T.C
KOCAELİ SAĞLIK VE TEKNOLOJİ
ÜNİVERSİTESİ DOĞA BİLİMLERİ VE
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ
PROGRAMI

GEOMETRİK PROBLEMLER

HAZIRLAYANLAR

ANIL ERDOĞAN – 220501006
GitHub Link: https://github.com/Kwalmm

AKIN TURAN – 220501013
GitHub Link: https://github.com/coldwraith44

DERS SORUMLUSU
ARŞ. GÖR. Hüseyin Tarık DURU

Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli
info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

30.12.2023

İÇİNDEKİLER

ÖZET

Bu çalışmada bizden istenen geometrik problemlerin çözümü bunlara dair olan sınıfların yardımıyla çözülmüştür. Proje bir adet test dosyası ve 4 adet sınıfımızı oluşturan dosyalardan oluşturmaktadır.

1. GİRİŞ

Geometrik olarak verilmiş bazı noktasal, şekilsel vb. Problemlerin c++ üzerinde uyarlanmasını göreceğiz.

2. YÖNTEM

Problemlerin nasıl çözüldüğüne burada birlikte bakacağız.

2.1) Sınıfların tanımlanması

2.1.1) Nokta sınıfı

Nokta sınıfı noktanın x ve y kordinatlarını tutar.

2.1.2) Doğru parçası sınıfı
2 Nokta nesnesi arasındaki bir doğru parçasını tutar.

2.1.3) Daire sınıfı
Bu sınıf bir nokta nesnesi ve bir yarıçapı tutar.

2.1.4) Üçgen sınıfı
Son olarak bu sınıf 3 nokta nesnesini tutar.

2.2) Sonuç
Geometrik şekilleri ve bunların özelliklerini tutan sınıflar oluşturduk ve test dosyası ile birlikte üzerinde işlemler yaptık.

3) Kodun son hali
3.1) Nokta.h
```
#include <iostream>
#include <string>
#ifndef NOKTA_H
#define NOKTA_H

class Nokta {
private:
    double x;
    double y;

public:
    // parametresiz yapıcı
    Nokta() : x(0), y(0) {}

    // tek parametreli yapıcı
    Nokta(double val) : x(val), y(val) {}

    // iki parametreli yapıcı
    Nokta(double x, double y) : x(x), y(y) {}

    // başka bir noktayı alıp o noktanın bir kopyasını yeni nokta olarak üreten yapıcı
    Nokta(const Nokta& nokta) : x(nokta.x), y(nokta.y) {}

    // başka bir nokta ve iki double değişken alarak yeni bir nokta üretir
    Nokta(const Nokta& nokta, double ofset_x, double ofset_y) : x(nokta.x + ofset_x),
y(nokta.y + ofset_y) {}

    // x ve y koordinatları için get ve set metotları
    double getX() const { return x; }
    double getY() const { return y; }
    void setX(double x) { this->x = x; }
    void setY(double y) { this->y = y; }

    // aynı anda iki koordinat alan ve noktanın x ve y koordinatlarının değerlerini
değiştiren bir set metodu
    void set(double x, double y) {
        this->x = x;
        this->y = y;
    }
```

```
    // toString metodu noktanın koordinatlarının string gösterimini döndürür
    std::string toString() const {
        return "(" + std::to_string(x) + ", " + std::to_string(y) + ")";
    }

    // yazdir metodu: toString metodunu kullanarak ekrana koordinatları yazdırır
    void yazdir() const {
        std::cout << toString() << std::endl;
    }
};
#endif
```

