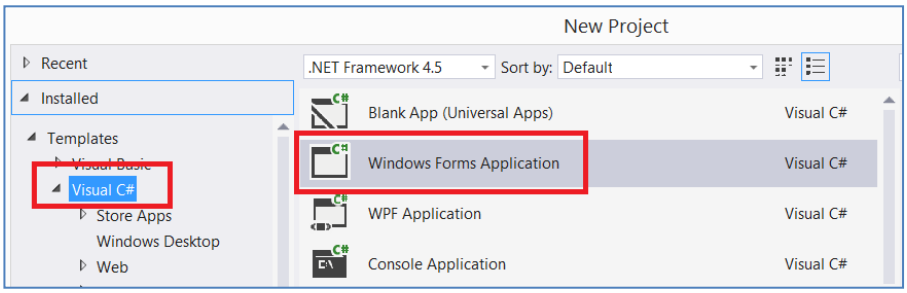
(2.Hafta)

C# PROGRAMLAMA İLE GÖRÜNTÜ İŞLEME

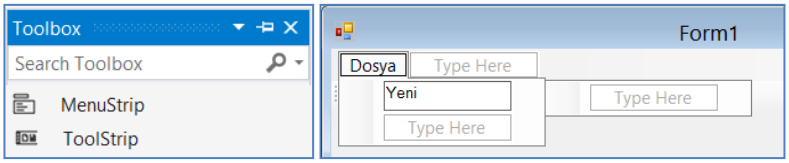
Görüntü işleme kapsamında geliştirilecek algoritmalar C# diliyle yazılacaktır. Bu amaçla bilgisayarımızda Visual Studio programının kurulu olması gerekmektedir. Öncelikle programın altyapının oluşturulup, dersler ilerledikçe içerisine kodlar eklenecektir.

Program Altyapısının Oluşturulması

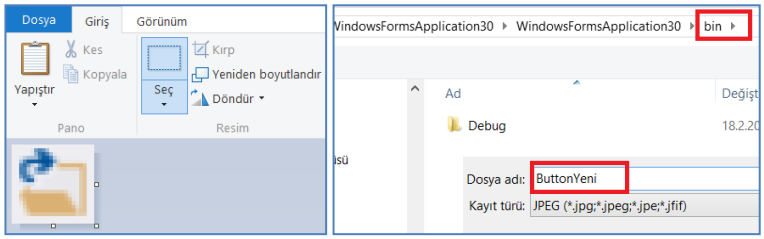
Geliştirilecek program Windows Form Application uygulaması olacaktır.



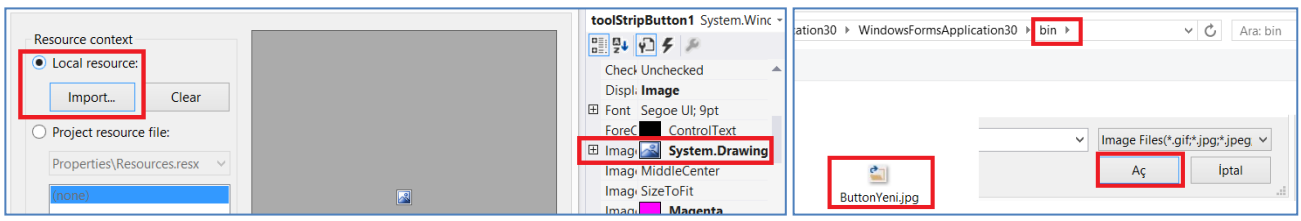
Program içerisinde Menü ve Araç çubuğu kullanılacaktır. Bu amaçla MenuStrip ve ToolStrip nesneleri kullanılacaktır.



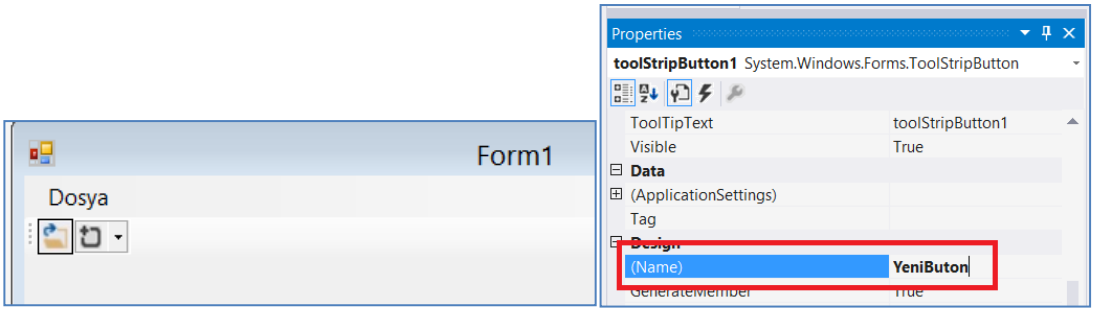
ToolStrip (araç çubuğu) üzerindeki butonların resimlerini atarken, Paintte oluşturulan örnek resimler projenin içerisindeki Bin klasörünün altına kopyalayalım.



Daha sonra ToolStrip üzerindeki butonu seçip Image özelliğinden butonu yükleyelim.



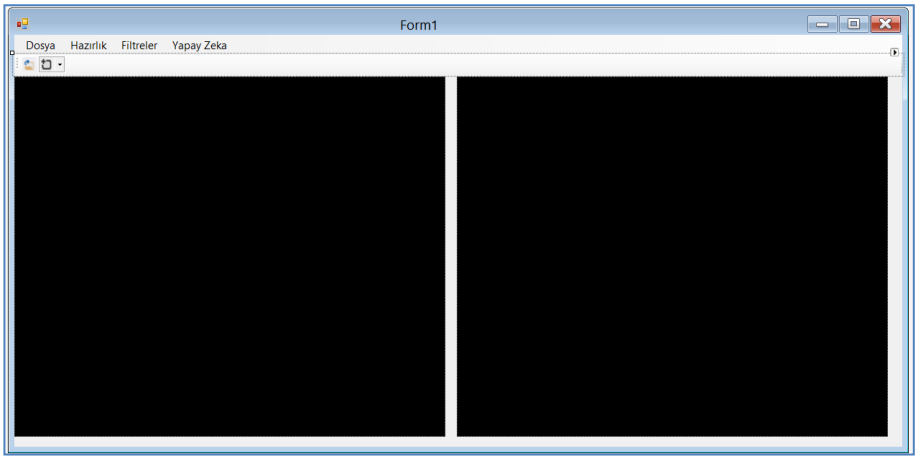
Butonun resmi yüklendikten sonra Properties penceresinden başka gerekli ayarlamalar yapılabilir. Örneğin butonun adı yapacağı işle ilgili olarak değiştirilebilir.



