**PROGRAMLAMA LABORATUVARI 1**

**1. PROJE**

Muhammet Ali BAKINÇ

Abdulhalik SARIÇİÇEK

Bilgisayar Mühendisliği Bölümü

Kocaeli Üniversitesi

**Özet**

Bu doküman Programlama Laboratuvarı 1 dersi 1. Projesi için çözümümüzü 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 projemizi hazırlarken kullandığımız kaynaklar ve proje derlenirken dikkat edilmesi gereken hususlar bulunmaktadır.

1. **Proje Tanımı**

Projede bizden istenen, C dili kullanarak belirli bir “.txt” uzantılı dosya içinde bulunan 2 boyutlu nokta verilerini okuyan, aldığı noktaları koordinat düzleminde gösteren ve bu noktaları çevreleyen minimum çember ile yakınından geçen eğriyi çıktı veren grafik uygulamasını yazmaktır.

Dosyalar sabit bir formatta verilmiştir. Noktalar süslü parantezler içerisinde ve noktalar arasını virgül ile ayıracak şekilde yazılmalıdır (ör. {{1,2}, {2,3}}).

Programın çalıştığı klasörde herhangi bir “.txt” uzantılı dosyanın olmaması halinde konsol üzerinde kullanıcıdan isteğe bağlı olarak noktaların girilmesi istenmektedir. Girilen noktalar dosyaya yazılıp kaydedilir.

Alınan nokta verileri minimum çevreleyen çember ve b-spline eğrisi için gerekli hesaplamaların yapılmasının ardından graphics.h kütüphanesi ile çizim yapılması için gerekli ölçülere dönüştürülür. Çizilen koordinat sistemi üzerinde noktalar ve değerleri belirtilir. Yapılan hesaplamalar sonucu çember için bulunan merkez ve yarıçap değerleri de dönüştürülerek ekrana çizdirilir. Son olarak b-spline eğrisinin değerleri de çizdirilerek işlem tamamlanır.

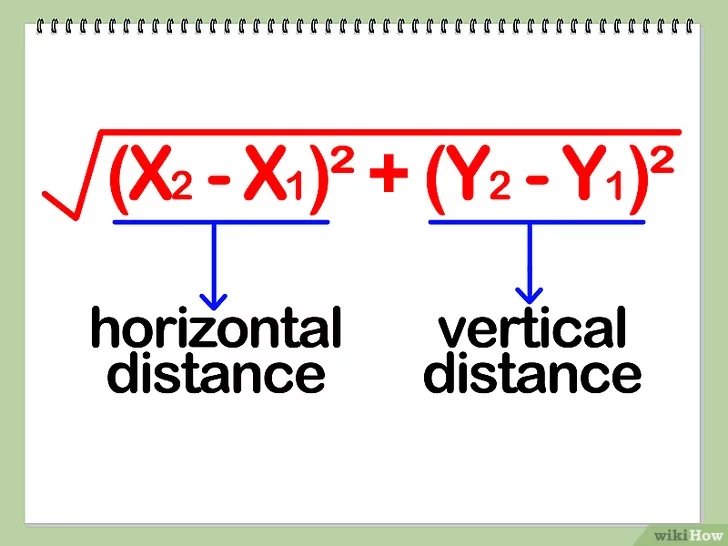
**2. Araştırmalar ve Yöntem**

Projede metin belgesini okuyabilmek için öncelikli olarak programın çalıştığı klasördeki dosyanın bulunabilmesi gerekiyor. Bunun için dosyanın belirlediğimiz bir isim ile kaydedilmiş olması gerekiyor. Böylelikle ismini bildiğimiz dosyayı da rahatlıkla okuyabildik.

Dosya içerisindeki noktaları rahatlıkla okuyabilmek için verilerin belirli bir formatta yazılması gerektiğine karar verdik. Ve bu formata uygun girilen veriyi belirli noktalarından parçalayan bir fonksiyon kullandık. Böylelikle noktaların basamak sayısı veya negatif, pozitif olma durumu fark etmeksizin okunabilmesini sağladık. Okunan değerler öncelikle *char* dizisine alındı. Daha sonra fonksiyon ile parçalandıktan sonra x ve y olarak iki ayrı *integer* dizilere atadık.

Daha sonra Minimum Çevreleyen Çember problemi için internetten algoritma fikirleri araştırmaya başladık. Edindiğimiz fikirler doğrultusunda şu aşamalar üzerinden ilerledik:

1. Çember için öncelikle geçici bir merkez belirlemeye karar verdik. Bunun için tüm noktaların x ve y değerlerinin ayrı ayrı ortalamasını alarak bir merkez belirledik.
2. Bu merkez ile diğer tüm noktalar arasındaki mesafeleri bir döngü ile hesaplayarak en uzak noktayı belirledik. Bunun için iki nokta arası mesafeyi hesaplayan matematik formülünü kullandık.

