Kaba kod

**İçindekiler**

[Data Klasörü 2](#__RefHeading___Toc107_753869283)

[I. Ddosya.cs 2](#__RefHeading___Toc109_753869283)

[II. Dsehir.cs 2](#__RefHeading___Toc113_753869283)

[Manager Klasörü 3](#__RefHeading___Toc115_753869283)

[I. DosyaManager 3](#__RefHeading___Toc130_753869283)

[II. KomşulukMatrisiManager 8](#__RefHeading___Toc134_753869283)

[III. SehirManager 9](#__RefHeading___Toc136_753869283)

[IV. YolBulucu 12](#__RefHeading___Toc180_753869283)

[Form Ekranları 16](#__RefHeading___Toc2056_2738574670)

[I. Gezgin-kargo-takip-problemi 16](#__RefHeading___Toc2061_2738574670)

# Data Klasörü

## Ddosya.cs

|  |
| --- |
| class DDosya  {  baslangicYolu = Application.StartupPath; // Başlangıç Dizini İşaret Edilir  path = baslangicYolu + "/assets/"; // Klasör Yolu  ciktilar = baslangicYolu + "/outputs/"; // Çıktıların tutulacağı klasörün yolu  konumDosyaYolu = path+ "konum.png"; //Konum iconunun yolu  haritaDosyaYolu = path+"harita.jpg"; // Harita resminin yolu  sehirlerDosyaYolu = path + "sehir.txt"; //Şehir bilgilerinin tutulduğu dosyanın yolu  kordinatDosyaYolu = path + "kordinat.txt"; //Harita kordinatlarının tutulduğu dosyanın yolu  ciktiDosyaYolu = ciktilar +  DateTime.Now.Year.ToString() + "-" +  DateTime.Now.Month.ToString() + "-" +  DateTime.Now.Day.ToString() + "-" +  DateTime.Now.Hour.ToString()+"-"+  DateTime.Now.Minute.ToString() +"-"+  DateTime.Now.Second.ToString() +  ".txt"; // Oluşturulan çıktı dosyasının adı ve yolu    public DDosya()  {  Directory.CreateDirectory(baslangicYolu+"/Outputs"); // Outputs klasörü yoksa oluşturma komutu  } |

## Dsehir.cs

|  |
| --- |
| class DSehir  {  public int plaka; //Hafızada tutulacak şehrin plakası  public int kordinatX; //Hafızada tutulacak şehrin x kordinatı  public int kordinatY; //Hafızada tutulacak şehrin y kordinatı  public string sehirAdi; //Hafızada tutulacak şehrin adı  public int komsuSayisi; //Hafızada tutulacak şehrin komşu sayısı  public DSehir[] komsular = new Dsehir[100]; //Hafızada tutulacak şehrin komşuları  public int[] komsuMesafe = new int[100]; //Hafızada tutulacak şehrin komşu mesafelri  public DSehir ileri; //Hafızada tutulacak şehrin ileri kayıt işlemi  public DSehir(int plaka, string sehirAdi) //Constructor  {  this.plaka = plaka; // Gelen plakayı plakaya eşitle  this.sehirAdi = sehirAdi; //Gelen sehir adını şehir adına eşitle  this.komsuSayisi = 0; // komşu sayısını 0 a eşitle  }  } |

