FEI – Traccia dell'esercitazione di laboratorio 04

[N.B. Si suppone di partire da un progetto con le modifiche e aggiunte richieste nelle esercitazioni precedenti correttamente completate]

1) Completare l'implementazione della classe TrasformazioneAffineGrayscale, che esegue trasformazioni affini su immagini grayscale. Suggerimenti:
   - □ Fare riferimento alle dispense per la descrizione della funzione di mapping e della tecnica di interpolazione;
   - □ Il metodo da implementare è Run(), ma data la complessità dell'operazione sarà opportuno aggiungere metodi privati che svolgano specifiche operazioni, ad esempio un metodo che esegua la funzione di mapping (dalle coordinate della nuova immagine a quelle dell'originale), e uno che esegua l'interpolazione (a partire dal valore dei 4 pixel individuati nell'immagine originale);
   - □ La classe deriva da TrasformazioneAffine, una classe generica e astratta che rappresenta la trasformazione affine su una immagine (di tipo TImage): si consiglia di aggiungere a tale classe tutti i membri (campi, metodi, …) che non siano dipendenti dal tipo di immagine, in modo da sfruttarli anche nell'eventuale implementazione della classe TrasformazioneAffineRgb;
   - □ La classe TrasformazioneAffine prevede due proprietà (ScaleFactorX, ScaleFactorY) corrispondenti ai fattori di scala sui due assi: per semplicità, si può supporre che siano uguali (ad esempio utilizzando il solo campo ScaleFactorX come unico fattore di scala s), applicando la funzione di mapping delle dispense.
   - □ Membri della classe Math che possono essere utili: Math.PI, Math.Sin, Math.Cos, Math.Floor (è ovviamente opportuno precalcolare il seno e coseno dell'angolo in modo da richiamare tali funzioni solo una volta);
   - □ Il risultato della funzione di mapping può cadere all'esterno dell'immagine originale (in tal caso si dovrebbe utilizzare il colore nella proprietà Background della classe); per non complicare eccessivamente il codice, può essere utile una funzione membro simile alla seguente, che accetti in input coordinate x,y qualsiasi, e restituisca automaticamente il valore del pixel corrispondente o il background.

```
private byte ImgOrBack(int x, int y)
{
   if (x >= 0 && x < InputImage.Width && y >= 0 && y < InputImage.Height)
      return InputImage[y, x];
   else return Background;
}
```

2) Completare l'implementazione della classe TrasformazioneAffineRgb, che esegue trasformazioni affini su immagini a colori RGB. L'implementazione può semplicemente consistere nell'utilizzo della classe TrasformazioneAffineGrayscale per ciascun canale.

3) [*Facoltativo*] L'implementazione proposta al punto 2) è semplice ma chiaramente inefficiente, in quanto esegue 3 volte la funzione di mapping e l'interpolazione per pixel con le stesse coordinate: implementare la classe TrasformazioneAffineRgb in modo più efficiente (senza utilizzare la classe TrasformazioneAffineGrayscale), ma evitando di replicare il codice nelle due classi (utilizzare la classe da cui entrambe derivano per tutto ciò che è comune a entrambe).