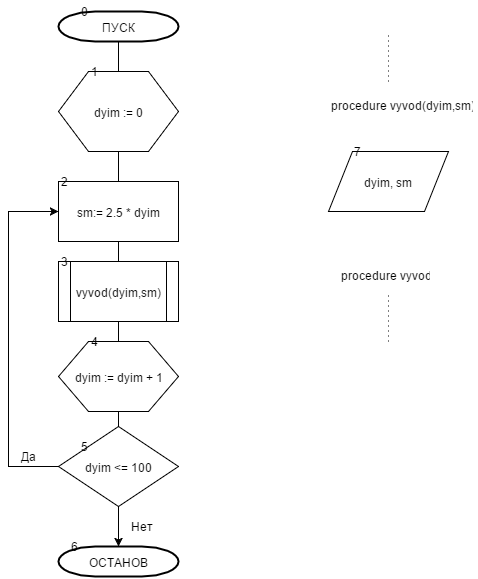
Лабораторная работа №5.

Задание 1.

1. Тема: Процедуры и функции.
2. Цель: научиться реализовывать вычисления при помощи процедур и функций на языке FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: Перевести дюймы в сантиметры от 0 до 100 дюймов. (1 дюйм=2.5 см). Результаты вывести в виде таблицы.
5. Математическая модель:

1 дюйм = 2,5 см.

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| dyim | integer | Дюймы и параметр цикла |
| sm | real | Сантиметры |
| vyvod | real | Процедура вывода |

1. Код программы:

program lr5\_task1;

var

dyim : integer;

sm : real;

procedure vyvod(dyim:integer;sm:real);

begin

writeln('Dyimy: ',dyim,' ','Sm: ',sm:2:1);

end;

begin

for dyim:= 0 to 100 do begin

sm := 2.5 \* dyim;

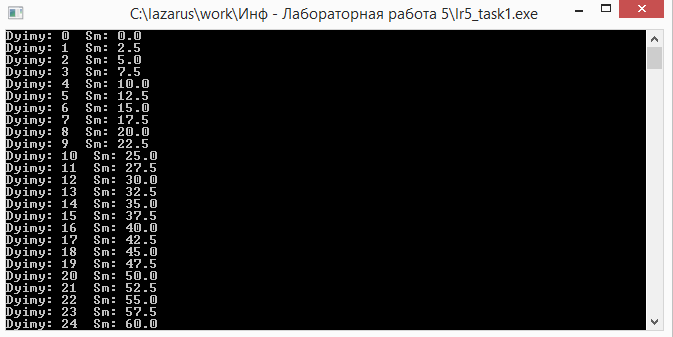
vyvod(dyim,sm);

end;

readln;

end.

1. Результат работы программы:



1. Анализ результатов вычисления:

Мы задаем цикл, в котором переменная dyim будет меняться от 0 до 100 и программа будет вычислять значение переменной sm для каждого значения dyim. После чего мы обращаемся к процедуре вывода результат на экран.

В нашем случае, процедура может быть с параметрами или без – на функциональности программы это никак не отражается.

1. Вывод: мы написали рабочую программу для перевода дюймов в сантиметры и оформили результаты в табличку.

Задание 2.

1. Тема: Процедуры и функции.
2. Цель: научиться реализовывать вычисления при помощи процедур и функций на языке FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.

Постановка задачи: Найти периметр фигуры **ABCD** по заданным сторонам **AB, AC, DC** и углам .

1. Математическая модель:

B

D

A

C

Треугольники ABC и BCD являются прямоугольными (по условию).