# Manager Klasörü

## DosyaManager

|  |
| --- |
| class DosyaManager  {  public DDosya Dosya = new Ddosya(); //Ddosya türünde bir nesne oluşturulur.  public void sehirListesiAl(SehirManager sehirManager) // Şehir listesinin alınması işlemi  {  string dosya\_yolu = @Dosya.sehirlerDosyaYolu; // şehirler.txt dosyasının yolu tanımlanır.  FileStream fs = new FileStream(dosya\_yolu, FileMode.Open, FileAccess.Read); //FileStream nenesioluşturulur.  StreamReader sw = new StreamReader(fs); //StreamReader türünde sw tanımlanır ve fs parametresi gönderilir.  string yazi = sw.ReadLine(); // ilk satır okunur.  char ayrac = ','; // ayraç için karakter belirtilir  while (yazi != null) // yazi boş olana kadar devam edilir  {  string[] satir = yazi.Split(ayrac); // ayraç ile satirdaki veriler ayrılır.  yazi = sw.ReadLine(); // yeni bir satır yazi nesnesine aktrarılır.  sehirManager.sehirEkle(Int32.Parse(satir[0]), satir[1]); // sehirManager nesnesinin sehirEkle methoduna ayraç ile ayrılan veriler gönderilir.  /\*  satir[0] → Plaka  satir[1] → Şehir Adı  \*/  }  sw.Close(); // StreamReader kapatılır.  fs.Close(); // FileStream kapatılır  }  public void komsuListesiAl(SehirManager sehirManager) // Komşu listesinin alınması  {  string dosya\_yolu = @Dosya.sehirlerDosyaYolu; // şehirler.txt dosyasının yolu tanımlanır.  FileStream fs = new FileStream(dosya\_yolu, FileMode.Open, FileAccess.Read); //FileStream nenesioluşturulur.  StreamReader sw = new StreamReader(fs); //StreamReader türünde sw tanımlanır ve fs parametresi gönderilir.  string yazi = sw.ReadLine(); // ilk satır okunur.  char ayrac = ','; // ayraç için karakter belirtilir  char ayrac2 = '-'; // ayraç2 için karakter belirtilir  while (yazi != null) // yazi boş olana kadar devam edilir  {  string[] satir = yazi.Split(ayrac); // ayraç ile satirdaki veriler ayrılır.  yazi = sw.ReadLine(); // yeni bir satır yazi nesnesine aktrarılır.  for (int i = 2; i < satir.Length; i++) // satirdaki verilerin 3. elemandan başlayarak sona kadar parça parça alınması için döngü kurulur.  {  string[] komsuMesafeAyir = satir[i].Split(ayrac2); //Her Parça ayraç2 ile ayrılarak komşu şehir ve mesafe bir birinden ayrılır  sehirManager.komsuEkle(satir[1], komsuMesafeAyir[0], Int32.Parse(komsuMesafeAyir[1]));  /\*  Komşu ekleme işlemi burada yapılır.  Satir[1] → Ana şehrin plakası  komsuMesafeAyir[0] → komşu şehrin adı  komsuMesafeAyir[0] → komşu şehrin mesafesi  \*/    }  }  sw.Close(); // StreamReader kapatılır.  fs.Close(); // FileStream kapatılır  }  public void kordinatEkle(SehirManager sehirManager)  {  string dosya\_yolu = Dosya.kordinatDosyaYolu; // Kordinat.txt yolu belirtilir.  FileStream fs = new FileStream(dosya\_yolu, FileMode.Open, FileAccess.Read); //FileStream nenesioluşturulur.  StreamReader sw = new StreamReader(fs); //StreamReader türünde sw tanımlanır ve fs parametresi gönderilir.  string yazi = sw.ReadLine(); // ilk satır okunur.  char ayrac = ','; // ayraç için karakter belirtilir  while (yazi != null) // yazi boş olana kadar devam edilir  {  string[] satir = yazi.Split(ayrac); // ayraç ile satirdaki veriler ayrılır.  sehirManager.kordinatEkle(Int32.Parse(satir[0]), Int32.Parse(satir[1]), Int32.Parse(satir[2]));  /\*  satir[0] → Şehrin Plakası  satir[0] → Şehrin X Kordinatı  satir[0] → Şehrin Y Kordinatı  \*/  yazi = sw.ReadLine(); // Yazıya yeni satır dahil edilir.    }  sw.Close(); // StreamReader kapatılır.  fs.Close(); // FileStream kapatılır  }  public void ciktiDosyasiOlustur(SehirManager sehirManager, List<Int32> arananListesiIndis,List<Int32> yol0, List<Int32> yol1, List<Int32> yol2, List<Int32> yol3, List<Int32> yol4, List<Int32> tumMesafeler)  {  string dosya\_yolu = Dosya.ciktiDosyaYolu; // Çıktı dosyasının yolu belirtiliyor  FileStream fs = new FileStream(dosya\_yolu, FileMode.Append, FileAccess.Write); // FS oluşturuluyor.  StreamWriter sw = new StreamWriter(fs); // SW oluşturuluyor  sw.WriteLine("Gidilmesi Gereken Şehirler"); // Dosyaya gidilmesi gereken şehirler yazılıyor.  sw.Write("Kocaeli(41)"); // En başa Kocaeli Ekleniyor.  for (int i = 0; i < arananListesiIndis.Count; i++) // Diğer şehirler listeye ekleniyor.  {  sw.Write(" - " + sehirManager.GetSehir(arananListesiIndis[i] + 1).sehirAdi.ToString()+"("+ sehirManager.GetSehir(arananListesiIndis[i] + 1).plaka.ToString() + ")");  }  sw.Write("- Kocaeli(41)"); // Kocaeli sona ekleniyor.  sw.WriteLine("\nEn Kısa Yollar"); // En kısa yollar yazısı dosyaya yazılıyor.  // Yol0 > 0 Yol0 a ait işlemler çıktı dosyasına yazılıyor  //Yol1 > 0 Yol1 a ait işlemler çıktı dosyasına yazılıyor  //Yol2 > 0 Yol2 a ait işlemler çıktı dosyasına yazılıyor  //Yol3 > 0 Yol3 a ait işlemler çıktı dosyasına yazılıyor  //Yol4 > 0 Yol4 a ait işlemler çıktı dosyasına yazılıyor } } |

