MODUL 2 – Operasi Citra Sederhana – Linier Brightness, Contrast, Inverse, dan Logarithmic Brightness

A. TUJUAN

- Mahasiswa dapat membuat aplikasi multiple form
- Mahasiswa dapat membuat Transformasi Linier Brightness menggunakan Scroll Bar
- Mahasiswa dapat membuat Contrast Citra menggunakan Scroll Bar
- Mahasiswa dapat membuat Inverse Citra
- Mahasiswa dapat membuat Transformasi Logarithmic Brightness

B. PETUNJUK

1. Awali setiap kegiatan praktikum dengan berdoa
2. Baca dan pahami tujuan, dasar teori, dan latihan-latihan praktikum dengan baik
3. Kerjakan tugas-tugas praktikum dengan baik, sabar dan jujur
4. Tanyakan kepada dosen apabila ada hal-hal yang kurang jelas

C. ALOKASI WAKTU: 6 jam pelajaran

D. DASAR TEORI

- Transformasi Linier Brightness

  Transformasi Linier Brightness merupakan operasi olah citra digital yang paling sederhana. Operasi ini melakukan penambahan nilai pixel jika nilai brightness dinaikkan, dan mengurangkan nilai pixel jika nilai brightness diturunkan.

  $g(x,y) = f(x,y) + b$, dimana g(x,y) adalah nilai pixel setelah transformasi, f(x,y) adalah nilai pixel asli, dan b adalah nilai brightness.

  Karena kesederhanaan operasi ini, maka komputasinya sangat cepat. Walaupun kecepatan komputasi tinggi, namun transformasi ini memiliki kelemahan. Transformasi linier brightness adalah transformasi satu arah, dimana citra setelah di transformasikan menggunakan operasi ini, tidak dapat dikembalikan ke nilai awalnya saat dilakukan reverse linier brightness. Sebagai contoh, jika pixel awal bernilai 240, kemudian ditransformasikan dengan penambahan nilai brightness 20, maka nilai pixel akhirnya adalah 260. Jika program dijalankan, maka akan terjadi error, karena nilai pixel hanya berada pada range 0 – 255. Begitu juga sebaliknya, jika dilakukan pengurangan yang hasilnya lebih kecil dari 0 akan terjadi error juga. Untuk mengatasi hal ini, biasanya dibuat fungsi Truncate yang tugasnya mencegah agar nilai pixel tidak lebih besar dari 255 atau lebih kecil dari 0. Jika nilai setelah operasi lebih besar dari 255 maka nilainya akan dipaksa (ditruncate) menjadi 255, dan jika nilai setelah operasi lebih kecil dari 0, maka nilainya akan dipaksa (ditruncate) menjadi 0. Dengan fungsi Truncate, nilai 240 dioperasikan brightness 20 akan menjadi 255 (bukan 260). Jika nilai 255 ini dioperasikan brightness -20 akan menjadi 235 (bukan 240), hal inilah yang membuat transformasi linier brightness tidak dapat direverse. Jadi fungsi truncate ini mencegah error karena nilai pixel tidak berada diluar range 0-255, tetapi akibatnya pixel yang sudah dibrightness tidak dapat

kembali sempurna. Solusi untuk masalah ini dapat diselesaikan menggunakan transformasi Log Brightness yang dibahas juga pada modul ini.
Algoritma untuk truncate dapat dilihat pada tahapan berikut:

Steps	Process
1	Start
2	baca nilai pixel f1
3	Jika f1 < 0, maka nilai f1 diubah menjadi 0
4	Jika f1 > 255, maka nilai f1 diubah menjadi 255
5	Stop

- Transformasi Contrast

  Transformasi Contrast lebih kompleks daripada Linier Brightness, otomatis lebih lama proses komputasinya. Persamaannya dengan linier Brightness, sama-sama tidak dapat dikembalikan ke bentuk awalnya. Sehingga fungsi Truncate tetapi harus digunakan untuk Contrast.
  Langkah awal yang dilakukan adalah menghitung Contrast Correction Factor menggunakan rumus berikut:

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

  Agar dapat bekerja, Nilai *F* disimpan dalam variable bertipe double dan bukan integer. Nilai C merepresentasikan level nilai contrast yang diinginkan.
  Tahap berikutnya adalah dengan melakukan perhitungan nilai contrast pada tiap pixel R, G, dan B.

