T.C

KOCAELİ SAĞLIK VE TEKNOLOJİ ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA FAKÜLTESİ

C++ ile Geometrik Problemeler

Hazırlayan

Bilgisayar mühendisliği

Yavuz Selim GÜRSOY

220501003

Yazılım mühendisliği

Talha TUNA

220502015

DERS SORUMLUSU

Prof. Dr. Hüseyin Tarık DURU

31 ARALIK 2023

Yeniköy Mahallesi Ilıca Caddesi No:29, Başiskele,Kocaeli

info@kocaelisaglik.edu.tr kocaelisaglik.edu.tr

İÇİNDEKİLER

1. ÖZET

Bu rapor, C++ dilinde oluşturulmuş ve geometrik problemleri çözen koddur. İlgili kodlar fonksiyonlar halinde ayrılmış ve her bir fonksiyonun işlevi, kullanımı ve iç çalışma mantığı ayrıntılı olarak açıklanmıştır.

.

2. GİRİŞ

Bu rapor, temel geometrik şekilleri ve bu şekillerin özelliklerini hesaplamak amacıyla geliştirilmiş olan bir C++ geometri kütüphanesini detaylı bir şekilde ele almaktadır. C++ programlama dili, bu kütüphanede kullanılan özellikleri ve fonksiyonları etkin bir şekilde destekler, bu da karmaşık geometrik hesaplamaların kolaylıkla gerçekleştirilmesine olanak tanır. Kütüphanede, noktalar, doğru parçaları, daireler ve üçgenler gibi geometrik şekillerin temsil edilmesi ve bu şekillerin özelliklerinin hesaplanması için sınıflar ve metodlar bulunmaktadır. Bu rapor, kütüphanenin yapısını, bileşenlerini ve işlevselliğini ayrıntılı olarak açıklamaktadır. Her bir geometrik sınıfın ve fonksiyonun işleyişi, kullanılan algoritmalar ve hesaplama yöntemleri bu raporun ilerleyen bölümlerinde derinlemesine incelenecektir.

3. YÖNTEM

Kütüphane, C++ programlama dili kullanılarak geliştirilmiştir. Her bir geometrik şeklin temsil edilmesi için ayrı sınıflar tanımlanmıştır: Dot, LineSegment, Circle ve Triangle. Her bir sınıf, geometrik bir şekli tanımlamak ve o şeklin özelliklerini hesaplamak için gereken temel metodlara sahiptir.

3.1 Geometry.h

- Dot Sınıfı:
  - Bu sınıf, iki boyutlu bir noktanın koordinatlarını temsil eder.
  - İki farklı yapılandırıcı bulunmaktadır: biri varsayılan, biri x ve y koordinatları için.
  - Kopya yapılandırıcıları ve ofset ile nokta oluşturma işlevleri de mevcuttur.
  - Koordinatları ayarlama ve almak için işlevler sağlar.
  - Noktanın dize temsilini ve ekrana yazdırılmasını sağlayan işlevler vardır.
- LineSegment Sınıfı:
  - Bu sınıf, iki nokta arasındaki doğru parçasını temsil eder.

- Başlangıç ve bitiş noktaları, uzunluk ve eğim bilgilerini içerir.
- Doğru parçasının uzunluğunu hesaplayan ve orta noktasını bulan işlevleri vardır.
- İki doğru parçasının kesişim noktasını ve dize temsilini sağlayan işlevleri de mevcuttur.

o Circle Sınıfı:

- Bu sınıf, merkezi ve yarıçapı olan bir daireyi temsil eder.
- Alan, çevre ve başka bir daireyle kesişim hesaplamalarını yapmak için işlevleri vardır.
- Merkez noktası ve yarıçap bilgilerini ayarlama ve almak için işlevleri vardır.
- Dairenin dize temsilini ve ekrana yazdırılmasını sağlayan işlevler vardır.

o Triangle Sınıfı:

- Bu sınıf, üç noktanın birleşimi ile oluşan bir üçgeni temsil eder.
- Alan, çevre ve üçgenin açılarını hesaplayan işlevleri vardır.
- Üç noktanın koordinatlarını ayarlama ve almak için işlevleri vardır.
- Üçgenin dize temsilini ve ekrana yazdırılmasını sağlayan işlevler vardır.

