Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №4

по курсу «Программирование»

**Работа в графическом режиме**

Выполнил студент группы ИВТ-11 \_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М./

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Чистяков Г. А./

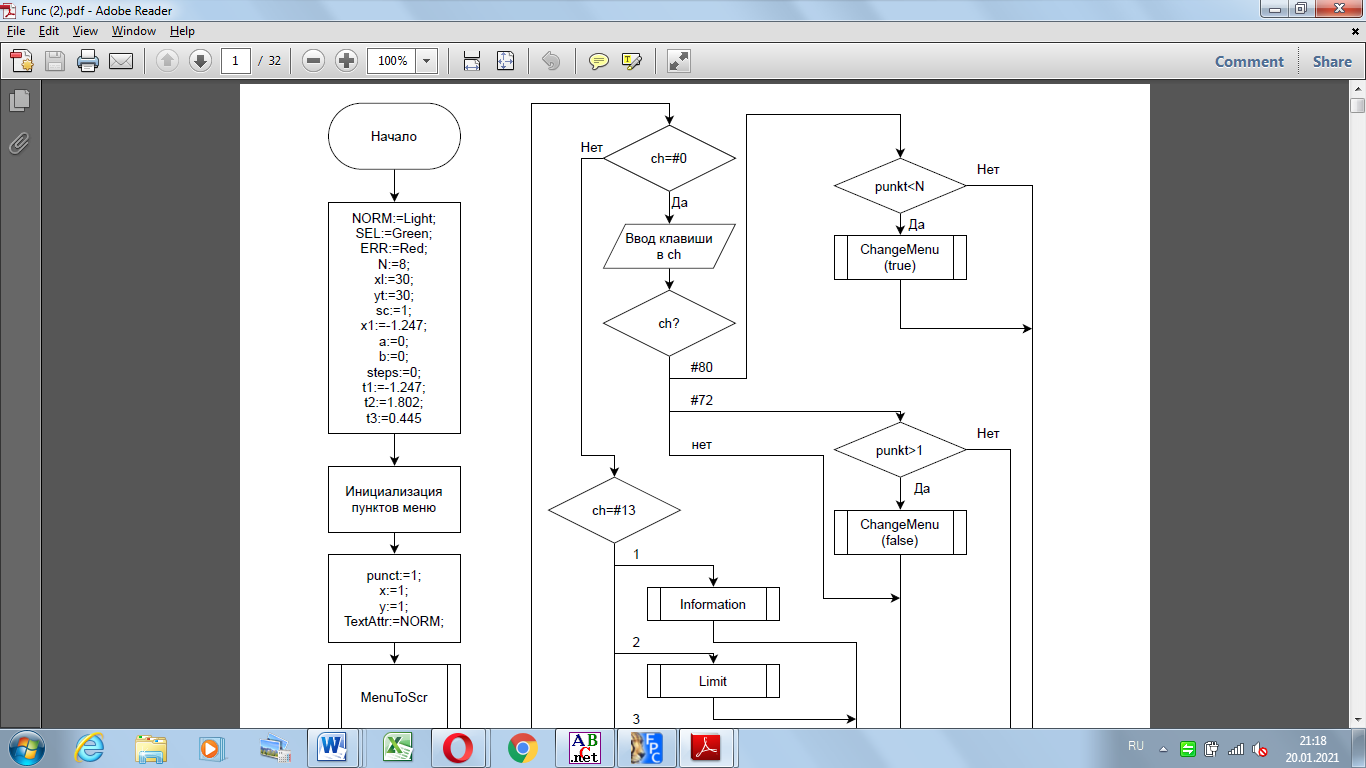
Киров 2021

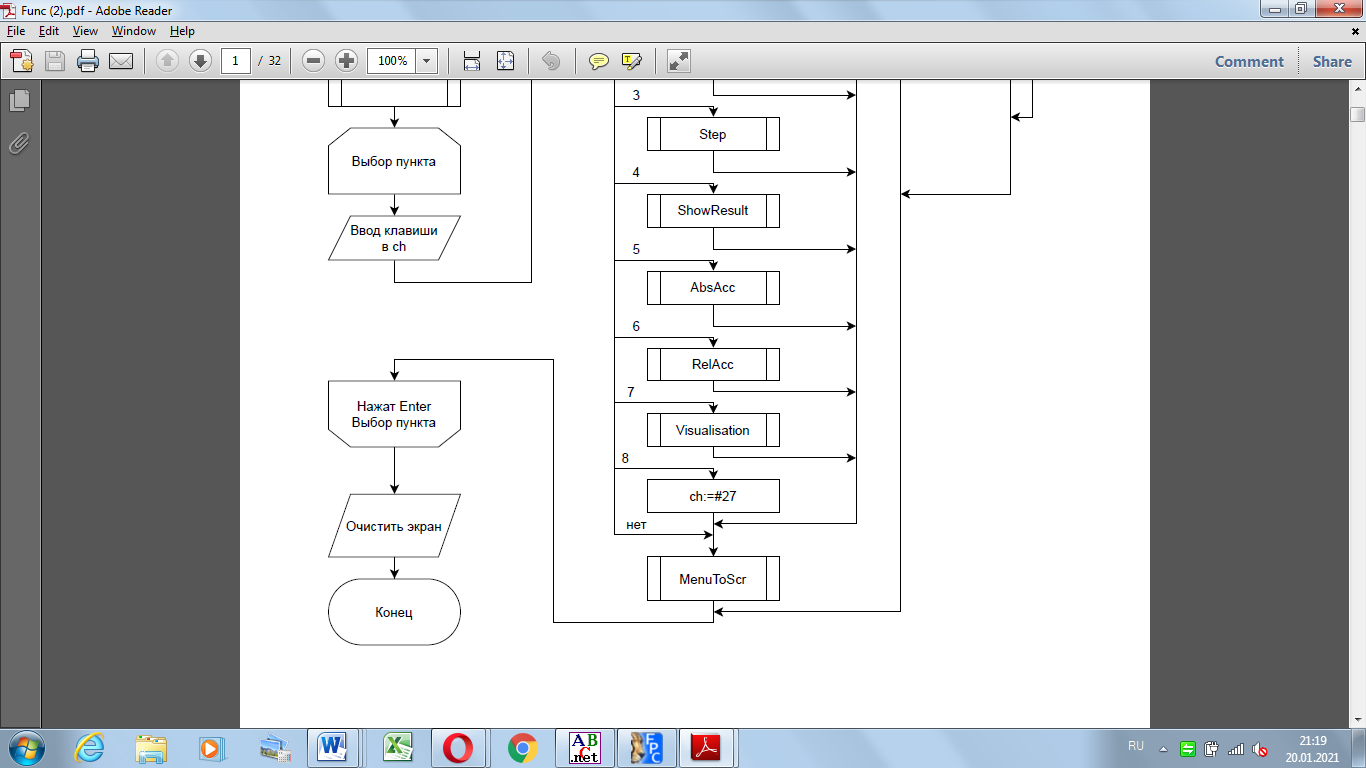
**Цель работы:** освоить принципы работы в графическом режиме; получить базовые навыки взаимодействия с графическими примитивами.

