T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ

ÖDEV KONUSU
GEOMETRİK PROBLEMLER

Hazırlayan

ÖĞRENCİ ADI

Enes KÜÇÜK-220501017

https://github.com/EnessKucukk

İbrahim Bener Karaca-220501019

https://github.com/BenerKaraca

DERS SORUMLUSU

Prof.Dr. Hüseyin Tarık DURU

TARİH 1.01.2024

İÇİNDEKİLER

1. ÖZET

Nokta, DogruParcasi, Daire ve Ucgen sınıfları, matematiksel kavramları temsil eden nesne tabanlı programlama örneklerini içerir. Nokta sınıfı x ve y koordinatlarını yönetirken aynı zamanda DogruParcasi sınıfı iki noktayı kullanarak doğru parçasının uzunluğunu ve orta noktasını hesaplar.

2. GİRİŞ

Ödevde bizden istenen noktalar şunlardır:

Tüm çalışmanın g1)Bir noktanın x ve y koordinatlarını nesne değişkeni olarak tutan (double tipinde tutulmalı) bir Nokta sınıfıoluşturun.

• Sınıf nesnelerinin 5 farklı şekilde başlatılabilmesi için beş yapıcıyı aşağıdaki gibi yazın:

• Parametresiz yapıcı: Noktanın koordinatlarını orijine ayarlar.

• Tek parametreli yapıcı: İki koordinata aynı değeri atar.

• İki parametreli yapıcı: x ve y koordinatları için sırasıyla iki double değişken alır.

• Başka bir noktayı alıp o noktanın bir kopyasını yeni nokta olarak üreten yapıcı.

• Başka bir nokta (Nokta nesnesi) ve iki double değişken (ofset_x ve ofset_y) alarak yeni bir nokta (Nokta nesnesi) üretir ve ofset değişkenlerini ilk giriş parametresi olarak alınan orijinal noktanın x ve y koordinatlarına ekler.

Yapıcılar koordinatları ayarlamak için uygun set metotlarını çağırmalıdır.

• x ve y koordinatları için get ve set metotlarını ayrı ayrı yazılmalıdır. (setX,setY,getX,getY).

• Aynı anda iki koordinat alan ve noktanın x ve y koordinatlarının değerlerini değiştiren bir set metotdu

yazılmalıdır.

• toString metodu, noktanın (Nokta nesnesi) koordinatlarının String gösterimini döndürür.

• yazdir metodu: toString metodunu kullanarak ekrana koordinatları yazdırır.

2) Nesne değişkenleri olarak bir doğru parçasının iki noktasını (Nokta nesnesi olarak) içeren,bir DogruParcasi sınıfı oluşturun.

• DogruParcasi nesnelerini başlatmak için aşağıdaki gibi üç yapıcı geliştirin:
• İki uç noktayı Nokta nesnesi olarak alan yapıcı.
• Başka bir DogruParcasi nesnesi alıp onun bir kopyasını yeni bir DogruParcasi nesnesi olarak oluşturan yapıcı.
• Bir Nokta nesnesi (doğru parçasının orta noktası olarak), parçanın uzunluğu (double) ve eğimi (double) değerlerini alarak, doğru parçasının 2 uç noktasının x ve y koordinatlarını hesaplayan yapıcı.
• İlgili get ve set metotlarını ekleyin.
• uzunluk metodu: DogruParcasi nesnesinin uzunluğunun double değişken olarak hesaplar ve döndürür.
• kesişimNoktası metodu: Bir Nokta nesnesini parametre olarak alır, bu noktadan doğru parçasına dik
olarak çizilecek doğru parçasının kesişme noktasını hesaplar ve bir Nokta nesnesi olarak döndürür.
• ortaNokta metodu: Doğru parçasının orta noktasını hesaplayan ve bir Nokta nesnesi olarak döndürür.
• toString yöntemi: geçerli LineSegment nesnesinin String olarak döndürme. Nesne sınıfının toString yöntemini kullanmalı.
• yazdir metodu: İki uç noktayı toString metodunu kullanarak ekrana yazdırır.