3.2) DogruParcasi.h

```
#include "Nokta.h"
#include <cmath>
#include <iostream>
#ifndef DOGRUPARCASI_H
#define DOGRUPARCASI_H
class DogruParcasi {
private:
    Nokta p1;
    Nokta p2;

public:
    // iki uç noktayı nokta nesnesi olarak alan yapıcı
    DogruParcasi(const Nokta& p1, const Nokta& p2) : p1(p1), p2(p2) {}

    // başka bir DogruParcasi nesnesi alıp onun bir kopyasını yeni bir DogruParcasi
nesnesi olarak oluşturan yapıcı
    DogruParcasi(const DogruParcasi& dp) : p1(dp.p1), p2(dp.p2) {}

    // bir nokta nesnesi,parçanın uzunluğu ve eğimi değerlerini alarak doğru
parçasının x ve y koordinatlarını hesaplayan yapıcı
    DogruParcasi(const Nokta& ortaNokta, double uzunluk, double egim) {
        double deltaX = uzunluk / 2 * cos(egim);
        double deltaY = uzunluk / 2 * sin(egim);
        p1.set(ortaNokta.getX() - deltaX, ortaNokta.getY() - deltaY);
        p2.set(ortaNokta.getX() + deltaX, ortaNokta.getY() + deltaY);
    }

    // ilgili get ve set metotları
    Nokta getP1() const { return p1; }
    Nokta getP2() const { return p2; }
    void setP1(const Nokta& p) { p1 = p; }
    void setP2(const Nokta& p) { p2 = p; }

    // uzunluk metodu: DogruParcasi nesnesinin uzunluğunun double değişken olarak
hesaplar ve döndürür
    double uzunluk() const {
        return sqrt(pow(p2.getX() - p1.getX(), 2) + pow(p2.getY() - p1.getY(), 2));
    }

    // 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
    Nokta kesisimNoktası(const Nokta& p) const {
        double m1 = (p2.getY() - p1.getY()) / (p2.getX() - p1.getX());
        double m2 = -1 / m1;
        double x = (m2 * p.getX() - m1 * p1.getX() + p1.getY() - p.getY()) / (m2 -
m1);
        double y = m2 * (x - p.getX()) + p.getY();
        return Nokta(x, y);
    }
```

```
    // ortaNokta metodu: doğru parçasının orta noktasını hesaplayan ve bir Nokta
nesnesi olarak döndürür
    Nokta ortaNokta() const {
        return Nokta((p1.getX() + p2.getX()) / 2, (p1.getY() + p2.getY()) / 2);
    }

    // toString yöntemi: geçerli LineSegment nesnesinin String olarak döndürme. Nesne
sınıfının toString yöntemini kullanmalı
    std::string toString() const {
        return "DogruParcasi: " + p1.toString() + " - " + p2.toString();
    }

    // yazdir metodu: iki uç noktayı toString metodunu kullanarak ekrana yazdırır
    void yazdir() const {
        std::cout << toString() << std::endl;
    }
};
#endif
```

3.3) Daire.h

```
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif
#ifndef DAIRE_H
#define DAIRE_H
#include "Nokta.h"
#include <cmath>
#include <iostream>

class Daire {
private:
    Nokta merkez;
    double yaricap;

public:
    // merkez ve yarıçapı parametre olarak alan yapıcı
    Daire(const Nokta& merkez, double yaricap) : merkez(merkez), yaricap(yaricap) {}

    // başka bir Daire nesnesi alıp onun bir kopyasını yeni bir daire nesnesi olarak
oluşturan yapıcı
    Daire(const Daire& d) : merkez(d.merkez), yaricap(d.yaricap) {}

    // başka bir daire nesnesi ve reel bir pozitif x değeri alarak parametre olarak
alınan daire nesnesini yarıçapı x ile çarpılmış olarak kopyalayan yapıcı
    Daire(const Daire& d, double x) : merkez(d.merkez), yaricap(d.yaricap* x) {}

    // alan metodu: dairenin alanını döndürür
    double alan() const {
        return M_PI * pow(yaricap, 2);
    }

    // cevre metodu: dairenin çevresini döndürür
    double cevre() const {
        return 2 * M_PI * yaricap;
    }

    // kesisim metodu: bir daire nesnesi alır ve parametre olarak gelen daire metodu
çağıran dairenin içinde ise 0 daireler birebir örtüşüyorsa 1 hiç kesişim yoksa 2
değerini döndürür
    int kesisim(const Daire& d) const {
        double mesafe = sqrt(pow(d.merkez.getX() - merkez.getX(), 2) +
pow(d.merkez.getY() - merkez.getY(), 2));
        if (mesafe < abs(yaricap - d.yaricap)) {
            return 0; // iç içe
```

```
        }
        else if (mesafe == abs(yaricap - d.yaricap) || mesafe == yaricap + d.yaricap)
{
            return 1; // birebir örtüşüyor
        }
        else {
            return 2; // hiç kesişmiyor
        }
    }

    // toString metodu: dairenin merkezi ve yarıçapını string halinde döndürür
    std::string toString() const {
        return "Daire: Merkez = " + merkez.toString() + ", Yarıçap = " +
std::to_string(yaricap);
    }

    // yazdir metodu: toString metodunu kullanarak ekrana direnin bilgilerini yazdırır
    void yazdir() const {
        std::cout << toString() << std::endl;
    }
};
#endif
```

