**KOCAELİ ÜNİVERSİTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

**PROGRAMLAMA LAB. I- I. Proje**

Minimum Çevreleyen Çember (Minimum Enclosing Circle)

İlke Can Ersöz-Egehan Demir

190202041@kocaeli.edu.tr 180202040@kocaeli.edu.tr

**ÖZET**

Bu projede bizden kullanıcı tarafından girilen N adet noktanın dosya üzerinden okunarak bu N adet noktayı içeren minimum çevreleyen yarıçaplı daireyi çizdirmemiz ve bu minimum çevreleyen çemberin yarıçapı ve merkezinin hesaplanmasını istenmektedir. Dosya üzerinden alınan tüm noktaları kapsayan ve en küçük çember çizdirilmiştir. Grafik ekranında 2 boyutlu koordinat düzlemi üzerinde minimum kapsayan çember ve bu çemberin merkez koordinatları ve yarıçapı yer alır. Ayrıca girilen N adet noktayı da bu düzlemde işaretlenmiş olarak görebiliriz. Konsol ekranında ise ayrıca bu koordinatlar ve yarıçap belirtilmiştir.

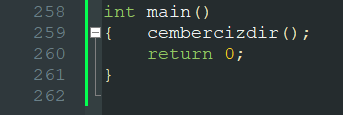
**GİRİŞ**

Proje hazırlanırken C Programlama Dili CodeBlocks geliştirme ortamında kullanıldı. Arayüz tasarımı içinse graphics.h kütüphanesi kullanıldı.

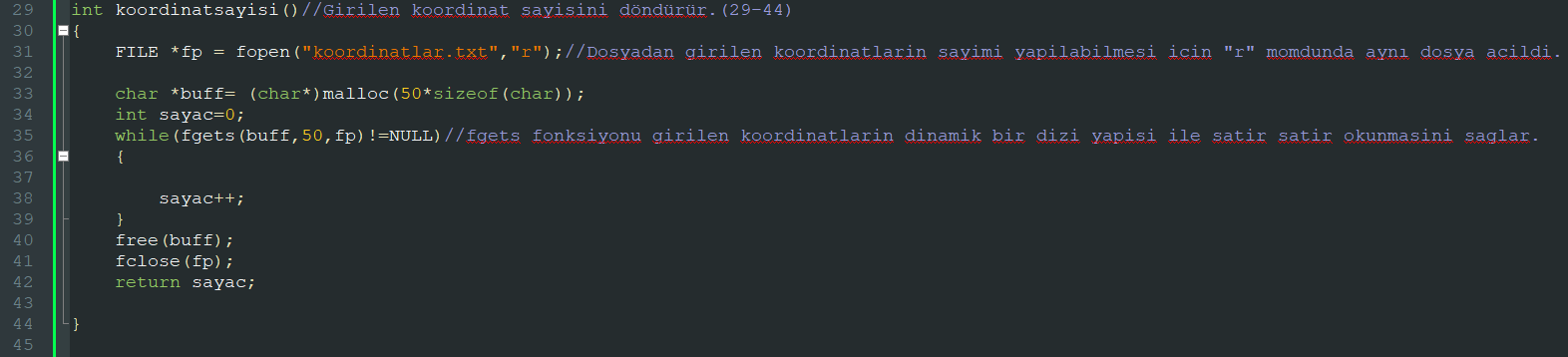
Programda *main* fonksiyonunda sadece *“cembercizdir*” fonksiyonunu çağırarak kod karmaşıklığının önüne geçilmeye çalışıldı. *cembercizdir* fonksiyonuna ek olarak *“koordinatal”* *“koordinatsayisi”*, *“uzaklik”* fonksiyonları ve bir adet “*cember” struct*’ı kullanıldı.”*koordinatal”, “koordinatsayisi*” ve “*uzaklik*” fonksiyonları “*cemberciz*” fonksiyonu içinde çağırıldı.