3) Dairenin merkezi (Nokta nesnesi olarak) ve yarıçapını nesne değişkenleri olarak tutan bir Daire sınıfını oluşturun.
• Daire nesnesini başlatmak için aşağıdaki gibi üç yapıcı geliştirin:
• Merkez (Nokta nesnesi olarak) ve yarıçapı parametre olarak alan yapıcı.
• Başka bir Daire nesnesi alıp onun bir kopyasını yeni bir Daire nesnesi olarak oluşturan yapıcı.
• Başka bir Daire nesnesi ve reel bir pozitif x değeri alarak, parametre olarak alınan Daire nesnesini yarıçapı x ile çarpılmış olarak kopyalayan yapıcı.
• alan metodu: Dairenin alanını döndürür.
• cevre metodu: Dairenin çevresini döndürür.
• kesisim metodu: Bir Daire nesnesi alır. Parametre olarak gelen daire metodu çağıran dairenin

içinde ise 0, daireler birebir örtüşüyorsa 1, hiç kesişim yoksa 2 döndürür.
• toString metodu: Dairenin merkezi ve yarıçapını String halinde döndürür.
• yazdir metodu: toString metodunu kullanarak ekrana direnin bilgilerini yazdırır.

4) Nesne değişkeni olarak 3 tane Nokta nesnesi içeren Ucgen sınıfını oluşturun.
• Ucgen nesnesini başlatmak için aşağıdaki gibi bir yapıcı yazın
• Üç tane Nokta nesnesi alan yapıcı
• İlgili get ve set yöntemlerini ekleyin
• toString metodu: Üçgenin String temsilini şu şekilde döndürür: "üçgen; onun noktaları"
• alan metodu: Bu üç noktanın temsil ettiği üçgenin alanını hesaplar ve döndürür.
• cevre metodu: üçgenin çevresini hesaplayın ve döndürün. Yöntem içerisinde üç doğru parçası oluşturun ve DogruParcasi sınıfının uzunluk metodunu kullanın. acilar metodu: Üçgenin açılarını üç öğeli double dizi olarak hesaplar ve döndürür.

NOT:
Gerekli test dosyasını hocalarımız bize iletecektir.

3. YÖNTEM

Bu projemizde bir C++ kodu yazacağız. Bizde anlaşılması kolay olması açısından gerekli sınıfları ayrı dosyalar halinde yazmaya karar verdik ve şu dosyaları oluşturduk

nokta.h ve nokta.cpp

DogruParcasi.h ve DogruParcasi.cpp

Daire.h ve Daire.cpp

Ücgen.h ve Ücgen.cpp

3.1 Tasarlama ‘H’

Bu bölümümüzde ödevizdeki tasarlama dosyalrının açıklayacağız

1.Nokta.h

```
#pragma once
#include <iostream>
#include <string>

class Nokta {
private:
    double x;
    double y;

public:
    Nokta();
    Nokta(double xy);
    Nokta(double x, double y);
    Nokta(const Nokta& nesne);
    Nokta(const Nokta& nesne, double ofset_x, double ofset_y);

    double getX() const;
    void setX(double x);

    double getY() const;
    void setY(double y);

    void set(double x, double y);

    std::string toString() const;
    void yazdir() const;
};
```

-Nokta(): (0, 0) noktasını oluşturur.
-Nokta(double xy): (xy, xy) noktasını oluşturur.
-Nokta(double x, double y): (x, y) noktasını oluşturur.
-Nokta(const Nokta& nesne): Başka bir Nokta nesnesinin kopyasını oluşturur.
-Nokta(const Nokta& nesne, double ofset_x, double ofset_y): Ofsetlerle kopya oluşturur.
-getX(): x koordinatını alır.
-getY(): y koordinatını alır.
-setX(double x): x koordinatını ayarlar.
-setY(double y): y koordinatını ayarlar.
-set(double x, double y): her iki koordinatı ayarlar.
-oString(): Noktayı temsil eden bir dize döndürür.
-yazdir(): Noktanın koordinatlarını konsola yazdırır.