  $R' = F(R - 128) + 128$, dimana R adalah nilai pixel Red dan R' adalah nilai pixel Red akhir setelah dilakukan operasi Contrast. Karena nilai hasil dari operasi ini dapat berada diluar range 0-255, maka fungsi truncate digunakan. Ingat kembali jika fungsi truncate dipakai, maka operasi ini tidak dapat direverse.

  Berikut adalah pseudo-code dari rumus contrast diatas:

1	`factor = ……..`
2	`colour = GetPixelColour(x, y)`
3	`redBaru = ……..`
4	`greenBaru = ……..`
5	`blueBaru = ……..`
6	`SetPixelColour(x, y) = RGB(redBaru, greenBaru, blueBaru)`

- Inverse Citra

  Inverse Citra adalah proses yang digunakan untuk membalik nilai citra dengan cermin nilai pixel tengah. Hasil yang didapatkan dikenal dengan nama citra negative. Proses ini juga mudah sekali karena operasinya hanya mengurangkan nilai 255 dengan nilai pixel merah, hijau, dan biru.

  $g(x) = 255 - f(x)$, dimana g(x,y) adalah citra negative, dan f(x,y) adalah citra asli

  Jadi jika nilai pixel asli 255, dengan operasi ini hasilnya akan bernilai 0 (karena 255 – 255).

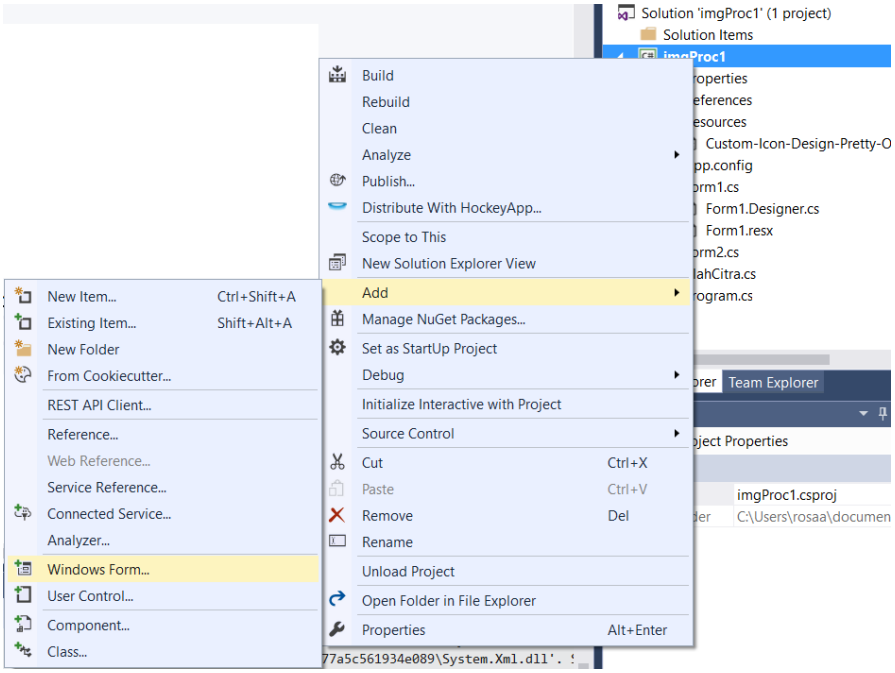
- Transformasi Logarithmic Brightness
  Operator Logarithmic adalah processor point sederhana dimana fungsi mappingnya dalam bentuk kurva logarithmic. Dengan kata lain, tiap nilai pixel diganti dengan nilai logarithmic. Kebanyakan implementasinya menggunakan natural logarithm atau logarithm basis-10. Fungsi mapping logarithmicnya adalah:
  $Q(i,j) = c\log(|P(i,j)|)$
  Karena logarithm tidak bias didefinisikan dengan nilai 0, kebanyakan implementasi dari operator ini ditambahkan dengan nilai 1 pada citra sebelum dihitung logarithmicnya. Operatornya kemudian menjadi:
  $Q(i,j) = c\log(1 + |P(i,j)|)$