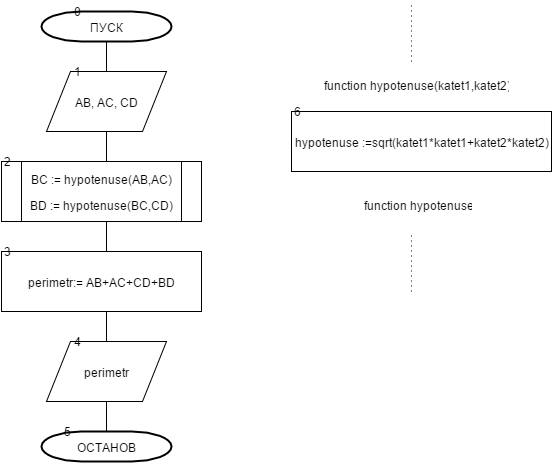
В треугольнике ABC нам известны два катета AB и AС. Значит, можно найти по теореме Пифагора гипотенузу BC.

В это же время BC является так же одним из катетов в прямоугольном треугольнике BCD, второй же известен по условию.

Найдем BD по теореме Пифагора.

Затем найдем периметр, сложив длины всех сторон.

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| AB | real | Сторона фигуры, катет |
| AC | real | Сторона фигуры, катет |
| CD | real | Сторона фигуры, катет |
| BD | real | Сторона фигуры, гипотенуза |
| BC | real | Гипотенуза, катет |
| perimetr | real | Периметр |
| Hypotenuse(katet1,katet2) | real | Функция вычисления гипотенузы по двум катетам |

1. Код программы:

program lr5\_task2;

var

AB,AC,CD,BD,BC,perimetr:real;

function hypotenuse(katet1,katet2:real):real;

begin

hypotenuse := sqrt(katet1\*katet1+katet2\*katet2);

end;

begin

writeln('Vvedite AB: '); readln(AB);

writeln('Vvedite AC: '); readln(AC);

writeln('Vvedite CD: '); readln(CD);

BC := hypotenuse(AB,AC);

BD := hypotenuse(BC,CD);

perimetr:= AB+AC+CD+BD;

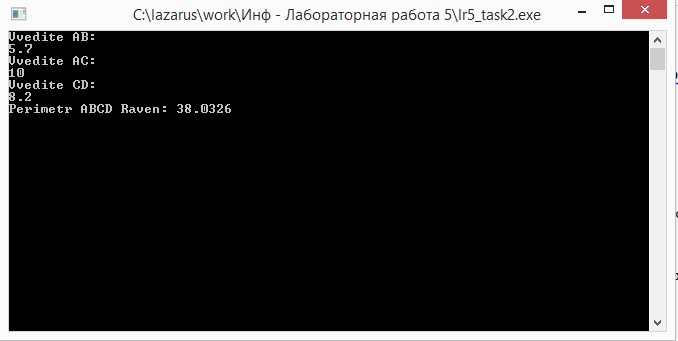
writeln('Perimetr ABCD Raven: ',perimetr:2:4);

readln;

end.

9.

Результат работы программы:



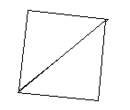
1. Анализ результатов вычисления: в качестве входных данных программа получает длины трех сторон фигуры. Потом она проводит вычисления (которые подробно описаны в пункте 4), прибегая к вызову функции.

После чего выводит результат (площадь) на экран.

1. Вывод: мы написали программу, которая ищет периметр фигуры на основании трех сторон и того факта, что фигура поделена на прямоугольные треугольники.

Задание 3.

1. Тема: Процедуры и функции.
2. Цель: научиться реализовывать вычисления при помощи процедур и функций на языке FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: Вычислить площадь фигуры, заданной сторонами и диагональю.
5. Математическая модель:



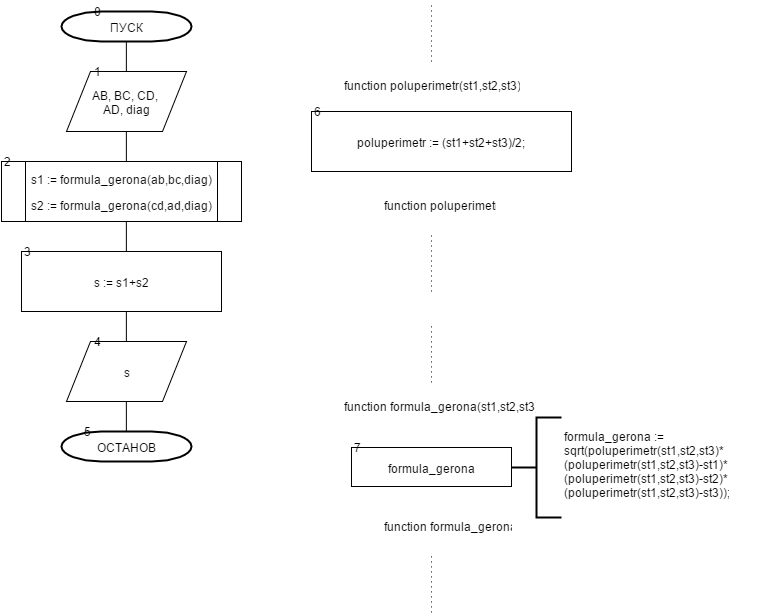
Фигура разбита на два треугольника, площадь которых мы можем найти по формуле Герона.

Где p – полупериметр,

a,b,c – стороны треугольника.

Потом сложим площади треугольников и получим площадь исходной фигуры.

1. Блок-схема:



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| ab | real | Сторона |
| bc | real | Сторона |
| cd | real | Сторона |
| ad | real | Сторона |
| diag | real | Диагональ |
| s1 | real | Площадь первого треугольника |
| s2 | real | Площадь второго треугольника |
| S | real | Площадь исходной фигуры |
| Poluperimetr(st1,st2) | real | Вычисление полупериметра. |
| formula\_gerona(st1,st2,st3) | real | Функция для вычисления площади. |

1. Код программы:

program lr5\_task3;

var

ab,bc,cd,ad,diag,s1,s2,S:real;

function poluperimetr(st1,st2,st3:real):real;

begin

poluperimetr := (st1+st2+st3)/2;

end;

function formula\_gerona(st1,st2,st3:real):real;

begin

formula\_gerona := sqrt(poluperimetr(st1,st2,st3)\*(poluperimetr(st1,st2,st3)-st1)\*(poluperimetr(st1,st2,st3)-st2)\*(poluperimetr(st1,st2,st3)-st3));

end;

begin

writeln('Vvedite storonu AB: '); readln(ab);

writeln('Vvedite storonu BC: '); readln(bc);

writeln('Vvedite storonu CD: '); readln(cd);

writeln('Vvedite storonu AD: '); readln(ad);

writeln('Vvedite diagonal: '); readln(diag);

s1 := formula\_gerona(ab,bc,diag);

s2 := formula\_gerona(cd,ad,diag);

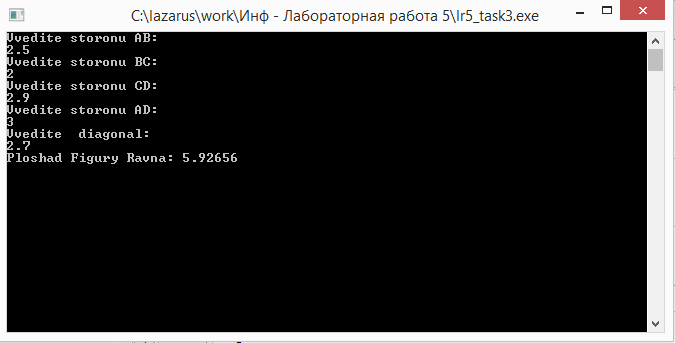
s := s1+s2;

writeln('Ploshad Figury Ravna: ',s:2:5);

readln();

end.

