T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ
YAZILIM MÜHENDİSLİĞİ PROGRAMI

ÖDEV KONUSU

GEOMETRİK PROBLEMLER

Hazırlayan
Senem ADALAN - 220502045
Sema Su YILMAZ - 220502016

DERS SORUMLUSU
PROF. DR. HÜSEYİN TARIK DURU

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

2 OCAK 2024

İÇİNDEKİLER

1. ÖZET

Bu C++ programı, geometrik problemleri çözmek için Nokta, DogruParcasi, Daire ve Ucgen adlı dört farklı sınıf içermektedir.Her sınıf, geometrik nesnelerin özelliklerini ve işlevselliğini temsil eder.Her sınıf özel yapıcılar, get/set metotları ve çeşitli geometrik hesaplamalar için fonksiyonlar içerir.

2. GİRİŞ

```
NOKTA SINIFI İÇİN TEST ÇIKTILARI:
Oluşturulan 5 noktanın x ve y değerleri:
(0.00, 0.00)
(5.00, 5.00)
(3.00, 4.00)
(3.00, 4.00)
(4.00, 3.00)
1. ve 2. noktaların güncel x ve y değerleri:
(10.00, 0.00)
(25.00, 30.00)

DOGRUPARCASI SINIFI İÇİN TEST SONUÇLARI:
Oluşturulan 3 doğru parçasının başlangıç ve bitiş noktaları:
(0.00, 0.00) - (3.00, 4.00)
(0.00, 0.00) - (3.00, 4.00)
(-0.54, 0.46) - (6.54, 7.54)
2. doğru parçasının başlangıç ve bitiş noktaları:
(0.00, 0.00) - (0.00, 0.00)
1. doğru parçası için uzunluk değeri: 5
1. doğru parçası için orta nokta: (1.50, 2.00)

DAIRE SINIFI İÇİN TEST SONUÇLARI:
Daire 1: Merkez: (0.00, 0.00), Yarıçap: 5.00
Alan: 78.5398, Çevre: 31.4159
Daire 2: Merkez: (0.00, 0.00), Yarıçap: 5.00
Alan: 78.5398, Çevre: 31.4159
Daire 3: Merkez: (0.00, 0.00), Yarıçap: 15.00
Alan: 706.858, Çevre: 94.2478
Daire 1 ve Daire 2 Kesişim Durumu: 1
Daire 1 ve Daire 3 Kesişim Durumu: 0

UCGEN SINIFI İÇİN TEST SONUÇLARI:
Üçgen; (0.00, 0.00), (4.00, 0.00), (0.00, 3.00)
Alan: 6
Çevre: 12
Açılar: 90, 36.8699, 53.1301
```

Bu projenin amacı geometrik problemleri modelleyen ve çözen bir C++ programı geliştirmektir.Projede kullanılan sınıflar (Nokta, DogruParcasi, Daire, Ucgen) geometrik nesnelerin temsilini sağlar ve bu nesnelerle ilgili temel işlemleri gerçekleştirmek üzere tasarlanmıştır.Kullanıcı, bu sınıfları kullanarak geometrik problemleri çözebilir ve farklı nesneler arasındaki ilişkileri keşfeder.

3. YÖNTEM

Aşağıda nokta.h dosyası verilmiştir.Bu C++ kodunda bir "Nokta" sınıfı tanımlanır.Header dosyasında tanımlanan işlemler aşağıdaki gibidir;

double x: Noktanın x koordinatını tutan özel bir üye değişken.

double y: Noktanın y koordinatını tutan özel bir üye değişken.

Nokta(): Noktayı varsayılan olarak (0,0) noktasına yerleştiren yapıcı fonksiyon.

Nokta(double value): Tek bir değer alarak hem x hem de y koordinatlarını belirten yapıcı fonksiyon.

Nokta(double xvalue, double yvalue): İki ayrı değer alarak x ve y koordinatlarını belirten yapıcı fonksiyon.

Nokta(const Nokta& other): Başka bir Nokta nesnesini kopyalayan kopya yapıcı fonksiyon.

Nokta(const Nokta& other, double ofset_x, double ofset_y): Diğer bir Nokta nesnesini belirtilen ofset değerleri ile kaydıran yapıcı fonksiyon.

void setX(double xvalue): x koordinatını ayarlayan fonksiyon.

double getX() const: x koordinatını döndüren fonksiyon.

void setY(double yvalue): y koordinatını ayarlayan fonksiyon.

double getY() const: y koordinatını döndüren fonksiyon.

void set(double xvalue, double yvalue): Hem x hem de y koordinatlarını aynı anda ayarlayan fonksiyon.

std::string toString(): Noktanın x ve y koordinatlarını içeren bir string döndüren fonksiyon.

void yazdir(): Noktanın koordinatlarını ekrana yazdıran fonksiyon.

```
#pragma once
#include <string>
using namespace std;

class Nokta {
private:
    double x;
    double y;

public:
    Nokta();
    Nokta(double value);
    Nokta(double xvalue, double yvalue);
    Nokta(const Nokta& other);
    Nokta(const Nokta& other, double ofset_x, double ofset_y);

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

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

    void set(double xvalue, double yvalue);

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

```

Aşağıda nokta.cpp dosyası verilmiştir.Bu C++ kodunda bir "Nokta" sınıfı ile işlemler gerçekleştirilir.Kaynak dosyasında tanımlanan işlemler aşağıdaki gibidir;

Nokta::Nokta(): Bu yapıcı fonksiyon, parametresiz olarak çağrıldığında bir Nokta nesnesini varsayılan değerlerle (0.0, 0.0) başlatır.

