PROGRAMLAMA LABORATUVARI 1

1. PROJE

Murat Karakurt

Ahmet Burhan Bulut

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

Özet

Bu doküman Programlama Laboratuvarı 1 dersi 1. Projesi için çözümü açıklamaya yönelik oluşturulmuştur. Dokümanda projenin tanımı, çözüme yönelik yapılan araştırmalar, kullanılan yöntemler, proje hazırlanırken kullanılan geliştirme ortamı ve kod bilgisi gibi programın oluşumunu açıklayan başlıklara yer verilmiştir. Doküman sonunda proje hazırlanırken kullanılan kaynaklar bulunmaktadır.

1. Proje Tanımı

Projede istenilen C programlama dili kullanılarak belirli bir klasörün içinden ".txt veya .csv" uzantılı dosya içerisinden girilen koordinatları dosyadan okuyarak koordinat düzleminde işaretler, bu noktalardan geçen minimum çevreleyen çemberi çizer, çemberin merkezini ve yarıçapını ekrana çıktı olarak verir. (Örnek Şekil 1 de verildiği gibi) Aynı zamanda bu noktalardan geçen B-Spline eğrilerini çizer.

Çıktı:

Merkez = {0,5, 0,5}

Yarıçap = 0,7071

Şekil 1. Örnek Çıktı

2. Giriş

Öncelikle belirlenen bir dosyaya koordinatlar girilir. Bu girdiler bir char dizisine aktarılır ardından dizideki tüm elemanların sayı olup olmadıkları kontrol edilir. Elemanlar sayı ise pozitif veya negatiflik durumları kontrol edilir. Dizideki elemanlar x ve y olarak iki farklı diziye aktarılır. Dizideki elemanlar x değerlerine göre sıralanır. Dizideki elemanlar find_farthermost fonksiyonuna gönderilir bu fonksiyonda en uzak iki nokta bulunur. Noktaların merkez noktası bulunması için circle_center fonksiyonuna gönderilir. disarda_mi fonksiyonu ile dışarıda nokta kalıp kalmadığı kontrol edilir. Dışarıda nokta kalmış ise noktaların ağırlık merkezi çemberin merkezi olarak kabul edilir. Merkez kabul edilen noktaları ve diğer girilen tüm noktalar new_center fonksiyonuna gönderilir. Fonksiyonda tüm üçlü kombinasyonlar denenerek en küçük çevreleyen çemberin merkezi ve yarıçapı bulunur. initwindow fonksiyonu kullanılarak grafik çizimi için 800*800 boyutunda yeni bir pencere açılır. for döngülerinin içerisinde sayılar bir char dizisine atanıp Sayilar fonksiyonuna gönderilir. Sayilar fonksiyonu koordinat eksenini ve sayıları açılan pencereye yazar. drawLine kullanılarak koordinat düzleminin x ve y eksenleri yerleştirilir. Noktalar, merkez ve yarıçap ölçeklendirme formülüne göre piksel cinsine dönüştürülür. draw_spline fonksiyonu Catmull-Rom Teoremi kullanılarak oluşturulmuştur. fillellipse fonksiyonu kullanılarak noktalar işaretlenir. circle fonksiyonu kullanılarak minimum çevreleyen çember çizdirilir.

3. KABA KOD

int main()

1.İlk olarak okunacak dosya açılır.
2. Dosyadaki tüm karakterler sayı olup olamamasına göre ayrıştırılır.
3.Ayrıştırılan noktalar x ve y koordinatlarına göre bir diziye aktarılır.
4.Diziler x değerlerine göre küçükten büyüğe göre sıralanır.
5.Bu noktalar find_farthermost fonksiyonuna gönderilerek en uzak iki nokta buldurulur.
6. circle_center fonksiyonu ile çemberin merkez noktası ve yarı çapı bulunur.
7.Dışarda bir nokta kalıp kalmadığını öğrenmek için ctrl fonksiyonu kullanılarak kontrol edilir.
8.Dışarıda nokta var ise new_center fonksiyonu ile yeni merkez noktası bulunur.
9.initwindow fonksiyonu ile yeni pencere açılır.
10.Pencere üzerine koordinat düzlemi ve x,y noktaları yazdırılır.
11.Noktalar ve merkez piksel cinsine dönüştürülür.
12.Çember ve B Spline eğrisi çizdilirir.

void double_distance()

1.Gönderilen noktalar arasındaki mesafeyi bulur.

int disarda_mi()