## KomşulukMatrisiManager

|  |
| --- |
| class KomsulukMatrisiManager {  public int[,] matris = new int[81, 81]; // 81 , 81 dizi oluşturduk  public KomsulukMatrisiManager()  {  for(int i = 0; i< 81; i++) // Tüm elemanlarına 0 değerini atadır.  for(int l = 0; l<81;l++)  matris[i, l] = 0;  }  public int[,] KomsulukMatrisi(SehirManager sehirManager) {  DSehir iter = sehirManager.ilk; //Dsehir tipinde bir gezici oluşturduk.  int main = 0; // Ana Şehir: başlangıç 0 dedik; (Plaka anlamı 1)  while(iter !=null) // gezici null olana kadar devam ettik. {  for(int i = 0; i < iter.komsuSayisi;i++) // gezicinin bulunduğu şehirdeki komşu sayısı kadar döngü döndürdük. {  matris[main, iter.komsular[i].plaka - 1] = iter.komsuMesafe[i];  /\*  main → Ana Şehir indisi  iter.komsular[i].plaka – 1 → Komşu şehirin indisi  \*/    main++; // Ana şehiri bir ileri götürdük. (Plakayı 1 arttırdık.)  iter = iter.ileri; // geziciyi bir ileri götürdük.  }  return matris; // Oluşan matrisi geri döndürdük.  }  } |