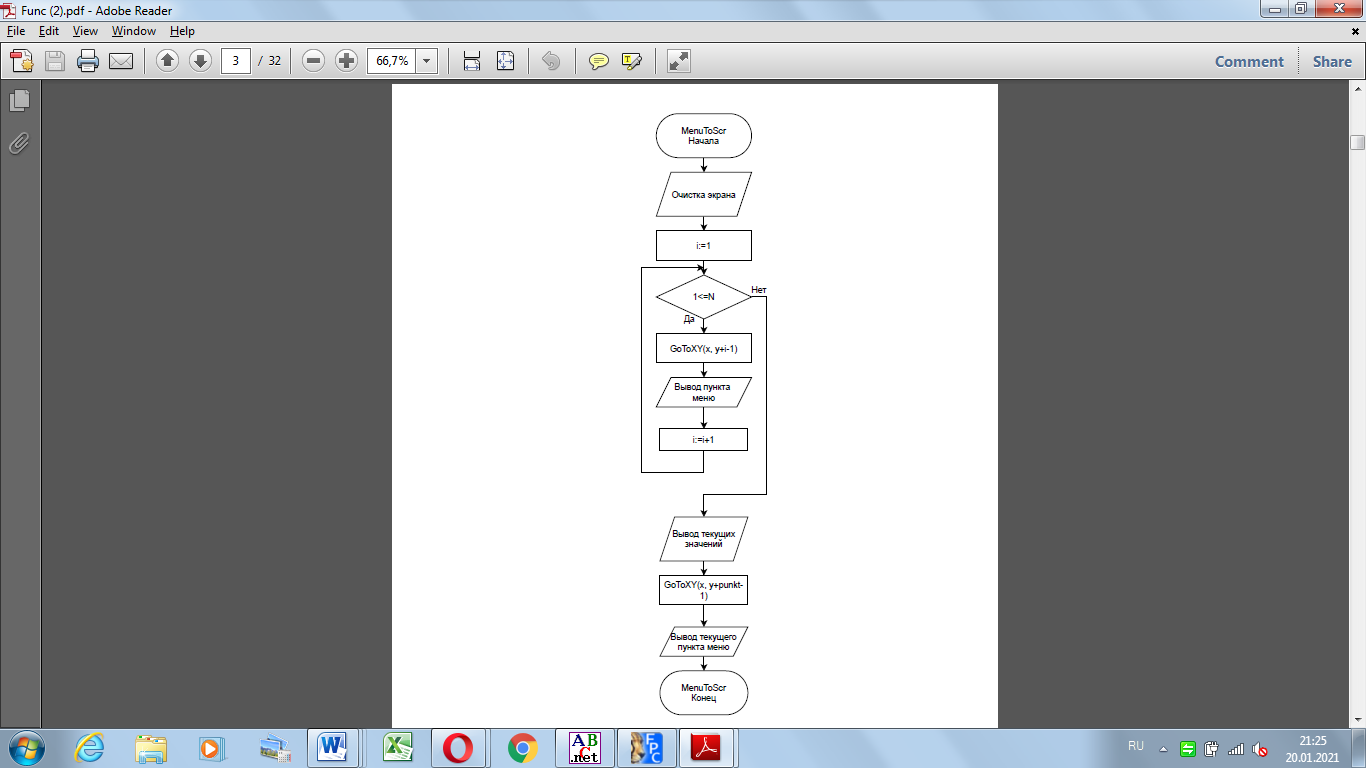
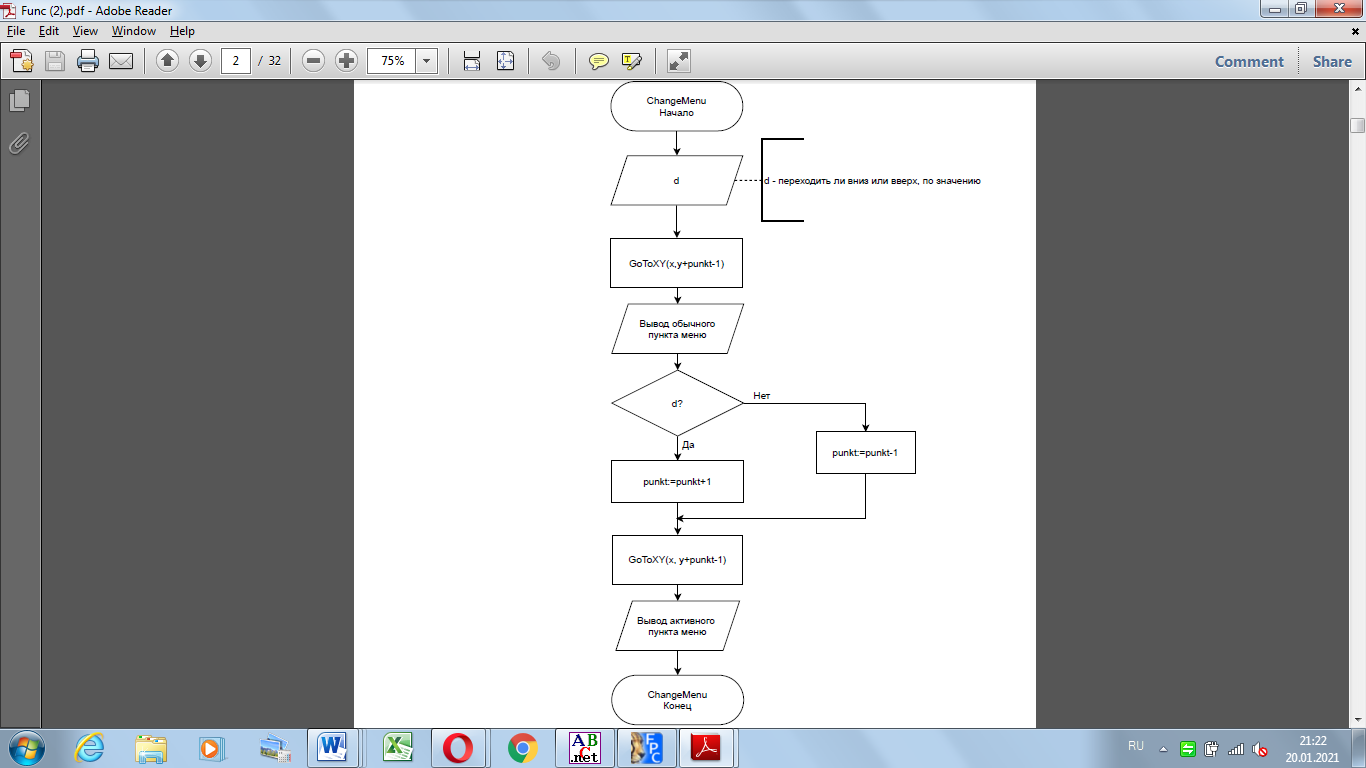
**Задание:**

