Práctica Evaluable Temas 8 y 9.

Manejo de ficheros y bibliotecas de uso frecuente

Objetivos.

- Repasar los conceptos estudiados hasta ahora.

Consideraciones iniciales.

- Deberéis realizar un único proyecto en Visual Studio o similar, con el nombre que se indica en este enunciado.
- Cada fichero fuente debe tener un comentario inicial explicando su utilidad. Se deben aclarar con comentarios también los fragmentos de código que no quede muy claro qué hacen.
- Como en otras prácticas evaluables anteriores, el proyecto debe compilar, ya que de lo contrario se evaluará con un 0.

Entrega.

Se debe entregar un archivo ZIP con el proyecto completo comprimido.

Nombre del archivo: **Grupo_Apellidos_Nombre_PracT8T9.zip**

Por ejemplo, si te llamas Andrés Pérez Gómez y eres de 1º de DAM B, el archivo debe llamarse *B_Perez_Gomez_Andres_PracT8T9.zip*. Dentro, debe contener el proyecto completo.

Desarrollo.

Ejercicio "Traductor de BASIC a C#".

Nombre del proyecto: "TraductorBASIC"

Puntuación máxima: 10 puntos

Presentación

El lenguaje BASIC es un lenguaje de programación de los años 70 y 80, de bajo nivel, que consistía en una secuencia de instrucciones ordenadas por un índice numérico, indicando lo que el programa debía hacer. Por ejemplo:

```
10 PRINT "Escribe un numero"
20 INPUT n1
30 PRINT "Escribe otro numero"
40 INPUT n2
50 PRINT "Su suma es"
60 PRINT n1+n2
```

Como puede verse, al inicio de cada instrucción hay un número, que permite ordenar la secuencia de instrucciones. En el ejemplo anterior, primero se ejecutaría la instrucción PRINT con el número 10, luego el INPUT con el número 20… y así sucesivamente. No es necesario que los números sean múltiplos de 10, pueden ser 1, 2, 3, 4, o 1, 6, 8, 9… lo importante es que el número de una instrucción vaya después de las que queramos que se ejecuten antes que ésa. Así, el siguiente código sería equivalente al anterior, aunque esté desordenado:

```
1 PRINT "Escribe un numero"
3 PRINT "Escribe otro numero"
5 PRINT "Su suma es"
6 PRINT n1+n2
2 INPUT n1
4 INPUT n2
```

Hay que tener en cuenta también que el código fuente puede tener líneas en blanco, que se deben descartar.

La práctica consistirá en implementar un traductor sencillo de instrucciones BASIC para construir el programa equivalente en lenguaje C#. Así, para el programa anterior deberíamos producir el siguiente código en C#:

```
using System;

class Programa

{

      static void Main()

      {

            Console.WriteLine("Escribe un numero");

            int n1 = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Escribe otro numero");

            int n2 = Convert.ToInt32(Console.ReadLine());

            Console.WriteLine("Su suma es");

            Console.WriteLine(n1+n2);

      }

}
```

Distinguiremos únicamente las instrucciones PRINT e INPUT del lenguaje BASIC. Por simplificar, en las instrucciones INPUT siempre almacenaremos datos en variables enteras, como en el ejemplo anterior. Las instrucciones PRINT, por otra parte, pueden o bien imprimir un texto fijo entre comillas dobles, o bien una operación matemática simple con variables.

El programa deberá hacer lo siguiente…

1. Procesamiento del código BASIC

(**0,5 puntos**) En primer lugar, el programa le pedirá al usuario un nombre de fichero de entrada, que contendrá las instrucciones en lenguaje BASIC del programa. Se deberá verificar en este punto que el fichero exista, antes de procesarlo.

(1,5 puntos) Se deberá definir una clase llamada **InstruccionBASIC**, que almacenará los datos de cada instrucción que se procese. En concreto, contendrá como atributos/propiedades el número de instrucción, el comando en sí (un tipo “enum” que podrá valer PRINT o INPUT), y los parámetros (un *string* con el resto de la instrucción). Además del constructor y propiedades get/set, esta clase deberá:

- Implementar la interfaz **IComparable<InstruccionBASIC>**, que permitirá ordenar dos instrucciones de menor a mayor número de instrucción
- Implementar un método llamado **Imprimir**, que volcará la instrucción a consola, con el formato de colores que se especifica más adelante.

(2 puntos) Una vez verificado que el fichero de entrada existe, se deberán procesar la instrucciones BASIC allí contenidas, y generar una lista (*List*) de objetos de tipo *InstruccionBASIC*, ordenada por número de instrucción.