**YÖNTEM**

Program ana fonksiyonda “*cembercizdir”*  fonksiyonunu çağırarak başlatılıyor

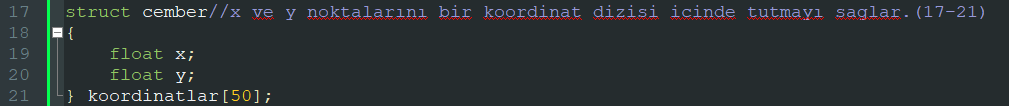


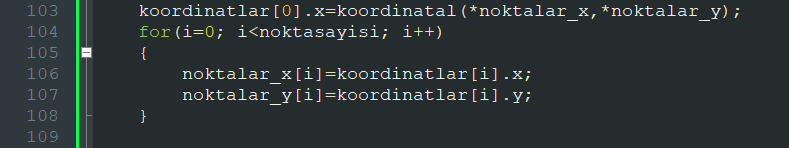
“*cembercizdir”* fonksiyonunda ilk olarak kullanıcıdan dosya üzerinden alınan noktaları saklama amacıyla 2 adet dizi oluşturuldu. Dosya üzerine girilen nokta sayısını belirlemek ve kodun diğer bölümlerinde kullanmak için *“koordinatsayisi”* fonksiyonu kullanıldı.



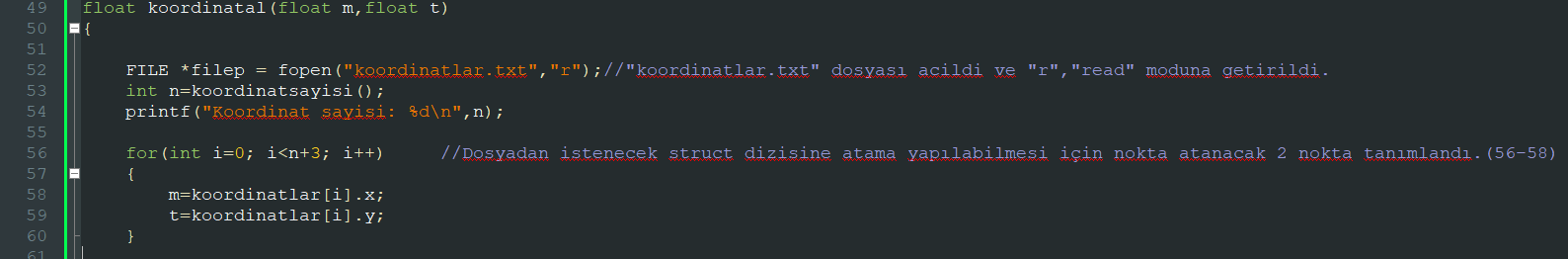
Yukarıdaki kod parçacığında ilk olarak *“koordinatlar”* adında kullanıcının istediği kadar nokta girmesini sağlayan *txt* formatında bir dosya açılması sağlandı .Eğer program çalışmaz ise programın başında neler yapılması gerektiği yorum satırları ile açıklandı. Daha sonra *fgets* fonksiyonu ile *txt* dosyası üzerinden alınan noktaların dinamik olarak dizide tutulması sağlandı. *free* fonksiyonu ile fazla tahsis edilen bellek boşa çıkarıldı.Döngüde ise *sayac* değişkeni ile nokta sayısı hesaplandı ve *“cembercizdir”* fonksiyonunda kullanılmak üzere return edildi.

Kullanıcının girdiği nokta sayısı hesaplandıktan sonra bu nokta sayısı kadar döndürülen bir döngüde *struct cember’i* kullanarak oluşturulan *koordinatlar[i] .x* ve *koordinatlar[i].y* dizilerini *noktalar\_x ve noktalar\_y dizilerine atanması sağlandı.*



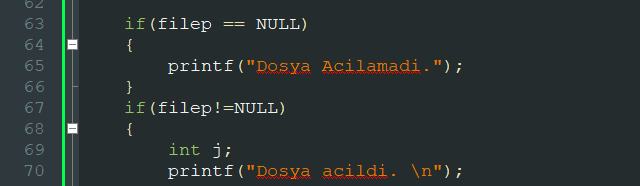
Yukarıdaki kod parçacıkları son paragrafta belirtilen işlemlerdir.