## SehirManager

|  |
| --- |
| class SehirManager  {  public DSehir ilk; //Dsehir tipinde ilk nesnesini oluşturduk.  public DSehir son; //Dsehir tipinde son nesnesini oluşturduk.  public SehirManager() // Constructor {  ilk = null; // ilk nesnesini null  son = ilk; // son nesnesini ilk eşitledik  }  public bool sehirEkle(int plaka,string sehirAdi) //Şehir ekleme işlemi  {  try{  DSehir node = new DSehir(plaka, sehirAdi); //Dsehir tipinde bir node oluşturduk  if (ilk == null) // ilk boş ise  {  ilk = node; // ilk node eşit  ilk.ileri = null; //ilk ilerisi null eşit  son = ilk; // son ilke eşit  }  else // ilk boş değil ise  {  son.ileri = node; // son ileri node eşit  node.ileri = null; // node ileri null eşit  son = node; // son node eşit  }  return true; // bool tipinde geri dönüş kayıt olumlu  }  catch  {  return false; // bool tipinde geri dönüş kayıt hatalı  }  }  public List<DSehir> sehirListesi() // Şehir Listesi  {  List<DSehir> sehirList = new List<DSehir>(); // Dsehir tipinde bir liste oluşturuldu  DSehir iter; // gezici oluşturuldu.  iter = ilk; // gezici ilke eşitlendi  while(iter != null) // gezici null olana kadar devam etti.  {  sehirList.Add(iter); // listeye şehir eklendi  iter = iter.ileri; // gezici 1 ileri gitti  }    return sehirList; // liste tipinde geri dönüş  }  public bool komsuEkle(string sahipSehir,string komsuSehir,int maliyet) //Komşu ekleme  {  DSehir sahipNode, komsuNode; // Ana şehir ve komşu şehir için node oluşturuldu  sahipNode = ilk; // ana şehire ilk node verildi.  komsuNode = ilk; // komşu şehre ilk node verildi.  while(sahipNode != null) // anaşehir null olana kadar devam etti.  {  if(sahipNode.sehirAdi == sahipSehir) // ana şehir node sindeki şehir adı metthoda gelen şehir adıyla aynı ise  break; // döngüden çık.  sahipNode = sahipNode.ileri; // değilse bir ileri git.  }  for(int i = 0; i<sahipNode.komsuSayisi;i++) // SahipNode komşu sayısı kadar döngü dön  {  if (sahipNode.komsular[i].sehirAdi == komsuSehir) // sahipnode daki komşular yeni gelen komşuya eşit ise  return false; // bool tipinde geri dönüş. Bu şehir komşu olarak eklenmiş.  }  while (komsuNode != null) // Komşunode null olana kadar döngüyü dön  {  if (komsuNode.sehirAdi == komsuSehir) //komşu node şehir adı methoda gelen şehir adına eşit ise  break; // döngüden çık  komsuNode = komsuNode.ileri; // komşu node bir ileri götür.  }  sahipNode.komsular[sahipNode.komsuSayisi] = komsuNode;  sahipNode.komsuMesafe[sahipNode.komsuSayisi] = maliyet;  sahipNode.komsuSayisi += 1;  /\*  Ana şehrin komşular arrayine güncel komşu sayısı indisine komşu node ekle. (Params. Geldi.)  Ana şehrin komşularMesafe arrayine güncel komşu sayısı indisine maliyet ekle (Params. Geldi.)  Ana şehrin komşu sayısını 1 arttır.  \*/  komsuNode.komsular[komsuNode.komsuSayisi] = sahipNode;  komsuNode.komsuMesafe[komsuNode.komsuSayisi] = maliyet;  komsuNode.komsuSayisi += 1;  /\*  Komşu şehrin komşular arrayine güncel sayısı indisine ana şehri ekle ekle. (Params. Geldi.)  Komşu şehrin komşularMesafe arrayine güncel komşu sayısı indisine maliyet ekle (Params. Geldi.)  Komşu şehrin komşu sayısını 1 arttır.  \*/  return true;  }  public DSehir GetSehir(string sehirAdi) // İsim ile şehir getir.  {  DSehir sehirBilgileri = ilk; // sehirBilgilerini ilk eşitle  while(sehirBilgileri != null) // Şehir bilgileri null olana kadar devam et.  {  if (sehirBilgileri.sehirAdi == sehirAdi) // sehirBilgilerinin sehirAdı gelen sehirAdına eşit ise  break; // döngüden çık  sehirBilgileri = sehirBilgileri.ileri; // ileri git.  }  return sehirBilgileri; // şehir bilgilerini geri döndür.  }  public DSehir GetSehir(int plaka) //Plaka ile şehir bulma  {  DSehir sehirBilgileri = ilk; // sehirBilgilerini ilk eşitle  while (sehirBilgileri != null) // Şehir bilgileri null olana kadar devam et.  {  if (sehirBilgileri.plaka == plaka) // sehirBilgilerinin plaka gelen plaka eşit ise  break; // döngüden çık  sehirBilgileri = sehirBilgileri.ileri; // ileri git.  }  return sehirBilgileri; // ileri git.  }  public void kordinatEkle(int plaka,int x,int y) // Kordinat Ekle  {  DSehir sehirBilgileri = ilk; //sehirBilgileri ni ilk eşitle  while(sehirBilgileri != null) //SehirBilgileri null olana kadar dön  {  if(sehirBilgileri.plaka == plaka) // ŞehirBilgileri Plakaya eşit ise  break; // Döngüden çık.  sehirBilgileri = sehirBilgileri.ileri; // ileri git.  }  sehirBilgileri.kordinatX = x; // Param. Olarak gelen x değerini sehirBilgilerine aktar.  sehirBilgileri.kordinatY = y; // Param. Olarak gelen y değerini sehirBilgilerine aktar.  }  } |