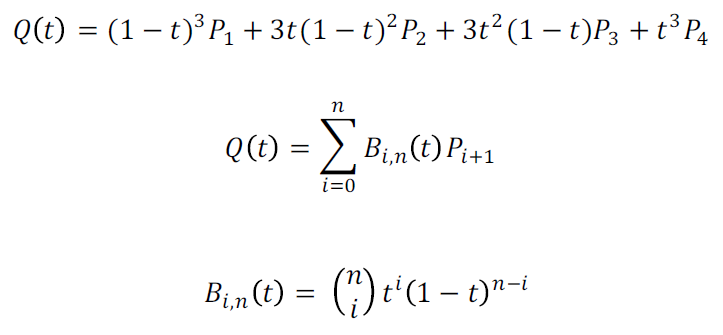


**Şekil 1.** İki Nokta Arası Mesafe Formülü

En uzak noktaya olan uzaklığı yarıçap kabul ederek tüm noktaları içine alan bir çember elde etmiş olduk.

1. Yine araştırmalarımız sonucu çemberi daha da küçültebilmek için merkez noktasının belirlenen en uzak noktaya yaklaştırılması gerektiğini fark ettik. Bunun için de merkez ile en uzak nokta arasındaki uzaklığın eğimini bulduk. Bu eğim ile orantılı olarak merkezin x ve y değerlerini, en uzak noktanın x ve y noktalarına yaklaştırdık. Her küçük yaklaştırma sonrası 2. adımı tekrar uygulayarak ikinci bir noktaya ulaşmaya çalıştık. Merkez ile en uzak nokta arasındaki mesafeyle, yeni bir noktanın merkeze olan uzaklığının eşit olduğu anda döngüyü tamamlayarak üzerinde iki nokta bulunan bir çember elde ettik.
2. Gözlemlerimiz sonucu çemberin üçüncü bir noktaya da ulaşabileceğini fark ettik. Bunun için çember üzerindeki iki nokta arasındaki mesafenin orta noktasını belirlememiz ve merkezi bu noktaya doğru yaklaştırmamız gerekliydi. Çember üzerindeki iki noktanın x ve y değerlerinin ortalamasını alıp bu orta noktayı belirledik. Daha sonra merkez ve bu orta nokta için 3. adımı uyguladık. Çember herhangi üçüncü bir noktaya ulaşması halinde bu döngü tamamlanır. Ulaşmadığı durumda ise merkez bu orta noktaya ulaşana dek yakınlaşır. Ve burada durur. Böylece olabilecek en küçük çember elde edilmiş olur.
3. Son adımda bulduğumuz yarıçap değerini grafiksel değerlere dönüştürüp çemberi çıktı olarak verdik.

Projemizin ikinci bir kısmı olan B-Spline eğrileri için internetten edindiğimiz bir formülü kullandık.



**Şekil 2.** B-Spline Eğrisi Formülü

Bu eğrinin çizilmesi için de şu adımları kullandık:

1. Tüm noktalarımızı koordinat üzerinde soldan sağa olacak şekilde sıraladık.
2. Nokta sayısı kadar “B(t)” değerlerinin toplamı olan “Q(t)” denkleminin sonucunu x ve y değerleri için hesaplayıp noktasal çıktı olarak verdik.
3. “t” değerinin çok küçük değişimiyle döngü içerisinde eğrinin tamamını çizdirmiş olduk.

**3. Geliştirme Ortamı**

Projemizi Windows sistemde, Dev-C++ üzerinde geliştirip GCC kullanarak derledik. C dilini ve çizim yapabilmek için graphics.h kütüphanesini kullandık.

1. **Kod Bilgisi**

**4.1. Akış Şeması**

Son sayfadadır.

**4.2. Algoritma**

Bu kısımda projenin genel algoritmasına açıklık getireceğiz.