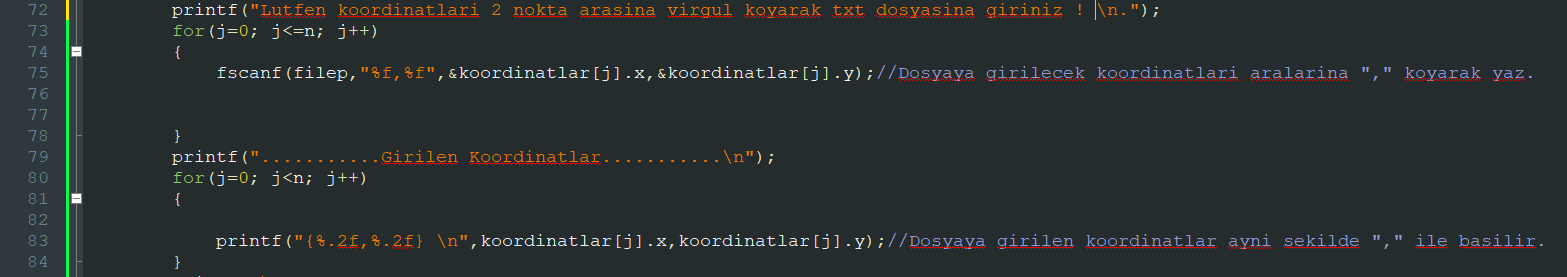
Son resimde gördüğümüz *koordinatal*  fonksiyonunda *noktalar\_x* ve *noktalar\_y’* nin yerleri gönderilerek alınan noktaların *cemberciz* fonksiyonunda noktalar\_x ve noktalar\_y ‘ nin içinde tutulması sağlandı.

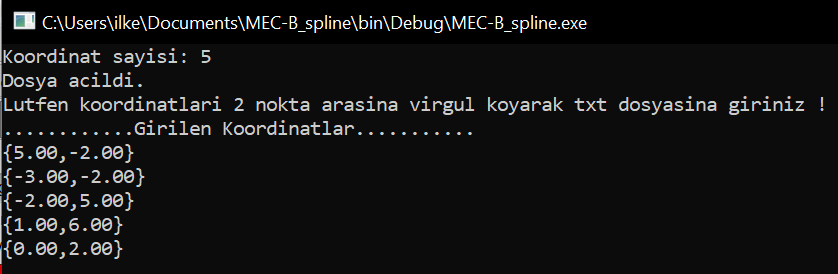


*koordinatal* fonksiyonunda “read” modunda dosya açıldı ve dosyadan alınan noktaların struct dizisine ataması yapılması için m ve t değişkenlerine atandı.

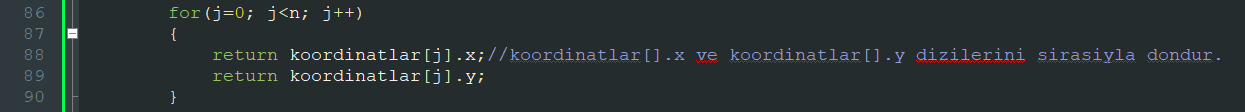
Daha sonra kullanıcının verileri gireceği dosyanın açılıp açılmadığı çeşitli hatalar ile karşılaşmamak için kontrol edildi.



Kontroller sağlanıp başarılı bir şekilde dosya açıldıktan sonra kullanıcının dosyaya nasıl girmesi gerektiği açıklandı ve *fscanf* fonksiyonu ile daha önceden yer ayrılmış diziye noktaların girilmesi sağlandı. Sonrasında konsol ekranında kullanıcının *txt* dosyasına girdiği noktaları kontrol etmesi ve görmesi için girdiği noktalar ve bu noktaların sayısı yazdırıldı.