Nokta::Nokta(double value): Bu yapıcı fonksiyon, tek bir değer alarak hem x hem de y koordinatlarını belirler. Bu durumda, her iki koordinat da aynı değere atanır.

Nokta::Nokta(double xvalue, double yvalue): Bu yapıcı fonksiyon, iki ayrı değer alarak x ve y koordinatlarını belirler.

Nokta::Nokta(const Nokta& other): Bu yapıcı fonksiyon, başka bir Nokta nesnesinin x ve y koordinatlarını alarak yeni bir Nokta nesnesini oluşturur.

Nokta::Nokta(const Nokta& other, double ofset_x, double ofset_y): Bu yapıcı fonksiyon, başka bir Nokta nesnesini ve ofset değerlerini alarak yeni bir Nokta nesnesini oluşturur. Bu ofset değerleri, belirtilen noktanın koordinatlarına eklenir.

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

// Parametresiz yapıcı
Nokta::Nokta() {
    setX(0.0);
    setY(0.0);
}

// Tek parametreli yapıcı
Nokta::Nokta(double value) {
    setX(value);
    setY(value);
}

// İki parametreli yapıcı
Nokta::Nokta(double xvalue, double yvalue) {
    setX(xvalue);
    setY(yvalue);
}

// Başka bir noktayı alıp o noktanın kopyasını yeni nokta olarak üreten yapıcı
Nokta::Nokta(const Nokta& other) {
    setX(other.getX());
    setY(other.getY());
}

// Başka bir nokta nesnesi ve ofset değerleri ile yeni nokta üreten yapıcı
Nokta::Nokta(const Nokta& other, double ofset_x, double ofset_y) {
    setX(other.getX() + ofset_x);
    setY(other.getY() + ofset_y);
}
```

void Nokta::setX(double xvalue): Bu metot, bir Nokta nesnesinin x koordinatını ayarlamak için kullanılır. xvalue parametresi, ayarlanmak istenen x koordinat değeridir.

double Nokta::getX() const: Bu metot, bir Nokta nesnesinin x koordinatını almak için kullanılır. Fonksiyon, x koordinatının değerini geri döndürür.

void Nokta::setY(double yvalue): Bu metot, bir Nokta nesnesinin y koordinatını ayarlamak için kullanılır. yvalue parametresi, ayarlanmak istenen y koordinat değeridir.

double Nokta::getY() const: Bu metot, bir Nokta nesnesinin y koordinatını almak için kullanılır. Fonksiyon, y koordinatının değerini geri döndürür.

```
36    // x koordinatı için set metodu
37   void Nokta::setX(double xvalue) {
38       x = xvalue;
39   }
40
41    // x koordinatı için get metodu
42   double Nokta::getX() const {
43       return x;
44   }
45
46    // y koordinatı için set metodu
47   void Nokta::setY(double yvalue) {
48       y = yvalue;
49   }
50
51    // y koordinatı için get metodu
52   double Nokta::getY() const {
53       return y;
54   }
```

void Nokta::set(double xvalue, double yvalue): Bu metot, hem x hem de y koordinatlarını aynı anda değiştirmek için kullanılır. xvalue ve yvalue parametreleri, yeni x ve y koordinat değerleridir. Bu metot, setX ve setY metodlarını çağırarak noktanın koordinatlarını günceller.

string Nokta::toString(): Bu metot, bir Nokta nesnesini string formatına çeviren bir metottur. fixed ve setprecision kullanılarak, noktanın x ve y koordinatlarını iki ondalık basamağa kadar düzenli bir şekilde gösterir. Sonuç olarak, bir string olarak döndürülür.

void Nokta::yazdir(): Bu metot, Nokta nesnesini ekrana yazdırmak için kullanılır. toString metodu ile elde edilen string ifade cout ile ekrana basılır, ardından bir satır atlaması sağlanır.