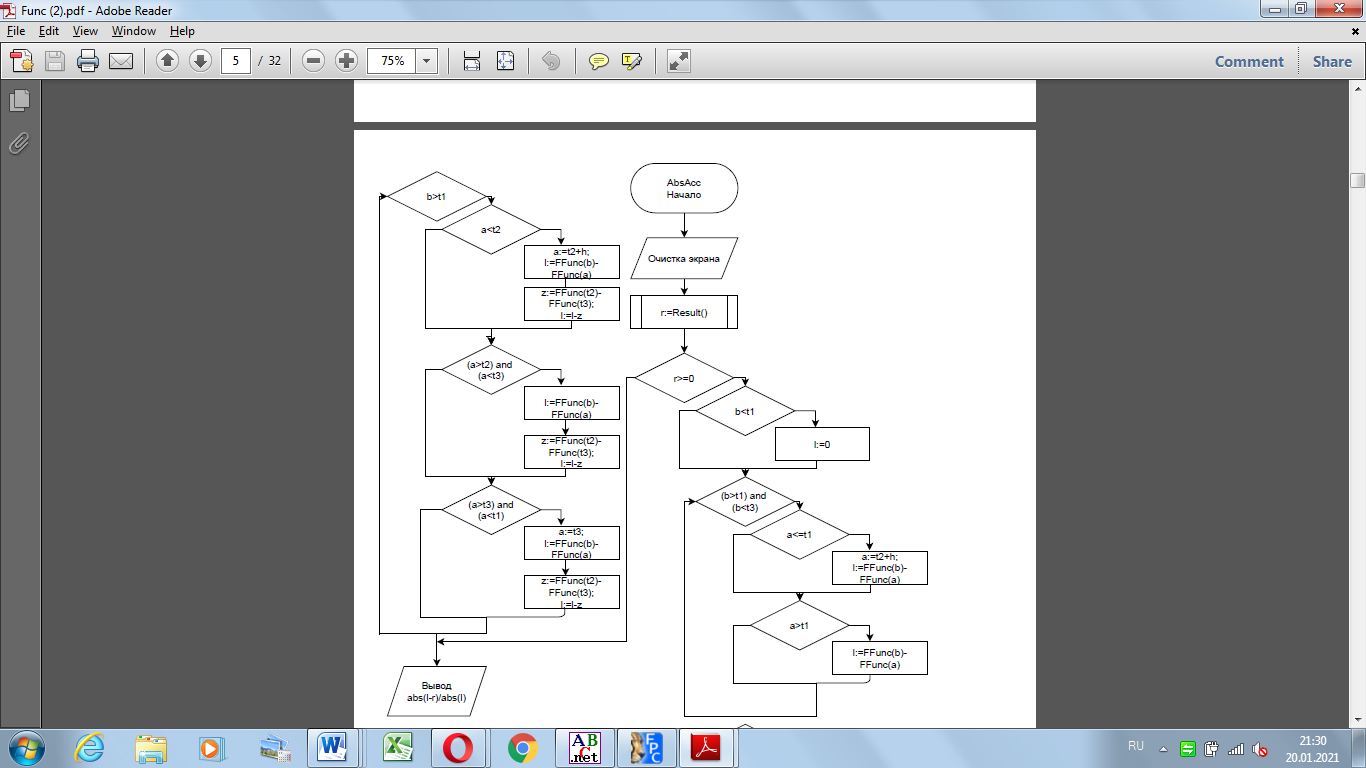
1. Дополнить программу, реализованную в ходе предыдущей лабораторной работы, режимом визуализации.  
2. Предусмотреть возможность вывода кривой, ограничивающей фигуру, на координатную плоскость.  
3. Реализовать следующие возможности и элементы: масштабирование графика, подписи на осях, вывод информации о задании.  
4. Реализовать не менее двух возможностей из представленных: независимое масштабирование по осям, штриховка вычисляемой площади, визуализация численного расчета интеграла.