DogruParcasi.h

```
#pragma once

#include "nokta.h"

class DogruParcasi {
private:
    Nokta xy1;
    Nokta xy2;

public:
    DogruParcasi(const Nokta& xy1, const Nokta& xy2);
    DogruParcasi(const DogruParcasi& nesne);
    DogruParcasi(const Nokta& ortaNokta, double uzunluk, double egim);
    void setP1(const Nokta& point);
    Nokta getXY1() const;
    void setXY1(const Nokta& nokta);

    Nokta getXY2() const;
    void setXY2(const Nokta& nokta);

    double uzunluk() const;

    Nokta kesisimNoktasi(const Nokta& dikNokta) const;

    Nokta ortaNokta() const;

    std::string toString() const;
    void yazdir() const;
};
```

- `Nokta xy1`: Doğru parçasının bir ucu için bir Nokta nesnesini temsil eder.
- `Nokta xy2`: Doğru parçasının diğer ucu için bir Nokta nesnesini temsil eder.
- `DogruParcasi (const Nokta& xy1, const Nokta& xy2)`: İki Nokta nesnesi alarak bir doğru parçası oluşturur.
- `DogruParcasi (const DogruParcasi& nesne)`: Bir başka DogruParcasi nesnesini kopyalar.
- `DogruParcasi (const Nokta& ortaNokta, double uzunluk, double egim)`: Belirtilen orta noktadan başlayarak, verilen uzunluk ve eğimle bir doğru parçası oluşturur.
- `void setP1 (const Nokta& point)`: Birinci ucu belirtilen Nokta nesnesi ile ayarlar.
- `Nokta getXY1 () const`: Birinci ucun Nokta nesnesini döndürür.
- `void setXY1 (const Nokta& nokta)`: Birinci ucun Nokta nesnesini belirtilen Nokta nesnesi ile ayarlar.
- `Nokta getXY2() const`: İkinci ucun Nokta nesnesini döndürür.
- `void setXY2(const Nokta& nokta)`: İkinci ucun Nokta nesnesini belirtilen Nokta nesnesi ile ayarlar.
- `double uzunluk() const`: Doğru parçasının uzunluğunu hesaplar ve döndürür.
- `Nokta kesisimNoktasi(const Nokta& dikNokta) const`: Verilen bir dikey doğruyla kesişim noktasını bulur.
- `Nokta ortaNokta() const`: Doğru parçasının orta noktasını bulur ve döndürür.

- `std::string toString() const`: Doğru parçasının bilgilerini içeren bir metin dizesini döndürür.
- `void yazdir() const`: Doğru parçasının bilgilerini ekrana yazdırır.

Daire.h

```
#pragma once
#include "nokta.h"

class Daire {
private:
    Nokta orjin;
    double r;

public:

    Daire(const Nokta& orjin, double r);
    Daire(const Daire& nesne);
    Daire(const Daire& nesne, double x);


    double alan() const;
    double cevre() const;
    int kesisim(const Daire& dairee) const;

    std::string toString() const;
    void yazdir() const;
};
```

- `Nokta orjin`: Dairenin merkezini temsil eden bir Nokta nesnesi.
- `double r`: Dairenin yarıçapını temsil eden bir sayı.
- `Daire(const Nokta& orjin, double r)`: Merkezi belirtilen bir nokta ve yarıçapı belirtilen bir değer olan bir daire oluşturur.
- `Daire(const Daire& nesne)`: Bir başka Daire nesnesini kopyalar.
- `Daire(const Daire& nesne, double x)`: X koordinatını belirtilen ofset ile değiştirerek yeni bir daire oluşturur.
- `double alan() const`: Dairenin alanını hesaplar ve döndürür.
- `double cevre() const`: Dairenin çevresini hesaplar ve döndürür.
- `int kesisim(const Daire& dairee) const`: Verilen başka bir daire ile kesişip kesişmediğini kontrol eder ve sonucu döndürür.
- `std::string toString() const`: Dairenin bilgilerini içeren bir metin dizesini döndürür.
- `void yazdir() const`: Dairenin bilgilerini ekrana yazdırır