Program çalıştığında, olduğu klasördeki “prolab1.txt” isimli dosyayı buluyor. Bulduğu dosyanın içindeki metin parçasını alarak bir *char* dizisine atıyor. Eğer dosya mevcut değilse uyarı veriyor ve konsol üzerinden kullanıcıdan veri girmesini istiyor. Aldığı veriyi klasöre kaydediyor. Alınan veriyi belirlenen karakterlerin ({,}) olduğu noktalardan *strtok* fonksiyonu ile parçalayarak, x ve y değerlerini *atoi* fonksiyonu ile integer dizilere attık. Bu işlemleri farklı bir fonksiyon içinde pointerlar aracılığıyla yaparak düzenli ve okunabilir verimli bir kod yazmaya çalıştık. Fonksiyondan nokta sayısını içeren bir değişkeni de daha sonra kullanmak için return ettik. Grafik çizimine başlamak için bir pencere oluşturduk. Bir fonksiyon ile de koordinat düzlemi çizdirdik. Bu işlemler için *graphics.h* kütüphanesini kullandık. Bu fonksiyonda -13 ile 13 arasında bir koordinat sistemi oluşturduk. Her değer arası 25 birim olacak şekilde döngüler ile noktaları ve değerlerini çıktı olarak verdik. Daha önce *xn* ve *yn* isimli *integer* dizilerde tuttuğumuz noktaları *px* ve *py* olarak farklı integer dizilere atadık. Bu atama sırasında yine döngüler aracılığıyla koordinat değerlerini grafik ölçeğine çevirmiş olduk. Daha sonra yine ayrı bir fonksiyonda dönüştürdüğümüz değerleri grafik üzerinde belirterek, değerlerini yazdırdık. Farklı bir fonksiyonda bütün noktaların x ve y değerlerinin ortalamasını alarak geçici bir merkez noktası belirledik. Bir sonraki aşamada en geniş fonksiyonumuz olan çemberi çizdiren fonksiyonu oluşturduk. Bu fonksiyon kendi içerisinde 3 aşamalı olarak ilerledi. İlk aşamada *for* döngüsü ile noktalar ile geçici merkez arasındaki mesafeyi hesapladık. Ve *if* kontrolü ile en uzak mesafeyi yarıçap olarak kabul ettik. İkinci aşamada merkez ile en uzak noktanın x ve y değerlerinin ortalamalarının oranı ile yarıçapın eğimini belirledik. Bu değerleri *double* olarak tutarak bölüm sonuçlarının doğru çıkmasını sağladık. Eğim değerinin negatif çıkması durumunu engellemek için yine *if* ile negatif değerleri pozitife çevirdik. Bir *while* döngüsü ile ikinci bir noktaya ulaşmaya çalıştık. Döngü ikinci bir nokta belirlenene kadar çalıştırıldı. Döngü içerisinde ilk olarak *if else*‘ler ile ilk noktamız ile merkez noktamızın birbirlerine göre konumlarını değerlendirdik. Bu konumlarına göre merkezin x ve y değerlerini 0.01 gibi küçük değişimlerle ilk noktaya doğru yaklaşmaya çalıştık. Her küçük değişim sonunda yarıçapı yeniden hesapladık. Sonrasında *for* döngüsü ile yeni merkezimiz ile diğer noktalar arasındaki mesafeleri hesapladık. Her hesaplamadan sonra if kontrolü ile yeni yarıçapın noktalardan birine olan uzaklığa eşit olması durumunu kontrol ettik. Böyle bir durum olduğu zaman *break* komutu ile while döngüsünden çıktık. Ve bu noktanın indeksini bir değişkene aldık. 3. Aşamada ise bulunan iki noktanın x ve y değerlerinin ortalaması alınarak, iki nokta arası mesafenin orta noktasını belirledik. Bu noktalar ile merkezin değerlerini kullanarak yeni eğim değeri hesapladık. Kullandığımız *while* döngüsünü bu aşama için de kullandık. Bu döngüde merkez ile bulduğumuz orta noktanın birbirleriyle konumlarını kontrol ettik. Tekrar for döngüsü ile üçüncü bir noktayı aradık. Ancak döngünün hemen sonlanmaması için bulduğumuz ilk iki noktayı *if* ile istisna tuttuk. Yalnız üçüncü bir noktanın bulunamaması durumunu kontrol etmek için for döngüsünden hemen sonra *if* ile merkezimizin orta noktaya ulaşıp ulaşmadığını kontrol ettik. Ulaşılması halinde break ile *while* döngüsünden ayrıldık. Bu kontrol sırasında merkez ile hedef nokta değerleri arasındaki farkın negatif olmaması için *fabs* fonksiyonu ile *double* değerlerin mutlak değerini de aldık. Fonksiyonun son kısmında ulaştığımız merkez ve yarıçap sonuçlarını grafiksel değerlerden, koordinat değerlerine tekrar çevirdik. Merkez noktasını da grafik üzerinde belirtip, yarıçap boyutunda çemberi çizdirdik. Aynı zamanda grafik ekranının üst kısmında ve konsol ekranında da değerleri çıktı olarak verdik. Çember işlemlerinin bitmesinin ardından son fonksiyonda b-spline eğrilerini çizdirdik. Bu fonksiyonda önce iç içe *for* döngüleriyle noktaları büyüklük sırasına uygun olarak yeniden sıraladık. Matematiksel denklemde kullanmak için faktöriyel ve kombinasyon için iki ayrı fonksiyon oluşturduk. Bu fonksiyonları kullanarak denklemin sonucunu hesapladık. Denklemde yer alan terimleri for döngüsü ile nokta sayısı kadar hesapladık ve toplamlarını bir değişkene atadık. İki ayrı toplam sonucunu x ve y değerleri için hesapladıktan sonra piksel nokta olarak çıktı verdik. Tüm bu işlemleri içeren en dış for döngüsünde denklemde yer alan t değerinin çok küçük değişimleri tekrarınca gerçekleştirerek eğrinin çizimini yapmış olduk.