1.Gönderilen noktanın merkeze uzaklığının yarıçaptan uzak olup olmadığını distance() fonksiyonu yardımı ile bulur.
2.Yarıçaptan büyük ise 0 değil ise 1 döndürür.

int new_center()

1.Tüm noktaların üçlü kombinasyonları alınır.
2.Üçlü kombinasyonlara göre merkez ve yarıçapı öteleyerek distance() ve in_inside() fonksiyonlarının yardımı ile minimum çevreleyen çemberin merkez noktasını bulur

void drawLine()

1. Bu fonksiyon aldığı değerlere göre koordinat düzleminin x ve y eksenlerini çizdirir.

void spline()

1. Alınan katsayıya göre 4 nokta seçer.
2. Her noktanın eğriye olan etkisini hesaplar.
3. Bu etkilere ve noktaların koordinatlarına göre bir nokta oluşturur.
4. Kullanıcın verdiği diziye bu noktanın değerlerini atar.

void draw_spline()

1. Verilen noktalar dizisinin değerlerini arrayx diye yeni bir integer dizisine kopyalar.
2. array_x2 dizisinin başına ve sonuna x ekseninin uçlarını ekler.
3. array_x2 elemanını x değişkenine atar.
4. array_y2 elemanını y değişkenine atar.
5. (double k = 0.00 ;k < uzunluk(array_x2)-1; k+= 0.01) için:
*. array adlı 2 elemanlı bir double dizisi oluşturur.
**. array dizisini ve katsayıyı spline() fonksiyonuna gönderir.
***. "Graphics.h" içerisindeki line() fonksiyonunu kullanarak x,y ve array[0],array[1] arasını çizdirir.
****. x değişkeninin değerini array[0] ile değiştirir.
*****. Y değişkeninin değerini array[1] ile değiştirir.

void sayilar()

1.Gönderilen noktaları koordinat düzlemine yazar.

int control()

1.Gönderilen noktanın oluşturulan çemberin dışarısında mı değil mi kontrolünü yapar.
2.Nokta dışarıda ise 1 değil ise 0 değeri döndürür.

double radius()

1.Gönderilen noktalar arasını çap kabul ederek çemberin yarıçapını döndürür.

void circle_center2()

1.Bütün x ve y noktalarının ağırlık merkezini bulup bu noktalayı merkez kabul eder.
2.Merkez noktalarına göre yarıçap bulur.
3.Bulduğu merkez ve yarıçapı fonksiyona gönderilen diziye atama yapar.

void circle_center()

1.Gönderilen noktaların orta noktasını merkez kabul eder.
2.Kabul edilen merkez noktasına göre yarıçap bulur .
3.Bulunan merkez ve yarıçapı fonksiyona gönderilen diziye atama yapar.

void find_farthermost()

1.Fonksiyona gönderilen koordinatlardan en uzak iki noktayı bulur.
2.Bu noktaların indexlerini fonksiyona gönderilen diziye atar.

4.AKIŞ DİYAGRAMI

Akış diyagramı main (tıklayınız).

Akış diyagramı fonksiyonlar(tıklayınız).

5. KARMAŞIKLIK ANALİZİ

```
void disarda_mi(double center_x,double center_y,int array_x[],int array_y[],int size,double radius,double array[])
{
    for(double i=-1; i<1; i+=0.01)
    {
        for(double j=-1; j<1; j+=0.01)
        {
            for(double z=-1; z<1; z+=0.01)
            {
                double x = center_x+i;
                double y =center_y+j;
                double z = radius+z;
                int gecerli_mi=1;
                for (int t=0; t<size; t++)
                {
                    if(in_inside(x,y,array_x[t],array_y[t],z) == 0)
                    {
                        gecerli_mi = 0;
                        break;
                    }
                }
                if(gecerli_mi == 1 && array[2]>z)
                {
                    array[0] = x;
                    array[1] = y;
                    array[2] = z;
                }
            }
        }
    }
```

8.000.000 nokta denenir. O(1)

1

O(n)

1

8000000*(n+2)=O(n)

```
void draw_spline(int array_x[],int array_y[],int uzunluk)
{
    int array_x2[uzunluk+2];
    int array_y2[uzunluk+2];

    array_x2[0] = 0;
    array_y2[0] = 400;
    for (int k=1; k<uzunluk+1; k++)
    {
        array_x2[k] = array_x[k-1];
        array_y2[k] = array_y[k-1];
    }

    array_x2[uzunluk+1] = 800;
    array_y2[uzunluk+1] = 400;

    double x = array_x2[1];
    double y = array_y2[1];
    array_x[uzunluk] = 400;
    array_y[uzunluk] = 400;
    for (double t = 0.00; t < double(uzunluk+2) - 3.0 ; t+= 0.001)
    {
        double array[2];
        spline(t, array, array_x2,array_y2);
        line(x, y, array[0], array[1]);
        x = array[0];
        y = array[1];
    }
}
```