```
56    // Aynı anda iki koordinatı alan ve noktanın x ve y koordinatlarını değiştiren set metodu
57   void Nokta::set(double xvalue, double yvalue) {
58       setX(xvalue);
59       setY(yvalue);
60   }
61
62    // toString metodu
63   string Nokta::toString() {
64       ostringstream oss;
65       oss << fixed << setprecision(2) << "(" << x << ", " << y << ")";
66       return oss.str();
67   }
68
69    // yazdir metodu
70   void Nokta::yazdir() {
71       cout << toString() << endl;
72   }
```

Aşağıda dogruparcasi.h dosyası verilmiştir.Bu C++ kodunda bir "DogruParcasi" sınıfı tanımlanır.Header dosyasında tanımlanan işlemler aşağıdaki gibidir;

Nokta basNokta: Doğru parçasının başlangıç noktasını temsil eden bir Nokta nesnesi.

Nokta sonNokta: Doğru parçasının bitiş noktasını temsil eden bir Nokta nesnesi.

DogruParcasi(const Nokta& baslangicNokta, const Nokta& bitisNokta): Başlangıç ve bitiş noktalarını alan bir yapıcı fonksiyon.

DogruParcasi(const DogruParcasi& other): Bir başka DogruParcasi nesnesini kopyalayan kopya yapıcı fonksiyon.

DogruParcasi(const Nokta& midNokta, double uzunluk, double aciDerece): Orta noktayı, uzunluğu ve açıyı kullanarak doğru parçası oluşturan yapıcı fonksiyon.

void setbasNokta(const Nokta& baslangicNokta): Başlangıç noktasını ayarlayan fonksiyon.

Nokta getbasNokta() const: Başlangıç noktasını döndüren fonksiyon.

void setsonNokta(const Nokta& bitisNokta): Son noktayı ayarlayan fonksiyon.

Nokta getsonNokta() const: Son noktayı döndüren fonksiyon.

void setP1(const Nokta& baslangicVeBitisNokta): Hem başlangıç hem de bitiş noktalarını aynı anda ayarlayan fonksiyon.

double uzunluk(): Doğru parçasının uzunluğunu hesaplayan fonksiyon.

Nokta DikKesisimNoktasiBul(const Nokta& verilenNokta) const: Verilen bir noktaya dik kesişen noktayı bulan fonksiyon.

Nokta ortaNokta(): Doğru parçasının orta noktasını bulan fonksiyon.

string toString(): Doğru parçasının başlangıç ve son noktalarını içeren bir string döndüren fonksiyon.

void yazdir(): Doğru parçasının başlangıç ve son noktalarını ekrana yazdıran fonksiyon.

```cpp
#pragma once
#include "nokta.h"
using namespace std;

//Nesne değişkenleri olarak bir doğru parçasının iki noktasını (Nokta nesnesi olarak) içeren DogruParcasi sınıfı
class DogruParcasi {
private:
    Nokta basNokta;
    Nokta sonNokta;

public:
    DogruParcasi(const Nokta& baslangicNokta, const Nokta& bitisNokta);
    DogruParcasi(const DogruParcasi& other);
    DogruParcasi(const Nokta& midNokta, double uzunluk, double aciDerece);

    void setbasNokta(const Nokta& baslangicNokta);
    Nokta getbasNokta() const;

    void setsonNokta(const Nokta& bitisNokta);
    Nokta getsonNokta() const;

    void setP1(const Nokta& baslangicVeBitisNokta);

    double uzunluk();
    Nokta DikKesisimNoktasiBul(const Nokta& verilenNokta) const;
    Nokta ortaNokta();
    string toString();
    void yazdir();
};
```

Aşağıda dogruparcasi.cpp dosyası verilmiştir.Bu C++ kodunda bir "DogruParcasi" sınıfı ile işlemler gerçekleştirilir.Kaynak dosyasında tanımlanan işlemler aşağıdaki gibidir;

DogruParcasi::DogruParcasi(const Nokta& baslangicNokta, const Nokta& bitisNokta): Bu yapıcı fonksiyon, iki Nokta nesnesini alarak doğru parçasının başlangıç ve bitiş noktalarını oluşturur. Gelen Nokta nesneleri, sınıfın üye değişkenleri basNokta ve sonNokta'ya atanır.

DogruParcasi::DogruParcasi(const DogruParcasi& other): Bu yapıcı fonksiyon, başka bir DogruParcasi nesnesini alarak, bu nesnenin kopyasını oluşturur. Gelen nesnenin üye değişkenleri, sınıfın üye değişkenlerine atanır.

DogruParcasi::DogruParcasi(const Nokta& midNokta, double uzunluk, double aciDerece): Bu yapıcı fonksiyon, bir Nokta nesnesi (midNokta), bir uzunluk (uzunluk) ve bir açı (aciDerece) alarak, bu bilgileri kullanarak doğru parçasının başlangıç ve bitiş noktalarını hesaplar. Açı, radyan cinsine çevrilir ve trigonometrik fonksiyonlar kullanılarak yatay ve dikey bileşenler hesaplanır. Sonuç olarak, başlangıç ve bitiş noktaları atanır.

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

// İki uç noktayı Nokta nesnesi olarak alan yapıcı
DogruParcasi::DogruParcasi(const Nokta& baslangicNokta, const Nokta& bitisNokta) {
    basNokta = baslangicNokta;
    sonNokta = bitisNokta;
}

// Başka DogruParcasi nesnesi alıp kopyasını yeni DogruParcasi nesnesi olarak oluşturan yapıcı
DogruParcasi::DogruParcasi(const DogruParcasi& other) {
    basNokta = other.basNokta;
    sonNokta = other.sonNokta;
}

// Bir Nokta nesnesi, uzunluk ve eğim alarak doğru parçasının uç noktalarını hesaplayan yapıcı
DogruParcasi::DogruParcasi(const Nokta& midNokta, double uzunluk, double aciDerece) {
    // Derecenin radyan cinsine dönüştürülmesi
    double aciRadyan = aciDerece * M_PI / 180.0;

    // Açıyı kullanarak eğim değerinin hesaplanması
    double egim = tan(aciRadyan);

    // Yatay ve dikey bileşenlerin hesaplanması
    double yatayBilesen = uzunluk * cos(aciRadyan) / 2.0;
    double dikeyBilesen = uzunluk * sin(aciRadyan) / 2.0;

    // Başlangıç noktasını ve bitiş noktasının ayarlanması
    basNokta = Nokta(midNokta.getX() - yatayBilesen, midNokta.getY() - dikeyBilesen);
    sonNokta = Nokta(midNokta.getX() + yatayBilesen, midNokta.getY() + dikeyBilesen);
}
```