3.2 Geometry.cpp

```
using namespace std;

Dot::Dot() {
    setCoordinates(0, 0);
}
Dot::Dot(double num) {
    setCoordinates(num, num);
}
Dot::Dot(const Dot &dotObj) {
    setCoordinates(dotObj.x, dotObj.y);
}
Dot::Dot(const Dot &dotObj, double ofsetX, double ofsetY) {
    double newX = ofsetX + dotObj.x;
    double newY = ofsetY + dotObj.y;

    setCoordinates(newX, newY);
}
Dot::Dot(double xIn, double yIn) {
    setCoordinates(xIn, yIn);
}
```

Ödev yönergelerine uygun olarak oluşturulmuş beş farklı nokta constructor'ı mevcuttur: İlk constructor, nokta nesnesinin x ve y koordinatlarını varsayılan olarak 0 olarak ayarlar. İkinci constructor, kullanıcı tarafından belirtilen x ve y koordinat değerlerini alarak nokta nesnesini oluşturur. Üçüncü constructor, belirtilen x ve y değerlerini kullanarak nokta nesnesini oluşturur. Dördüncü constructor, başka bir nokta nesnesinin x ve y koordinat değerlerini alarak yeni bir nokta nesnesini oluşturur. Beşinci constructor ise, belirtilen sayı değeriyle bir nokta nesnesinin x ve y koordinatlarını çarparak yeni bir nokta nesnesini oluşturur.

```
double Dot::getX() {
    return this -> x;
}

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

double Dot::getY() {
    return this -> y;
}

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

void Dot::setCoordinates(double xIn, double yIn) {
    setX(xIn);
    setY(yIn);
}
```

Bu yapı, sınıfın iç yapısının korunmasını ve veri bütünlüğünü sağlamak amacıyla sınıfın iç verilerine doğrudan erişimi engeller. Nokta sınıfına özel olarak tanımlanan setter ve getter fonksiyonlar, kullanıcının sınıf özelliklerine kontrollü bir şekilde erişmesini sağlar. Ayrıca, setCoordinates() fonksiyonu, tek seferde iki koordinata atama yapabilme özelliğine sahiptir.

```
string Dot::toString() {
    string returnX = to_string(getX());
    string returnY = to_string(getY());

    string returnString = "X: " + returnX + "\n" + "Y: " + returnY + "\n";

    return returnString;

}

void Dot::print() {
    cout << "Nokta nesnesinin koordinatlari:\n" << toString() << endl;
```

print() fonksiyonu, toString() fonksiyonundan dönen string değeri alarak bu değeri ekrana yazdırır; toString() fonksiyonu ise noktaları bir string değer olarak döndürür.

Doğru parçası sınıfının üç farklı constructor'ı vardır: İlk constructor, iki nokta nesnesi alarak bu noktalar arasında bir doğru parçası nesnesi oluşturur. İkinci constructor, mevcut bir doğru parçası nesnesini alarak bu nesnenin özelliklerine göre yeni bir doğru parçası nesnesi oluşturur. Üçüncü constructor ise bir orta nokta, eğim ve uzunluk değerlerini alır; bu değerlere göre gerekli hesaplamaları yaparak bir doğru parçası nesnesi oluşturur.

```
LineSegment::LineSegment(const Dot& dotObj1, const Dot& dotObj2){
    setDot1(dotObj1);
    setDot2(dotObj2);
}
LineSegment::LineSegment(LineSegment &lineObj) {
    setDot1(lineObj.getDot1());
    setDot2(lineObj.getDot2());
}
LineSegment::LineSegment(Dot &dotObj, double lengthIn, double slopeIn) {
    double dotObj2x = dotObj.getX() + lengthIn / sqrt(1 + pow(slopeIn, 2));
    double dotObj2y = dotObj.getY() + slopeIn * (dotObj.getX() - dotObj2x);

    dot2.setX(dotObj2x);
    dot2.setY(dotObj2y);
}



double LineSegment::calcLength() {
    double lengthCalc = sqrt(pow((getDot1().getX() - getDot2().getX()),2)
                        + pow((getDot1().getY() - getDot2().getY()),2));
    return lengthCalc;
```

