO` y la posición del borde izquierdo está limitada por un valor `TOPE`, también dado como constante (para evitar que el dibujo de pista se salga del ancho de la línea en la consola). La velocidad de refresco viene dada por un valor `delta` (en milisegundos) solicitado al usuario antes de empezar el juego. Así pues, el estado del juego en cada frame viene determinado por las constantes `ANCHO` y `TOPE`, la variable `delta` y las siguientes variables:

- `pistaIzq`: posición del borde izquierdo de la pista (relativa a la columna izquierda de la pantalla). Tiene que cumplirse $0 \leq$ `pistaIzq` $\leq$ `TOPE`. El borde derecho se calculará a partir de este valor y la constante `ANCHO`.
- `posCoche`: posición actual *del centro del coche* `'o'` (también relativa a la columna izquierda). Nótese que el coche ocupa tres posiciones (`<o>`) que deben estar dentro de la pista para evitar colisiones.

- *colIzda, colDcha*: booleanos que indican si hay colisión con el borde izquierdo o derecho respectivamente.

Por ejemplo, en este frame:

```
|····<o>··|
```

tendríamos `pistaIzq=7` (posición del primer `|` respecto al margen izquierdo), `ANCHO=9`, `posCoche=13` (posicion del caracter central `o` respecto al margen izquierdo de la consola), y `colIzda=false, colDcha=false`.

En cada vuelta del bucle las tareas a realizar son (**en este orden**):

- **Recoger el input de usuario**. Si hiciésemos la lectura de input utilizando el tradicional `Console.ReadLine()`, la entrada sería *bloqueante*, es decir, el programa pararía hasta recibir entrada de teclado y luego *intro*. Para que el juego fluya utilizaremos una lectura *no bloqueante*: si se pulsa una tecla se recoge el valor y continúa la ejecución; si no hay pulsación continúa la ejecución. Esto puede hacerse del siguiente modo:

```
string dir;
if (Console.KeyAvailable) {     // detección de tecla pulsada
  string dir = (Console.ReadKey(true)).KeyChar.ToString(); // lectura de tecla
  if (dir == "a") ...
  else if (dir == "d") ...
  ...
  // vaciado buffer de entrada: en un frame solo se guarda la primera tecla pulsada
  while (Console.KeyAvailable) Console.ReadKey(true);
}
```

  De este modo si hay pulsación de tecla, ejecutará el if y obtendrá el valor en la cadena `dir` y si no, no entra en el *if*.

- **Lógica del juego**. En cada frame, aleatoriamente la pista puede describir una curva a izquierda o derecha, o continuar recta (con la misma probabilidad). Para generar números aleatorios se puede inicializar el generador al principio del programa con:

```
Random rnd = new Random();
```

  Y después, cada vez que necesitemos un número aleatorio haremos:

```
int k = rnd.Next (i, j);
```

  Así, en `k` obtenemos un valor aleatorio en el intervalo `[i,j-1]` (atención a los límites!) que podemos utilizar para determinar las curvas. Deberemos controlar que la pista no se salga de la pantalla con el `TOPE` establecido.

- **Control de colisiones**. Una vez determinadas las nuevas posiciones del coche y de la pista en los pasos anteriores, hay que determinar si se ha producido colisión entre el coche y alguno de los bordes de la pista, en cuyo caso se pondrá a `true` la correspondiente variable `colDcha` o `colDcha`. Estas varialbes determinan el final del juego (finaliza el bucle principal).

- **Renderizado gráfico**. Consiste en dibujar en pantalla **una línea** correspondiente al estado actual de juego. En concreto, para cada frame se dibujarán (por este orden) los blancos previos al inicio de la pista, el borde izquierdo '`|`', los espacios '`·`' hasta el coche, el propio coche `<o>`, los espacios '`·`' hasta el borde derecho de la pista y el propio borde derecho `|`.

- **Tiempo de retardo**. Para que el usuario tenga tiempo de reacción y el juego sea *jugable*, en cada vuelta del bucle introducimos un tiempo de retardo `delta` (en milisegundos). La forma más simple de hacerlo es:

```
System.Threading.Thread.Sleep(delta);
```

Con todo lo explicado, el programa tendrá el siguiente aspecto:

```
using System;
namespace coches{
  class MainClass  {
    const int ANCHO = 9; // ancho de la pista
    const int TOPE = 40;
    Random rnd = new Random(); // generador de aleatorios para cambio de direccion

    public static void Main (string[] args) {
      // incialización del estado
      int pistaIzq = 20,                    // pos inicial de la pista
        posCoche = pistaIzq+1+ANCHO/2,      // pos inicial del coche
        delta;                              // ms de refresco por frame
      bool colIzda = false, colDcha = false; // colisiones
      // peticion de delta
      ...
      // fin inicialización

      // renderizado del estado de juego
      ...
      while (...) {  // bucle ppal
        // procesamiento del input de usuario, movimiento del coche
        ...
        // lógica del juego, siguiente trozo de pista
        ...
        // control de colisiones con bordes de la pista
        ...
        // renderizado del estado de juego
        ...
        // retardo entre frames
        System.Threading.Thread.Sleep(delta);
      } // fin bucle ppal
      // renderizado final, con dibujo de colisión
      ...
    }
}}}
```

Nótese que hay dos renderizados (uno antes del bucle principal y otro dentro del bucle), que se codificarán exactamente igual (copy-paste). El renderizado final, fuera del bucle principal, será distinto ya que debe dibujar las colisiones: en el margen de la pista donde hay colisión correspondiente, se reemplaza el símbolo `'|'` por `'*'`.

Una posible partida completa puede tener este aspecto en pantalla (el color del texto puede cambiarse con `Console.ForegroundColor = ...` y volver a dejarlo normal con `Console.ResetColor()`):

Extensiones del juego Una vez finalizada la versión básica del juego implementaremos algunas extensiones:

- Permitir al usuario terminar el juego con la tecla `'q'` sin necesidad de colisión.
- Variar dinámicamente la velocidad del juego alterando el valor de `delta` (hacerlo de manera razonable para que el juego sea *jugable*).
- (Opcional) Puede hacerse el juego más divertido permitiendo que el ancho de la pista pueda variar aleatoriamente, en particular estrechándose. Para ello en vez de una única variable `pista` necesitaremos dos variables que almacenen los márgenes izquierdo y derecho. Un estrechamiento consistiría en desplazar uno o ambos márgenes hacia el interior.