**4.3. Kaba Kod**

1. Dosyayı aç.

2. Eğer *dosya*==NULL ise kullanıcıdan veri iste. Girilen veriyi oku ve dosyaya yaz.

3. Değilse dosyadan oku *inputtxt’e* yaz.

4. *inputtxt*’i atama\_ve\_nokta\_sayisi’na gönder ve sonucu *noktasayisi’na* ata.

5. Pencere aç.

6. *koordinat\_cizdir* fonksiyonunu çağır.

7. *noktalari\_yaz* fonksiyonuna *px, py, xn, yn* ve *noktasayisi’nı* gönder.

8. *ortalama* fonksiyonuna *\*pxort, \*pyort, noktasayisi, xn* ve *yn* gönder.

9. *x1*=175 ve *y1*=700

10. *cemberi\_hesapla\_ve\_ciz* fonksiyonuna *px, py, x1, y1, \*pxort, \*pyort, noktasayisi* gönder.

10. *b\_spline* fonksiyonuna *px, py, xn, yn, noktasayisi* gönder.

11. Programı bitir.

**4.4. Karmaşıklık Hesabı**

Zaman karmaşıklığı hesabını fonksiyonlar üzerinden yaparsak:

* *int faktor(int x){}* için **4n+4**
* *double kom(int x, int y){}* için **12n+20** (faktor fonksiyonu kullanır)
* *void koordinat\_cizdir(int d){}* için **478**
* *void noktalari\_yaz(double px[], double py[], int xn[], int yn[], int noktasayisi){}* için **8n2+131n+3**
* *int atama\_ve\_nokta\_sayisi(char inputtxt[50], int xn[], int yn[]){}* için **8n+7**
* *void ortalama(double &xort, double &yort, int noktasayisi, int xn[], int yn[]){}* için **6n+8**
* *void cemberi\_hesapla\_ve\_ciz (double px[], double py[], float x1, float y1, double &xort, double &yort, int noktasayisi){}* için **32n2+1682n-129543**
* *void b\_spline(double px[], double py[], int xn[], int yn[], int noktasayisi){}* için **120,031n2+320034n+45028**
* Tüm bu fonksiyonların kullanıldığı *main* fonksiyonu için ise **120,071n2+321,861n-83,994**’tür.

Bu sonuçlara bağlı olarak Big O notasyonu **O (n2)**’dir.

**4.5. İstatistik**

Program kodu boşluksuz ve yorumsuz yaklaşık 300 satır tek dosyadan oluşmaktadır. Kod düzenini sağlamak için yaklaşık 75 boş satır kullanılmıştır.

Okuyucuya izlenim oluşturması için yorum satırlarına yer verilmiştir. Kullandığımız kütüphaneler ve ne için kullandığım kabaca aşağıdaki gibidir:

*<stdio.h>*

Çıktı ve girdi almak için, *fopen, scanf, printf, sprintf, fgets* fonksiyonları için

<*stdlib.h*>

*atoi* fonksiyonu için

*<string.h>*

Dosyadan alınan veriyi parçalayan *strtok* fonksiyonu için

*<math.h>*

*pow, sqrt, fabs* gibi fonksiyonlar için

<graphics.h>

Grafik çıktısı yapmak için

**4.6. Programın Derlenmesi**

Programın kaynak kodu tek dosyadan oluşmaktadır. Bu dosyayı Dev-C++, CodeBlocks ya da GCC kullanarak derleyebilirsiniz. Derlerken dikkat edilmesi gereken mesele ise kullandığınız IDE’ye *graphics.h* kütüphanesinin eklenmiş olması gereklidir.

**Kaynakça**

**1.** Introduction to Computer Graphics,

<http://saksagan.ceng.metu.edu.tr/courses/ceng477/files/pdf/week_12.pdf>

**2.** İki nokta arası uzaklık formülü,

<https://webders.net/659/iki-nokta-arasi-uzaklik.html>

**3.** Grafik kütüphanesi fonksiyonları,

<http://eitworld.com/c-graphics/functions/>

**4.** Minimum Çevreleyen Çember Problemi

<https://www.cs.mcgill.ca/~cs507/projects/1998/jacob/problem.html>