Ayrıca, calcLength() fonksiyonu, doğru parçasının uzunluğunu hesaplamak için doğru parçasının iki noktası arasındaki uzunluğu iki noktası bilinen doğru parçasının uzunluğu formülünü kullanarak hesaplar.

```
Dot LineSegment::calcIntersectPoint(Dot &dotObj) {
    setSlope((dot2.getY() - dot1.getY()) / (dot2.getX() - dot1.getX()));

    double slope2 = -1 / getSlope();

    double xIntersect = (getSlope() * dot1.getX() - slope2 * dotObj.getX()
                        + dotObj.getY() - dot1.getY()) / (getSlope() - slope2);

    double yIntersect = getSlope() * (xIntersect - dot1.getX()) + dot1.getY();

    return Dot(xIntersect, yIntersect);
}



Dot LineSegment::calcMidpoint() {
    double midX = dot1.getX() + dot2.getX();
    double midY = dot1.getY() + dot2.getY();

    return Dot(midX/2.0, midY/2.0);
}
```

calcIntersectPoint() fonksiyonu, kendisine parametre olarak verilen nokta nesnesinden, doğru parçasına dik olarak çizilen başka bir doğru parçasının doğru parçasını hangi noktadan keseceğini belirler ve bu kesim noktasını döndürür. Ayrıca, calcMidpoint() fonksiyonu doğru parçası nesnesinin orta noktasını hesaplar ve bu orta noktayı bir nokta nesnesi olarak döndürür.

```
void LineSegment::setDot1(const Dot& dotObj) {
    this -> dot1 = dotObj;
}
Dot LineSegment::getDot1() {
    return this -> dot1;
}


void LineSegment::setDot2(const Dot& dotObj) {
    this -> dot2 = dotObj;
}
Dot LineSegment::getDot2() {
    return this -> dot2;
}


void LineSegment::setLength(double lengthIn) {
    this -> length = lengthIn;
}
double LineSegment::getLength() {
    return this -> length;
}


void LineSegment::setSlope(double slopeIn) {
    this -> slope = slopeIn;
}
double LineSegment::getSlope() {
    return this -> slope;
}
```

Doğru parçasına ait nokta nesnelerini ayarlamak için setter getter fonksiyonları.

setDot1(const Dot& dotObj) ve getDot1(): dot1 özelliğini ayarlamak ve almak için

.

setDot2(const Dot& dotObj) ve getDot2(): dot2 özelliğini ayarlamak ve almak için.

setLength(double lengthIn) ve getLength(): length özelliğini ayarlamak ve almak için.

setSlope(double slopeIn) ve getSlope(): slope özelliğini ayarlamak ve almak için.

```
string LineSegment::toString() {
    string dot1Str = dot1.toString();
    string dot2Str = dot2.toString();

    string outputStr = "Baslangic noktasi:\n" + dot1Str + "\nBitis noktasi:\n"+ dot2Str;
    return outputStr;

}


void LineSegment::print() {
    cout << "Dogru parcasi ozellikleri:\n" << toString();
}
```

toString() metodu, LineSegment sınıfının özelliklerini bir string olarak döndürür. İlk olarak, dot1 ve dot2 noktalarının string temsilini (yani koordinatlarını) almak için toString() metodunu kullanır. Ardından, bu noktaların string temsillerini kullanarak bir outputStr oluşturur. Bu string, başlangıç ve bitiş noktalarını içerir. Son olarak, outputStr string'ini döndürür.

print() metodu, LineSegment sınıfının özelliklerini ekrana yazdırır. İlk olarak, "Dogru parcasi ozellikleri:" şeklinde bir başlık ekrana yazdırılır. Ardından, toString() metodunu çağırarak doğru parçasının özelliklerini içeren bir string alınır. Bu string, cout (standart çıkış) ile ekrana yazdırılır.