2. Salida del programa

Cuando ya se tenga el listado procesado de instrucciones BASIC, el programa deberá:

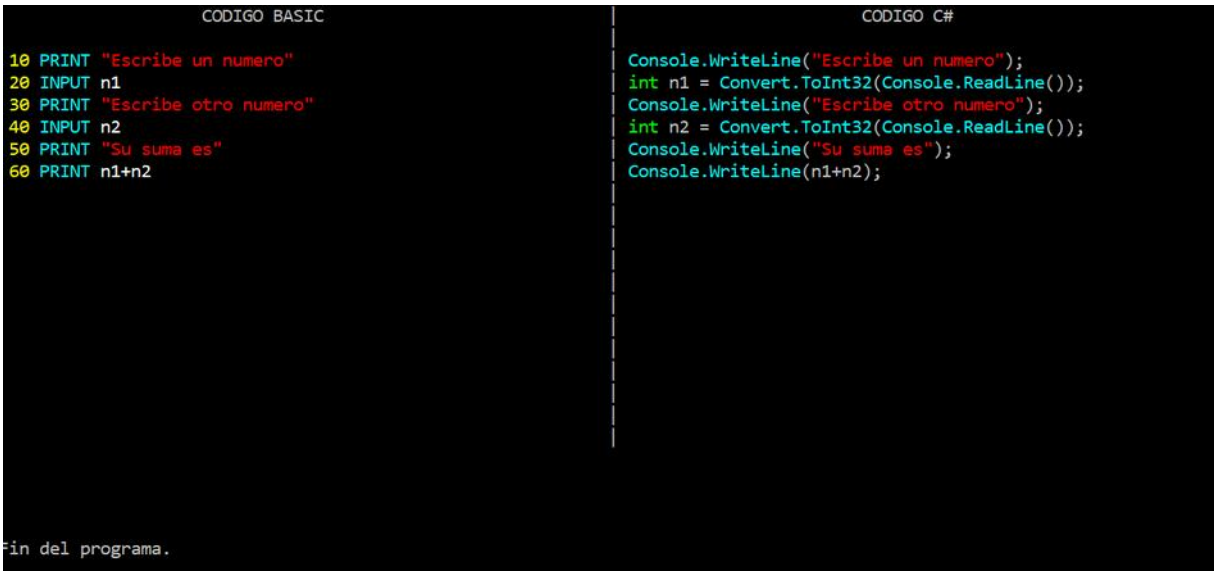
- **(2 puntos)** Mostrar en la mitad izquierda de la consola el código BASIC formateado de acuerdo al código de colores que se especifica a continuación.
- **(2 puntos)** Generar un archivo de salida con el mismo nombre que el de entrada, pero con el sufijo ".cs", con el código C# generado para ese fichero de entrada. Por ejemplo, si el fichero de entrada se llamaba "codigoBASIC.txt", el de salida se llamará "codigoBASIC.txt.cs". Dicho código resultado debe tener una instrucción "using System;" al principio, una clase con su método *Main*, y dentro, el código BASIC traducido a C#, como se muestra en el ejemplo anterior, debidamente indentado (4 espacios por cada nivel de indentación).
- **(2 puntos)** Mostrar en la mitad derecha de la consola el código C# generado (sin contar la clase y el *Main*, sólo las instrucciones traducidas), con el código de colores que se indica a continuación.

Salida por pantalla y código de colores

En pantalla se deberá mostrar tanto el código BASIC procesado (mitad izquierda) como el código C# generado (mitad derecha), siguiendo este código de colores:

- Las instrucciones propias de cada lenguaje (*PRINT* e *INPUT* para BASIC, y *Console.WriteLine, Console.ReadLine* y *Convert.ToInt32* para C#), se deben mostrar en color cyan.
- Los tipos simples (*int*) se deben mostrar en verde
- Los números de instrucción BASIC se deben mostrar en amarillo
- Las cadenas de texto (incluyendo comillas) se deben mostrar en rojo
- Todo lo demás, se debe mostrar en blanco o gris claro.

Para el código BASIC del ejemplo anterior, la salida por pantalla debe ser similar a esta, incluyendo la línea de división intermedia a mitad de pantalla:



Se asumirá que el código BASIC y su resultado no van a exceder de unas 20 líneas, y que serán lo suficientemente cortos (poco anchos) para que quepan en su mitad de la consola.

Observaciones generales de la práctica.

- Además de los atributos indicados para cada clase, se pueden añadir otros atributos, constructores o métodos que se consideren oportunos. Posteriormente, se valorará si esos elementos añadidos son de utilidad o no.
- Recordad, que se **valorarán negativamente algunas malas prácticas** de programación, como por ejemplo:
  - La *suciedad* de código, en lo referente a lo explicado en el Tema 9 del bloque 1 del módulo de Entornos de Desarrollo:
    - Nombres poco apropiados de variables, funciones o clases (ya se venía penalizando con anterioridad).
    - Espaciado y alineación vertical (separación entre funciones y entre bloques de código con propósito diferente).
    - Espaciado y alineación horizontal (incluyendo que las líneas de código no excedan del ancho recomendado).
  - Utilización incorrecta de las estructuras de control, estructuras repetitivas, las instrucciones,... Algunos casos típicos que:
    - Utilización incorrecta de la estructura de control "*switch-case*".
    - La instrucción "*goto*" no debe utilizarse salvo en "*switch-case*".
    - Los bucles "*for*" solo deberían utilizarse cuando se conoce el principio y fin del bucle, es decir, cuántas repeticiones van a realizarse.
  - Las funciones que devuelvan algún tipo de dato (*return*), no deberían tener más de punto de salida, solo deberían tener una única instrucción *return*.
  - La definición inapropiada de atributos.
  - La definición inapropiada de constructores.
  - El acceso inapropiado de cualquier atributo de clase.
  - La repetición de código que pueda ser implementado en funciones.
  - La escasa modularidad del código (por ejemplo, que el bloque *Main* sea innecesariamente grande y no esté dividido en funciones auxiliares apropiadas).