E. LATIHAN PRAKTIKUM

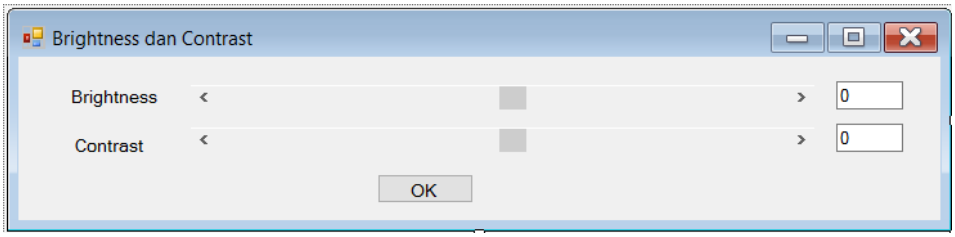
1. Membuat Brightness menggunakan multiple form dan Control Scroll Bar

Percobaan ini akan melanjutkan membuat Brightness pada percobaan di modul 1. Pada modul 1, Brightness diset menggunakan nilai konstanta tertentu dan menghasilkan warna baru. Pada percobaan ini, nilai Brightness akan diset menggunakan Horizontal Scroll Bar atau Textbox dari form baru.

1. Siapkan Form 2 yang digunakan sebagai kendali dari Brightness dan Contrast untuk form pertama. Dari Solution Exlorer, pilih project, klik kanan Add Windows Form.

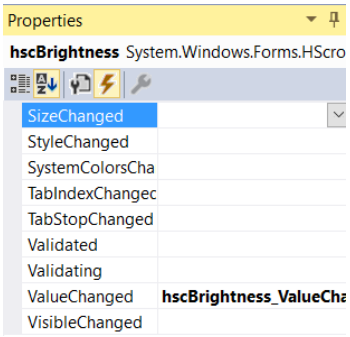


2. Buat Form control seperti pada contoh berikut.



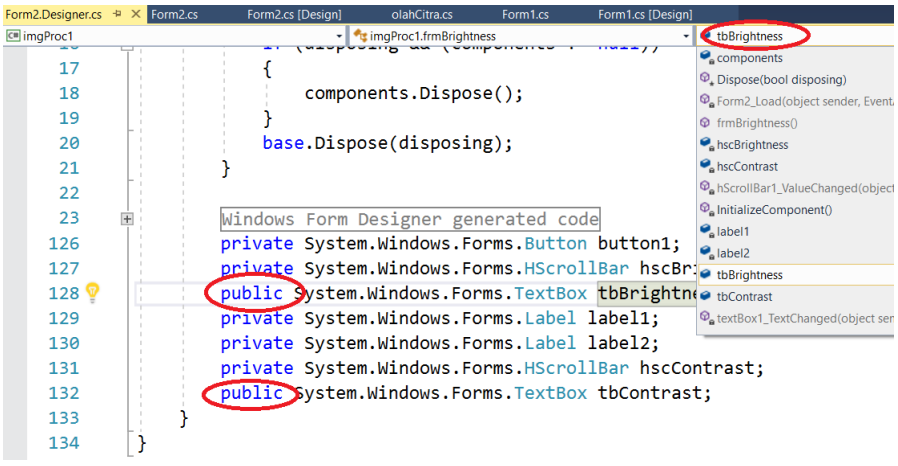
- Set Form Text menjadi “Brightness dan Contrast”. Beri nama Form2 dengan nama frmBrightness.