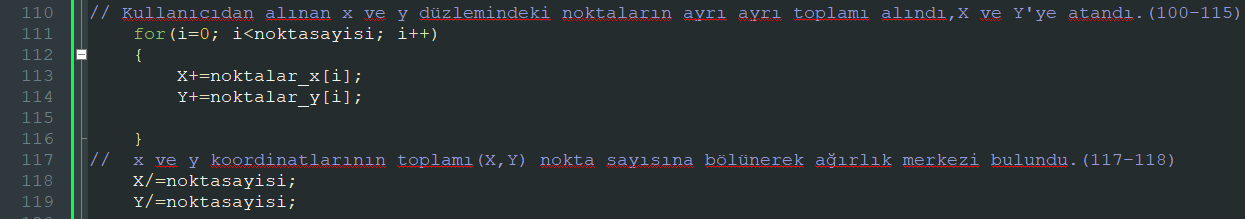


*koordinatal* fonksiyonun en son basamağı olarak da cembercizdir fonksiyonunda kullanılması amacıyla *koordinatlar[ ].x* ve *koordinatlar[ ].y* dizileri döndürüldü.

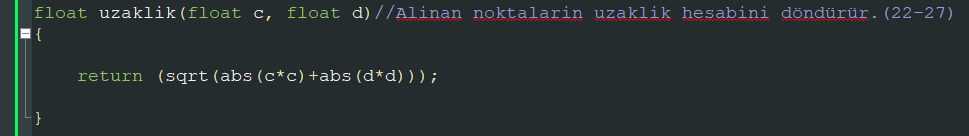


Şu ana kadar özetlemek gerekirse noktalar dosya üzerinden alındı, ayrı dinamik dizilerde *cembercizdir* fonksiyonunda kullanılacak şekilde tutuldu ve kullanıcıya gösterildi.

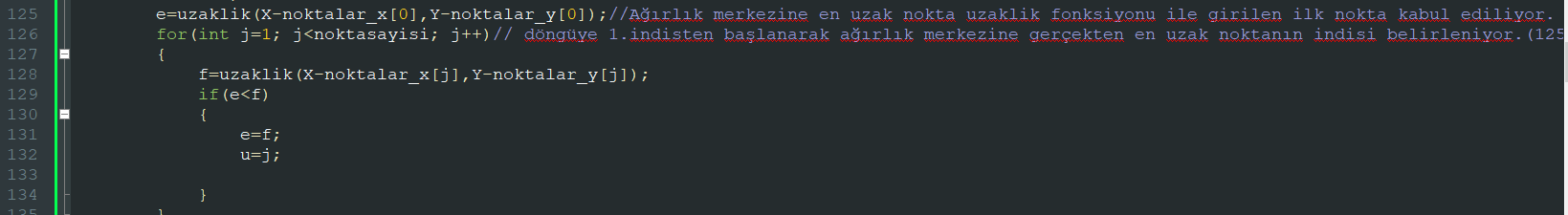
Yarıçap ve merkez koordinatları bulunurken ilk olarak *noktalar[x]* ve *noktalar[y]* içinde saklanan koordinatların apsisleri ve ordinatların toplamı *X* ve *Y* değişkenine atandı. Sonra *X* ve *Y* nokta sayısına bölünerek kullanıcının girdiği koordinatların ağırlık merkezi bulunmuş oldu.



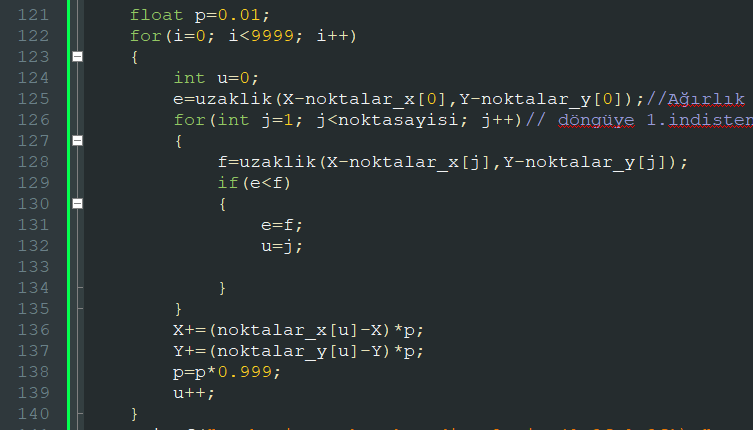
X ve Y değişkenlerine tüm koordinatların merkezi atandıktan sonra ağırlık merkezine en uzak nokta bulma amaçlandı. Bu en uzak nokta bulunurken ilk olarak girilen nokta (0.indeks) ağırlık merkezine en uzak kabul edilerek başlandı. Ayrıca bu en uzak nokta ile ağırlık merkezi arasındaki uzaklık *e* değişkenine atandı. Bu iki nokta arası uzaklığı bulmak içinse uzaklik fonksiyonu kullanıldı.