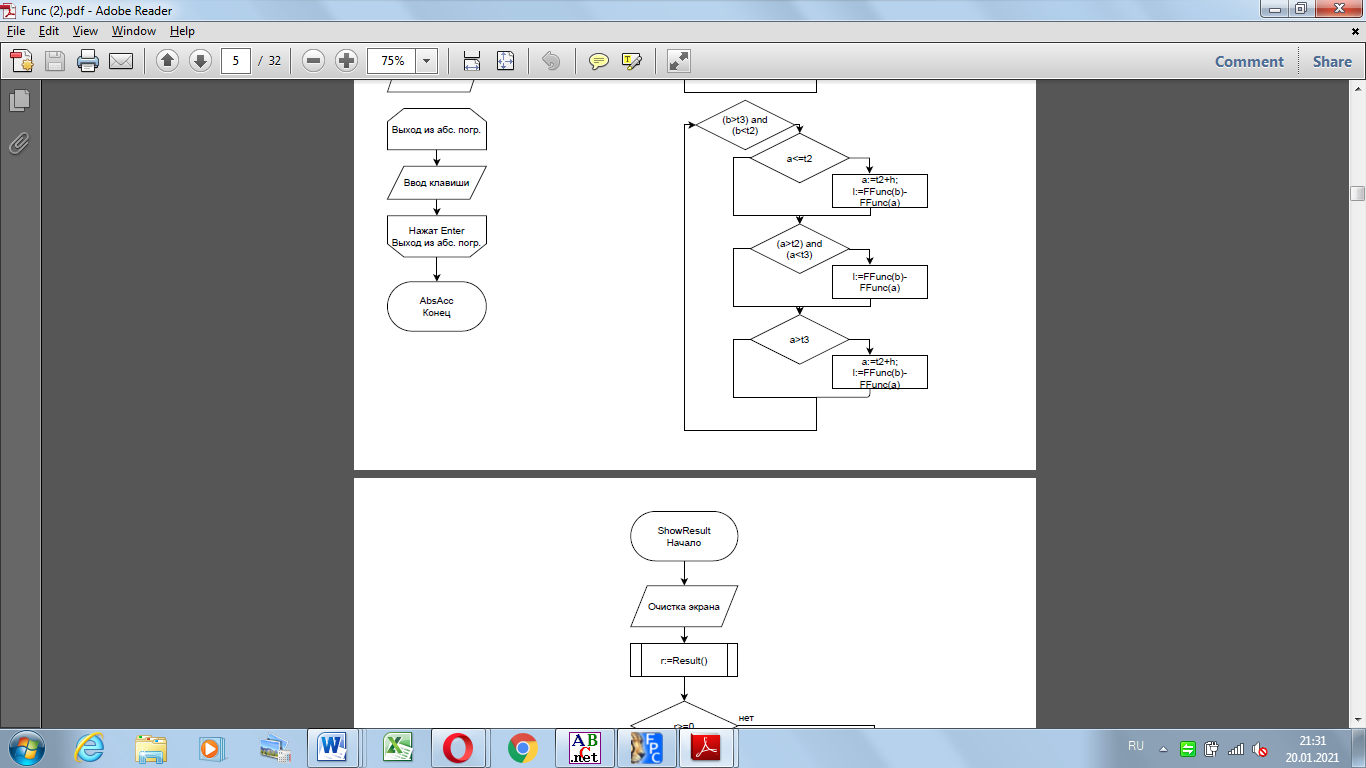
**Схема алгоритма:**