1

n

(n-1)*1000+n=O(n)

1

(n-1)*1000

```
int control(double center_x,double center_y,int array_x[],int array_y[],int size,double radius)
{
    int temp =0;
    for (int i=0; i<size; i++)
    {
        double dist1 = distance(center_x,center_y,array_x[i],array_y[i]);

        if(dist1>radius)
        {
            return 1;
        }
    }
    return 0;
}
```

n

O(n)

```
void circle_center2 (int array_x[],int array_y[],int size,double array[3])
{
    int x_toplam=0,y_toplam =0;
    double temp =0,rad=0;
    for (int i=0; i<size; i++)
    {
        x_toplam += array_x[i];
        y_toplam += array_y[i];
    }
    double center_x = x_toplam/size;
    double center_y = y_toplam/size;

    for (int i=0; i<size; i++)
    {
        double dist = distance(center_x,center_y,double(array_x[i]),double(array_y[i]));
        if (dist >rad)
            rad = dist;
    }

    array[0]=center_x;
    array[1]=center_y;
    array[2]=rad;
}
```

n+n=O(n)

n

1

n

```
for (int k=0; k<i/2; k++)
{
    fillellipse(400 + (array_x[k]*20),400 - (array_y[k] *20),5,5);
}
for (int k=0; k<i/2; k++)
{
    array_x[k] = 400 + (array_x[k]*20);
    array_y[k] = 400 - (array_y[k] *20);
}
```

n+n=O(n)

```
void find_farthermost(int array1[],int array2[],int size,int array3[2])
{
    int temp2 =0,temp3 =0,i,j;
    double temp=0.0,dist =0.0;

    for (i=0; i<size-1; i++)
    {
        for (j=i+1; j<size; j++)
        {
            dist = distance(array1[i],array2[i],array1[j],array2[j]);
            if (dist>temp)
            {
                temp = dist;
                temp2 = i;
                temp3 = j;
            }
        }
    }
    array3[0] = temp2;
    array3[1] = temp3;
}
```

n-1

n-1,n-2,n-3,....0

(n-1)*n/2=O(n^2)

```
for (int k=0; k<j; k++)
{
    if (isdigit(array[k]))
    {
        if (isdigit(array[k]) && isdigit(array[k+1]))
        {
            int toplam = 10*(array[k] - '0') + (array[k+1] - '0');
            array3[i] = toplam;
            if(array[k-1] == '-')
            {
                array3[i] *= -1;
            }
            k++;

        }
        else
        {
            array3[i] = array[k] - '0';
        }
        if(array[k-1] == '-')
        {
            array3[i] *= -1;
        }
        i++;


    }
```

O(n)

```
for (int k=0; k<i/2-1; k++)
{
    for(int l=k+1; l<i/2; l++)
    {
        if(array_x[k]>array_x[l])
        {
            int temp = array_x[k];
            array_x[k] = array_x[l];
            array_x[l] = temp;
            int temp2 =array_y[k];
            array_y[k] = array_y[l];
            array_y[l] = temp2;
        }
    }
}
```

n-1

n-1,n-2,n-3,....,0

1

(n-1)*n/2=O(n^2)

```
for(int a=20; a<400; a+=20)
{
    if (s>=10)
    {
        char a2 = '0' + (s%10);
        char a1 = '0' + (s/10);
        c[0] ='-';
        c[1] = a1;
        c[2] = a2;
        c[3] = '\0';
        s++;
        Sayilar(0,uzunluk/2,WHITE,genislik/2,(uzunluk/2)+a,c);
    }
    else
    {
        char o = '0' + s;
        c[0] ='-';
        c[1] = o;
        c[2] = '\0';
        Sayilar(0,uzunluk/2,WHITE,genislik/2,(uzunluk/2)+a,c);
        s++;
    }
}
```

1

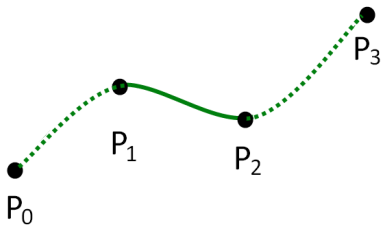
O(n)

Karmaşıklığı en büyük olan find_farthermost fonksiyonudur. Bu yüzden tüm kodun karmaşıklığı bu fonksiyonun karmaşıklığına eşit olup O(n^2)' dir.