void DogruParcasi::setbasNokta(const Nokta& baslangicNokta): Bu metot, bir Nokta nesnesi alarak doğru parçasının başlangıç noktasını ayarlar.

Nokta DogruParcasi::getbasNokta() const: Bu metot, doğru parçasının başlangıç noktasını döndürür.

void DogruParcasi::setsonNokta(const Nokta& bitisNokta):Bu metot, bir Nokta nesnesi alarak doğru parçasının bitiş noktasını ayarlar.

Nokta DogruParcasi::getsonNokta() const:Bu metot, doğru parçasının bitiş noktasını döndürür.

void DogruParcasi::setP1(const Nokta& baslangicVeBitisNokta): Bu metot, bir Nokta nesnesi alarak hem başlangıç hem de bitiş noktalarını aynı anda ayarlar. Yani, başlangıç noktasını belirler ve bitiş noktasını başlangıç noktasına eşitler.

double DogruParcasi::uzunluk(): Bu metot, doğru parçasının uzunluğunu hesaplamak için kullanılır. İki nokta arasındaki uzaklığı hesaplamak için Euclidean uzaklık formülünü kullanır.

```
34
35   // Başlangıç noktası için set metodu
36   void DogruParcasi::setbasNokta(const Nokta& baslangicNokta) {
37       basNokta = baslangicNokta;
38   }
39
40   // Başlangıç noktası için get metodu
41   Nokta DogruParcasi::getbasNokta() const {
42       return basNokta;
43   }
44
45   // Bitiş noktası için set metodu
46   void DogruParcasi::setsonNokta(const Nokta& bitisNokta) {
47       sonNokta = bitisNokta;
48   }
49
50   // Bitiş noktası için get metodu
51   Nokta DogruParcasi::getsonNokta() const {
52       return sonNokta;
53   }
54
55   // Başlangıç ve bitiş noktalarının tutulması için yazılan setP1 metodu
56   void DogruParcasi::setP1(const Nokta& baslangicVeBitisNokta) {
57       basNokta = baslangicVeBitisNokta;
58       sonNokta.setX(basNokta.getX());
59       sonNokta.setY(basNokta.getY());
60   }
61
62   // Uzunluk hesaplama metodu
63   double DogruParcasi::uzunluk() {
64       double deltaX = sonNokta.getX() - basNokta.getX();
65       double deltaY = sonNokta.getY() - basNokta.getY();
66       return sqrt(pow(deltaX, 2) + pow(deltaY, 2));
67   }
```

Başlangıç ve bitiş noktaları arasındaki vektörü temsil eden dogruParcaX ve dogruParcaY hesaplanır.

Doğru parçasının birim vektörü, doğru parçasının uzunluğuna bölünerek hesaplanır.

Verilen noktanın doğru parçasına dik olan vektör, verilen noktanın başlangıç noktasına olan vektör farkını temsil eder.

Bu adımda iç çarpım kullanılarak, verilen noktanın doğru parçasına dik olan uzaklığı hesaplanır.

Doğru parçasının üzerindeki kesişim noktası, başlangıç noktasına dik olan uzaklığın doğru parçasının başlangıç noktasından olan vektörle çarpılmasıyla hesaplanır.

Son olarak, hesaplanan kesişim noktası değerleri kullanılarak yeni bir Nokta nesnesi oluşturulur ve bu nokta nesnesi döndürülür.

```
69    // KesişimNoktası hesaplama metodu
70    Nokta DogruParcasi::DikKesisimNoktasiBul(const Nokta& verilenNokta) const {
71        // Doğru parçasının uzunluk vektörünün hesaplanması
72        double dogruParcaX = sonNokta.getX() - basNokta.getX();
73        double dogruParcaY = sonNokta.getY() - basNokta.getY();
74
75        // Doğru parçasının birim vektörünün hesaplanması
76        double dogruParcaUzunluk = sqrt(dogruParcaX * dogruParcaX + dogruParcaY * dogruParcaY);
77        double birimDogruParcaX = dogruParcaX / dogruParcaUzunluk;
78        double birimDogruParcaY = dogruParcaY / dogruParcaUzunluk;
79
80        // Verilen noktanın doğru parçasına dik olan vektörünün hesaplanması
81        double dikVectorX = verilenNokta.getX() - basNokta.getX();
82        double dikVectorY = verilenNokta.getY() - basNokta.getY();
83
84        // Verilen noktanın doğru parçasına dik olan uzaklığının hesaplanması
85        double uzaklik = dikVectorX * birimDogruParcaX + dikVectorY * birimDogruParcaY;
86
87        // Doğru parçasının üzerindeki kesişim noktasının hesaplanması
88        double kesisimNoktasiX = basNokta.getX() + uzaklik * birimDogruParcaX;
89        double kesisimNoktasiY = basNokta.getY() + uzaklik * birimDogruParcaY;
90
91
92        // Yeni Nokta nesnesini oluştur ve döndür
93        return Nokta(kesisimNoktasiX, kesisimNoktasiY);
94    }
```