1. Результат работы программы:

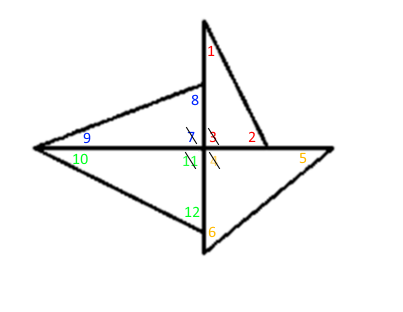


1. Анализ результатов вычисления: мы вводим четыре стороны и диагональ, разбивающую фигуру на два треугольника. Предварительно задав функции, вычисляющую площадь треугольника по формуле Герона и полупериметр треугольника, мы обращаемся к ним в процессе вычисления. Причем функция нахождения полупериметра вызывается из функции нахождения площади.
2. Вывод: мы написали программу для нахождения площади фигуры по четырём сторонам и диагонали.

Задание 4.

1. Тема: Процедуры и функции.
2. Цель: научиться реализовывать вычисления при помощи процедур и функций на языке FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: написать программу для вычисления площади фигуры, которая задана координатами точек.
5. Математическая модель:

Для удобства пронумеруем вершины треугольников, из которых состоит фигура, следующим образом:



Поскольку координаты точек 7, 3, 4, 11 совпадают, обозначим эти общие координаты как xCentr и yCentr соответственно.

Найдем длины сторон каждого треугольника по формуле:

*x1,y1,x2,y2* – координаты конца и начала отрезка.

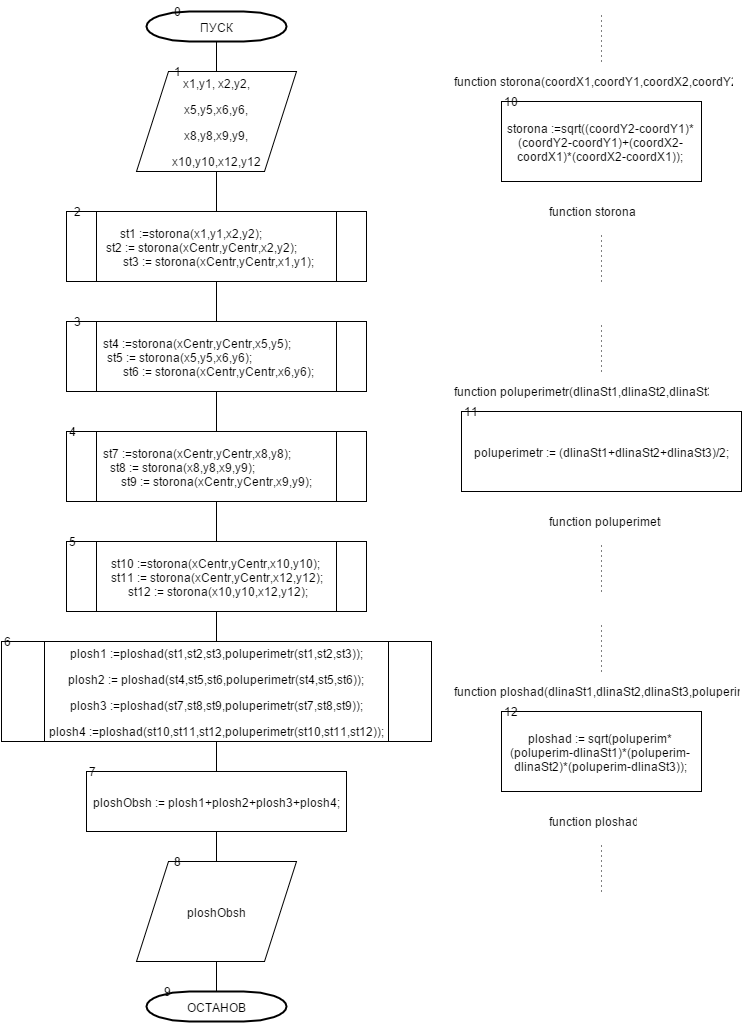
Затем найдём площадь каждого отдельно взятого треугольника по формуле Герона:

*p* – полупериметр.

*a,b,c* - длины сторон треугольника.

Найдя площадь каждого треугольника, мы складываем их и получаем площадь исходной фигуры.