- Tambahkan Dua Label diset text menjadi “Brightness” dan “Contrast”.
- Tambahkan Dua Horizontal Scroll Bar diset name sebagai “hscBrightness” dan “hscContrast”. Set Value sebagai “0”. Set nilai Maksimumnya dengan nilai 127, dan nilai minimumnya dengan nilai -127. Set event ValueChanged dari hscBrightness dan hscContrast. Event ini digunakan untuk mengubah nilai textbox jika terjadi perubahan nilai pada ScrollBar. Pilih ScrollBar, klik tombol Event (gambar petir) pada properties, Double click pada ValueChanged, dan tambahkan kode berikut. Karena nilai ScrollBar adalah numeric, maka harus dikonversi menjadi String menggunakan method *ToString();* .




```
private void hscBrightness_ValueChanged(object sender, EventArgs e)
        {
            tbBrightness.Text = hscBrightness.Value.ToString();
        }
```

- Tambahkan Dua TextBox diset name sebagai “tbBrightness” dan “tbContrast”. Set textnya menjadi “0”. Nilai angka pada TextBox sesuai dengan nilai angka dari Horizontal Scroll Bar. Jika nilai Scroll Bar berubah, maka nilai textboxnya juga berubah. Jika nilai textboxnya diubah, maka nilai scroll bar juga ikut berubah. Jika nilai textbox “null” atau kosong atau “-“, maka nilai scrollbar adalah “0”. Jika nilai TextBox lebih kecil dari -127, maka akan diset menjadi -127. Jika nilai TextBox lebih besar dari 127, maka akan diset menjadi 127. Set “tbBrightness” dan “tbContrast” sebagai public Control, agar dapat diakses oleh form utama. Perhatikan Screenshot untuk mengubah Control private menjadi public. Saat textbox berubah, maka nilai ScrollBar juga berubah. Tambahkan kode berikut pada event “TextChanged”:



Nilai ScrollBar sesuai dengan nilai TextBox jika berada pada range -127 hingga 127. Karena TextBox bertipe Text dan ScrollBar bertipe Numeric, maka nilai TextBox harus dikonversi ke Numeric (int16). Jika diinputkan selain angka, atau diluar range yang ditentukan, maka akan muncul error MessageBox. Lakukan langkah yang sama pada tbContrast.

```
private void tbBrightness_TextChanged(object sender, EventArgs e)
{
   if ((tbBrightness.Text == "") || (tbBrightness.Text == "-"))
   {
      hscBrightness.Value = 0;
      tbBrightness.Text = "0";
   }
   else if ((Convert.ToInt16(tbBrightness.Text) <= 127) &&
(Convert.ToInt16(tbBrightness.Text) >= -127))
   {
      hscBrightness.Value = Convert.ToInt16(tbBrightness.Text);
   }
   else
   {
      MessageBox.Show("Input nilai Error");
      tbBrightness.Text = "0";
   }
}
```

- Tambahkan Satu Button set text menjadi “OK”. Set name sebagai “tbOK”. Set Property DialogResult dari Button menjadi “OK”. Jika tombol OK ditekan, maka frmBrightness akan ditutup. Double click pada Tombol OK, ketikkan kode berikut. Kode ini digunakan untuk menutup form jika nilai Brightness sudah diset.

```
private void tbOK_Click(object sender, EventArgs e)
{
   this.Close();
}
```

3. Double-click frmBrightness pada File Form2.cs [Design]. kode berikut digunakan untuk memberi nilai awal tbBrightness dan tbContrast menjadi “0”:

```
private void Form2_Load(object sender, EventArgs e)
{
   tbBrightness.Text = hscBrightness.Value.ToString();
   tbContrast.Text = hscContrast.Value.ToString();
}
```

4. Brightness yang akan dibuat pada percobaan ini adalah Brightness Linier, dimana nilainya bertambah atau berkurang sesuai dengan variable nilai yang diisikan. Akan ada kemungkinan nilai Brightness setelah ditambahkan variable, dapat bernilai lebih besar dari 255. Demikian juga jika dikurangkan dengan variable tertentu, maka nilai pixelnya akan lebih kecil dari 0. Nilai pixel yang lebih besar dari 255 ataupun lebih kecil dari 0, akan menyebabkan program error. Untuk mencegah terjadinya error, perlu dibuat fungsi truncate yang akan membuat nilai pixel menjadi 255 jika nilai pixel setelah dilakukan brightness lebih besar dari 255, dan membuat nilai pixel menjadi 0 jika nilai pixel setelah dilakukan brightness lebih kecil dari 0. Karena proses brightness dilakukan