Daha sonra noktaların oluşturduğu dizinin 1.indisinden kontrol etmeye başlanarak gerçekten ağırlık merkezine (X, Y) en uzak noktanın kaçınca indiste bulunacağı hesaplandı ve *u* değişkeninde tutuldu. Yeni en uzak nokta ve o noktanın indisi belirlendikten sonra e değişkenine de bu uzaklık atandı.



Ağırlık merkezine en uzak nokta ve bu noktanın dinamik olarak tutulan dizide kaçıncı indekste bulunduğu belli olduktan sonra *e* uzaklığı(X-noktalar\_X[u] ,Y-noktalar\_y[u]) yarıçap ve ağırlık merkezini çemberin merkezi kabul edilen bir çember çizilseydi bu çember kesin olarak tüm noktaları kapsardı fakat en küçük çember olduğu kesinleşmezdi. Bu çizilen çemberi daha da küçültmek için dolayısıyla ağırlık merkezini en küçük çemberin merkezine yaklaştırmak için 9999 kere dönen bir döngü içinde ilk yaklaşma miktarı 0.1 olan ve *p* değişkeninde saklanan yaklaşma miktarı oluşturuldu. *p* değişkeni her yaklaşma sonrası kendisinin 0,999 katına çıkarak yani bir nevi limiti alınarak küçültüldü. *X* ve *Y* değişkenlerine ise her seferinde en uzak nokta ile arasındaki fark bulunup azalan bir p ile çarpılması ve bu yaklaşma miktarının her seferinde kendilerine eklenmesiyle *X* ve *Y* gerçek merkez noktası oldular.



En küçük ve tüm noktaları kapsayan çemberin bulunmasıyla X,Y merkez koordinatı e uzaklığı ise yarıçap oldu. Kullanıcıya bu bilgiler gösterildi

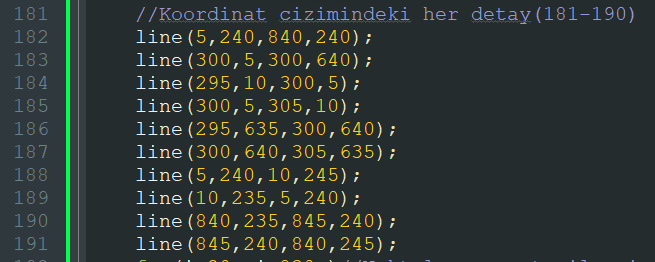
.

Çemberin çizilmesi için gerekli tüm veriler alındıktan ve en küçük çemberin bilgileri bulunduktan sonra çizim işlemlerine başlandı.