Butona tıkladığımızda ona ait olan fonksiyon kodları açılacaktır.

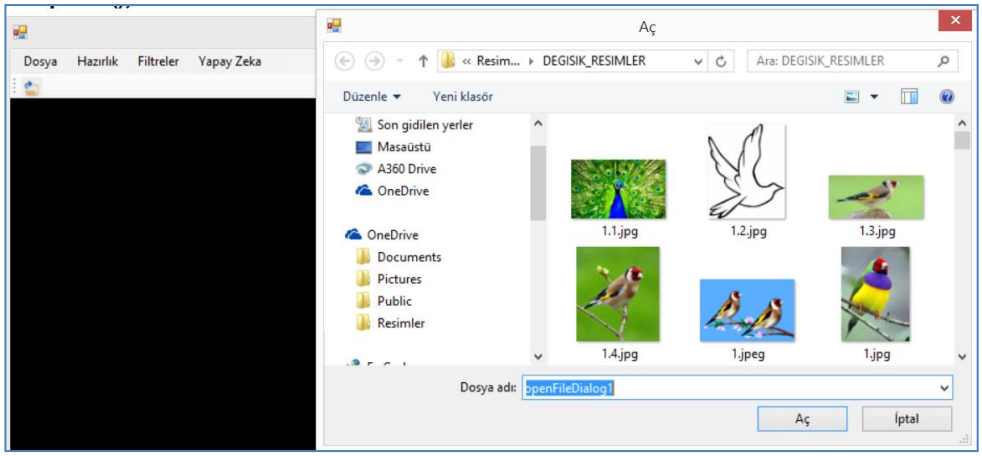
```
public partial class Form1 : Form
{
    1 reference
    public Form1()
    {
        InitializeComponent();
    }

    1 reference
    private void YeniButon_Click(object sender, EventArgs e)
    {

    }
}
}
```

Programımızın tasarımını üstte bir menu çubuğu, altında ise Araç çubuğu olacak şekilde tasarlayalım. Form üzerinde ise iki tane Picturebox olsun. Bunlardan birincisi orjinal resmi, diğeri ise dönüştürülmüş resmi göstersin. Program ilerledikçe ekleme ve çıkarmalar olacaktır. Bu altyapı üzerinde devam edelim.



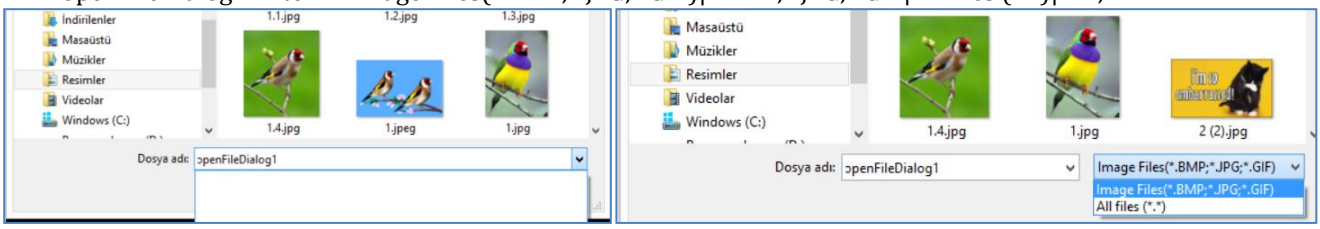
RESİM YÜKLEME

Bilgisayarımızdaki herhangi bir resmi PictureBox1 yüklemek için aşağıdaki kodları kullanabiliriz. Bunu fonksiyon şeklinde yazarsak hem menüden hem de araç çubuğunda aynı kodlar kullanılarak resim yüklenebilir. Burada geliştirilen programda iki tane picturebox kullanılmaktadır. Birincisinde orjinal resim (giriş resmi), ikincisinde ise dönüştürülmüş resim (çıkış resmi) görüntülenecektir.



Dosya açarken uzantıların görüntülenmesi süzmek için uygun olacaktır. Bunun için aşağıdaki kodları kullanabiliriz.

```
openFileDialog1.DefaultExt = ".jpg";
openFileDialog1.Filter = "Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";
```



Eğer resim yükleme çalışırken openFileDalog açıldıktan resim seçilmeyip "iptal" tuşuna basılırsa, resimyolu değişkeni "null" olacağı için program hata verir. Bunu engellemek için Try-Catch kodları içine yazılması gerekir.

```
private void DosyaAc_menu_Click(object sender, EventArgs e)
{
 DosyaAc();
}
private void DosyaAc_toolbar_Click(object sender, EventArgs e)
{
 DosyaAc();
}
 //DOSYA AÇ ----------------------------
 public void DosyaAc()
 {
      try
      {
            openFileDialog1.DefaultExt = ".jpg";
            openFileDialog1.Filter = "Image Files(*.BMP;*.JPG;*.GIF)|*.BMP;*.JPG;*.GIF|All files (*.*)|*.*";

            openFileDialog1.ShowDialog();

            String ResminYolu = openFileDialog1.FileName;
            pictureBox1.Image = Image.FromFile(ResminYolu);
      }
      catch { }
 }
```

Görüntülenen resim çeşitli boyutlarda olabilir. Bu nedenle resmin picturebox içerisinde düzgün görüntülenmesi için SizeMode seçeneğini aşağıdaki seçeneklerden uygun olanla yapmak gerekir. Burada;