1. Блок-схема: См. на следующей странице



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| xCentr | real | Координата центра по оси OX |
| yCentr | real | Координата центра по оси OY |
| St1 | real | Длина стороны 1 |
| … | … | … |
| St12 | real | Длина стороны 12 |
| X1 | real | Координата вершины 1 по оси OX |
| Y1 | real | Координата вершины 1 по оси OY |
| … | … | … |
| X12 | real | Координата вершины 12 по оси OX |
| Y12 | real | Координата вершины 12 по оси OY |
| plosh1 | real | Площадь треугольника 1 |
| … | … | … |
| Plosh4 | real | Площадь треугольника 4 |
| ploshObsh | real | Общая площадь фигуры |

1. Код программы:

program lr5\_task4\_2222;

var

st1,st2,st3,st4,st5,st6,st7,st8,st9,st10,st11,st12,

xCentr,yCentr,

x1,y1,

x2,y2,

x5,y5,

x6,y6,

x8,y8,

x9,y9,

x10,y10,

x12,y12,

plosh1, plosh2, plosh3, plosh4, ploshObsh : real;

function storona(coordX1,coordY1,coordX2,coordY2:real):real;

begin

storona := sqrt((coordY2-coordY1)\*(coordY2-coordY1)+(coordX2-coordX1)\*(coordX2-coordX1));

end;

function poluperimetr(dlinaSt1,dlinaSt2,dlinaSt3:real):real;

begin

poluperimetr := (dlinaSt1+dlinaSt2+dlinaSt3)/2;

end;

function ploshad(dlinaSt1,dlinaSt2,dlinaSt3,poluperim:real):real;

begin

ploshad := sqrt(poluperim\*(poluperim-dlinaSt1)\*(poluperim-dlinaSt2)\*(poluperim-dlinaSt3));

end;

begin

xCentr := 0; yCentr := 0;

{x1 := 0; y1 := 7;

x2 := 2; y2 := 0;

x5 := 5; y5 := 0;

x6 := 0; y6 := -4;

x8 := 0; y8 := 2;

x9 := -9; y9 := 0;

x10 := -9; y10 := 0;

x12 := 0; y12 := -3;} // Быстрое присваивание тестовых значений

writeln('Vvedite koordinatu x1: '); readln(x1); // 0 Примерные значения

writeln('Vvedite koordinatu y1: '); readln(y1); // 7

writeln('Vvedite koordinatu x2: '); readln(x2); // 2

writeln('Vvedite koordinatu y2: '); readln(y2); // 0

writeln('Vvedite koordinatu x5: '); readln(x5); // 5

writeln('Vvedite koordinatu y5: '); readln(y5); // 0

writeln('Vvedite koordinatu x6: '); readln(x6); // 0

writeln('Vvedite koordinatu y6: '); readln(y6); // -4

writeln('Vvedite koordinatu x8: '); readln(x8); // 0

writeln('Vvedite koordinatu y8: '); readln(y8); // 2

writeln('Vvedite koordinatu x9: '); readln(x9); // -9

writeln('Vvedite koordinatu y9: '); readln(y9); // 0

writeln('Vvedite koordinatu x10: '); readln(x10); // -9

writeln('Vvedite koordinatu y10: '); readln(y10); // 0

writeln('Vvedite koordinatu x12: '); readln(x12); // 0

writeln('Vvedite koordinatu y12: '); readln(y12); // -3

st1 := storona(x1,y1,x2,y2); // Длины сторон первого треугольника

st2 := storona(xCentr,yCentr,x2,y2);

st3 := storona(xCentr,yCentr,x1,y1);

st4 := storona(xCentr,yCentr,x5,y5); // Длины сторон второго треугольника

st5 := storona(x5,y5,x6,y6);

st6 := storona(xCentr,yCentr,x6,y6);

st7 := storona(xCentr,yCentr,x8,y8); // Длины сторон третьего треугольника

st8 := storona(x8,y8,x9,y9);

st9 := storona(xCentr,yCentr,x9,y9);

st10 := storona(xCentr,yCentr,x10,y10); // Длины сторон четвёртого треугольника

st11 := storona(xCentr,yCentr,x12,y12);

st12 := storona(x10,y10,x12,y12);

plosh1 := ploshad(st1,st2,st3,poluperimetr(st1,st2,st3));

plosh2 := ploshad(st4,st5,st6,poluperimetr(st4,st5,st6));

plosh3 := ploshad(st7,st8,st9,poluperimetr(st7,st8,st9));

plosh4 := ploshad(st10,st11,st12,poluperimetr(st10,st11,st12));