Nokta DogruParcasi::ortaNokta(): Bu metot, doğru parçasının orta noktasını hesaplar. Basitçe, başlangıç noktasının ve bitiş noktasının x ve y koordinatlarını toplar, her birini ikiyle böler ve bu değerleri kullanarak yeni bir Nokta nesnesi oluşturur.

string DogruParcasi::toString(): Bu metot, doğru parçasının başlangıç noktası ve bitiş noktasının toString metotlarını kullanarak birleştirir ve bu şekilde doğru parçasının string temsilini oluşturur.

void DogruParcasi::yazdir(): Bu metot, toString metotunu kullanarak doğru parçasının string temsilini oluşturur ve ardından bu stringi ekrana yazdırır.

```cpp
// Orta noktayı hesaplayan metot
Nokta DogruParcasi::ortaNokta() {
    return Nokta((basNokta.getX() + sonNokta.getX()) / 2, (basNokta.getY() + sonNokta.getY()) / 2);
}

// Nokta sınıfındaki toString metodu kullanılarak yapılan toString yöntemi
string DogruParcasi::toString() {
    return basNokta.toString() + " - " + sonNokta.toString();
}

// Ekrana yazdırma metodu
void DogruParcasi::yazdir() {
    cout << toString() << endl;
}

```

Aşağıda daire.h dosyası verilmiştir.Bu C++ kodunda bir "Daire" sınıfı tanımlanır.Header dosyasında tanımlanan işlemler aşağıdaki gibidir;

Nokta merkezNokta: Bir dairenin merkezini temsil eden bir Nokta nesnesi.

double yaricap: Bir dairenin yarıçapını tutan bir double değişkeni.

Daire(const Nokta& merkez, double yaricapvalue): Merkez noktayı ve yarıçapı alan yapıcı fonksiyon.

Daire(const Daire& other): Başka bir Daire nesnesini kopyalayan kopya yapıcı fonksiyon.

Daire(const Daire& other, double x_carpani): X koordinatındaki bir çarpanı kullanarak başka bir Daire nesnesini kopyalayan yapıcı fonksiyon.

double alan(): Dairenin alanını hesaplayan fonksiyon.

double cevre(): Dairenin çevresini hesaplayan fonksiyon.

int kesisim(const Daire& other): Verilen başka bir Daire ile kesişim durumunu kontrol eden fonksiyon (0: Kesişim yok, 1: İç içe, 2: Dışarda, 3: Çakışık).

string toString(): Dairenin merkezini ve yarıçapını içeren bir string döndüren fonksiyon.

void yazdir(): Dairenin merkezini ve yarıçapını ekrana yazdıran fonksiyon.

```cpp
#pragma once
#include "nokta.h"
using namespace std;

//Dairenin merkezi (Nokta nesnesi olarak) ve yarıçapını nesne değişkenleri olarak tutan Daire sınıfı
class Daire {
private:
    Nokta merkezNokta;
    double yaricap;

public:
    Daire(const Nokta& merkez, double yaricapvalue);
    Daire(const Daire& other);
    Daire(const Daire& other, double x_carpani);

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

    string toString();
    void yazdir();
};
```

Daire::Daire(const Nokta& merkez, double yaricapvalue): Bu, merkez noktasını ve yarıçapı parametre olarak alan yapıcı metodudur. Dairenin merkezi ve yarıçapı bu metot aracılığıyla belirlenir.

Daire::Daire(const Daire& other): Başka bir daire nesnesini kopyalayarak yeni bir daire nesnesi oluşturan kopya yapıcı metodudur.

Daire::Daire(const Daire& other, double x_carpani): Başka bir daire nesnesini ve bir çarpan değerini alarak, parametre alınan daire nesnesinin yarıçapını çarpanla çarpılmış olarak kopyalayan yapıcı metodudur.

double Daire::alan(): Bu metot, dairenin alanını hesaplar ve geri döndürür. Alan hesaplamasında, π sayısı ve yarıçap kullanılarak alan formülü uygulanmıştır.