Circle(Dot& dotObj, double radiusIn): Bu kurucu metot, bir daire oluştururken verilen bir merkez noktası ve yarıçap değerini kullanarak bir daireyi başlatır. İlk olarak, yarıçapı ayarlamak için setRadius() metodunu ve ardından merkez noktayı ayarlamak için setMidpoint() metodunu çağırır.

```
Circle::Circle(Dot& dotObj, double radiusIn) {
    setRadius(radiusIn);
    setMidpoint(dotObj);
}
Circle::Circle(Circle &circleObj) {
    setRadius(circleObj.getRadius());
    setMidpoint(circleObj.getMidpoint());
}
Circle::Circle(Circle &circleObj, double x) {
    setMidpoint(circleObj.getMidpoint());
    setRadius(circleObj.getRadius() * x);
}
```

Circle(Circle &circleObj): Bu kurucu metot, mevcut bir Circle nesnesinin özelliklerini (yani yarıçap ve merkez nokta) kullanarak yeni bir Circle nesnesi oluşturur. Öncelikle, kopyalanan dairenin yarıçapını almak için getRadius() metodunu ve ardından merkez noktasını almak için getMidpoint() metodunu çağırır. Sonra, bu değerleri kullanarak yeni dairenin özelliklerini ayarlar.

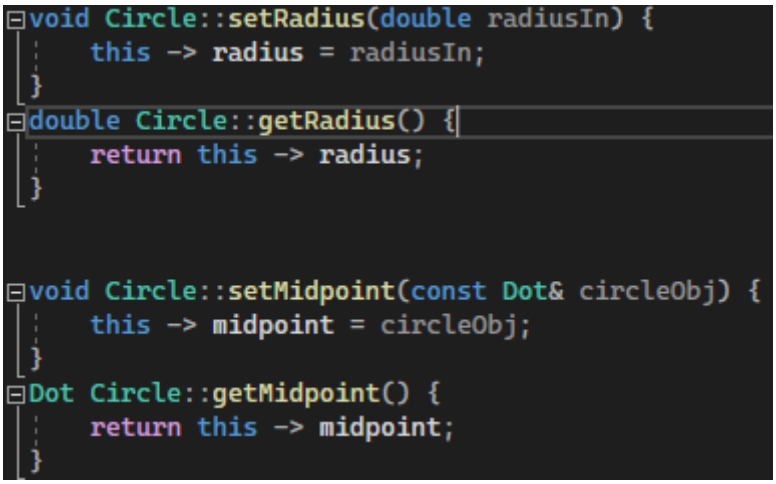
Circle(Circle &circleObj, double x): Bu kurucu metot, mevcut bir Circle nesnesinin merkez noktasını kullanarak yarıçapını belirtilen bir kat sayıyla çarparak yeni bir Circle nesnesi oluşturur. İlk olarak, merkez noktayı almak için getMidpoint() metodunu çağırır. Ardından, kopyalanan dairenin yarıçapını almak için getRadius() metodunu kullanır, bu yarıçapı verilen x değeriyle çarparak yeni yarıçap değerini hesaplar. Son olarak, bu yeni yarıçapı ayarlamak için setRadius() metodunu çağırır.

```
double Circle::calcArea() {
    double area = M_PI * pow(getRadius(),2);
    return area;

}



double Circle::calcCircumference() {
    double circumference = 2 * M_PI * getRadius();
    return circumference;


}


int Circle::calcIntersection(Circle &circleObj) {
    double radiusCombined = circleObj.getRadius() + getRadius();
    double distanceBetweenMidpoints = sqrt(pow(getMidpoint().getX() - circleObj.getMidpoint().getX(), 2) +
                                           pow(getMidpoint().getY() - circleObj.getMidpoint().getY(), 2));

    if (distanceBetweenMidpoints == 0 && circleObj.getRadius() == getRadius()){
        return 1;
    }

    else if (distanceBetweenMidpoints < radiusCombined){
        return 0;
    }

    else{
        return 2;
    }
}
```

calcArea(): Bu fonksiyon, dairenin alanını hesaplar. Alan, π (pi) sayısının dairenin yarıçapının karesi ile çarpılmasıyla elde edilir.

calcCircumference(): Bu fonksiyon, dairenin çevresini hesaplar. Çevre, 2π (pi) sayısının dairenin yarıçapıyla çarpılmasıyla elde edilir.