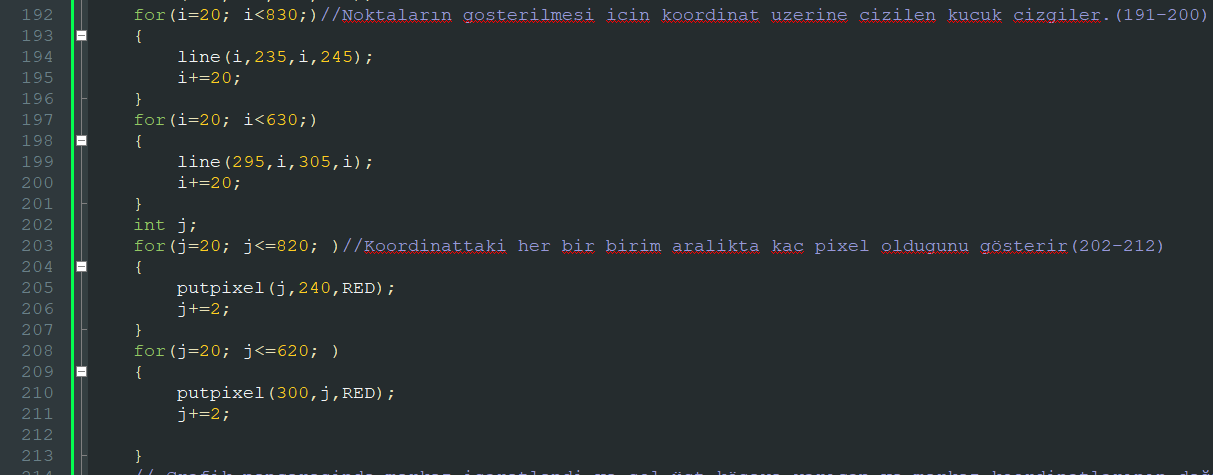
Çizime ilk olarak koordinat sistemini grafik ekranına tüm detayları ile oturtarak ve grafik penceresini açma fonksiyonunu kullanarak başlandı. İlk önce 2 boyutlu düzlemde x eksenine 40, y eksenine 30 adet nokta yazıldı. Bu noktalar için x ve y düzleminde ayrı ayrı kullanılmak üzere *sayi1* ve *sayi2* dizileri oluşturuldu. Bu dizilerin oluşturulmasındaki amaç grafik ekranına doğrudan tam sayı değerler yazdırılmamasıdır. Çözüm olarak bu diziler sprintf fonksiyonu ile diziye noktalar gönderildi ve *outtextxy f*onksiyonu kullanarak koordinat düzleminin üstündeki sayılar yazıldı.



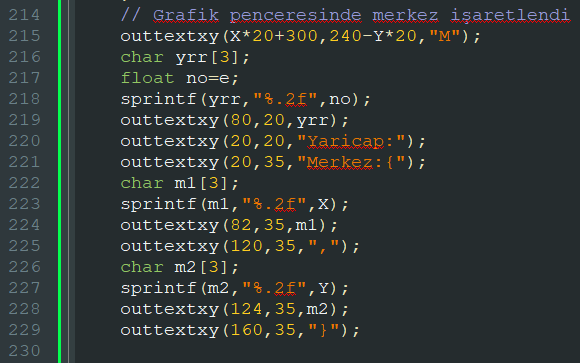
Noktalar yerleştirildikten sonra graphics.h kütüphanesi fonksiyonlarından *line* fonksiyonu ile 2 boyutlu koordinat düzlemi çizildi.



Koordinat düzlemi için son olarak ise kullanıcının girdiği noktaların koordinat düzleminde görmesi için ve çemberin geçtiği noktaların koordinatlarının belli olması için sayıların gösterildiği küçük çizgiler çizildi.



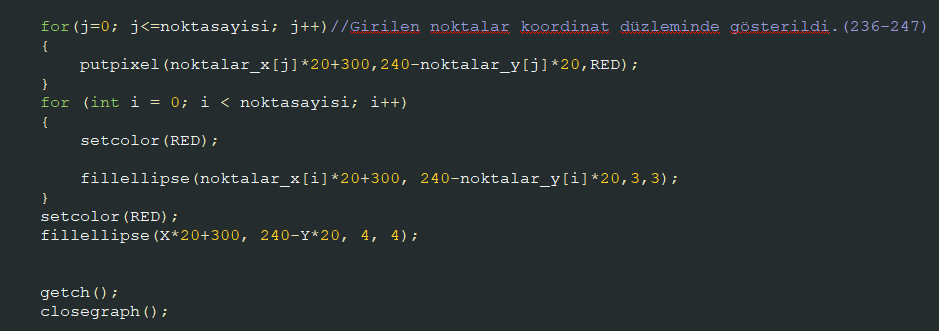
Yorum satılarında da açıklandığı gibi pikseller 20 kat büyütülerek grafik penceresi daha kullanılışlı hala getirildi. Daha sonra grafik penceresine çemberin merkez koordinatları ve yarıçap bilgisi kullanıcının bilgi alması için yazıldı. Yine tam sayı yazamama sorunundan dolayı dizi oluşturup *sprintf* fonksiyonu kullanıldı.