Ucgen.h

```
#pragma once
#include "nokta.h"
#include "dogruparcasi.h"

class Ucgen {
private:
    Nokta xy1;
    Nokta xy2;
```

```
    Nokta xy3;

public:

    Ucgen(const Nokta& xy1, const Nokta& xy2, const Nokta& xy3);


    Nokta getXY1() const;
    void setXY1(const Nokta& xy);

    Nokta getXY2() const;
    void setXY2(const Nokta& xy);

    Nokta getXY3() const;
    void setXY3(const Nokta& xy);


    std::string toString() const;
    double alan() const;
    double cevre() const;
    double* acilar() const;
};
```

- `Nokta xy1`, `Nokta xy2`, `Nokta xy3`: Üçgenin üç köşe noktasını temsil eden Nokta nesneleri. Bu noktalar, üçgenin konumunu belirtir.
- `Ucgen(const Nokta& xy1, const Nokta& xy2, const Nokta& xy3)`: Üçgenin üç köşe noktasını belirleyerek bir üçgen oluşturan kurucu fonksiyon.
- `Nokta getXY1() const`: Üçgenin birinci köşe noktasını döndüren getter fonksiyonu.
- `void setXY1(const Nokta& xy)`: Üçgenin birinci köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.
- `Nokta getXY2() const`: Üçgenin ikinci köşe noktasını döndüren getter fonksiyonu.
- `void setXY2(const Nokta& xy)`: Üçgenin ikinci köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.
- `Nokta getXY3() const`: Üçgenin üçüncü köşe noktasını döndüren getter fonksiyonu.
- `void setXY3(const Nokta& xy)`: Üçgenin üçüncü köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.
- `std::string toString() const`: Üçgenin köşe noktalarını içeren bir metin dizesini döndüren fonksiyon.
- `double alan() const`: Üçgenin alanını hesaplayan ve döndüren fonksiyon.
- `double cevre() const`: Üçgenin çevresini hesaplayan ve döndüren fonksiyon.
- `double* acilar() const`: Üçgenin iç açılarını hesaplayarak, bir dizi içinde döndüren fonksiyon.

3.2 Gerçekleme ‘cpp’

Nokta.cpp

```
#include "nokta.h"
#include <iostream>
#include <iomanip>

Nokta::Nokta() : x(0), y(0) {}

Nokta::Nokta(double xy) : x(xy), y(xy) {}

Nokta::Nokta(double x, double y) : x(x), y(y) {}

Nokta::Nokta(const Nokta& nesne) : x(nesne.x), y(nesne.y) {}

Nokta::Nokta(const Nokta& nesne, double ofset_x, double ofset_y) : x(nesne.x +
ofset_x), y(nesne.y + ofset_y) {}

double Nokta::getX() const {
    return x;
}

void Nokta::setX(double x) {
    this->x = x;
}

double Nokta::getY() const {
    return y;
}

void Nokta::setY(double y) {
    this->y = y;
}

void Nokta::set(double x, double y) {
    this->x = x;
    this->y = y;
}

std::string Nokta::toString() const {
    return "(" + std::to_string(x) + ", " + std::to_string(y) + ")";
}

void Nokta::yazdir() const {
    std::cout << std::fixed << std::setprecision(2) << "(" << x << ", " << y <<
")" << std::endl;
}
```

- `double x`: Noktanın x koordinatını tutan özel bir veri üyesi.

- `double y`: Noktanın y koordinatını tutan özel bir veri üyesi.

- `Nokta()`: Varsayılan kurucu fonksiyon, x ve y koordinatlarını 0.0 olarak ayarlar.

- `Nokta(double xy)`: Tek bir parametre alarak x ve y koordinatlarına aynı değeri atayan kurucu fonksiyon.

- `Nokta(double x, double y)`: İki parametre alarak x ve y koordinatlarını belirtilen değerlere ayarlayan kurucu fonksiyon.

- `Nokta(const Nokta& nesne)`: Bir `Nokta` nesnesini kopyalayan kopya kurucu fonksiyon.