3.4) Ucgen.h

```
#include "Nokta.h"
#include "DogruParcasi.h"
#include <cmath>
#include <iostream>
#ifndef UCGEN_H
#define UCGEN_H
class Ucgen {
private:
    Nokta p1;
    Nokta p2;
    Nokta p3;

public:
    // üç tane Nokta nesnesi alan yapıcı
    Ucgen(const Nokta& p1, const Nokta& p2, const Nokta& p3) : p1(p1), p2(p2), p3(p3)
{}

    // ilgili get ve set metotları
    Nokta getP1() const { return p1; }
    Nokta getP2() const { return p2; }
    Nokta getP3() const { return p3; }
    void setP1(const Nokta& p) { p1 = p; }
    void setP2(const Nokta& p) { p2 = p; }
    void setP3(const Nokta& p) { p3 = p; }

    // toString metodu: Üçgenin string temsilini şu şekilde döndürür: üçgen , onun
noktaları
    std::string toString() const {
        return "Ucgen; onun noktaları: " + p1.toString() + ", " + p2.toString() + ", "
+ p3.toString();
    }

    // alan metodu: bu üç noktanın temsil ettiği üçgenin alanını hesaplar ve döndürür
    double alan() const {
        return abs((p1.getX() * (p2.getY() - p3.getY()) + p2.getX() * (p3.getY() -
p1.getY()) + p3.getX() * (p1.getY() - p2.getY())) / 2.0);
    }
    // cevre metodu: üçgenin çevresini hesaplar ve döndürür. Yöntem içerisinde üç
doğru parçası oluşturur ve DogruParcasi sınıfının uzunluk metodunu kullanırd
    double cevre() const {
```

```
        DogruParcasi dp1(p1, p2);
        DogruParcasi dp2(p2, p3);
        DogruParcasi dp3(p3, p1);
        return dp1.uzunluk() + dp2.uzunluk() + dp3.uzunluk();
    }

    // acilar metodu: üçgenin açılarını üç öğeli double dizi olarak hesaplar ve
döndürür
    double* acilar() const {
        double* acilar = new double[3];
        double a = DogruParcasi(p2, p3).uzunluk();
        double b = DogruParcasi(p1, p3).uzunluk();
        double c = DogruParcasi(p1, p2).uzunluk();
        acilar[0] = acos((b * b + c * c - a * a) / (2 * b * c));
        acilar[1] = acos((a * a + c * c - b * b) / (2 * a * c));
        acilar[2] = acos((a * a + b * b - c * c) / (2 * a * b));
        return acilar;
    }
};
#endif
```

4) Kaynakça

https://stackoverflow.com
https://medium.com/
https://blogs.sas.com
https://www.sanfoundry.com
https://chat.openai.com
https://bing.ai.com