calcIntersection(Circle &circleObj): Bu fonksiyon, mevcut dairenin bir başka daire ile kesişip kesişmediğini kontrol eder ve bu kesişimin durumuna göre bir değer döndürür: Eğer daireler tamamen üst üste binmişse (merkezleri aynı ve yarıçapları eşitse), bu durumda daireler birbirinin içindedir ve fonksiyon 1 değerini döndürür. Eğer daireler birbirinden daha küçük bir mesafede ise (dışbükey durumda bir daire diğerinin içindeyse veya dışında bir kısmı varsa), fonksiyon 0 değerini döndürür. Eğer daireler birbirine hiç temas etmiyorsa, fonksiyon 2 değerini döndürür.



setRadius(double radiusIn): Bu fonksiyon, bir Circle nesnesinin yarıçapını ayarlamak için kullanılır. Fonksiyon, parametre olarak aldığı radiusIn değerini sınıfın radius özelliğine atar.

getRadius(): Bu fonksiyon, bir Circle nesnesinin yarıçap değerini döndürmek için kullanılır. Fonksiyon, sınıfın radius özelliğinin değerini geri döndürür.

setMidpoint(const Dot& circleObj): Bu fonksiyon, bir Circle nesnesinin merkez noktasını ayarlamak için kullanılır. Fonksiyon, parametre olarak aldığı circleObj (bir Dot nesnesi) değerini sınıfın midpoint özelliğine atar.

getMidpoint(): Bu fonksiyon, bir Circle nesnesinin merkez noktasını döndürmek için kullanılır. Fonksiyon, sınıfın midpoint özelliğinin değerini geri döndürür.

toString(): dairenin merkezi ve yarıçapını string bir değere atar ve bu değeri geri döndürür.

print(): toString fonksiyonunu kullanarak dairenin özelliklerini yazdırır.

```
Triangle::Triangle(Dot &dotObj1, Dot &dotObj2, Dot &dotObj3) {
    setDot1(dotObj1);
    setDot2(dotObj2);
    setDot3(dotObj3);
}
```

Triangle::Triangle(Dot &dotObj1, Dot &dotObj2, Dot &dotObj3): Bu, Triangle sınıfının üç noktalı (köşeli) bir constructor'ını tanımlar. Bu constructor, üç farklı Dot nesnesi (köşe noktalar) alır ve bu noktaları Triangle nesnesinin üç köşe noktası olarak ayarlar. setDot1(dotObj1);: İlk köşe noktasını (dot1) dotObj1 olarak ayarlar. setDot2(dotObj2) ;: İkinci köşe noktasını (dot2) dotObj2 olarak ayarlar. setDot3(dotObj3);: Üçüncü köşe noktasını (dot3) dotObj3 olarak ayarlar .Kısacası üçgen sınıfının başlatılması için parametre olarak üç nokta nesnesi alan constructor.

```
double Triangle::calcArea() {
    LineSegment line1(getDot1(), getDot2());
    LineSegment line2(getDot1(), getDot3());
    LineSegment line3(getDot2(), getDot3());

    double a = line1.calcLength();
    double b = line2.calcLength();
    double c = line3.calcLength();

    double s = (a + b + c) / 2.0;
    double area = sqrt(s * (s - a) * (s - b) * (s - c));

    return area;
}
```

Bu kod parçası, Triangle sınıfına ait bir calcArea fonksiyonunu tanımlar ve bu fonksiyon, üçgenin alanını hesaplamak için gerekli hesaplamaları yapar. İlk olarak, üçgenin üç kenarını temsil eden LineSegment nesneleri oluşturulur. Ardından her bir kenarın uzunluğu hesaplar. Üçgenin yarı çevresi bu uzunluklarını kullanarak hesaplar. Son olarak, bu yarı çevre değeri ve kenar uzunlukları kullanılarak üçgenin alanını hesaplar ve bu alan değeri fonksiyon tarafından döndürür.

```
double Triangle::calcCircumference() {
    LineSegment line1(getDot1(), getDot2());
    LineSegment line2(getDot1(), getDot3());
    LineSegment line3(getDot2(), getDot3());

    return line1.calcLength() + line2.calcLength() + line3.calcLength();
}
```