## YolBulucu

|  |
| --- |
| public class YolBulucu {  public static readonly int NO\_PARENT = -1; //Komşu yoksa -1  public List<Int32> kenarlar = new List<int>(); // kenarlar listesi oluştur.  public List<Int32> arananSehirMesafeleri = new List<int>(); //arananSehirMesafeleri listesi oluştur.  public int enKisaMesafe = 0; //enKisaMesafe 0 olarak oluştur.  public int toplamMinMesafe = 0; //toplamMinMesafe 0 olarak oluştur.  public List<Int32> tamYol = new List<int>(); // yol güzergahları için tamYol listesi oluştur.  public bool kont= true; // kontr true olarak oluştur.  public YolBulucu() {  kenarlar.Clear(); //Kenarlar listesini temizle  arananSehirMesafeleri.Clear(); //arananSehirMesafeleri listesini temizle  enKisaMesafe = 0; // enKisaMesafe 0 eşitle  toplamMinMesafe = 0; // toplamMinMesafe 0 eşitle  tamYol.Clear(); // tamYol listesini temizle  kont = true; // kont true eşitle  }  public void topluDijikstra(int[,] komsulukMatrisi, int baslangicKenari, List<Int32> bitisKenarlari,int status) //Tam bir listenin dijikstra yapılması. {  int minMesafe = Int32.MaxValue; //minMesafeyi en büyük int sayıya eşitle  int minMesafeIndis = Int32.MaxValue; //minMesafeIndis en büyük int sayıya eşitle  for (int i = 0; i < bitisKenarlari.Count; i++) //bitisKenarları sayısı kadar dön. {  dijkstra(komsulukMatrisi, baslangicKenari, bitisKenarlari[i],status); // dijikstra methoduna git.  if (arananSehirMesafeleri[i] < minMesafe) // arananSehirMesafeleri[i] < minMesafe ise {  minMesafe = arananSehirMesafeleri[i]; // minMesafe yi arananSehirMesafeleri[i] yap  minMesafeIndis = i; // minMesafeIndis I ye eşitle  }  }  if (bitisKenarlari.Count == 0) // bitisKenarı 0 eşit ise  return; // döngüden çık.  else { // değil ise  kenarlar.Clear(); // kenarlar listesini temizle  dijkstra(komsulukMatrisi, baslangicKenari, bitisKenarlari[minMesafeIndis], status); // dijikstra yap  for (int i = 1; i < kenarlar.Count; i++) // kenarlar listesi kadar dön  tamYol.Add(kenarlar[i]); //tamYol listesine kenarı ekle  toplamMinMesafe += minMesafe; // toplamMinMesafe nin üzerine minMesafeyi ekle  baslangicKenari = bitisKenarlari[minMesafeIndis]; //baslangicKenarini eşitle  bitisKenarlari.RemoveAt(minMesafeIndis); //bitisKenarlarından minMesafeIndis ini çıkar  arananSehirMesafeleri.Clear(); // arananSehirListesini temizle  if(bitisKenarlari.Count == 0 && kont == true) // bitisKenarı 0 ve kont true ise  {  bitisKenarlari.Add(40); bitis kenarına kocaeli ekle  kont = false; // kont false yap  }  topluDijikstra(komsulukMatrisi, baslangicKenari, bitisKenarlari,status); // topluDijikstra methoduna tekrar git (recursive)  } }  public void dijkstra(int[,] komsulukMatrisi,int baslangicKenari,int bitisKenari, int status) //Dijikstra {  int nKenar = komsulukMatrisi.GetLength(0); //nKenar oluştur ve komşuluk matrisinin 2. kısmının uzunluğunu al  int[] enKisaMesafeler = new int[nKenar]; //enKisaMesafeler int tipinde oluştur.  bool[] eklenenler = new bool[nKenar]; // eklenenleri tutmak için bool tipinde array oluştur  for (int sehirIndex = 0; sehirIndex < nKenar;  sehirIndex++) // sehirIndex < nKenar olana kadar dön  {  enKisaMesafeler[sehirIndex] = int.MaxValue; // enKisaMesafeler En büyük int yap  eklenenler[sehirIndex] = false; // eklenenler false yap  }  enKisaMesafeler[baslangicKenari] = 0; //enKısaMesafeler 0 yap  int[] parents = new int[nKenar]; //komşular arrayi oluştur  parents[baslangicKenari] = NO\_PARENT; // komşular arrayinde başlangıç kenarını -1 yap.  for (int i = 1; i < nKenar; i++) // 1 den başlayıp nkenar kadar dön {  int enYakinKenar = -1;  int EnKisaMesafe = int.MaxValue;  for (int sehirIndex = 0;  sehirIndex < nKenar; // sehirIndex nKenar olana kadar dön.  sehirIndex++)  {  if (!eklenenler[sehirIndex] &&  enKisaMesafeler[sehirIndex] <  EnKisaMesafe) //eklenenler false ve enKisaMesafeler[sehirIndex] < enKisaMesafe ise  {  enYakinKenar = sehirIndex;  EnKisaMesafe = enKisaMesafeler[sehirIndex];  } }  eklenenler[enYakinKenar] = true; // eklenenler[enYakınKenar] true yap  for (int sehirIndex = 0;  sehirIndex < nKenar;  sehirIndex++) // sehirIndex < nKenar olana kadar dön  {  int kenarMesafesi = komsulukMatrisi[enYakinKenar, sehirIndex];  if (kenarMesafesi > 0 && ((EnKisaMesafe + kenarMesafesi) <  enKisaMesafeler[sehirIndex])) //kenarMesafesi > 0 ve ((EnKisaMesafe + kenarMesafesi) < enKisaMesafeler[sehirIndex]) ise  {  parents[sehirIndex] = enYakinKenar;  enKisaMesafeler[sehirIndex] = EnKisaMesafe +  kenarMesafesi;  } } }  cozumuEkranaYaz(baslangicKenari, enKisaMesafeler, parents,bitisKenari,status);//cozumuEkranaYaz  }  public void cozumuEkranaYaz(int baslangicKenari,  int[] mesafeler,  int[] parents,  int bitisKenari,  int status)  {  int nKenar = mesafeler.Length; //mesafeler uzunlugunu nKenar Ekle  for (int sehirIndex = 0; sehirIndex < nKenar; sehirIndex++) // sehirIndex < nKenar olana kadar dön {  if (sehirIndex != baslangicKenari && sehirIndex == bitisKenari) {  arananSehirMesafeleri.Add(mesafeler[bitisKenari]);  cozumYolunuEkranaYaz(sehirIndex, parents);  } } }  public void cozumYolunuEkranaYaz(int gecerliKenar,int[] parents)  {  if (gecerliKenar == NO\_PARENT)  return;  cozumYolunuEkranaYaz(parents[gecerliKenar], parents);  kenarlar.Add(gecerliKenar);  } } |