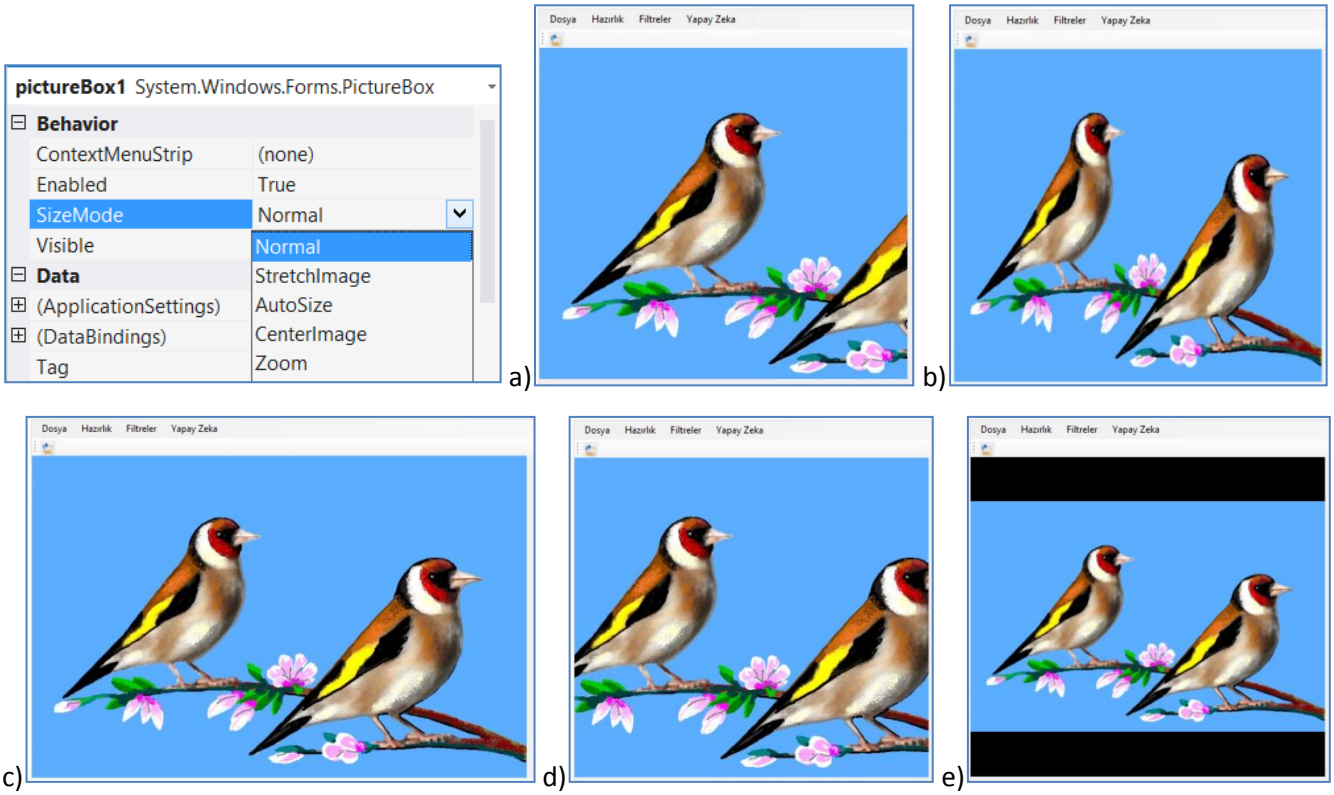
a) Normal (resmin sığdığı kadar kısmı normal görüntülenir

b) StrechImage (resim çerçevenin içine tam sığacak şekilde daraltılır yada genişletilir,

c) AutoSize (çerçeve resmin boyutlarını alır)

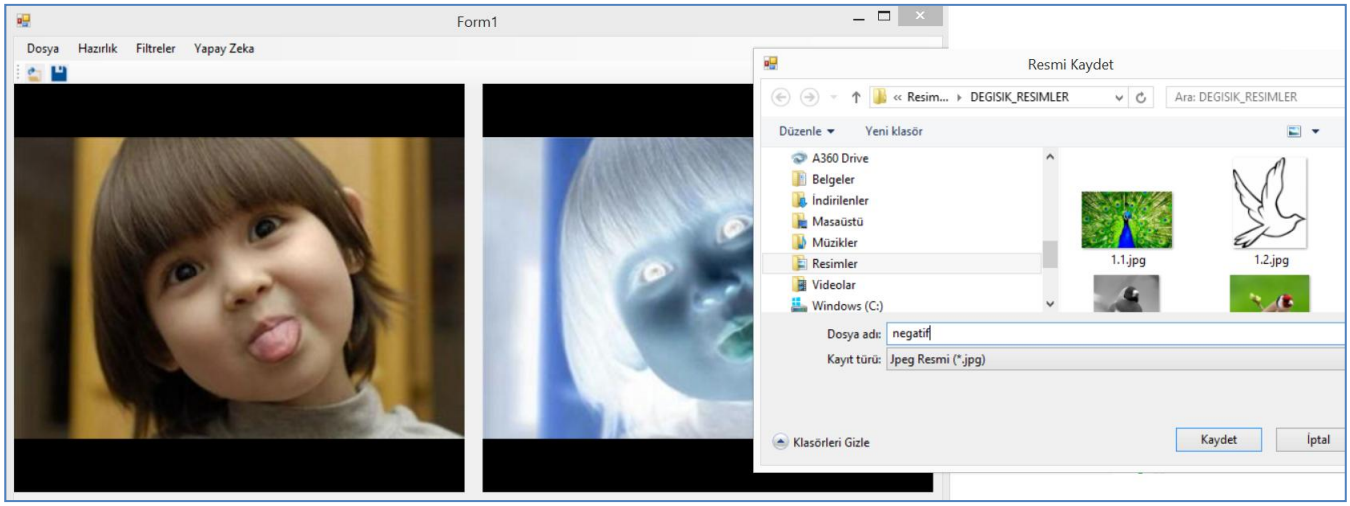
d) CenterImage (resmin orta bölgesi çerçeve içinde normal şekilde görüntülenir, sığmayan kısımlar dışarıda kalır,

e) Zoom (resmin orantısı bozulmadan tamamı çerçeve içinde görüntülenir).



RESMİ KAYDETME

Birinci picturebox da görüntülenen orjinal resim, belli filtrelerden geçirildikten sonra ikinci picturebox da görüntülenecektir. Görüntülenen bu ikinci resim gerektiğinde bilgisayara kaydedilebilmelidir. Bu amaçla aşağıdaki kodları kullanabiliriz. Burada üç tip resim kaydetme yapılmaktadır. Bunlar jpeg, bitmap ve gif resimleridir.




```
//RESMİ KAYDETME----------------------------------------
public void ResmiKaydet()
{
  SaveFileDialog saveFileDialog1 = new SaveFileDialog();
  saveFileDialog1.Filter = "Jpeg Resmi|*.jpg|Bitmap Resmi|*.bmp|Gif Resmi|*.gif";
  saveFileDialog1.Title = "Resmi Kaydet";
```

```
        saveFileDialog1.ShowDialog();

        if (saveFileDialog1.FileName != "") //Dosya adı boş değilse kaydedecek.
        {
            // FileStream nesnesi ile kayıtı gerçekleştirecek.
            FileStream DosyaAkisi = (FileStream)saveFileDialog1.OpenFile();

            switch (saveFileDialog1.FilterIndex)
            {
                case 1:
                    pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Jpeg);
                    break;

                case 2:
                    pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Bmp);
                    break;

                case 3:
                    pictureBox2.Image.Save(DosyaAkisi, System.Drawing.Imaging.ImageFormat.Gif);
                    break;
            }

            DosyaAkisi.Close();
        }
    }
```