Bu kod parçası, Triangle sınıfına ait bir calcCircumference fonksiyonunu tanımlar. Bu fonksiyon, üçgenin çevresini hesaplamak için üç kenarın uzunluklarını toplar ve bu toplamı döndürür. İlk olarak, üçgenin üç kenarını temsil eden LineSegment nesneleri oluşturulur. Her bir kenarın uzunluğu bu nesneler üzerinden calcLength fonksiyonu ile hesaplanır. Son olarak, bu üç kenar uzunluğunun toplamı, yani üçgenin çevresi, fonksiyon tarafından geri döndürülür.

```cpp
std::array<double,3> Triangle::calcAngles() {
    array <double, 3> resultArray{};

    LineSegment line1(getDot1(), getDot2());
    LineSegment line2(getDot1(), getDot3());
    LineSegment line3(getDot2(), getDot3());

    double a = line1.calcLength();
    double b = line2.calcLength();
    double c = line3.calcLength();

    double aRadian = std::acos((b * b + c * c - a * a) / (2 * b * c));
    double bRadian = std::acos((a * a + c * c - b * b) / (2 * a * c));
    double cRadian = std::acos((a * a + b * b - c * c) / (2 * a * b));

    double aDegrees = aRadian * (180.0 / M_PI);
    double bDegrees = bRadian * (180.0 / M_PI);
    double cDegrees = cRadian * (180.0 / M_PI);

    resultArray[0] = aDegrees;
    resultArray[1] = bDegrees;
    resultArray[2] = cDegrees;

    return resultArray;
}
```

Bu kod parçası, Triangle sınıfının üyeleri içerisindeki bir fonksiyonun tanımını içerir. Bu fonksiyon, üçgenin üç iç açısının derece cinsinden değerlerini hesaplar ve bu değerleri bir dizi içinde döndürür. Fonksiyon, üçgenin kenarlarını ve bu kenarların uzunluklarını kullanarak üçgenin açılarını hesaplar. Sonrasında bu açıları dereceye dönüştürür ve bir dizi içerisinde saklar.

```cpp
void Triangle::setDot1(const Dot &dotObj) {
    this -> dot1 = dotObj;
}
Dot Triangle::getDot1() {
    return this -> dot1;
}

void Triangle::setDot2(const Dot &dotObj) {
    this -> dot2 = dotObj;
}
Dot Triangle::getDot2() {
    return this -> dot2;
}

void Triangle::setDot3(const Dot &dotObj) {
    this -> dot3 = dotObj;
}
Dot Triangle::getDot3() {
    return this -> dot3;
}
```

Bu kod parçası, Triangle sınıfı için nokta değerlerini ayarlama (set) ve sorgulama (get) işlevlerini tanımlar. dot1, dot2, ve dot3 adında üç farklı nokta üyesi vardır. Her bir üye için ayrı ayrı set ve get fonksiyonları bulunmaktadır. Kısacası Üçgen sınıfına özel setter getter fonksiyonlar.

```cpp
string Triangle::toString() {

    string returnString =
            "\nUcgen noktalari:\nA:\n" + getDot1().toString() +
            "\nB:\n" + getDot2().toString() +
            "\nC:\n" + getDot3().toString();
    return returnString;
}

void Triangle::print() {
    cout << "Ucgen ozellikleri: " << toString();
}
```

toString() fonskiyonunu kullanarak üçgen nesnesinin özelliklerini ekrana yazdıran fonksiyon.

print() Üçgen nesnesinin özelliklerini string bir değere atıp bu değeri döndüren fonksiyon.