double Daire::cevre(): Bu metot, dairenin çevresini hesaplar ve geri döndürür. Çevre hesaplamasında, π sayısı ve yarıçap kullanılarak çevre formülü uygulanmıştır.

```cpp
#define _USE_MATH_DEFINES
#include "daire.h"
#include <cmath>
#include <iostream>
#include <iomanip>
#include <sstream>

// Merkez (Nokta nesnesi olarak) ve yarıçapı parametre olarak alan yapıcı.
Daire::Daire(const Nokta& merkez, double yaricapvalue) {
    merkezNokta = merkez;
    yaricap = yaricapvalue;
}

// Başka bir Daire nesnesi alıp kopyasını yeni bir Daire nesnesi olarak oluşturan yapıcı.
Daire::Daire(const Daire& other) {
    merkezNokta = other.merkezNokta;
    yaricap = other.yaricap;
}

// Başka bir Daire nesnesi ve x değeri alarak, parametre alınan Daire nesnesini yarıçapı x ile çarpılmış olarak kopyalayan yapıcı
Daire::Daire(const Daire& other, double x_carpani) {
    merkezNokta = other.merkezNokta;
    yaricap = other.yaricap * x_carpani;
}

// Alan hesaplama metodu
double Daire::alan() {
    return M_PI * pow(yaricap, 2);
}

// Çevre hesaplama metodu
double Daire::cevre() {
    return 2 * M_PI * yaricap;
}
```

Mesafe Hesaplanması:double mesafe adında bir değişken oluşturulur ve iki daire merkezi arasındaki mesafe bu değişkene atanır.

Tam Örtüşme Durumu:Eğer iki dairenin merkezleri birbirine eşit ve yarıçapları da eşitse (tamamen örtüşme durumu), return 1; ile 1 değeri döndürülür.

İç İçe Geçme Durumu:Eğer iki daire arasındaki mesafe, yarıçapların farkından daha küçükse, daireler birbirine iç içe geçmiştir ve return 0; ile 0 değeri döndürülür.

Normal Kesişme Durumu:Eğer iki daire arasındaki mesafe, iki dairenin yarıçaplarının toplamından daha küçük ve yarıçapların farkından daha büyükse daireler birbirine normal şekilde kesişir ve return 1; ile 1 değeri döndürülür.

Hiç Kesişme Yoksa:Hiçbir durum sağlanmıyorsa, daireler hiç kesişmemiştir ve return 2; ile 2 değeri döndürülür.

```
36    // Kesişim kontrol metodu
37    int Daire::kesisim(const Daire& other) {
38        // İki daire arasındaki mesafenin hesaplanması
39        double mesafe = sqrt(pow(merkezNokta.getX() - other.merkezNokta.getX(), 2) +
40            pow(merkezNokta.getY() - other.merkezNokta.getY(), 2));
41
42        // Daireler tamamen örtüşüyorsa
43        if (mesafe == 0 && yaricap == other.yaricap) {
44            return 1;
45        }
46
47        // Daireler iç içe geçmişse
48        if (mesafe < abs(yaricap - other.yaricap)) {
49            return 0;
50        }
51
52        // Daireler kesişiyorsa
53        if (mesafe < yaricap + other.yaricap && mesafe > abs(yaricap - other.yaricap)) {
54            return 1;
55        }
56
57        // Hiç kesişim yoksa
58        return 2;
59    }
```

toString Metodu:Bu metod, bir dairenin bilgilerini bir string olarak döndürür.oss adında bir ostringstream (output string stream) nesnesi oluşturulur.oss nesnesine dairenin merkezinin toString metodu ile elde edilen string ve yarıçapın bilgisi eklenir.Yarıçap, fixed ve setprecision manipülatörleri kullanılarak ondalık kısımda iki basamağa sabitlenir.oss.str() ile ostringstream nesnesinin içeriği bir string olarak alınarak döndürülür.

yazdir Metodu:Bu metod, toString metodunu çağırarak dairenin bilgilerini ekrana yazdırır. cout << toString() << endl; ifadesi ile toString metodunun çıktısı ekrana yazdırılır.

```
61   // toString metodu
62   string Daire::toString() {
63       ostringstream oss;
64       oss << "Merkez: " << merkezNokta.toString() << ", Yarıçap: " << fixed << setprecision(2) << yaricap;
65       return oss.str();
66   }
67
68   // yazdir metodu
69   void Daire::yazdir() {
70       cout << toString() << endl;
71   }
```

Aşağıda ucgen.h dosyası verilmiştir.Bu C++ kodunda bir "Ucgen" sınıfı tanımlanır.Header dosyasında tanımlanan işlemler aşağıdaki gibidir;

Nokta nokta1: Bir üçgenin birinci noktasını temsil eden bir Nokta nesnesi.

Nokta nokta2: Bir üçgenin ikinci noktasını temsil eden bir Nokta nesnesi.

Nokta nokta3: Bir üçgenin üçüncü noktasını temsil eden bir Nokta nesnesi.

Ucgen(const Nokta& n1, const Nokta& n2, const Nokta& n3): Üç Nokta nesnesini alan yapıcı fonksiyon.

void setNokta1(const Nokta& n1): Birinci noktayı ayarlayan fonksiyon.

Nokta getNokta1() const: Birinci noktayı döndüren fonksiyon.

void setNokta2(const Nokta& n2): İkinci noktayı ayarlayan fonksiyon.

Nokta getNokta2() const: İkinci noktayı döndüren fonksiyon.

void setNokta3(const Nokta& n3): Üçüncü noktayı ayarlayan fonksiyon.