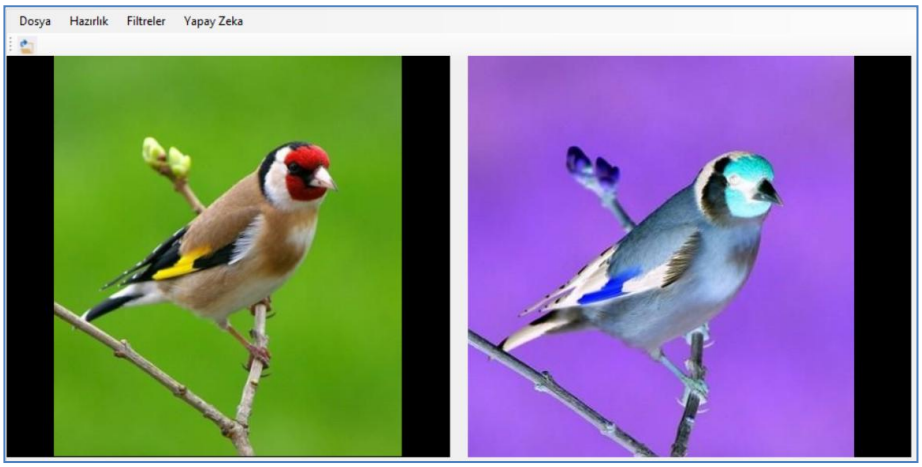
RESMİN NEGATİFİNİ ALMA (Negative)

Bu işlem basit olarak şu şekilde yapılır. *Her piksel renk değeri bir döngü içinde 255 sayısından çıkarıldığında geriye kalan değer, negatif rengi verecektir*. Renkli resimlerde ise 3 renk de aynı işleme tabi tutulur.

Örneğin; siyah renkli bir pikselin değeri 0 dır. Bu pikselin değeri 255-0=255 olarak ayarlanırsa bu renk de beyaz olacaktır. Yada beyaz yakın bir değer olan 212 değerini dönüştürürsek, 255-212=43 olacaktır. Bu da siyaha yakın bir renktir. Bunu matematiksel olarak ifade edersek, g(x;y) giriş resminin Kırmızı piksel değeri, f[x;y] çıkış resminin kırmızı piksel değeri olacaktır.

$$f(x; y) = 255 - g(x; y)$$

Programlama




```
    //NEGATİFİNİ ALMA ----------------------------------
    public Bitmap ResminNegatifiniAl()
    {
        Color OkunanRenk, DonusenRenk;
        int R = 0, G = 0, B = 0;
```

```
        int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı. İçerisine görüntü yüklendi.
        int ResimYuksekligi = GirisResmi.Height;
         CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi ile aynı olur. Tanımlaması globalde yapıldı.

        int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
        for (int x = 0; x < ResimGenisligi; x++)
        {
           j = 0;
           for (int y = 0; y < ResimYuksekligi; y++)
           {
              OkunanRenk = GirisResmi.GetPixel(x, y);
              R = 255 - OkunanRenk.R;
              G = 255 - OkunanRenk.G;
              B = 255 - OkunanRenk.B;
              DonusenRenk = Color.FromArgb(R, G, B);
              CikisResmi.SetPixel(i, j, DonusenRenk);
              j++;
           }
           i++;
        }

        return CikisResmi;
     }
```

GRİ TON RESİM (Siyah-Beyaz) (Grayscale)

Griton resimler renkler olmadan sadece ışığın yoğunluğunu gösteren resimlerdir. Eski teknoloji siyah beyaz resimlere benzer.

Bu resimler bilgisayarda 8 bit formatında saklanır. Bunun anlamı her piksel 8 bit ikili kod ile saklanır. Yani 2^8=256 olarak gösterilirse 0-255 arasında değer alır. 0 siyah rengi gösterirken, 255 beyaz rengi gösterecektir.

Bir griton resim 800x600 piksel boyutlarında olursa içerisinde 480.000 piksel olacaktır. Her piksel 8 bit=1 byte hafızada yer alacağına göre resmin tamamı 480 kByte=0,48 MByte yer kaplayacaktır. Aynı resim renkli olsayda, her renk (RGB-RedGreenBlue) benzer şekilde 256 ton renk alarak 2^8*3=2^{24} yani 24 bit yer kaplar. Bu durumda aynı resim bu sefer 3 katı yer kaplar.

Griton resimleri, kenar tespiti yada eşik uygulamaları gibir farklı amaçlar için ileriki uygulamlarımızda kullanacağız.

Renkli resmin Gri-Ton a dönüştürülmesi

Renkli sayısal bir görüntüyü gri-ton bir görüntüye dönüştürme işlemi aslında RGB renk modelinde belirtilen her bir renk bandına karşı düşen gri-ton görüntülerin ölçeklendirilmesinden başka bir şey değildir. Normalde üç rengin değerini toplayıp üçe bölerek gri tonu elde edebiliriz. Fakat bu gözümüzün farklı renkleri farklı algılama hassasiyetini tam yansıtma. Bu nedenle aşağıdaki formüllerde verildiği gibi ölçekleme yapılmalıdır. İlk verilen formül en gerçekçi olanıdır.

$$GriDegeri = 0.299 \times R + 0.587 \times G + 0.114 \times B$$

$$GriDegeri = 0.21 \times R + 0.71 \times G + 0.071 \times B$$

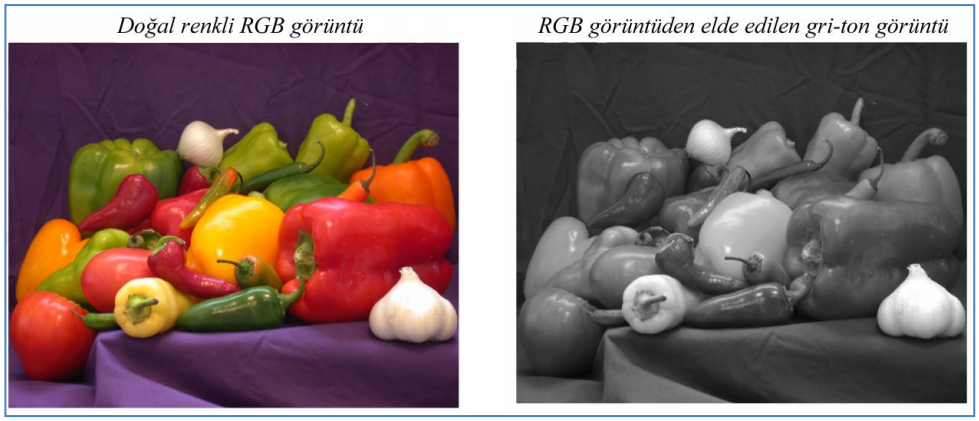
$$GriDegeri = (R + G + B)/3$$

Gri renk değeri bulunduktan sonra, R,G,B renk değerlerinin hepsi aynı renk koduna sahip olmalıdır. Yani;