# Form Ekranları

## Gezgin-kargo-takip-problemi

|  |
| --- |
| namespace ProLab\_21 {  public partial class Form1 : Form {  //gerekli değikenlerin tanımlanması  //komşuluk matrisi methodunun çağırılması  public Form1() {  InitializeComponent();  DrawArea = new Bitmap(pictureBox1.Size.Width, pictureBox1.Size.Height);  // harita ekranının oluşturulması  // sehir, komşuluk ve kordinat bilgilerinin alınması  // şehir adlarının checklistbox’a aktarılması  // değişkene komşuluk matrisinin eklenmesi  }  private void LabelKonumlandir() {  // Konum işaretci labellerinin konumlarının gizlenmesi  }  // yol0 ,yol1 ,yol2 ,yol3 ,yol4 ,yol5 ,tumToplamMesafe dizilerinin oluşturulması  private void YolBulClick() {  // mesafeListesiForm undaki listbox2 nin silinmesi  // yol0 ,yol1 ,yol2 ,yol3 ,yol4 ,yol5 ,tumToplamMesafe verilerinin içindeki verilerin silinmesi  // haritanın temizlenmesi  // yeni haritanın eklenmesi  //labelKonumlandır methodunun çağrılması  // listbox2 temizlenmesi  // arananListesiIndis temizlenmesi  // ArananListesiOlustusur methodunun çağrılması  // YolBul metoduna arananListesiIndis, komşuluk matrisi ve 0 parametrelerinin gönderilmes  // CheckListbox’da seçilen tüm şehirlerinin seçiminin kaldırılması  // yol0.Count > 0 ve yol1.Count > 0 listbox1’e “Tüm Şehirler” itemi eklenmesi  // yol0.Count > 0 listbox1’e “1. Yol” itemi eklenmesi ve 0 numaralı itemi selected edilmesi  // yol1.Count > 0 listbox1’e “2. Yol” itemi eklenmesi  // yol2.Count > 0 listbox1’e “3. Yol” itemi eklenmesi  // yol3.Count > 0 listbox1’e “4. Yol” itemi eklenmesi  // yol4.Count > 0 listbox1’e “5. Yol” itemi eklenmesi  // EnKisaMesafeyiLabeleYaz methodunun çağrılması  //yol0’dan büyük ise dosyaManager classının ciktiDosyasiOlustur methoduna sehirManager, arananListesiIndis, yol0, yol1, yol2, yol3, yol4, tumToplamMesafe parametrelerinin gönderilmesi  }  private void EnKisaMesafeyiLabeleYaz() {  // minMesafe ye en büyük int değer atanması  //tumMesafelerListesinin eleman sayısı kadar dönen bir döngü oluştur var minMesafe en küçüğü bulacak şekilde dön ve bulunan en küçük mesafeyi label5 e yaz.  }  private void ArananListesiOlustusur()  {  if (checkedListBox1.CheckedItems.Count > 9 ||checkedListBox1.CheckedItems.Count < 1)  { //checkListbox da seçilen elemean sayısı eleman sayısı 9 büyükse veya 1 den küçükse ise  richTextBox2.Text = ""; //richTextBox2’yi temizle  MessageBox.Show("Limit dışı seçim yaptınız.\nSeçilen il sayısı: " + checkedListBox1.CheckedItems.Count, "Maksimum Sınır Aşıldı", MessageBoxButtons.OK, MessageBoxIcon.Error); //Uyarı mesajı ver  }  else //checkListbox da seçilen elemean sayısı eleman sayısı 9 küçükse veya 1 den büyükse  {  listBox2.Items.Add("Kocaeli"); //listBox2 ye Kocaeli’yi ekle  mesafeListesiForm.listBox2.Items.Add("Kocaeli"); //MesafeListesi formuna Kocaeliyi Ekle  for (int i = 0; i < checkedListBox1.CheckedItems.Count; i++)  { // checkedListBox1 seçilen eleman sayısı kadar dön.  listBox2.Items.Add(checkedListBox1.CheckedItems[i].ToString()); //Seçilen elemanı listbox2 ye ekle  mesafeListesiForm.listBox2.Items.Add(checkedListBox1.CheckedItems[i].ToString());  