Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**Пермский национальный исследовательский**

**политехнический университет**

ЭТФ

ИТАС

Факультет\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КОБ

Специальность \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

* **лабораторной работе № 13**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*наименование лабораторной работы*)

Выполнил:

студент группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КОБ-23-1С

Горкунов М. В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*фамилия*, *инициалы*)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*)

Проверил:

доцент кафедры ИТАС Тарутин А.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*оценка*)

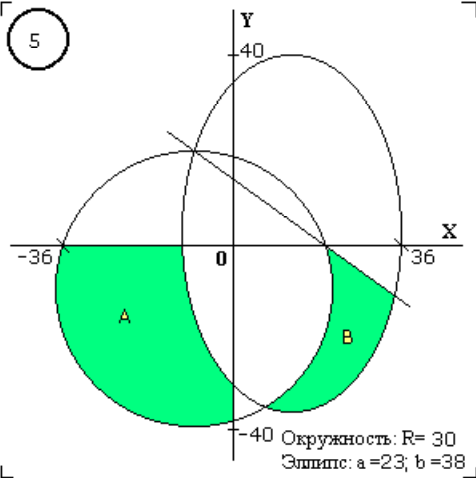
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(*подпись*)

**Цель лабораторной работы:** разработать алгоритм и программу вычисления площади областей ограниченных кривыми второго и первого порядков.

**Содержание лабораторной работы:** В лабораторной работе было необходимо использовать различные методы вычислений площади под кривой такие как: метод разбиения на прямоугольники и метод разбиения на трапеции. Предварительно найти уравнения кривых ограничивающие области и критические точки для вычисления площади. Число разбиений N1, N2 и N3 задать самостоятельно. Точность результатов приближенных вычислений – 0,001. Предусмотреть в программе повтор описанных действий с новым значением числа разбиений N в ответ на соответствующий запрос программы. Полученные результаты вывести на экран в виде таблицы.

На рисунке изображены две закрашенные области площадь которых нужно найти.

Для решения задачи понадобятся следующие **формулы:**

Уравнение окружности:

Уравнение эллипса:

Уравнение прямой:

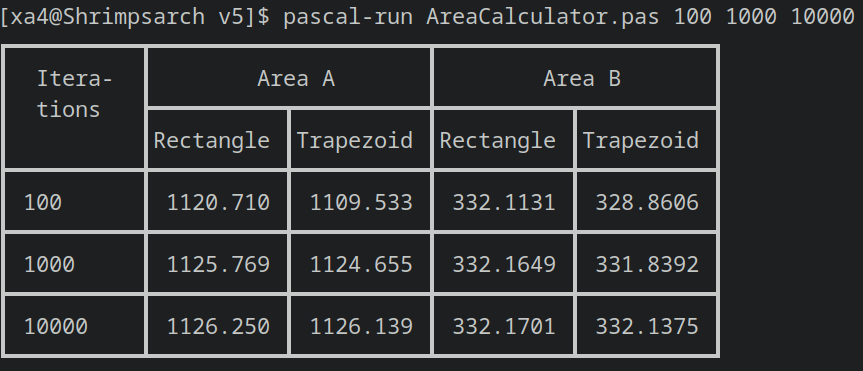
Уравнение вычисление площади методом разбиения на прямоугольники:

Уравнение вычисление площади методом разбиения на трапеции:

**Алгоритм программы:**

1. Программа будет принимать N1, N2, N3, ... Nn, аргументы для расчётов и построения таблицы.
2. В теле программы итерируемся по всем аргументам и приводим их к типу Integer если не получается, то игнорируем и начинаем новую итерацию.
3. Приведённый к типу Integer аргумент используем как количество разбиений n.
4. Методом простого сканирования находим крайнюю точку (х конечный) чтобы найти dx.
5. Полученный результат делим на n и получаем dx.
6. По формулам представленным выше находим площадь областей двумя методами.
7. Добавляем в stringBuilder строку таблицы с результатами.
8. Выполняем те же действия для каждого значения N.
9. Выводим полученные результаты в виде таблицы.