Son olarak da çember ve kullanıcının girdiği noktaların yerleri çizdirildi. *circle* komutu kullanırken X, Y, e sayılarına öteleme yapıldı. Bu öteleme grafik ekranının 0,0 noktasının pencerenin sol üst köşesi olmasının yarattığı sorunun çözümüdür. Aynı şekilde piksellerin 20 kat büyütülmesi de grafik penceresinin daha şık görünmesi içindir.



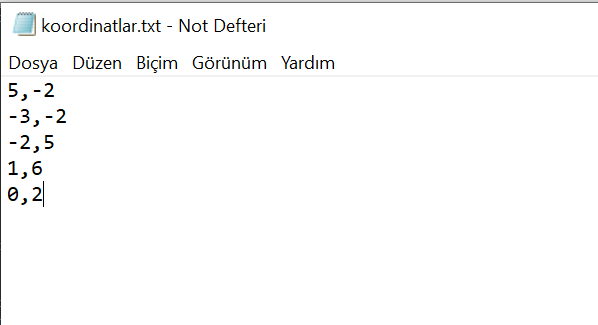
Kullanıcının girdiği tüm noktaları görmesi ve yapılan hesaplamaların doğruluğunu kontrol etmek için tüm girilen noktalar koordinat düzleminde *putpixel* fonksiyonu kullanılarak gösterildi. fillelipse ve kırmızı renk seçeneği sayesinde noktaların yeri daha belirginleşti.



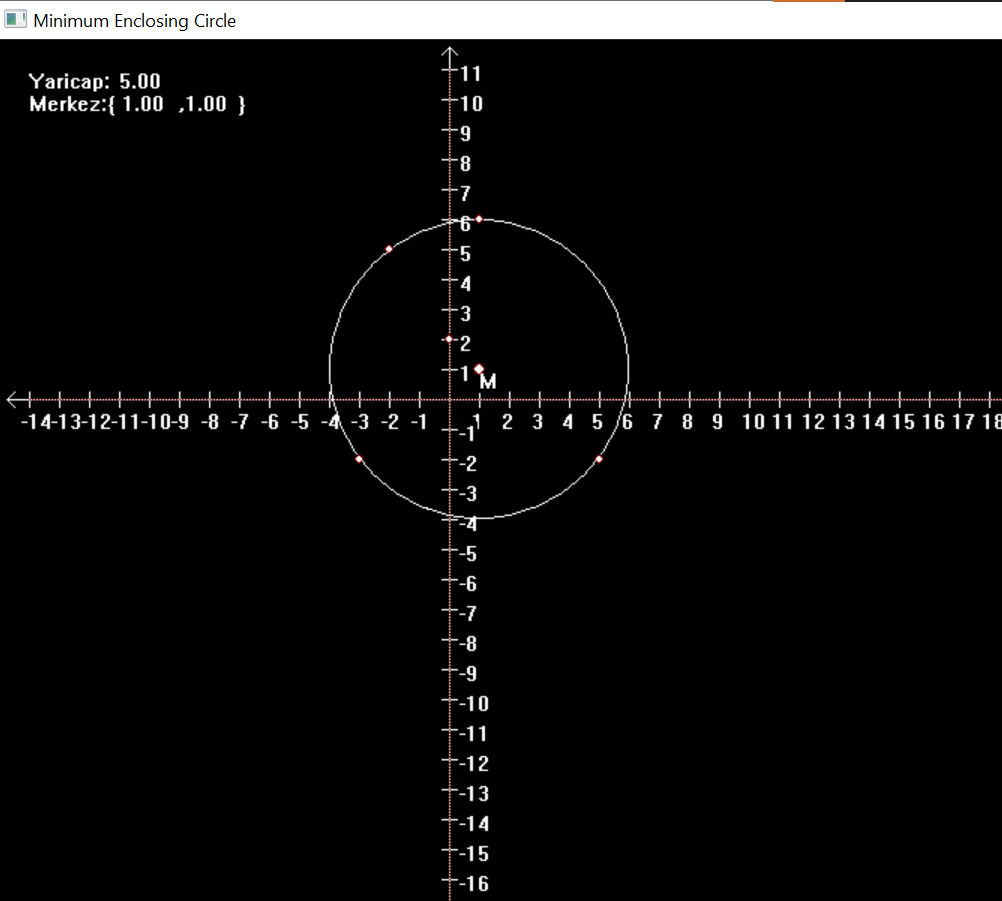
Son olarak grafik penceresi ve *cembercizdir* fonksiyonu kapatıldı.