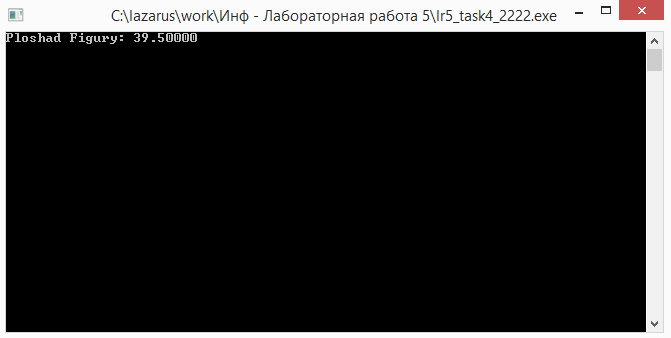
ploshObsh := plosh1+plosh2+plosh3+plosh4;

writeln('Ploshad Figury: ',ploshObsh:2:5);

readln;

end.

1. Результат работы программы:



1. Анализ результатов вычисления:

Этапы вычисления подробно расписаны в мат. модели. Программа работает корректно.

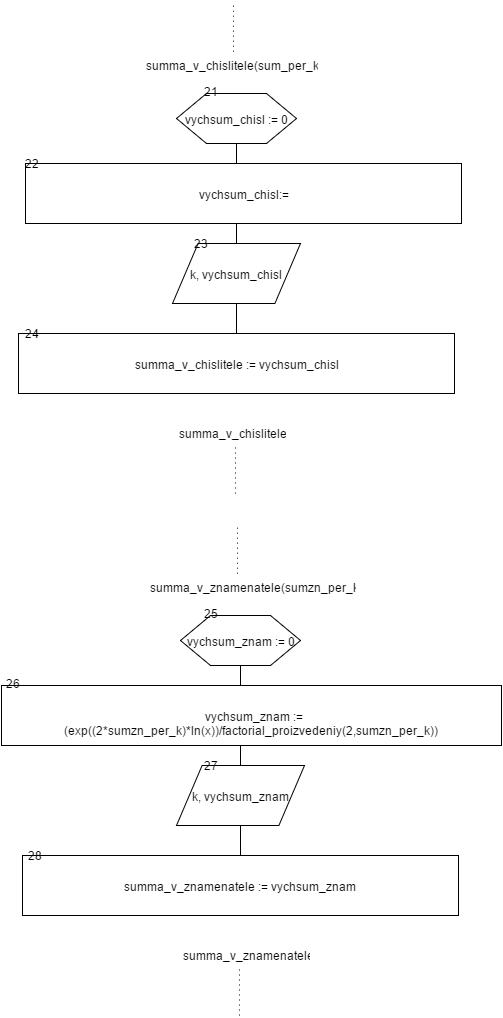
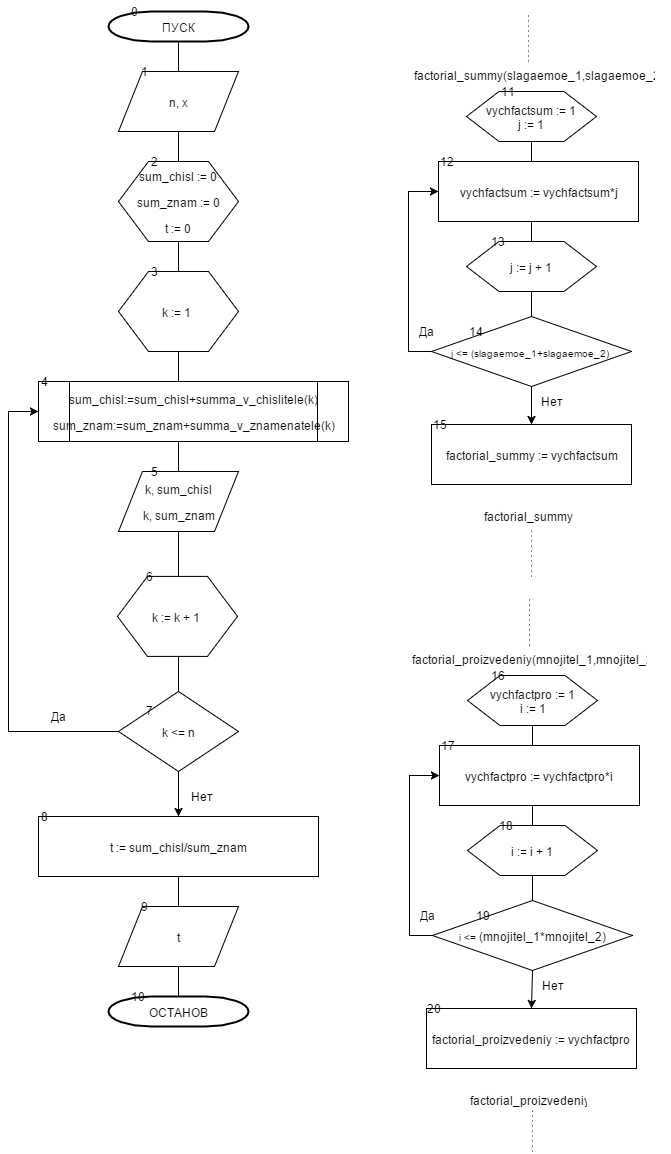
1. Вывод: написать программу для вычисления площади фигуры, которая задана координатами точек.