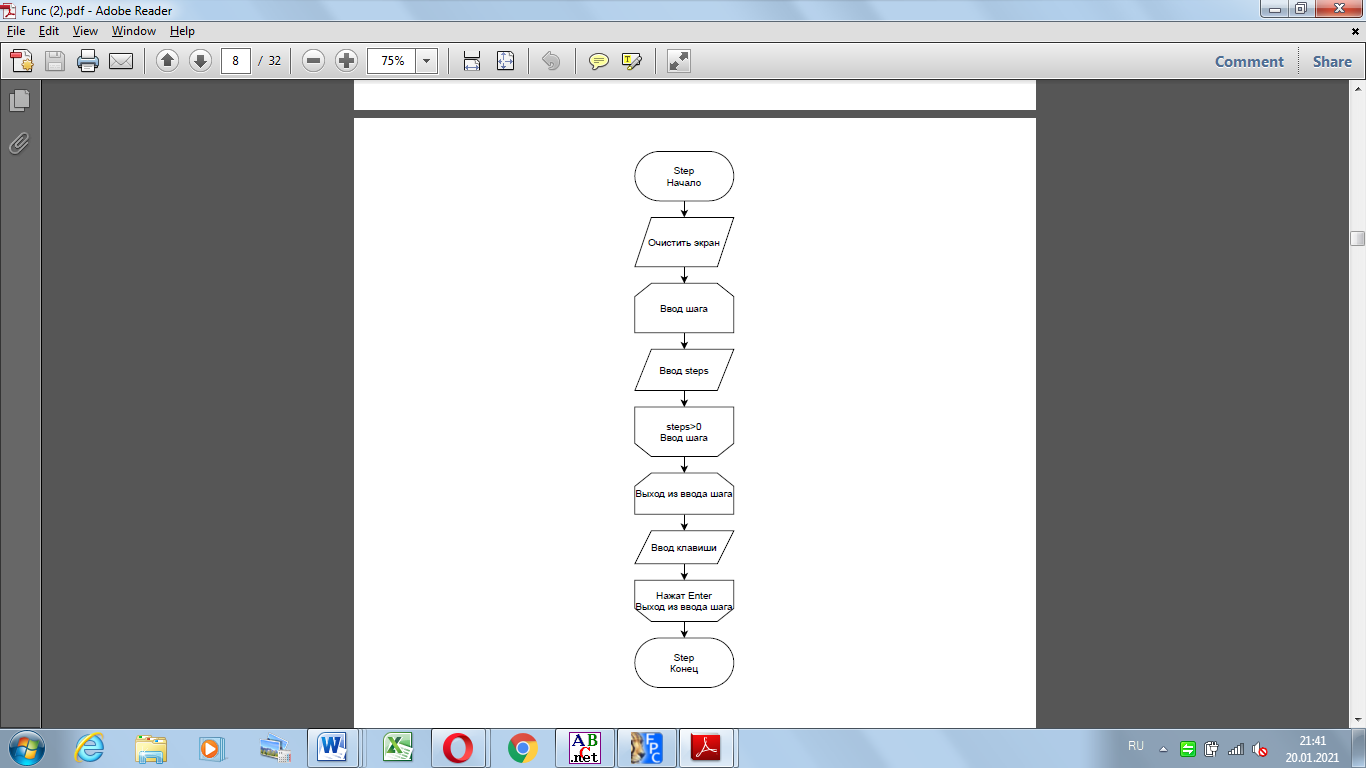
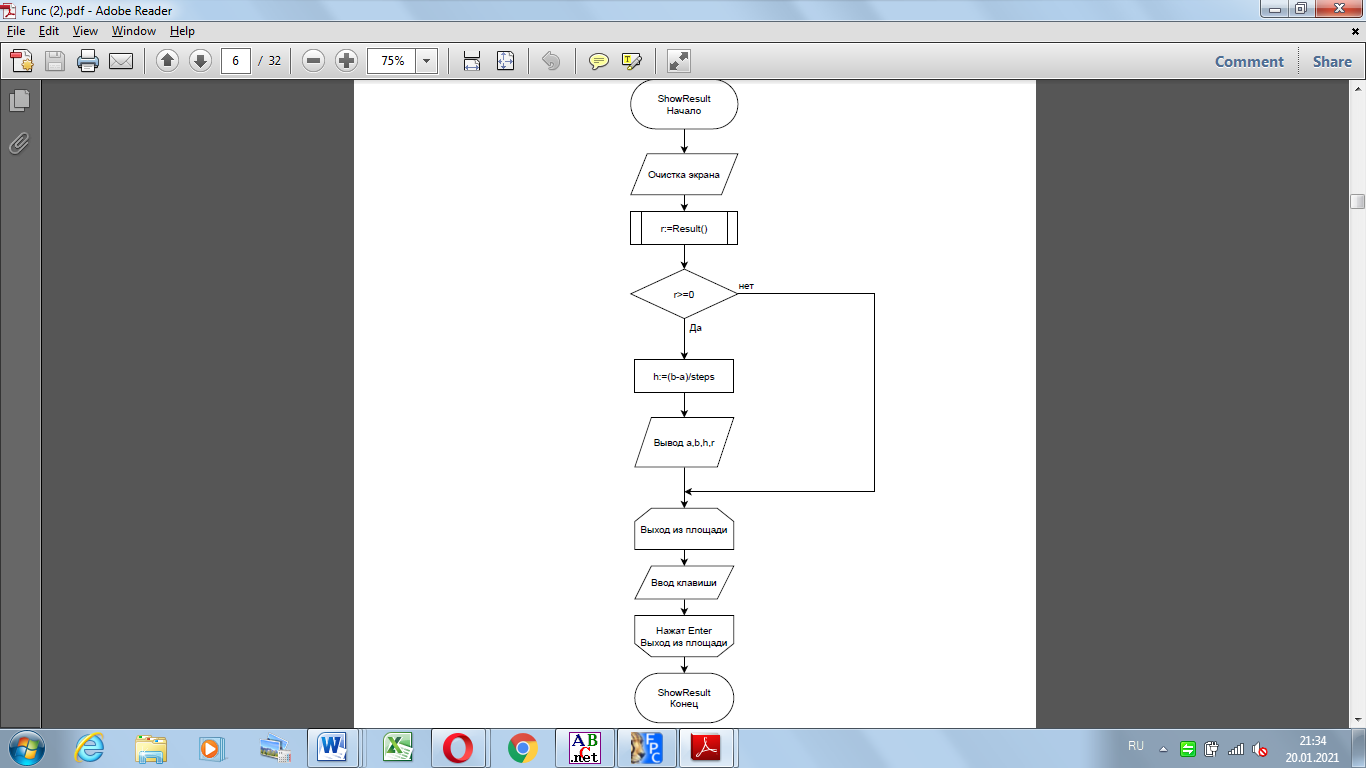