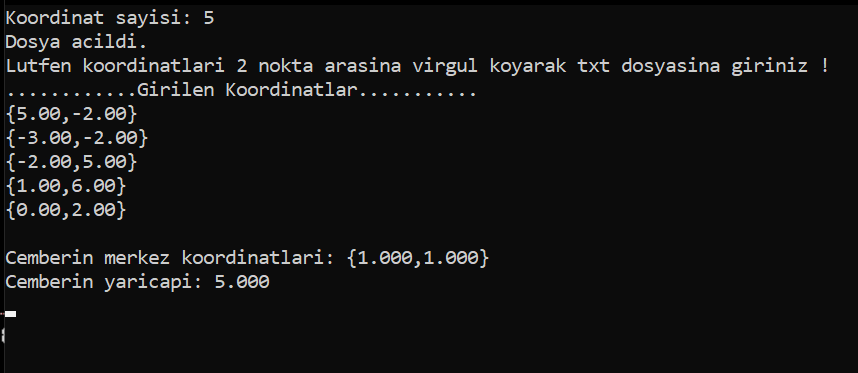
**SONUÇ**

Kullanıcı ilk olarak programın üstünde olası hatalara karşı bilinçlendirildi. Daha sonra açılan txt dosyasına verilerini anlatıldığı gibi girdi.



Bu dosyayı kaydederek programı çalıştırdı ve çember çizdilirildi.





**Algoritmanın Yalancı Kodu**

Koordinatları al

Noktaların ortasını bul

Orta noktaya en uzak noktayı ilk girilen nokta kabul et

if

Daha uzak bir nokta var mı kontrol et.

Var ise bunu yarıçap kabul et.

Var ise bu noktanın indisini bul

En uzak nokta ile ağırlık merkezi arasındaki uzaklığı her adımda küçült

X+=(noktalar\_x[u]- X) \*p;

Y+=(noktalar\_y[u]- Y) \*p;

Yaklaşma miktarını her seferinde X ve Y ‘ye bir daha ekle

Minimum 2 nokta arası uzaklık eşit oluncaya kadar devam et.

X,Y en küçük çemberin merkez koordinatları olur.

Zaman Karmaşıklığı: O (N) \* O(N) = O(N^2)

**Kaynakça**

1-Algoritma:

<https://www.geeksforgeeks.org/minimum-enclosing-circle-set-1/>

<https://www.geeksforgeeks.org/minimum-enclosing-circle-set-2/>

<http://cagataykiziltan.net/programin-calisma-hizi-ve-algoritma-verimliligi/zaman-karmasikligi-ve-buyuk-o-notasyonu-time-complexity-and-big-o-notation/>

codeforces.com

http://www.lee-mac.com/mecfunction.html

Onur Gök-Suhap Şahin Sunumları

2-C dili ve graphics.h için :

<https://www.bilgigunlugum.net/prog/cprog/c_dinamik>

<https://www.programmingsimplified.com/c/graphics.h>

<https://fatihalparslann.wordpress.com/2017/02/02/c-dilinde-graphics-h-kutuphanesi-1-bolum/>

Çeşitli youtube kanalları