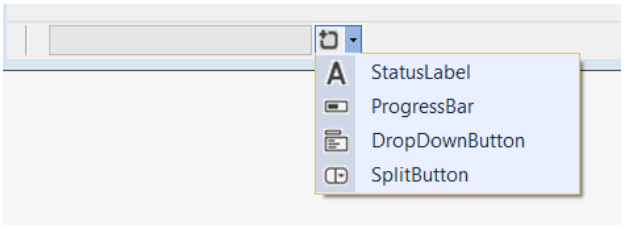
pada Form utama, maka fungsi truncate juga dilakukan pada Form utama. Perhatikan kode berikut:

```
private static int truncate(int x)
{
    if (x > 255) x = 255;
    else if (x < 0) x = 0;
    return x;
}
```

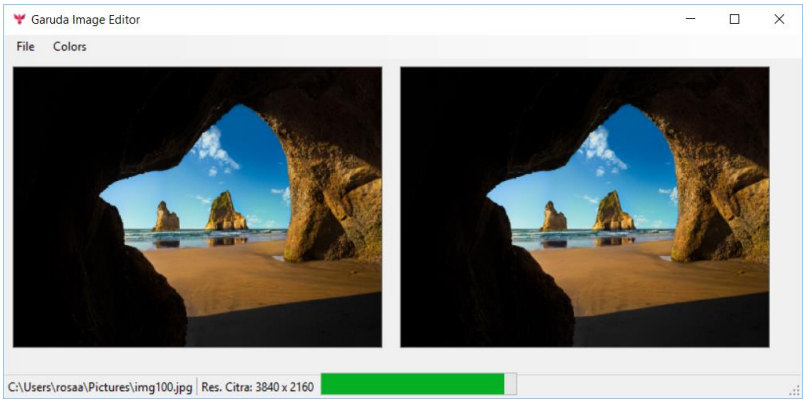
5. Pada Form Utama, tambahkan kode berikut saat menuStrip Brightness dan Contrast di klik. Bagian awal dari kode berfungsi untuk memeriksa apakah citra yang akan diolah sudah dibuka atau belum, jika belum maka akan muncul MessageBox yang memberi pesan "Tidak ada citra yang akan diolah". FrmBrightness di buat sebagai object baru "frm2" seperti terlihat pada baris 70. Jika tombol OK dari frm2 ditekan (baris 71), maka proses brightness akan dijalankan.

```
62      private void brightnessContrastToolStripMenuItem_Click(object sender, EventArgs e)
63      {
64          if (pbInput.Image == null)
65          {
66              MessageBox.Show("Tidak Ada citra yang akan diolah");
67          }
68          else
69          {
70              frmBrightness frm2 = new frmBrightness();
71              if (frm2.ShowDialog() == DialogResult.OK)
72              {
73                  Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
74                  int nilaiBrightness = Convert.ToInt16(frm2.tbBrightness.Text);
75                  for (int i = 0; i < b.Width; i++)
76                  {
77                      for (int j = 0; j < b.Height; j++)
78                      {
79                          Color c1 = b.GetPixel(i, j);
80                          int r1 =
81                          int g1 =
82                          int b1 =
83                          b.SetPixel(i, j, Color.FromArgb(r1, g1, b1));
84                      }
85                  }
86                  this.pbOutput.Image = b;
87              }
88          }
89      }
```