Задание 5.

1. Тема: Процедуры и функции.
2. Цель: научиться реализовывать вычисления при помощи процедур и функций на языке FreePascal.
3. Используемое оборудование: ПК, среда программирования Lazarus.
4. Постановка задачи: вычислить значение данного выражения.
5. Математическая модель:



1. Блок-схема: см. на следующей странице.



1. Список идентификаторов:

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Смысл |
| sum\_chisl | real | Сумма в числителе |
| sum\_znam | real | Сумма в знаменателе |
| t | real | Искомое значение |
| factorial\_summy() | real | Функция, вычисляющая факториал от суммы. |
| factorial\_proizvedeniy() | real | Функция, вычисляющая факториал от произведения. |
| summa\_v\_chislitele() | real | Функция, вычисляющая сумму в числителе. |
| summa\_v\_znamenatele() | real | Функция, вычисляющая сумму в знаменателе. |
| n | integer | Входные данные |
| x | integer | Входные данные |
| k | integer | Параметр цикла |
| vychfactsum | real | Локальная переменная функции. Промежуточный результат вычислений. |
| j | integer | Локальная переменная функции. Параметр цикла. |
| vychfactpro | real | Локальная переменная функции. Промежуточный результат вычислений. |
| i | integer | Локальная переменная функции. Параметр цикла. |
| vychsum\_chisl | real | Локальная переменная функции. Вывод на экран промежуточного результата (отладка, проверка и т.д.) |
| vychsum\_znam | real | Локальная переменная функции. Вывод на экран промежуточного результата (отладка, проверка и т.д.) |

1. Код программы:

**program** lr5\_task5;

**var**

sum\_chisl,sum\_znam,t: real;

n,x,k : integer;

**function** factorial\_summy(slagaemoe\_1,slagaemoe\_2:longint):real;

**var**

j : integer;

vychfactsum : real;

**begin**

vychfactsum := 1;