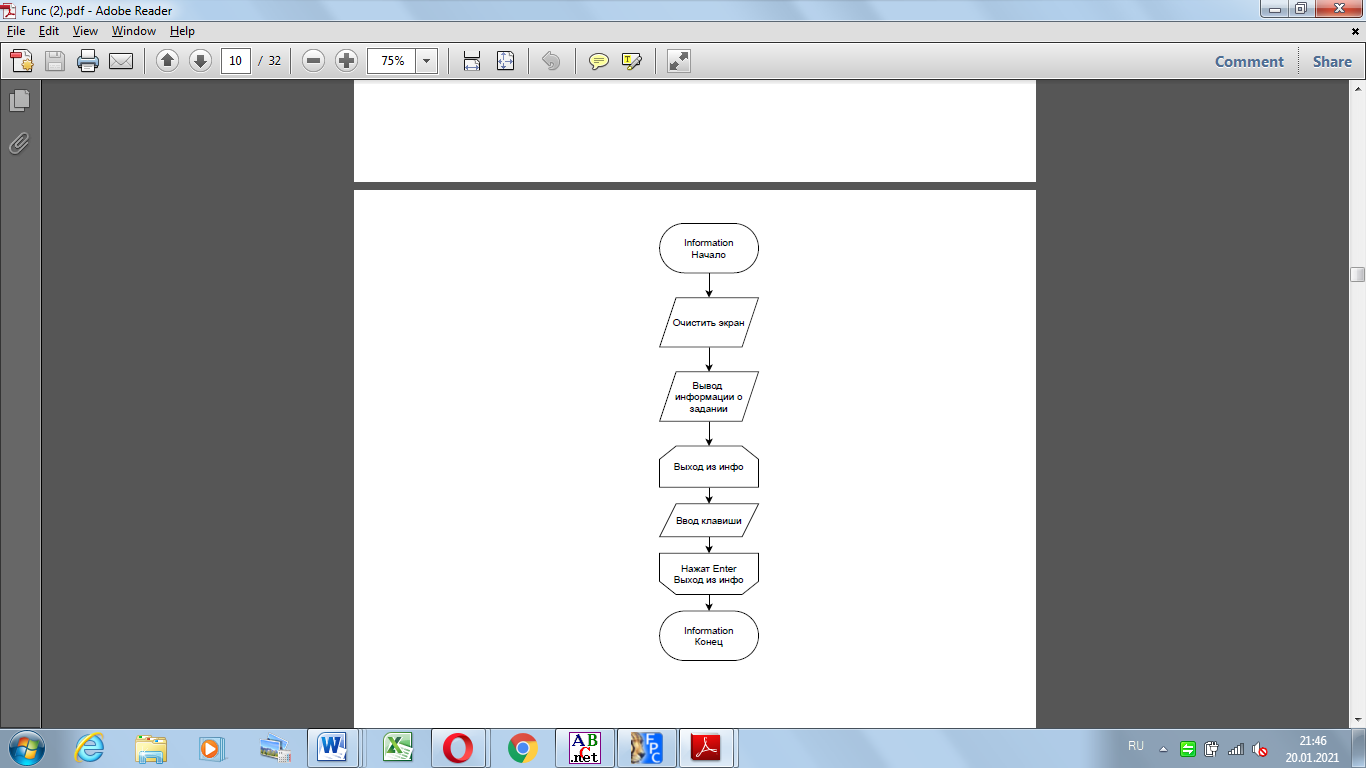


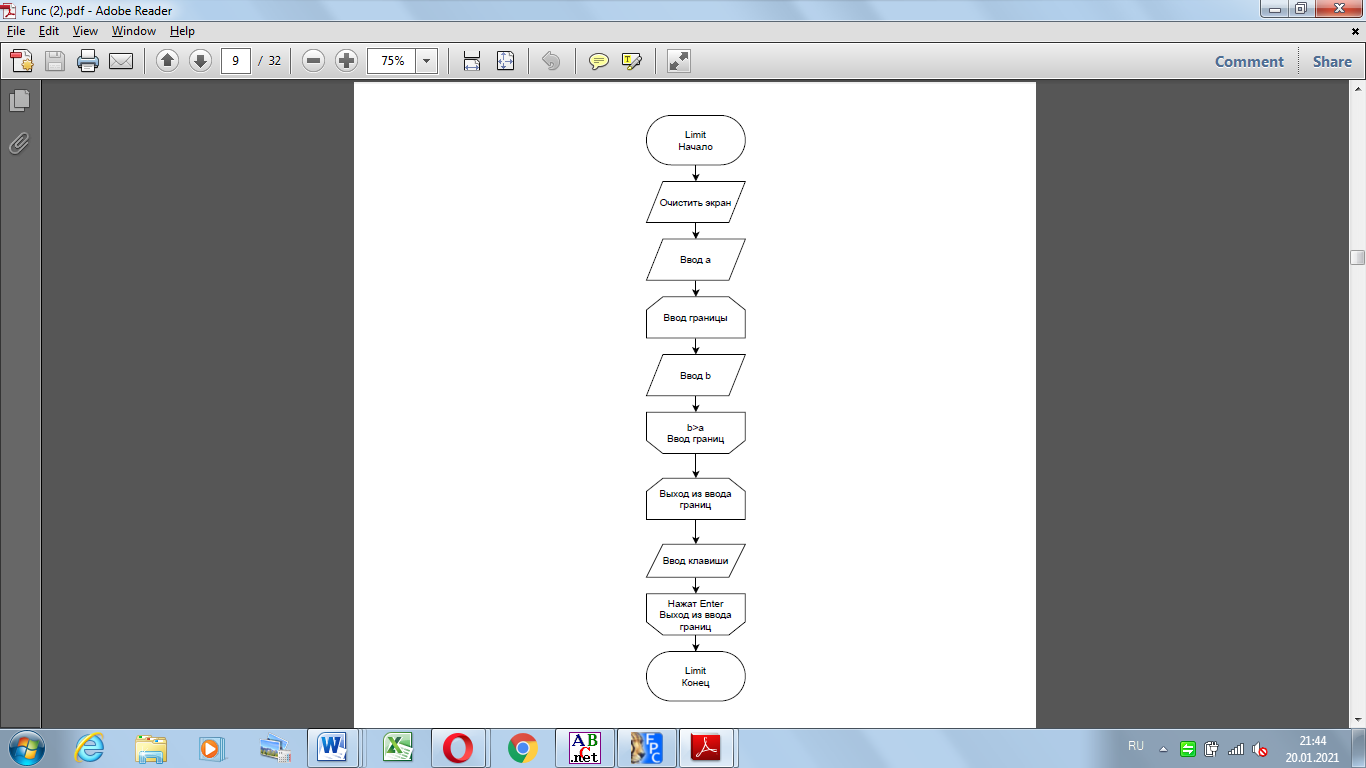
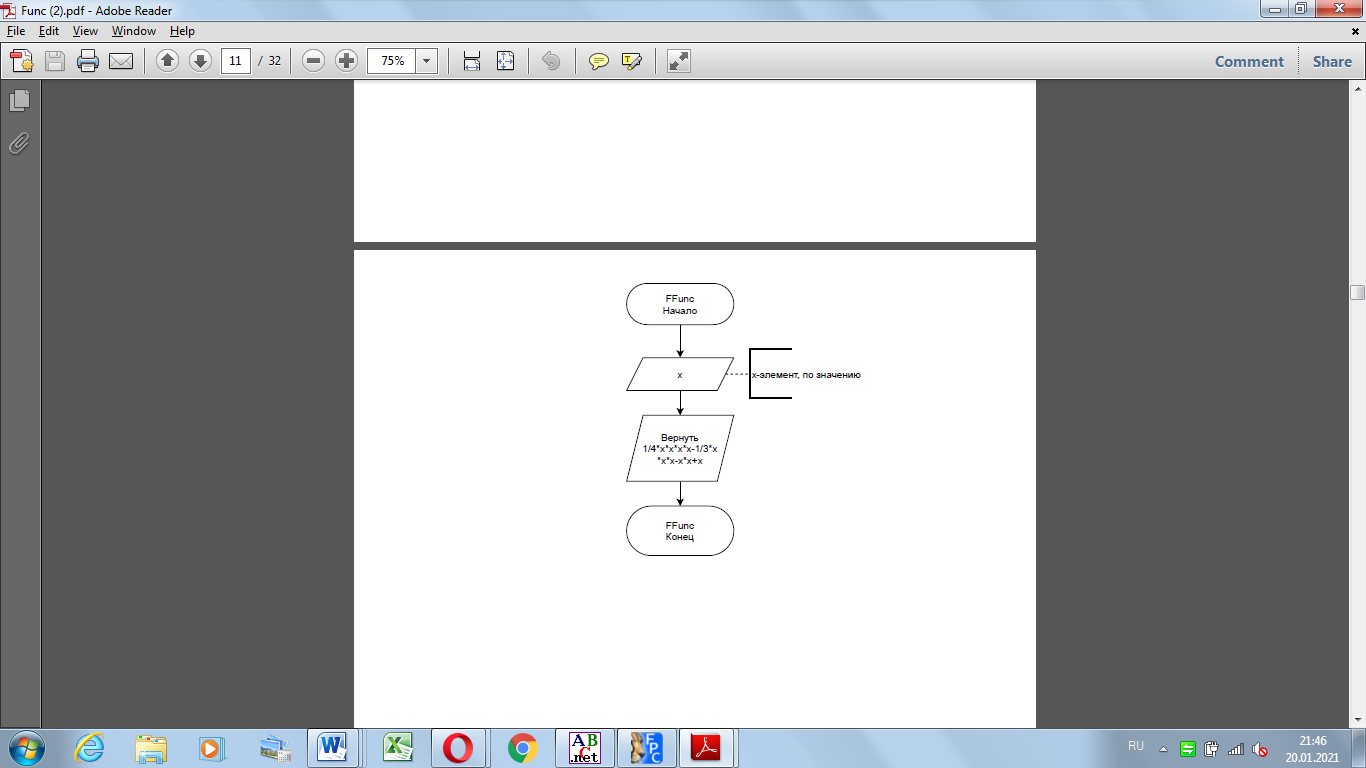