- `Nokta(const Nokta& nesne, double ofset_x, double ofset_y)`: Belirtilen ofset değerleri ile bir `Nokta` nesnesini kopyalayan kopya kurucu fonksiyon.

- `double getX() const`: x koordinatının değerini döndüren getter fonksiyonu.

- `void setX(double x)`: x koordinatının değerini ayarlayan setter fonksiyonu.

- `double getY() const`: y koordinatının değerini döndüren getter fonksiyonu.

- `void setY(double y)`: y koordinatının değerini ayarlayan setter fonksiyonu.

- `void set(double x, double y)`: Hem x hem de y koordinatlarını belirtilen değerlere ayarlayan fonksiyon.

- `std::string toString() const`: Noktanın x ve y koordinatlarını içeren bir metin dizesini döndüren fonksiyon.

- `void yazdir() const`: Noktanın x ve y koordinatlarını ekrana yazdıran fonksiyon.

Ucgen.cpp

```cpp
#include "ucgen.h"
#include "dogruparcasi.h"
#include <cmath>
#include <iostream>


Ucgen::Ucgen(const Nokta& xy1, const Nokta& xy2, const Nokta& xy3)
    : xy1(xy1), xy2(xy2), xy3(xy3) {}

Nokta Ucgen::getXY1() const {
    return xy1;
}
void Ucgen::setXY1(const Nokta& xy) {
    xy1 = xy;
}

Nokta Ucgen::getXY2() const {
    return xy2;
}
void Ucgen::setXY2(const Nokta& xy) {
    xy2 = xy;
```

```
}
Nokta Ucgen::getXY3() const {
    return xy3;
}

void Ucgen::setXY3(const Nokta& xy) {
    xy3 = xy;
}
std::string Ucgen::toString() const {
    return "Ucgen; " + xy1.toString() + ", " + xy2.toString() + ", " +
xy3.toString();
}

double Ucgen::alan() const {


    return 0.5 * DogruParcasi(xy1, xy2).uzunluk() * xy3.getY() - xy1.getY();
}

double Ucgen::cevre() const {

    double a = DogruParcasi(xy1, xy2).uzunluk();
    double b = DogruParcasi(xy2, xy3).uzunluk();
    double c = DogruParcasi(xy3, xy1).uzunluk();
    return a + b + c;
}
double* Ucgen::acilar() const {

    double* aciDizisi = new double[3];
    double a = DogruParcasi(xy1, xy2).uzunluk();
    double b = DogruParcasi(xy2, xy3).uzunluk();
    double c = DogruParcasi(xy3, xy1).uzunluk();

    aciDizisi[0] = acos((b * b + c * c - a * a) / (2 * b * c)) * (180.0 / 3.14);
    aciDizisi[1] = acos((c * c + a * a - b * b) / (2 * c * a)) * (180.0 / 3.14);
    aciDizisi[2] = acos((a * a + b * b - c * c) / (2 * a * b)) * (180.0 / 3.14);

    return aciDizisi;
}
```

- `Ucgen(const Nokta& xy1, const Nokta& xy2, const Nokta& xy3)`: Üçgenin üç köşe noktasını belirleyerek bir üçgen oluşturan kurucu fonksiyon.
- `Nokta getXY1() const`: Üçgenin birinci köşe noktasını döndüren getter fonksiyonu.
- `void setXY1(const Nokta& xy)`: Üçgenin birinci köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.
- `Nokta getXY2() const`: Üçgenin ikinci köşe noktasını döndüren getter fonksiyonu.
- `void setXY2(const Nokta& xy)`: Üçgenin ikinci köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.
- `Nokta getXY3() const`: Üçgenin üçüncü köşe noktasını döndüren getter fonksiyonu.
- `void setXY3(const Nokta& xy)`: Üçgenin üçüncü köşe noktasını belirtilen Nokta nesnesi ile ayarlayan setter fonksiyonu.