**for** j := 1 **to** (slagaemoe\_1+slagaemoe\_2) **do**

vychfactsum := vychfactsum\*j;

factorial\_summy := vychfactsum;

**end**;

**function** factorial\_proizvedeniy(mnojitel\_1,mnojitel\_2:longint):real;

**var**

i : integer;

vychfactpro : real;

**begin**

vychfactpro := 1;

**for** i := 1 **to** (mnojitel\_1\*mnojitel\_2) **do**

vychfactpro := vychfactpro\*i;

factorial\_proizvedeniy := vychfactpro;

**end**;

**function** summa\_v\_chislitele(sum\_per\_k:integer):real;

**var**

vychsum\_chisl : real;

**begin**

vychsum\_chisl := 0;

vychsum\_chisl := (exp((2\*sum\_per\_k+1)\*ln(x))/factorial\_summy(2\*sum\_per\_k,1));

writeln('Pri K = ',k,' vyr pod sum v chisl. = ',vychsum\_chisl:2:9);

summa\_v\_chislitele := vychsum\_chisl;

**end**;

**function** summa\_v\_znamenatele(sumzn\_per\_k:integer):real;

**var**

vychsum\_znam : real;

**begin**

vychsum\_znam := 0;

vychsum\_znam := (exp((2\*sumzn\_per\_k)\*ln(x))/factorial\_proizvedeniy(2,sumzn\_per\_k));

writeln('Pri K = ',k,' vyr pod sum v znam. = ',vychsum\_znam:2:9);

summa\_v\_znamenatele := vychsum\_znam;

**end**;

**begin**

writeln('Vvedite N: '); readln(n);

writeln('Vvedite X: '); readln(x);

sum\_chisl := 0;

sum\_znam := 0;

t := 0;

writeln();

**for** k := 1 **to** n **do begin**

sum\_chisl := sum\_chisl + summa\_v\_chislitele(k);

sum\_znam := sum\_znam + summa\_v\_znamenatele(k);

writeln('Pri K = ',k,' sum v chisl. = ',sum\_chisl:2:9);

writeln('Pri K = ',k,' sum v znam. = ',sum\_znam:2:9);

writeln();

**end**;

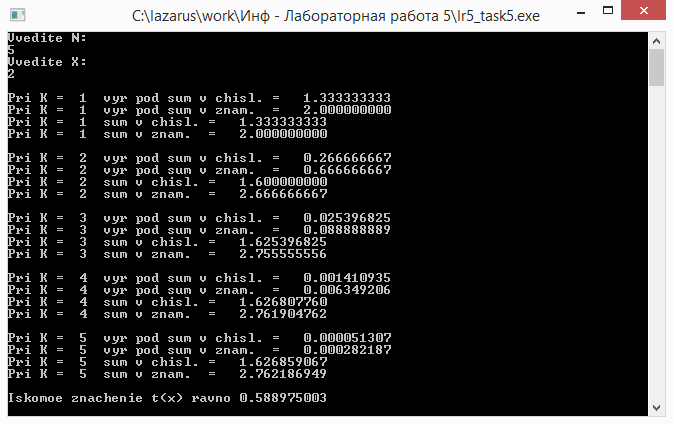
t := sum\_chisl/sum\_znam;

writeln('Iskomoe znachenie t(x) ravno ',t:2:9);

readln();

**end**.

1. Результат работы программы:



1. Анализ результатов вычисления:

Мы задаём функции для нахождения:

Факториала выражения (a+b) – в общем виде.

Факториала выражения (a\*b) – в общем виде.

Значения суммы в числителе.

Значения суммы в знаменателе.

Введём значения X и N. Обнулим необходимые переменные.

После чего мы задаем цикл с параметром К, который изменятеся от 1 до n. Посчитаем суммы, выводя на экран промежуточные значения, чтобы облегчить себе проверку и отладку.

Найдем значение переменной t по исходной формуле и выведем его на экран.

1. Вывод: мы написали программу, которая вычисляет значение данного выражения при заданных X и N.