$$R=GriDegeri,\ G=GriDegeri,\ B=GriDegeri$$

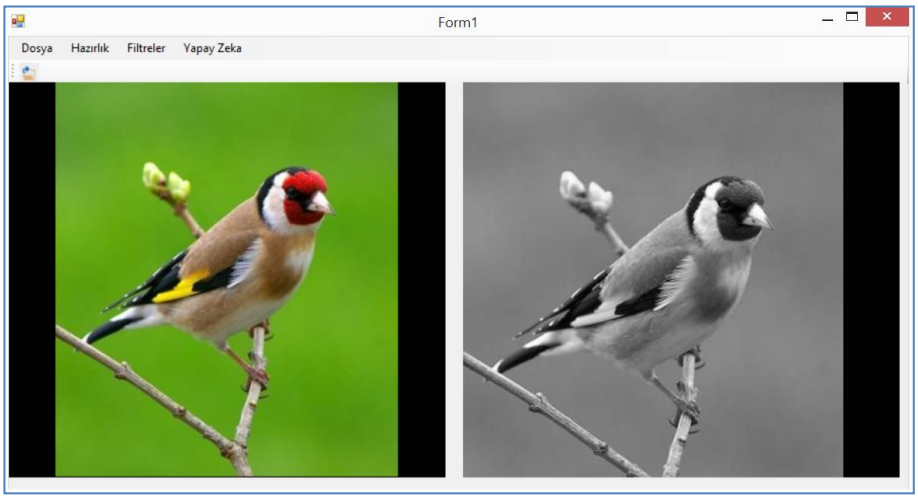
Tabii bu şekliyle resim yine 24 bit olarak saklanmış olur (hafızada renkli resim gibi yer tutar). Bunu 8 bit olarak saklamak gerekir.

{Araştırma: 24 bit resim 8 bit olarak nasıl saklanır. Yani renkler üç tane formatta değilde tek renk formatında nasıl saklanır. Konuyu araştırınız.Her iki formatla saklanan Gri-ton resmin hafızada kapladığı alanın farklı olduğunu gösteriniz.}



Şekil. Doğal renkli görüntüden Gri ton görüntünün elde edilmesi.

Programlama

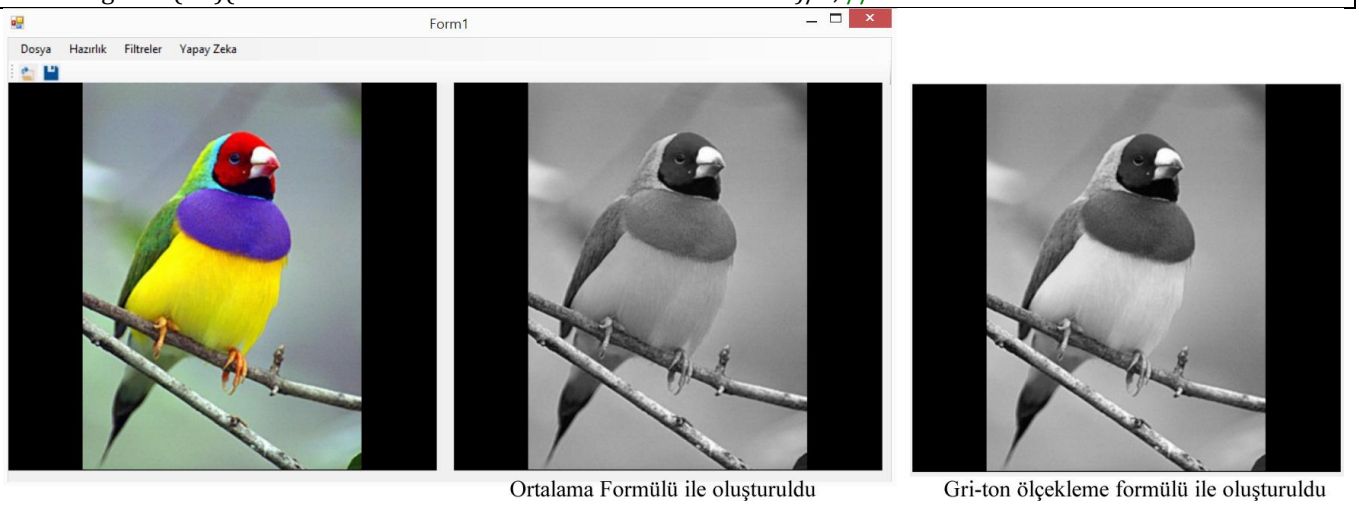



```
    //GRİ TONLAMA ------------------------------------
    public Bitmap ResmiGriTonaDonustur()
    {
      Color OkunanRenk, DonusenRenk;
      int R = 0, G = 0, B = 0;

      int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı. Fonksiyonla gelmedi.
      int ResimYuksekligi = GirisResmi.Height;
      CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi ile aynı olur.

      int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
      for (int x = 0; x < ResimGenisligi; x++)
      {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
          OkunanRenk = GirisResmi.GetPixel(x, y);
```

```
            int GriDegeri = Convert.ToInt16(OkunanRenk.R*0.299 + OkunanRenk.G*0.587 + OkunanRenk.B*0.114);
//Gri-ton formülü
            R = GriDegeri;
            G = GriDegeri;
            B = GriDegeri;
            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(i, j, DonusenRenk);
            j++;
          }
          i++;
        }

        return CikisResmi;
    }
```

Gri-ton formülü olarak 3 kanal (R,G,B) renk ortalamasını alarak da gri-ton a dönüştürebiliriz. Ama bu formül daha az gerçekçi bir sonuç verecektir. Bunun için yukarıdaki programda ilgili satırı aşağıdaki kodlarla değiştirmeliyiz.

```
int GriDegeri = (int)(OkunanRenk.R + OkunanRenk.G + OkunanRenk.B)/3; //Ortalama Gri-ton formülü
```