- `std::string toString() const`: Üçgenin köşe noktalarını içeren bir metin dizesini döndüren fonksiyon.
- `double alan() const`: Üçgenin alanını hesaplayan fonksiyon. Alan hesaplanırken, bir kenarın uzunluğu ve üçüncü köşe noktasının y koordinatı kullanılır.
- `double cevre() const`: Üçgenin çevresini hesaplayan fonksiyon.
- `double* acilar() const`: Üçgenin iç açılarını hesaplayarak bir dizi içinde döndüren fonksiyon. Hesaplama, üç kenar uzunluğu ve kosinüs teoremi kullanılarak gerçekleştirilir. Bu açılar derece cinsinden döndürülür.

Dogruparcasi.cpp

```
#include "dogruparcasi.h"
#include <cmath>

DogruParcasi::DogruParcasi(const Nokta& xy1, const Nokta& xy2) : xy1(xy1),
xy2(xy2) {}

DogruParcasi::DogruParcasi(const DogruParcasi& nesne) : xy1(nesne.xy1),
xy2(nesne.xy2) {}

DogruParcasi::DogruParcasi(const Nokta& ortaNokta, double uzunluk, double egim) {

    double aci = atan(egim);
    double gX = uzunluk / 2 * cos(aci);
    double gY = uzunluk / 2 * sin(aci);

    xy1 = Nokta(ortaNokta.getX() - gX, ortaNokta.getY() - gY);
    xy2 = Nokta(ortaNokta.getX() + gX, ortaNokta.getY() + gY);
}

void DogruParcasi::setP1(const Nokta& point) {
    xy1 = point;
}
Nokta DogruParcasi::getXY1() const {
    return xy1;
}

void DogruParcasi::setXY1(const Nokta& nokta) {
    xy1 = nokta;
}
Nokta DogruParcasi::getXY2() const {
    return xy2;
}
void DogruParcasi::setXY2(const Nokta& nokta) {
    xy2 = nokta;
}

double DogruParcasi::uzunluk() const {
    return sqrt(pow(xy2.getX() - xy1.getX(), 2) + pow(xy2.getY() - xy1.getY(),
2));
}

Nokta DogruParcasi::kesisimNoktasi(const Nokta& dikNokta) const {
    Nokta kesisim;
```

```
    double m1 = (xy2.getY() - xy1.getY()) / (xy2.getX() - xy1.getX());
    double m2 = (dikNokta.getY() - xy1.getY()) / (dikNokta.getX() - xy1.getX());


    kesisim.setX((m1 * xy1.getX() - m2 * dikNokta.getX() + dikNokta.getY() -
xy1.getY()) / (m1 - m2));
    kesisim.setY(m1 * (kesisim.getX() - xy1.getX()) + xy1.getY());

    return kesisim;
}

Nokta DogruParcasi::ortaNokta() const {

    return Nokta((xy1.getX() + xy2.getX()) / 2, (xy1.getY() + xy2.getY()) / 2);
}

std::string DogruParcasi::toString() const {
    return xy1.toString() + " ve " + xy2.toString();
}

void DogruParcasi::yazdir() const {
    std::cout << toString() << std::endl;
}
```

- `DogruParcasi(const Nokta& xy1, const Nokta& xy2)`: İki nokta arasında bir doğru parçası oluşturan kurucu fonksiyon.
- `DogruParcasi(const DogruParcasi& nesne)`: Bir başka `DogruParcasi` nesnesini kopyalayan kopya kurucu fonksiyon.
- `DogruParcasi(const Nokta& ortaNokta, double uzunluk, double egim)`: Belirtilen merkezi noktaya, uzunluğa ve eğime sahip bir doğru parçası oluşturan kurucu fonksiyon.
- `void setP1(const Nokta& point)`: Birinci noktayı belirtilen noktaya ayarlayan setter fonksiyonu.
- `Nokta getXY1() const`: Birinci noktanın koordinatlarını döndüren getter fonksiyonu.
- `void setXY1(const Nokta& nokta)`: Birinci noktanın koordinatlarını belirtilen nokta ile ayarlayan setter fonksiyonu.
- `Nokta getXY2() const`: İkinci noktanın koordinatlarını döndüren getter fonksiyonu.
- `void setXY2(const Nokta& nokta)`: İkinci noktanın koordinatlarını belirtilen nokta ile ayarlayan setter fonksiyonu.
- `double uzunluk() const`: Doğru parçasının uzunluğunu hesaplayan fonksiyon.
- `Nokta kesisimNoktasi(const Nokta& dikNokta) const`: Belirtilen dik doğru ile bu doğru parçasının kesişim noktasını hesaplayan fonksiyon.
- `Nokta ortaNokta() const`: Doğru parçasının orta noktasını hesaplayan fonksiyon.
- `std::string toString() const`: Doğru parçasının koordinatlarını içeren bir metin dizesini döndüren fonksiyon.
- `void yazdir() const`: Doğru parçasının koordinatlarını ekrana yazdıran fonksiyon.