arananListesiIndis.Add(sehirManager.GetSehir(checkedListBox1.CheckedItems[i].ToString()).plaka – 1); //arananListesiIndis listesine gelen şehrin adını ve plakasını ekle  }  listBox2.Items.Add("Kocaeli"); //Kocaeli’yi listbox2 ye ekle  mesafeListesiForm.listBox2.Items.Add("Kocaeli"); //Kocaeli’yi mesafeListesiForm listbox2 ye ekle  } }  private void Form1\_Load(object sender, EventArgs e) {  //Komşuluk matrisini richTextBox’e ekle  //Harita resmini pictureBox ekle  //lDizi[i] arrayini label tipinde 120 tane oluştur. Ve resim olarak konum resmini koy }  private void HaritadaCiz(int x,int y,int x1,int y1,int status,int color1,int color2,int color3) {  SolidBrush Normal = new SolidBrush(Color.FromArgb(color1, color2, color3)); // Normal Kalem  SolidBrush Kalin = new SolidBrush(Color.FromArgb(0, 175, 0)); // Kalın Kalem  Pen mypen = new Pen(Kalin, 2); // Kalem oluştur ve Kalin ozellige sahip olsun  // Gelen Paramaetre 0 ise peni Normal Pen yap  // Gelen Paramaetre 1 ise peni Kalin Pen yap  g.DrawLine(mypen, x, y, x1, y1); // Kordinatları çiz }  private void YolBul(List<Int32> arananIndisListesi,int[,] matris,int tur) {  int[,] newMatris = new int[81, 81]; // 81,81 lik yeni bir array oluştur  //kMatris deki verileri newMatris’e aktar  List<Int32> arananIndisListesiCopy = new List<int>(); //arananIndis bir kopyasını oluştur ve verileri ona kopyala  if (tur < 5) // tur sayısı 5’den küçükse {  YolBulucu t = new YolBulucu(); //yolBulucu tipinde bir nesne oluştur.  t.topluDijikstra(newMatris, 40, arananIndisListesiCopy, 0); // bu nesnenin topluDijikstrasını verileri gönder  richTextBox2.Text += sehirManager.GetSehir(41).sehirAdi +"("+ sehirManager.GetSehir(41).plaka.ToString()+") "; // Bulunan yolları richTextbox’e ekle  for (int i = 0; i < t.tamYol.Count; i++) // tamyol uzunluğu kadar dön {  //tur sayısı 0 ise yol0 verileri ekle  //tur sayısı 1 ise yol1 verileri ekle  //tur sayısı 2 ise yol2 verileri ekle  //tur sayısı 3 ise yol3 verileri ekle  //tur sayısı 4 ise yol4 verileri ekle  // Dsehir tipinde oncekiSehirBilgisi değişkenini oluştur.  // I == 0 ise oncekiSehirBilgisine Kocaeliyi ekle  // I != 0 ise tamYol verisinden gelen indisi sehir olarak bul ve oncekiSehirBilgisine ekle  // richTextBox verileri yaz  }  //Bulunan yolla ilgili mesafe değerini richTextBox ekle  // tumToplamMesafe listesine bulunan mesafeyi ekle  //tamYol uzunluğu 5 >= ise  {  BurayaDon: //goto oluştur.  Random random = new Random(); //Random tipinde bir nesne oluştur  //sayi tipinde bir int değişken oluştur ve random aralığı tamYol listesinin uzunluğı kadar olsun  for(int i = 0; i< arananListesiIndis.Count;i++) // arananListesiIndis uzunluğu kadar dön  //tamYol[sayi] ile arananListesi[i] birbirlerine eşit ise goto ya git  //newMatris de o alanı 0 ile değiştir  //yolBul methoduna git params (arananListesiIndis, newMatris, tur + 1)  } } }  private void YolCizClick() {  //Mesafe listesindeki listboxları ve labelleri temizle  //tabPage5 sayfasına direk git.  //haritayı temizle ve yeniden oluştur.  // labelKonumlandir metodunu çağır  // listBox1 de seçilen bir item yok ise hata mesajı ver  // varsa {  int tempIndex = 0; //tempIndex değişkeni oluştur  List<Int32> temp = new List<int>(); // temp listesi oluştur  //Listbox1 seçilen item = “1Yol” ise temp’e yol0’ı eşitle mesafelerList form label5’ine 1.yol yaz ve tempIndex 0 yap  //Listbox2 seçilen item = “2.Yol” ise temp’e yol0’ı eşitle mesafelerList form label5’ine 2.yol yaz ve tempIndex 1 yap  //Listbox1 seçilen item = “3.Yol” ise temp’e yol0’ı eşitle mesafelerList form label5’ine 3.yol yaz ve tempIndex 2 yap  //Listbox1 seçilen item = “4.Yol” ise temp’e yol0’ı eşitle mesafelerList form label5’ine 4.yol yaz ve tempIndex 3 yap    //Listbox1 seçilen item = “5.Yol” ise temp’e yol0’ı eşitle mesafelerList form label5’ine 5.yol yaz ve tempIndex 4 yap  //Listbox1 seçilen item = “Tüm Yollar” ise TumYollarCiz metodunu çağır.  //Listbox1 hiçbiri değişse hata mesajı var  //Listbox1 seçilen item != “Tüm Yollar” ise {  //temp uzunluğu kadar dön {  //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // mesafeListesiForm’un listbox’ina şehir adlarını ekle  //HaritadaCiz metoduna git params(OncekiSehirBilgisi.kordinatX, OncekiSehirBilgisi.kordinatY, sehirBilgisi.kordinatX, sehirBilgisi.kordinatY,0, 29, 53, 87)  //lDizi[i] konumunu sehirin konumu yap. }  //mesafeListesi nin label2 sine mesafeyi yaz ve mesafeListesi formunu ShowDialog tipinde aç.  } } }  private void TumYollarCiz()  {  //yol0 uzunluğu kadar dön  DSehir OncekiSehirBilgisi; //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // tumToplamMesafe[0]== minMesafe ise kalın çiz ve labelleri konumlandır.  // tumToplamMesafe[0]!= minMesafe ise normal çiz  //yol1 uzunluğu kadar dön  DSehir OncekiSehirBilgisi; //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // tumToplamMesafe[0]== minMesafe ise kalın çiz ve labelleri konumlandır.  // tumToplamMesafe[0]!= minMesafe ise normal çiz  //yol2 uzunluğu kadar dön  DSehir OncekiSehirBilgisi; //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // tumToplamMesafe[0]== minMesafe ise kalın çiz ve labelleri konumlandır.  // tumToplamMesafe[0]!= minMesafe ise normal çiz  //yol3 uzunluğu kadar dön  DSehir OncekiSehirBilgisi; //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // tumToplamMesafe[0]== minMesafe ise kalın çiz ve labelleri konumlandır.  // tumToplamMesafe[0]!= minMesafe ise normal çiz  //yol4 uzunluğu kadar dön  DSehir OncekiSehirBilgisi; //OncekiSehir bilgisi değişkeni oluştur  //i = 0 ise OncekiSehirBilgisi’ne Kocaeliyi Ekle  //i != 0 ise OncekiSehirBilgisi’ne temp deki datayı ekle  // tumToplamMesafe[0]== minMesafe ise kalın çiz ve labelleri konumlandır.  // tumToplamMesafe[0]!= minMesafe ise normal çiz    //enKisa mesayi mesafeListesiForm’ye yaz en kisa yolu mesafeListesiForm listbox1 ekle gidilecek yolu mesafeListesiForm listbox2 ekle  // mesafeListesiForm showDialog tipinde aç }  private void YolBulBtn\_Click(object sender, EventArgs e) { YolBulClick(); }  private void YolCizBtn\_Click(object sender, EventArgs e) { YolCizClick(); } } } |

## yol-bilgileri

|  |
| --- |
| //Bu form üzerinde kod yazılmadı |