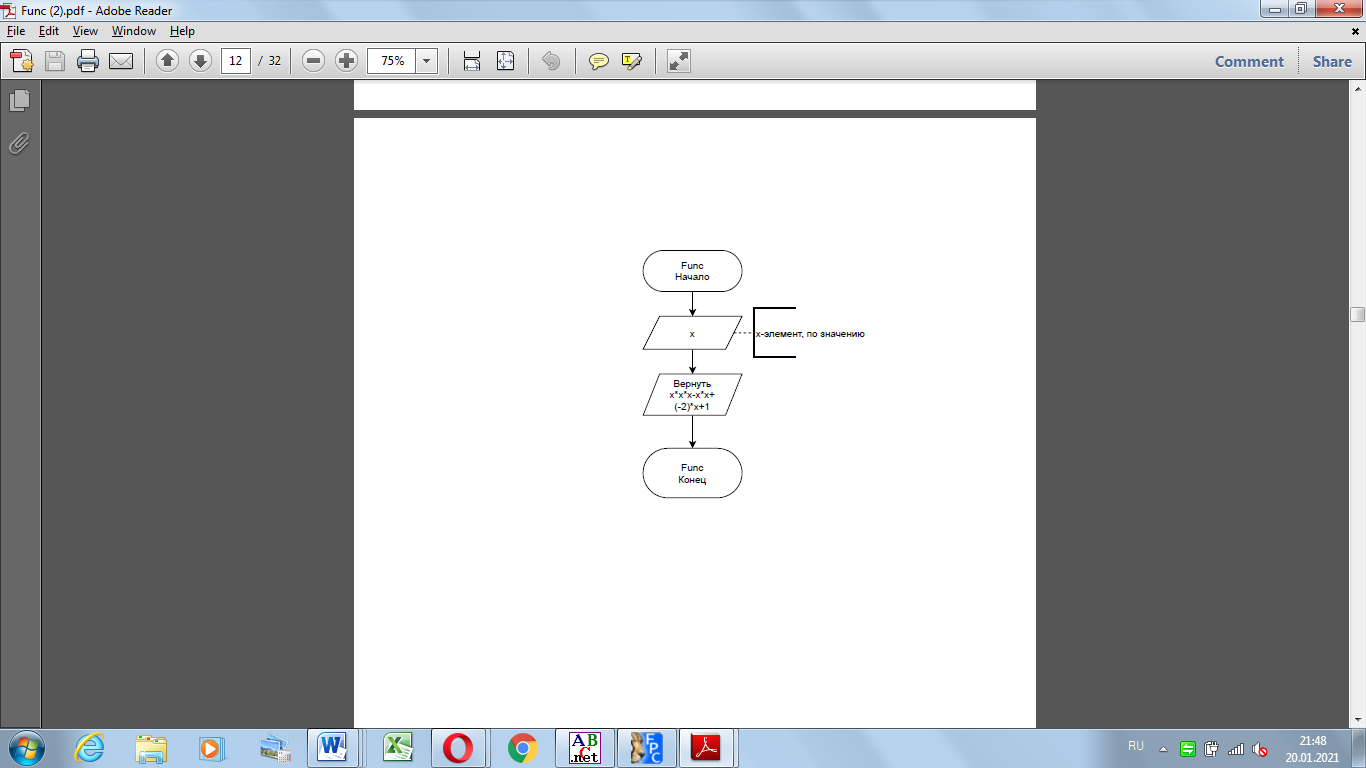


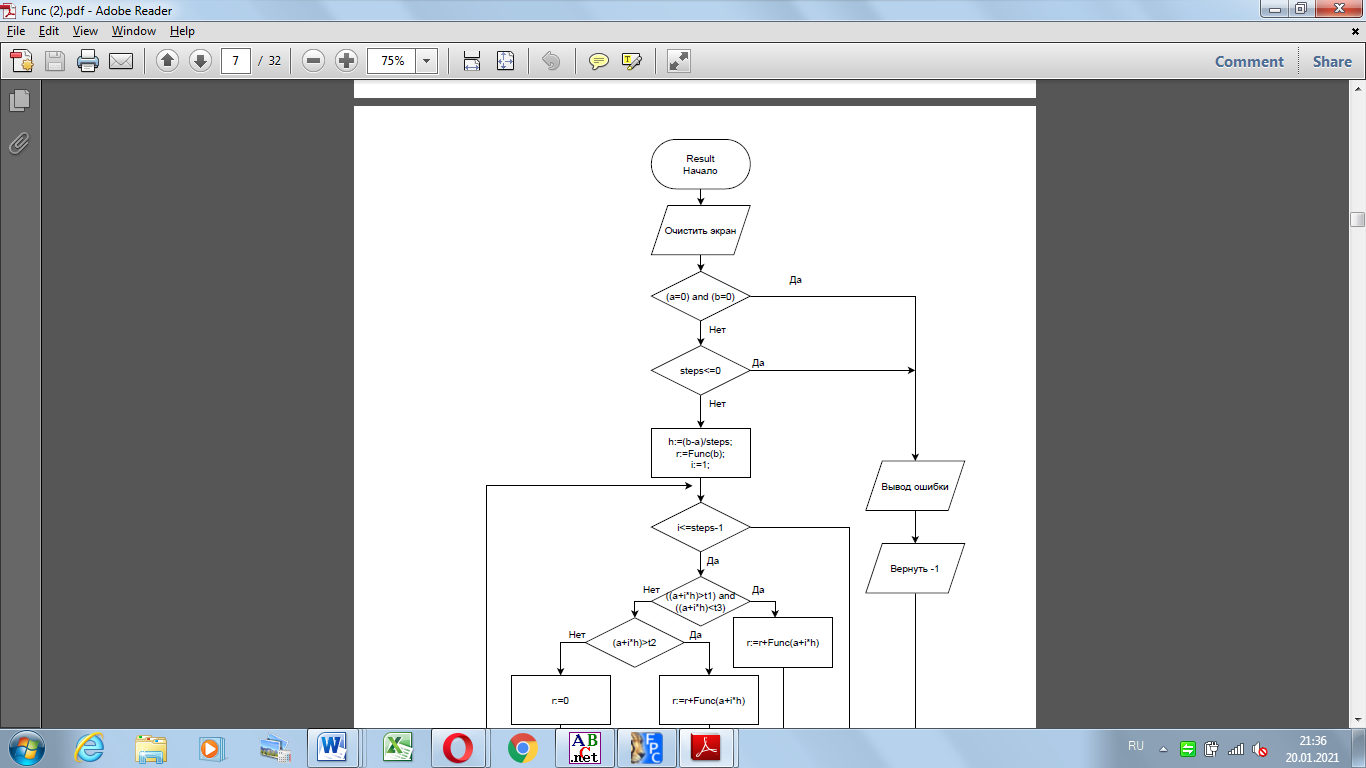