Nokta getNokta3() const: Üçüncü noktayı döndüren fonksiyon.

std::string toString(): Üçgenin noktalarını içeren bir string döndüren fonksiyon.

double alan(): Üçgenin alanını hesaplayan fonksiyon.

double cevre(): Üçgenin çevresini hesaplayan fonksiyon.

double* acilar(): Üçgenin açılarını dizi olarak döndüren fonksiyon.

```cpp
#pragma once
#include "nokta.h"
#include "dogruparcasi.h"
using namespace std;

// Nesne değişkeni olarak 3 tane Nokta nesnesi içeren Üçgen sınıfı
class Ucgen {
private:
    Nokta nokta1;
    Nokta nokta2;
    Nokta nokta3;

public:
    Ucgen(const Nokta& n1, const Nokta& n2, const Nokta& n3);

    void setNokta1(const Nokta& n1);
    Nokta getNokta1() const;

    void setNokta2(const Nokta& n2);
    Nokta getNokta2() const;

    void setNokta3(const Nokta& n3);
    Nokta getNokta3() const;

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

#define _USE_MATH_DEFINES: Bu önişlemci komutu, <cmath> başlık dosyasını kullanarak matematiksel işlemler yapar.

#include "ucgen.h": Bu satır, ucgen.h adlı başka bir başlık dosyasını içeriye dahil eder.

#include <cmath>: Matematiksel işlemleri gerçekleştirmek için gerekli olan C++ standardı matematik kütüphanesini içeriye dahil eder.

Ucgen::Ucgen(const Nokta& n1, const Nokta& n2, const Nokta& n3): Üç noktayı alan bir üçgen sınıfı yapıcısıdır. Bu, üç noktanın nokta1, nokta2 ve nokta3 üye değişkenlere atanmasını sağlar.

void Ucgen::setNokta1(const Nokta& n1): nokta1 üye değişkenini belirtilen Nokta nesnesi ile güncelleyen bir metod.

Nokta Ucgen::getNokta1() const: nokta1 üye değişkenini döndüren bir sabit metod.

void Ucgen::setNokta2(const Nokta& n2): nokta2 üye değişkenini belirtilen Nokta nesnesi ile güncelleyen bir metod.

Nokta Ucgen::getNokta2() const: nokta2 üye değişkenini döndüren bir sabit metod.

void Ucgen::setNokta3(const Nokta& n3): nokta3 üye değişkenini belirtilen Nokta nesnesi ile güncelleyen bir metod.

Nokta Ucgen::getNokta3() const: nokta3 üye değişkenini döndüren bir sabit metod

```cpp
#define _USE_MATH_DEFINES
#include "ucgen.h"
#include <cmath>

// Üç tane Nokta nesnesi alan yapıcı
Ucgen::Ucgen(const Nokta& n1, const Nokta& n2, const Nokta& n3) {
    nokta1 = n1;
    nokta2 = n2;
    nokta3 = n3;
}

// nokta1 için set metodu
void Ucgen::setNokta1(const Nokta& n1) {
    nokta1 = n1;
}

// nokta1 için get metodu
Nokta Ucgen::getNokta1() const {
    return nokta1;
}

// nokta2 için set metodu
void Ucgen::setNokta2(const Nokta& n2) {
    nokta2 = n2;
}

// nokta2 için get metodu
Nokta Ucgen::getNokta2() const {
    return nokta2;
}

// nokta3 için set metodu
void Ucgen::setNokta3(const Nokta& n3) {
    nokta3 = n3;
}
```

string Ucgen::toString(): Bu metot, üçgenin bilgilerini bir string olarak döndürür. Üçgenin köşe noktalarını ve bu noktaların toString metodunu kullanarak birleştirir.

double Ucgen::alan(): Bu metot, üçgenin alanını Heron formülünü kullanarak hesaplar. İlk olarak, her bir kenarın uzunluğunu DogruParcasi sınıfından oluşturulan nesneler aracılığıyla hesaplar. Daha sonra, Heron formülünü kullanarak üçgenin alanını hesaplar ve geri döndürür.

double Ucgen::cevre(): Bu metot, üçgenin çevresini hesaplar. Her bir kenarın uzunluğunu DogruParcasi sınıfından oluşturulan nesneler aracılığıyla hesaplar ve bu uzunlukları toplayarak üçgenin çevresini elde eder.