6. YÖNTEM

Program C dilini ve “Graphics.h” kütüphanesini kullanmaktadır.
Program CodeBlocks derleyicisi üzerinden yazılmıştır.
Program Minimum çevreleyen çemberi çizdirirken bütün noktaların 3’lü kombinasyonunu hesaplar. Bu kombinasyonlara göre çemberin merkezi ötelerken yarıçapın uzunluğunu değiştirir. Çemberin minimum çevreleyen çember olup olmadığını kontrol eder. Minimum çevreleyen çemberi bulduktan sonra ekrana çizdirir. Ekranda dosyadan alınan koordinatları ve çemberin merkezi de gösterilir.

Program tüm noktalardan geçen eğriyi çizerken Catmull-Rom Teoremi kullanır.



Şekil 2. Catmull-Rom Teoremi

Bu teorem her bir eğriyi çizdirirken 4 nokta alır. Baş ve son noktalar kontrol noktalarıdır. t katsayısı ortadaki noktalar arasında 0-1 aralığında değer alır.
Qi-1 = -t3 + 2t2 – t
Qi = 3t3 – 5t2 + 2
Qi+1 = -3t3 + 4t2 + t
Qi+2 = t3 - t2
Noktaların hepsinin bu t değerine göre çizdirilecek eğriye bir etkisi vardır. Bu etkilere göre eğrinin t noktasındaki koordinatı belirlenir.
X(t) = (Pi-1(x) . Qi-1(t)) . (Pi(x) . Qi(t)) . (Pi(x) . Qi(t)). (Pi(x) . Qi(t)) / 2
Y(t) = (Pi-1(y) . Qi-1(t)) . (Pi(y) . Qi(t)) . (Pi(y) . Qi(t)). (Pi(y) . Qi(t)) / 2

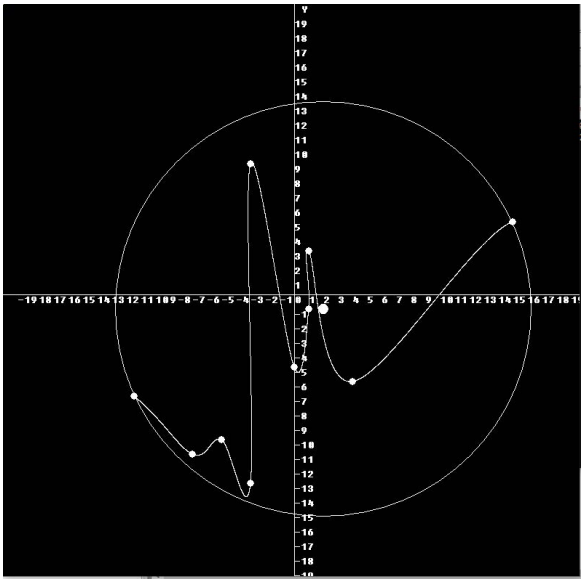
Bu koordinatlara göre eğri graphics.h kütüphanesinin içindeki line fonksiyonu ile çizdirilir.

7.DENEYSEL SONUÇLAR

Noktalar: (-5,-10),(-3,-13),(1,-1),(-3,9),(-7,-11), (15,5),(1,3),(0,-5),(-11,-7),(4,-6)

```
Merkez : {2.000000,-1.000000}
Yaricap: 14.317821
```

Şekil 3.1 Yarıçap ve Merkez Çıktısı



Şekil 3.2 Çizdirilen çember ve eğri.

8. SONUÇ

Program alınan koordinatlara göre en küçük çevreleyen çemberi ve eğriyi sorunsuz çizdirmiştir.

9. KAYNAKÇA

1. https://web.stanford.edu/class/archive/cs/cs106b/cs106b.1126/materials/cppdoc/graphics.html
2. http://www.ttbilgin.com/2018-2019-guz/bilmuh-giris/hafta4/algoritmalar-ve-akis-diyagramlari.pdf
3. https://www.youtube.com/watch?v=9_aJGUTePYo
4. http://web.cse.ohio-state.edu/~wang.3602/courses/cse581-2012-spring/2D.pdf
5. https://www.techcrashcourse.com/2015/08/c-program-draw-hut-color-screen-graphics.html