Ortalama Formülü ile oluşturuldu

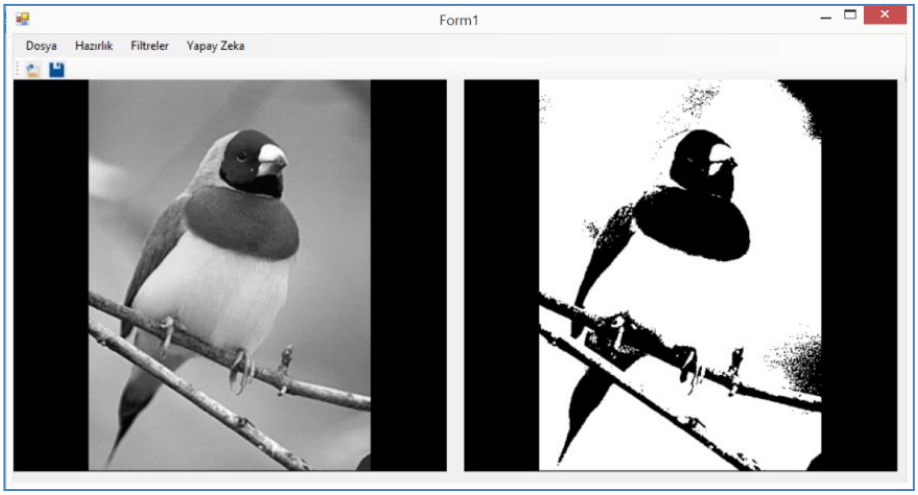
Gri-ton ölçekleme formülü ile oluşturuldu

EŞİKLEME (Thresholding)

Bu teknikte resim içerisindeki her pikselin renk değeri belli bir eşik değeri ile kıyaslanır. Eğer bu eşik değerinin altında ise bir renk, üzerinde ise başka bir renk atanır. Örneğin gri-ton bir resmi bu filtreden geçirirken, renk değeri 128 gibi bir eşik değerin altındaysa 0-siyah olarak, üzerinde ise 255-beyaz olarak atanabilir. Matematiksel olarak aşağıdaki gibi gösterilebilir.

$$f(x,y) = \begin{cases} 0 & eğer\ g(x,y) < T = 128 \\ 255 & diğer \end{cases}$$

Programlama




```
    //EŞİKLEME ----------------------------------
    public Bitmap ResmiEsiklemeYap()
    {
      Color OkunanRenk, DonusenRenk;
      int R = 0, G = 0, B = 0;

      int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
      int ResimYuksekligi = GirisResmi.Height;
       CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi
ile aynı olur.

      int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
      for (int x = 0; x < ResimGenisligi; x++)
      {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
          OkunanRenk = GirisResmi.GetPixel(x, y);
          if(OkunanRenk.R>=128)
            R = 255;
          else
            R = 0;

          if (OkunanRenk.G >= 128)
            G = 255;
          else
            G = 0;
```

```
            if (OkunanRenk.B >= 128)
               B = 255;
            else
               B = 0;

            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(i, j, DonusenRenk);
            j++;
         }
         i++;
      }

      return CikisResmi;
   }
```

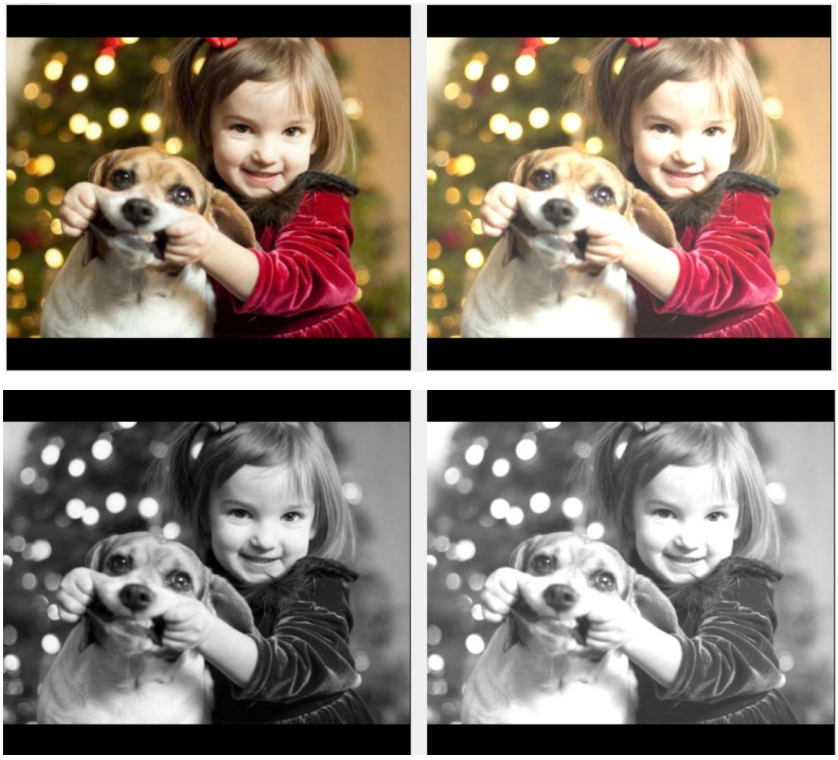
PARLAKLIK (BRIGHTNESS)

Bir resmin parlaklılığını artırma yada rengini açma, beyaz tona doğru ilerlemek için renk kanalları üzerine aynı büyüklükte eşdeğer bir sayı eklemekle gerçekleştirilir. Resmi koyulaştırmak için tam tersi sayı çıkarılır. Burada dikkat edilmesi gereken üzerine eklenen sayılarla sonuç 255 değerini geçmemelidir yada çıkarılan sayılar nedeniyle 0 altına düşmemelidir. Şu şekilde formülüze edebiliriz.