```
37   // nokta3 için get metodu
38  Nokta Ucgen::getNokta3() const {
39      return nokta3;
40  }
41
42   // toString metodu
43  string Ucgen::toString() {
44      return "Üçgen; " + nokta1.toString() + ", " + nokta2.toString() + ", " + nokta3.toString();
45  }
46
47   // Alan hesaplama metodu
48  double Ucgen::alan() {
49      DogruParcasi kenar1(nokta1, nokta2);
50      DogruParcasi kenar2(nokta2, nokta3);
51      DogruParcasi kenar3(nokta3, nokta1);
52
53      // Heron's formülü kullanılarak üçgen alanının hesaplanması
54      double u = (kenar1.uzunluk() + kenar2.uzunluk() + kenar3.uzunluk()) / 2;
55      return sqrt(u * (u - kenar1.uzunluk()) * (u - kenar2.uzunluk()) * (u - kenar3.uzunluk()));
56  }
57
58   // Üç doğru parçası oluşturularak DogruParcasi sınıfındaki uzunluk metodu ile çevre hesaplama işlemi
59  double Ucgen::cevre() {
60      DogruParcasi kenar1(nokta1, nokta2);
61      DogruParcasi kenar2(nokta2, nokta3);
62      DogruParcasi kenar3(nokta3, nokta1);
63
64      return kenar1.uzunluk() + kenar2.uzunluk() + kenar3.uzunluk();
65  }
```

static double aciDizisi[3];: Bu metot içinde kullanılan bir static double dizisidir. Bu dizi, hesaplanan üç iç açıyı saklamak için kullanılır ve metot dışına kalıcı bir ömür kazandırmak amacıyla static anahtar kelimesi ile tanımlanmıştır.

DogruParcasi kenar1(nokta1, nokta2);,DogruParcasi kenar2(nokta2, nokta3);, DogruParcasi kenar3(nokta3, nokta1);: Üç kenarı temsil eden DogruParcasi nesneleri oluşturulur. Bu nesneler, kenar1, kenar2 ve kenar3 olarak adlandırılmıştır.

Açıların Hesaplanması:Her bir açı kosinüs teoremi kullanılarak hesaplanır.acos fonksiyonu, bir açının kosinüsünün tersini (yani açıyı) döndürür.Her açı, üç kenarın uzunlukları ile ilgili formül kullanılarak hesaplanır.Hesaplanan açılar aciDizisi dizisine atanır.

for (int i = 0; i < 3; ++i): Bu döngü, hesaplanan açıları radyandan dereceye dönüştürür. Açıların önceki hesaplamaları radyan cinsindendir, bu nedenle dereceye çevirmek için her açıyı 180 / M_PI ile çarparız.

return aciDizisi;: Bu diziyi döndürür. C++'da dizilerin doğrudan return edilmesine izin verilmediği için bir diziyi işaretçi (pointer) ile döndürmek gereklidir

```
67   // Üçgenin iç açılarını hesaplayan acilar metodu
68   double* Ucgen::acilar() {
69       static double aciDizisi[3];
70
71       DogruParcasi kenar1(nokta1, nokta2);
72       DogruParcasi kenar2(nokta2, nokta3);
73       DogruParcasi kenar3(nokta3, nokta1);
74
75       // Kosinüs teoremi kullanılarak üçgenin iç açılarını indexleme işlemi ile hesaplama
76       aciDizisi[0] = acos((kenar1.uzunluk() * kenar1.uzunluk() + kenar3.uzunluk() * kenar3.uzunluk() -
77           kenar2.uzunluk() * kenar2.uzunluk()) /
78           (2 * kenar1.uzunluk() * kenar3.uzunluk()));
79       aciDizisi[1] = acos((kenar1.uzunluk() * kenar1.uzunluk() + kenar2.uzunluk() * kenar2.uzunluk() -
80           kenar3.uzunluk() * kenar3.uzunluk()) /
81           (2 * kenar1.uzunluk() * kenar2.uzunluk()));
82       aciDizisi[2] = acos((kenar2.uzunluk() * kenar2.uzunluk() + kenar3.uzunluk() * kenar3.uzunluk() -
83           kenar1.uzunluk() * kenar1.uzunluk()) /
84           (2 * kenar2.uzunluk() * kenar3.uzunluk()));
85
86       // Radyan cinsinden açıları dereceye çevirme
87       for (int i = 0; i < 3; ++i) {
88           aciDizisi[i] = aciDizisi[i] * 180 / M_PI;
89       }
90
91       return aciDizisi;   // C++ dizilerin doğrudan return edilmesine izin vermediğinden pointer ile return etme işleminin gerçekleştirilmesi
92   }
```

GitHub Bağlantıları

Senem ADALAN

https://github.com/SenemADALAN/Geometrik-Problemler/tree/main/Geometrik%20Problemler

Sema Su YILMAZ

SemaSuYILMAZ/Geometrik-Problemler: C++ ile geometrik işlemlerin gerçekleştirilmesi (github.com)

4. SONUÇ VE ÖĞRENİLEN DERSLER

Bu projeyi geliştirirken set ve get fonksiyonlarının kullanım amaçlarını öğrendik.set ve get fonksiyonları ile daha çok işlem gerçekleştirdiğimiz için veri kapsülleme işlemleri hakkında da bilgi sahibi olduk. Veri kapsülleme ve information hiding işlemlerinde kullanılan const ifadesinin kullanım alanlarını öğrendik. Default constructor ve parametreli olarak yazılan constructor'lar ile işlem yaparak yapıcı fonksiyonları daha iyi biçimde anlamış olduk.

5. KAYNAKÇA

https://geomatdata.blogspot.com/2018/07/heron-teoreminin-ispat.html

https://acikders.ankara.edu.tr/pluginfile.php/79557/mod_resource/content/0/MAT%20114-8.pdf

https://www.derspresso.com.tr/matematik/dogrunun-analitigi/denklem

https://chat.openai.com

Nesneye Yönelik Programlama Ders Notları — Dr.Öğr.Üyesi Ulaş Vural