3.3 GeometryTest.cpp

```
using namespace std;
int main() {
    Dot dot0;
    Dot dot1(5);
    Dot dot2(7,11);
    Dot dot3(dot1);
    Dot dot4(dot2, 6, 8);

    dot0.print();
    dot1.print();
    dot2.print();
    dot3.print();
    dot4.print();


    cout << "--------------------------------------------------------------------------------\n";
```

Dot sınıfından beş nesne oluşturulur. İlk iki nokta varsayılan ve tek parametreli yapılandırıcılarla, sonraki üç nokta farklı parametrelerle oluşturulmuştur. Oluşturulan nesnelerin koordinatları ekrana yazdırılır.

```
LineSegment line1(dot0, dot1);
LineSegment line2(line1);
LineSegment line3(dot2, 8, 0.5);

line1.print();
cout << "\nLine1 nesnesinin orta noktasi: " << endl << line1.calcMidpoint().toString() << endl;
line2.print();
cout << "\nLine2 nesnesinin uzunlugu: " << endl << line2.calcLength() << endl << endl;
line3.print();
cout << "\nNokta1 nesnesinden Line3 nesnesine dik olarak cizilecek dogru parcasinin kesisme noktasi: " << endl << line3.calcIntersectPoint(dot1).toString() << endl;


cout << "--------------------------------------------------------------------------------\n";
```

Üç adet LineSegment nesnesi oluşturulur. İlk nesne, dot0 ve dot1 noktaları arasında. İkinci nesne, line1 nesnesinin bir kopyası olarak oluşturulur. Üçüncü nesne, dot2 ile başlangıç noktası ve 8 ile uzunluğu olan bir doğru parçasıdır. Oluşturulan LineSegment nesnelerinin özellikleri ekrana yazdırılır: İlk line1 nesnesinin detayları ve orta noktası. Kopyalanan line2 nesnesinin detayları. Üçüncü line3 nesnesinin detayları ve dot1 noktasından geçen ve line3 ile kesişen doğru parçasının kesişim noktası.

```
Circle circle1(dot4, 8);
Circle circle2(circle1);
Circle circle3(circle2, 9);

circle1.print();
cout << "\nCircle1 nesnesinin alani: " << circle1.calcArea() << endl;
circle2.print();
cout << "\nCircle2 nesnesinin cevresi: " << circle2.calcCircumference() << endl << endl;
circle3.print();
cout << "\nCircle3 nesnesi Circle2 nesnesinin icinde mi, disinda mi, yoksa ortusuyor mu?" << endl << circle3.calcIntersection(circle2) << endl;



cout << "--------------------------------------------------------------------------------\n";
```

Circle sınıfının özelliklerini ve yöntemlerini kullanarak üç daire nesnesi oluşturur, bu nesnelerin özelliklerini ve hesaplanan değerlerini ekrana yazdırır.

```
Triangle triangle1(dot2, dot4, dot0);

triangle1.print();
cout << "\nTriangle1 nesnesinin alani: " << endl << triangle1.calcArea() << endl << endl;
cout << "\nTriangle1 nesnesinin cevresi: " << endl <<triangle1.calcCircumference() << endl << endl;
cout << "\nTriangle1 nesnesinin acilari: " << endl << "A: " << triangle1.calcAngles()[0] << endl
                                                 << "B: " << triangle1.calcAngles()[1] << endl
                                                 << "C: " << triangle1.calcAngles()[2] << endl;
```

Bu kod parçası, Triangle sınıfının özelliklerini ve yöntemlerini kullanarak bir üçgen nesnesi oluşturur. Oluşturulan üçgen nesnesinin özelliklerini ekrana yazdırır, alanını, çevresini ve açılarını hesaplar ve bu değerleri ekrana yazdırır.

4. SONUÇ VE ÖĞRENİLEN DERSLER

Bu proje sayesinde C++ programlama dilinin kullanımı, büyük çaplı projelerin taslaklarının nasıl oluşturulabileceği ve bu projelere dair resmi evrakların nasıl tamamlanabileceği hakkında deneyim ve bilgi sahibi olduk. Ayrıca, ekip çalışması ve kod standartları konularına da dikkat ettik.

5. KAYNAKÇA

https://en.cppreference.com/w/c/locale/setlocale
https://stackoverflow.com/questions/117293/does-using-const-on-function-parameters-have-any-effect-why-does-it-not-affect
https://www.w3schools.com/cpp/cpp_pointers.asp

6. GITHUB BAĞLANTILARI

https://github.com/TalhaTuna2

https://github.com/Yavuz-Selim-Gursoy