$$f(x,y)= g(x,y)(R\mp b, G\mp b, B\mp b)$$

Program Kodları

Birinci resimler orjinal resim, ikinciler rengi 50 değerinde açılmış resimdir.



```
   //PARLAKLIĞINI DEĞİŞTİRME ---------------------------------------
   public Bitmap ResminParlakliginiDegistir()
   {
      int R = 0, G = 0, B = 0;
      Color OkunanRenk, DonusenRenk;
      Bitmap GirisResmi, CikisResmi;
```

```
        GirisResmi = new Bitmap(pictureBox1.Image);

        int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
        int ResimYuksekligi = GirisResmi.Height;
        CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi ile aynı olur.

        int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
        for (int x = 0; x < ResimGenisligi; x++)
        {
          j = 0;
          for (int y = 0; y < ResimYuksekligi; y++)
          {
            OkunanRenk = GirisResmi.GetPixel(x, y);
          //Rengini 50 değeri ile açacak.
            R = OkunanRenk.R + 50;
            G = OkunanRenk.G + 50;
            B = OkunanRenk.B + 50;

            //Renkler 255 geçtiyse son sınır olan 255 alınacak.
            if (R > 255) R = 255;
            if (G > 255) G = 255;
            if (B > 255) B = 255;

            DonusenRenk = Color.FromArgb(R, G, B);
            CikisResmi.SetPixel(i, j, DonusenRenk);
            j++;
          }
          i++;
        }
        return CikisResmi;
    }
```

KARŞITLIK (KONTRAST)

Görüntünün kontrasını ayarlama biraz daha karmaşık bir uygulamadır. İlk adım, aşağıdaki formülle verilen bir kontrast düzeltme faktörünü hesaplamaktır:

$$F = \frac{259(C + 255)}{255(259 - C)}$$

F: Kontras düzeltme faktörü (double)

C=Kontras seviye katsayısı

Algoritma'nın doğru çalışması için kontrast düzeltme faktörü (F) değerinin bir tamsayı değil, ondalık sayı (double) olması gerekir. Formüldeki C değeri, istenilen kontrast seviyesini belirtir.

Bir sonraki adım, gerçek kontrast ayarını yapmaktır. Aşağıdaki formüller kırmızı (R), yeşil (G), mavi (M) renkler için verilmiştir.

$$R' = F(R - 128) + 128$$

$$G' = F(G - 128) + 128$$

$$B' = F(B - 128) + 128$$

Bu formüllerden çıkan R,G,B renk değerlerinin 0-255 geçerli renk aralığında olması gerekir.

Kırmızı, yeşil ve mavi değerlerin 0'dan 255'e kadar geçerli bir aralıkta olmasını sağlar. Kontrast seviyesi (C) ise +255 ile -255 arasında olmalıdır. Pozitif değerler kontrast miktarını artırırken, Negatif değerler kontrast miktarını düşürecektir.

Program Kodları

Aşağıdaki resimler normal, +128 Kontrast ve -128 kontrast miktarlarını göstermektedir.

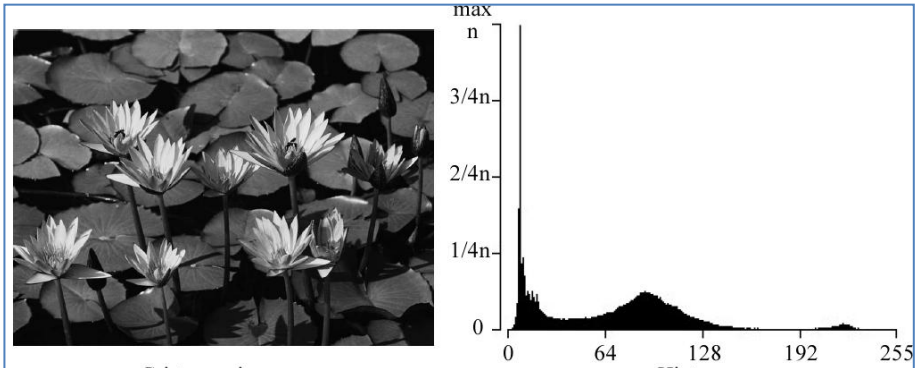
```
    public Bitmap Karsitlik()
    {

      int R = 0, G = 0, B = 0;
      Color OkunanRenk, DonusenRenk;
      Bitmap GirisResmi, CikisResmi;
      GirisResmi = new Bitmap(pictureBox1.Image);

      int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
      int ResimYuksekligi = GirisResmi.Height;
      CikisResmi = new Bitmap(ResimGenisligi, ResimYuksekligi); //Cikis resmini oluşturuyor. Boyutları giriş resmi
ile aynı olur.

      double KontrastSeviyesi = -128;

      double KontrastFaktoru = (259 * (KontrastSeviyesi + 255)) / (255 * (259 - KontrastSeviyesi));

      int i = 0, j = 0; //Çıkış resminin x ve y si olacak.
      for (int x = 0; x < ResimGenisligi; x++)
      {
        j = 0;
        for (int y = 0; y < ResimYuksekligi; y++)
        {
          OkunanRenk = GirisResmi.GetPixel(x, y);
          R = OkunanRenk.R;
          G = OkunanRenk.G;
          B = OkunanRenk.B;

          R = (int)((KontrastFaktoru  * (R- 128)) + 128);
          G = (int)((KontrastFaktoru * (G - 128)) + 128);
          B = (int)((KontrastFaktoru * (B - 128)) + 128);

          //Renkler sınırların dışına çıktıysa, sınır değer alınacak.
          if (R > 255) R = 255;
          if (G > 255) G = 255;
          if (B > 255) B = 255;

          if (R < 0) R = 0;
          if (G < 0) G = 0;
          if (B < 0) B = 0;

          DonusenRenk = Color.FromArgb(R, G, B);
          CikisResmi.SetPixel(i, j, DonusenRenk);
          j++;
        }
        i++;
```

```
    }

    return CikisResmi;


}
```

HİSTOGRAM

Resim üzerinde eşikleme işlemi yapıldığında belli özellikleri ortaya çıkarmak için, uygun eşik değeri kullanmamız gerekir. Fakat bu eşik değerinin hangi değer olması gerektiğini tam olarak bilemeyiz. İşte tam bu noktada histogram konusu devreye girmektedir.

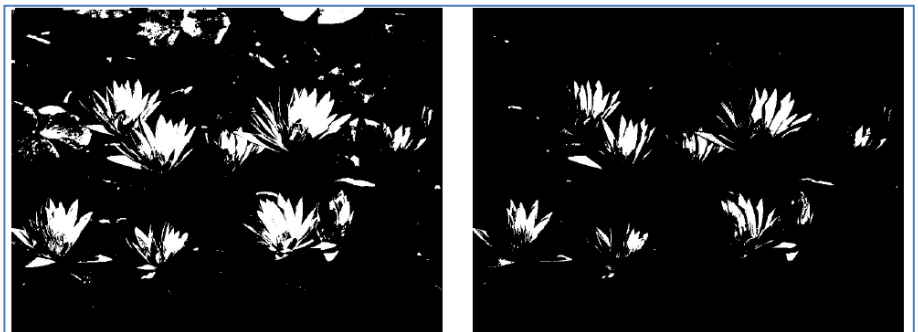
Görüntü histogramı, gri-ton resmin üzerindeki piksel bilgilerini gösteren bir çubuk grafiktir. Grafiğin yatay x-ekseni 0-255 arasındaki gri ton değerlerini gösterir. y-ekseni ise bu değerlere sahip piksel sayılarını gösterir.