6. Tahap berikutnya dari percobaan ini adalah menggunakan control Progress Bar untuk melihat progress dari proses olah citra. Progress Bar dapat ditambahkan dari ToolBox secara langsung, atau dari Status Strip. Untuk percobaan ini, penulis menggunakan Progress Bar dari Status Strip. Set Visible dari Progress Bar sebagai "False". Set Name sebagai "ProgressBar1". Sesaat sebelum proses olah citra dimulai, ProgressBar1 Visible diset "True". Setelah Selesai proses, visible dari ProgressBar1 diset kembali menjadi "False". Untuk Algoritma Progress Bar, Penulis mengupdate nilai Progress Bar pada tiap nilai perulangan lebar citra. Perhatikan kode berikut yang merupakan tambahan dari kode sebelumnya untuk menampilkan status Progress Bar.




```
Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
int nilaiBrightness = Convert.ToInt16(frm2.tbBrightness.Text);
progressBar1.Visible = true;
for (int i = 0; i < b.Width; i++)
{
   for (int j = 0; j < b.Height; j++)
   {
     Color c1 = b.GetPixel(i, j);
     int r1 = ……..;
     int g1 = ……..;
     int b1 = ……..;
     b.SetPixel(i, j, Color.FromArgb(r1, g1, b1));
   }
   progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
}
progressBar1.Visible = false;
this.pbOutput.Image = b;
```



Berikut adalah citra yang diproses menggunakan brightness nilai 50.

2. Membuat Contrast

Contrast dan Brightness akan diproses secara bersamaan pada frmBrightness. Karena diproses bersamaan, maka kode proses brightness yang telah dibuat di percobaan sebelumnya ditambahkan dengan kode yang sesuai dengan algoritma Contrast yang telah dibahas pada bagian awal dari modul ini. Tipe data dari Contrast Factor adalah Double, sehingga proses olah citra dilakukan dengan mengubah semua tipe datanya menjadi double, begitu juga fungsi dari truncate. Kemudian setelah proses selesai dan saat menyimpan nilai dalam bentuk pixel, tipe data dikembalikan dalam bentuk integer. Berikut kode yang digunakan untuk membuat Contrast.

```
private static double truncate(double x)
{
   if …….
   else if …….
   return x;
}
```

```
frmBrightness frm2 = new frmBrightness();
if (frm2.ShowDialog() == DialogResult.OK)
{
   ……
   ……
   ……
   ……
   ……
   ……
   ……
   ……
   ……
   ……
}
```

Berikut adalah tampilan hasil citra yang diproses contrast nilai 100.

3. Membuat Inverse Citra

Inverse citra dilakukan dengan melakukan pembalikan nilai citra. Jika nilai citra adalah maksimal (255), maka nilai inversenya adalah minimal (0). Sangat mudah sekali untuk melakukan pembalikan nilai citra, yaitu dengan mengurangi nilai 255 dengan nilai pixel yang dioperasikan. Berikut adalah kodenya:

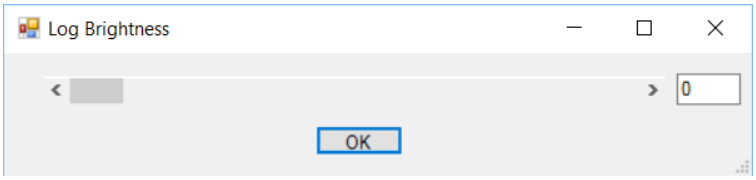
```
private void inverseToolStripMenuItem_Click(object sender, EventArgs e)
{
   if (pbInput.Image == null)
   {
      MessageBox.Show("Tidak Ada citra yang akan diolah");
   }
   else
   {
      Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
      progressBar1.Visible = true;
      for (int i = 0; i < b.Width; i++)
      {
         for (int j = 0; j < b.Height; j++)
         {
            Color c1 = b.GetPixel(i, j);
            int r1 = ……..;
            int g1 = ……..;
            int b1 = ……..;
            b.SetPixel(i, j, Color.FromArgb(r1, g1, b1));
         }
         progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
      }
      progressBar1.Visible = false;
      this.pbOutput.Image = b;
   }
}
```

Berikut adalah hasil citra yang diproses menggunakan Inverse.