Результат работы:

Код программы:

const r = 30.0;

const a = 23.0;

const b = 38.0;

const circleX = -7.7157287525381;

const circleY = -10;

const ellipseX = 13.0318780471981;

const ellipseY = 2;

const k = -0.719447193774024;

const c = 14.779801779947;

const circleEllipseIntersectionX = -7.225;

const circleEllipseIntersectionY = 19.996;

const circleEllipseIntersectionX2 = 8.485;

const circleEllipseIntersectionY2 = -35.249;

const ellipseLineXIntersection = 34.806;

const circleLineXIntersection = 20.5685424949238;

const formatA = ':0000.000';

const formatB = ':000.0000';

type CalculatingAreaFunction = function(

func : Func<Double, Double>;

lowerBound, upperBound : Double;

iterations : Integer) : Double;

function CalculateAreaUnderCurveByRect(

func : Func<Double, Double>;

lowerBound, upperBound : Double;

iterations : Integer)

: Double;

begin

var area : real = 0;

var dx := (upperBound - lowerBound) / iterations;

for var i := 0 to iterations - 1 do begin

area += dx \* func(dx \* i + lowerBound)

end;

Result := abs(area);

end;

function CalculateAreaUnderCurveByTrapezoid(

func : Func<Double, Double>;

lowerBound, upperBound : Double;

iterations : Integer)

: Double;

begin

var area : real = 0;

var previouseValue := func(0);

var dx := (upperBound - lowerBound) / iterations;

for var i := 1 to iterations - 2 do begin

area += func(dx \* i + lowerBound);

end;

area += (func(lowerBound) + func(upperBound)) / 2;

Result := abs(area \* dx);

end;

function GetAArea(

iterations : Integer;

GetAreaUnderCurve : CalculatingAreaFunction)

: Double;

begin

var aArea := GetAreaUnderCurve(

x -> circleY - sqrt(r\*\*2 - (x - circleX)\*\*2),

circleX - r, 0,

iterations);

aArea += GetAreaUnderCurve(

y -> circleX - sqrt(r\*\*2 - (y - circleY)\*\*2),

circleY, 0,

iterations);

aArea -= GetAreaUnderCurve(

x -> ellipseY - a / b \* sqrt(b\*\*2 - (x - ellipseX)\*\*2),

ellipseX - a, 0,

iterations);

aArea -= (-circleX + r) \* -circleY;

Result := aArea;

end;

function GetBArea(

iterations : Integer;

getAreaUnderCurve : CalculatingAreaFunction)

: Double;

begin

var bArea := getAreaUnderCurve(

x -> k \* x + c,

circleX, 2,

iterations);

bArea += getAreaUnderCurve(

x -> ellipseY - a / b \* sqrt(b\*\*2 - (x - ellipseX)\*\*2),

circleEllipseIntersectionX2,

ellipseLineXIntersection,

iterations);

bArea -= getAreaUnderCurve(

x -> circleY - sqrt(r\*\*2 - (x - circleX)\*\*2),

circleEllipseIntersectionX2,

circleLineXIntersection,

iterations);

Result := bArea;

end;

begin

var tableBuilder := new StringBuilder();

tableBuilder

.AppendLine('┏━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━┓')

.AppendLine('┃ Itera- ┃ Area A ┃ Area B ┃')

.AppendLine('┃ tions ┣━━━━━━━━━━┳━━━━━━━━━━╋━━━━━━━━━━┳━━━━━━━━━━┫')

.AppendLine('┃ ┃Rectangle ┃Trapezoid ┃Rectangle ┃Trapezoid ┃');

for var i := 1 to ParamCount do begin

var iterations : Integer;

if not Integer.TryParse(ParamStr(i), iterations) then

continue;

var aAreaByRect := GetAArea(

iterations,

CalculateAreaUnderCurveByRect);

var aAreaByTrapezoid := GetAArea(

iterations,

CalculateAreaUnderCurveByTrapezoid);

var bAreaByRect := GetBArea(

iterations,

CalculateAreaUnderCurveByRect);

var bAreaByTrapezoid := GetBArea(

iterations,

CalculateAreaUnderCurveByTrapezoid);

tableBuilder.AppendLine('┣━━━━━━━━━━╋━━━━━━━━━━╋━━━━━━━━━━╋━━━━━━━━━━╋━━━━━━━━━━┫')

.AppendLine(string.Format(

'┃ {0,-8} ┃ {1'+formatA+'} ┃ {2'+formatA+'} ┃ {3'+formatB+'} ┃ {4'+formatB+'} ┃',

iterations, aAreaByRect, aAreaByTrapezoid, bAreaByRect, bAreaByTrapezoid));

iterations \*= 10;

end;

tableBuilder.AppendLine('┗━━━━━━━━━━┻━━━━━━━━━━┻━━━━━━━━━━┻━━━━━━━━━━┻━━━━━━━━━━┛');

writeln(tableBuilder.ToString());

end.