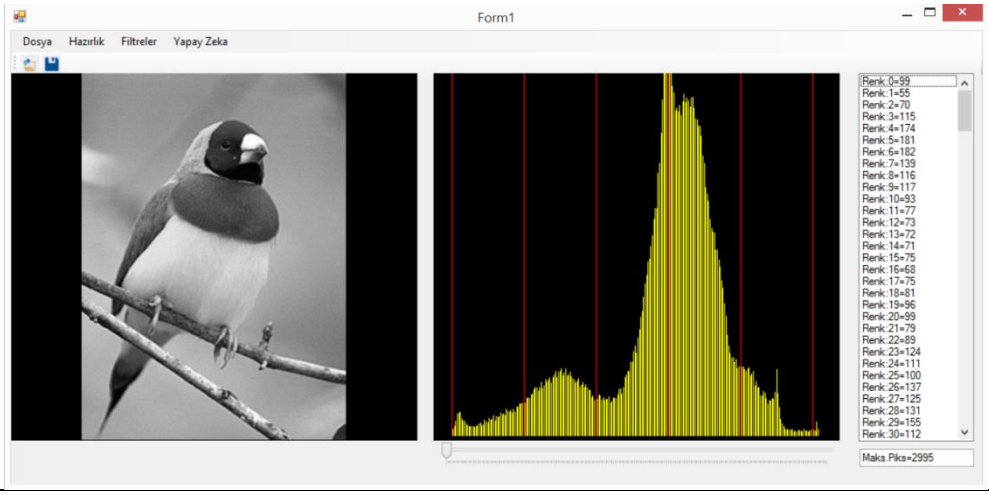
Gri-ton resim Histogram

Aşağıdaki grafikte yatay eksen 0-255 arası eşit 4 parçaya bölünmüştür. Dikey eksen ise hangi renk tonundan en fazla adet varsa onun sayısına bağlı olarak (n ile gösterilmiştir) eşit kısımlara bölünmüştür. Buna göre en fazla piksel adedi 10 renk tonu civardında olduğu görülmektedir.

Resme bakıldığında renklerin koyu piksellerden oluştuğu gözükmektedir. Bu durum histogramın sol tarafa siyaha yakın bölgede toplanmasına yol açmıştır. Aynı zamanda histogramın orta bölgeye yakın bir yerde bir zirve yaptığını görmekteyiz. Bu bölgedeki renkler zambakların yapraklarının renginden kaynaklanmaktadır (suyun üstündeki geniş yapraklar). 128 den sonra renk sayılarının azaldığını görmekteyiz. Bunlar açık renkli tonlardır. Aşağıdaki gibi yapılacak iki farklı eşikle (128 veya 192) renklerin hangi nesnelere ait olduğunu görebiliriz.


128 den sonra eşik uygulanmıştır. 192 den sonra eşik uygulanmıştır.

Programlama




```
    //HİSTOGRAM ----------------------------------
    public void ResminHistograminiCiz()
    {

      ArrayList DiziPiksel = new ArrayList();

      int OrtalamaRenk = 0;
      Color OkunanRenk ;
      int R = 0, G = 0, B = 0;
      Bitmap GirisResmi; //Histogram için giriş resmi gri-ton olmalıdır.
      GirisResmi = new Bitmap(pictureBox1.Image);

      int ResimGenisligi = GirisResmi.Width; //GirisResmi global tanımlandı.
      int ResimYuksekligi = GirisResmi.Height;

      int i = 0; //piksel sayısı tutulacak.
      for (int x = 0; x < GirisResmi.Width; x++)
      {
        for (int y = 0; y < GirisResmi.Height; y++)
        {
          OkunanRenk = GirisResmi.GetPixel(x, y);
          OrtalamaRenk = (int)(OkunanRenk.R + OkunanRenk.G + OkunanRenk.B) / 3; //Griton resimde üç kanal rengi aynı değere sahiptir.

          DiziPiksel.Add(OrtalamaRenk); //Resimdeki tüm noktaları diziye atıyor.
        }

      }

      int [] DiziPikselSayilari = new int[256];
      for (int r = 0; r < 255; r++) //256 tane renk tonu için dönecek.
      {
        int PikselSayisi=0;
        for (int s = 0; s < DiziPiksel.Count ; s++) //resimdeki piksel sayısınca dönecek.
        {
            if (r == Convert.ToInt16(DiziPiksel[s]))
              PikselSayisi++;
        }
        DiziPikselSayilari[r] = PikselSayisi;
      }

      //Değerleri listbox'a ekliyor.
      int RenkMaksPikselSayisi = 0; //Grafikte y eksenini ölçeklerken kullanılacak.
      for (int k = 0; k <= 255; k++)
      {
```

```
            listBox1.Items.Add("Renk:" + k + "=" + DiziPikselSayilari[k]);

            if(DiziPikselSayilari[k]>RenkMaksPikselSayisi)
            {
               RenkMaksPikselSayisi=DiziPikselSayilari[k];
            }
         }

         //Grafiği çiziyor.
         Graphics CizimAlani;
         Pen Kalem1 = new Pen(System.Drawing.Color.Yellow, 1);
         Pen Kalem2 = new Pen(System.Drawing.Color.Red, 1);
         CizimAlani = pictureBox2.CreateGraphics();

         pictureBox2.Refresh();
         int GrafikYuksekligi = 400;
         double OlcekY = RenkMaksPikselSayisi / GrafikYuksekligi, OlcekX = 1.6;
         for (int x = 0; x <= 255; x++)
         {
            CizimAlani.DrawLine(Kalem1, (int)(20 + x * OlcekX), GrafikYuksekligi, (int)(20 + x * OlcekX),
(GrafikYuksekligi - (int)(DiziPikselSayilari[x] / OlcekY)));
            if(x%50==0)
               CizimAlani.DrawLine(Kalem2, (int)(20 + x * OlcekX), GrafikYuksekligi, (int)(20 + x * OlcekX), 0);

         }
         textBox1.Text = "Maks.Piks=" + RenkMaksPikselSayisi.ToString();

      }
```