4. Tranformasi Log Brightness Citra

Pada Percobaan pertama modul ini, anda telah membuat Brightness linier. Brightness ini memiliki kelemahan yaitu jika proses Brightness dilakukan, maka akan sulit untuk mengembalikan lagi citra ke bentuk asalnya. Contoh jika pixel bernilai 200 dan diberikan brightness linier senilai 70, maka citra akhirnya bernilai 255 (maksimal nilai citra 255, tidak mungkin pixel bernilai 270). Apabila nilai 255 diberikan nilai -70 untuk mengembalikan lagi kebentuk semula, hasilnya adalah 185 (bukan 200). Hal ini terjadi karena efek truncate. Pada Log Brightness hal ini tidak akan terjadi karena log brightness tidak membutuhkan fungsi truncate. Aplikasi Image Editor yang popular saat ini seperti GIMP dan Photoshop biasanya menggunakan Log Brightness dan bukan Linier Brightness.

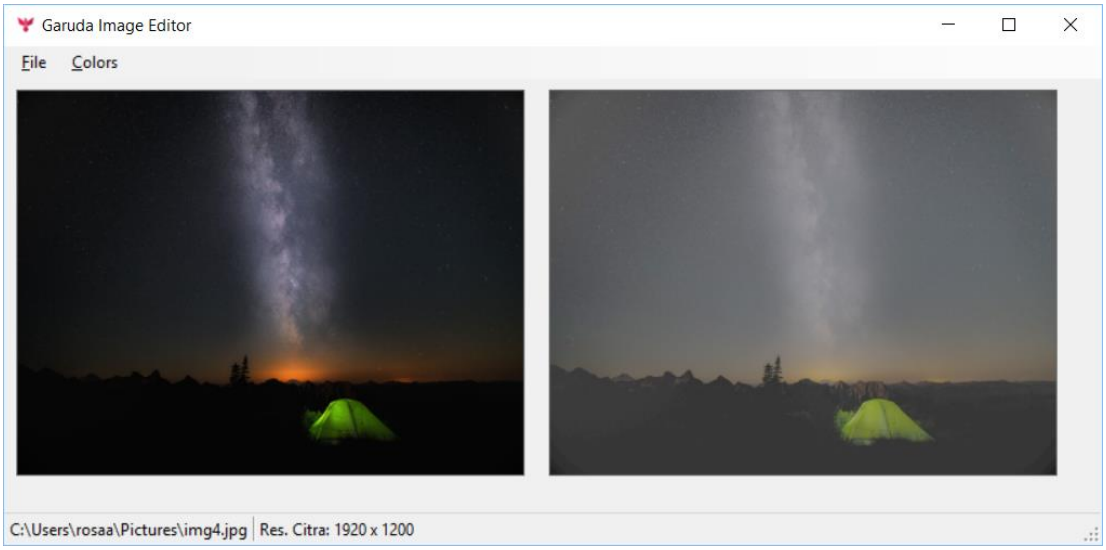
Buat Form baru yang digunakan sebagai masukan nilai Log Brightness. Nilai masukannya berada pada range 0 – 105 (karena nilai log basis 10 berada pada range 0 – 2,408). Tipe data untuk operasi Log Brightness adalah Double.



Perhatikan kode berikut:

```
frmLogBr frm3 = new frmLogBr();
if (frm3.ShowDialog() == DialogResult.OK)
{
   Bitmap b = new Bitmap((Bitmap)this.pbInput.Image);
   double nilaiBrightness = Convert.ToDouble(frm3.tbLogBr.Text);
   progressBar1.Visible = true;
   for (int i = 0; i < b.Width; i++)
   {
      for (int j = 0; j < b.Height; j++)
      {
         Color c1 = b.GetPixel(i, j);
         double r1 =    ..........
         double g1 =    ..........
         double b1 =    ..........
         b.SetPixel(i, j, Color.FromArgb(Convert.ToInt16(r1),
Convert.ToInt16(g1), Convert.ToInt16(b1)));
      }
      progressBar1.Value = Convert.ToInt16(100 * (i + 1) / b.Width);
   }
   progressBar1.Visible = false;
   this.pbOutput.Image = b;
}
```

Berikut adalah hasil citra yang diproses menggunakan log brightness 70.



A. TUGAS PRAKTIKUM

Kerjakan tahapan-tahapan percobaan yang dituliskan pada modul ini.

Buat transformasi Log Contrast.

--- SELAMAT BELAJAR ---