Daire.cpp

```cpp
#include "daire.h"
#include <cmath>
#include <iostream>


Daire::Daire(const Nokta& orjin, double r) : orjin(orjin), r(r) {}
Daire::Daire(const Daire& nesne) : orjin(nesne.orjin), r(nesne.r) {}
Daire::Daire(const Daire& nesne, double x) : orjin(nesne.orjin), r(nesne.r * x)
{}

double Daire::alan() const {
    return 3.14 * r * r;
}
double Daire::cevre() const {
    return 2 * 3.14 * r;
}

int Daire::kesisim(const Daire& dairee) const {
    double uzaklik = sqrt(pow(orjin.getX() - dairee.orjin.getX(), 2) +
pow(orjin.getY() - dairee.orjin.getY(), 2));
    if (uzaklik + dairee.r < r) {
        return 0;
    } else if (uzaklik <= r + dairee.r) {
        return 1;
    } else {
        return 2;
    }
}

std::string Daire::toString() const {
    return "Merkez: " + orjin.toString() + ", Yaricap: " + std::to_string(r);
}

void Daire::yazdir() const {
    std::cout << toString() << std::endl;
}
```

- `Daire(const Nokta& orjin, double r)`: Belirtilen orijin noktası ve yarıçap ile bir daire oluşturan kurucu fonksiyon.
- `Daire(const Daire& nesne)`: Bir başka `Daire` nesnesini kopyalayan kopya kurucu fonksiyon.
- `Daire(const Daire& nesne, double x)`: Belirtilen orijin noktasını ve yarıçapı, belirtilen oranla büyüten kurucu fonksiyon.
- `double alan() const`: Dairenin alanını hesaplayan fonksiyon (A = pi* r^2).
- `double cevre() const`: Dairenin çevresini hesaplayan fonksiyon (C = 2 pi* r).
- `int kesisim(const Daire& dairee) const`: İki dairenin kesişim durumunu belirten fonksiyon. 0: İç içe daireler, 1: Kenetlenen daireler, 2: Tamamen ayrık daireler.
- `std::string toString() const`: Dairenin bilgilerini içeren bir metin dizesini döndüren fonksiyon.
- `void yazdir() const`: Dairenin bilgilerini ekrana yazdıran fonksiyon.

4. SONUÇ VE ÖĞRENİLEN DERSLER

Bu ödemizde bizden istenenilenleri yaptıktan sonra şunları öğrenmiş olduk

-Sınıf kullanımı

-Sınıfların birbirine entegresi

-Sınıfları Gerekli fonksiyonlar ile bağlanması

-Sınıfları gerekli şlemlerde veri almak için kullanmak

Ve en önemlisi bu ödevde C++ bilgimiz artmıştır ve pratiğimiz artmıştır.

5. KAYNAKÇA

https://www.yusufsezer.com.tr/cpp-class-ornekleri/

https://aslihanakbiyik.medium.com/c-i%CC%87le-nesne-y%C3%B6nelimli-programlama-9605e3a150c

https://canerkaradag.com/cpp/cpp-classes-and-objects/

https://www.turkhackteam.org/konular/c-kutuphanelerin-kullanim-alanlari.1576471/