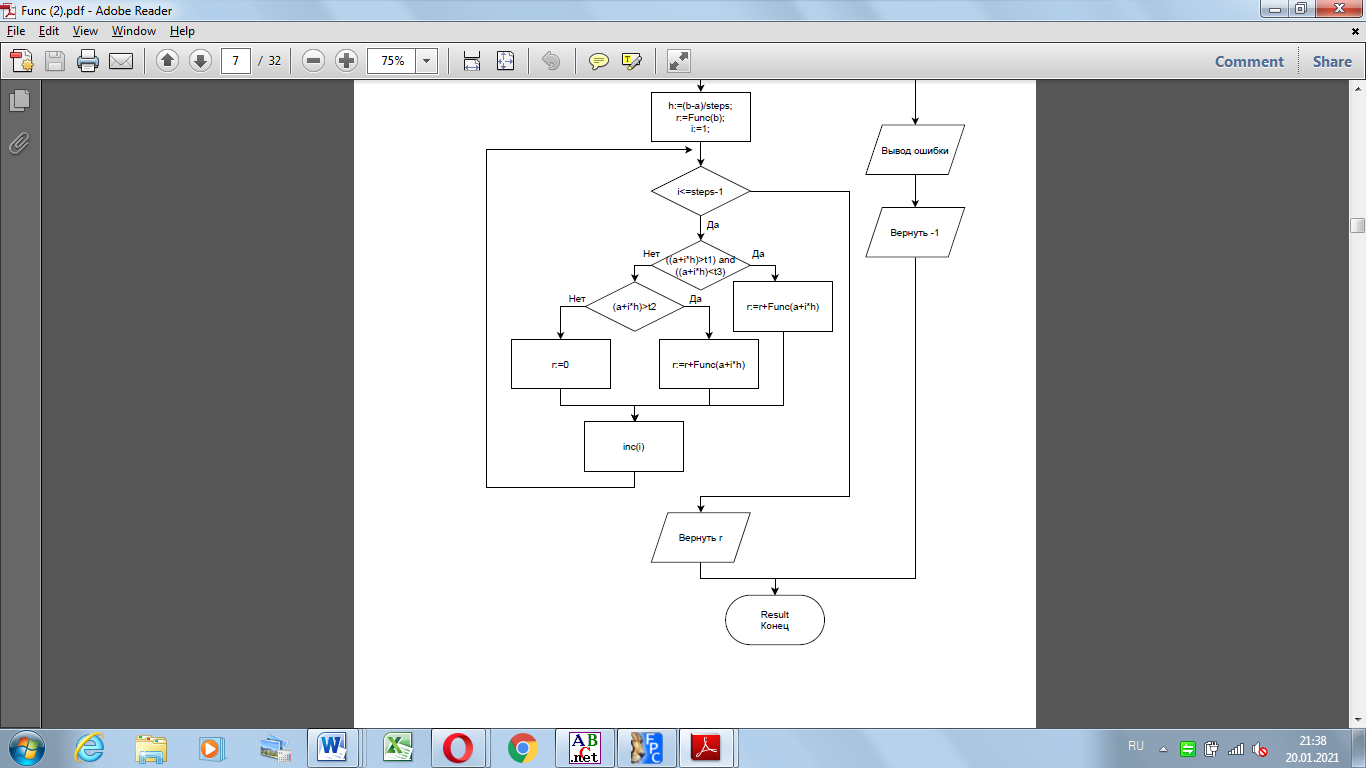


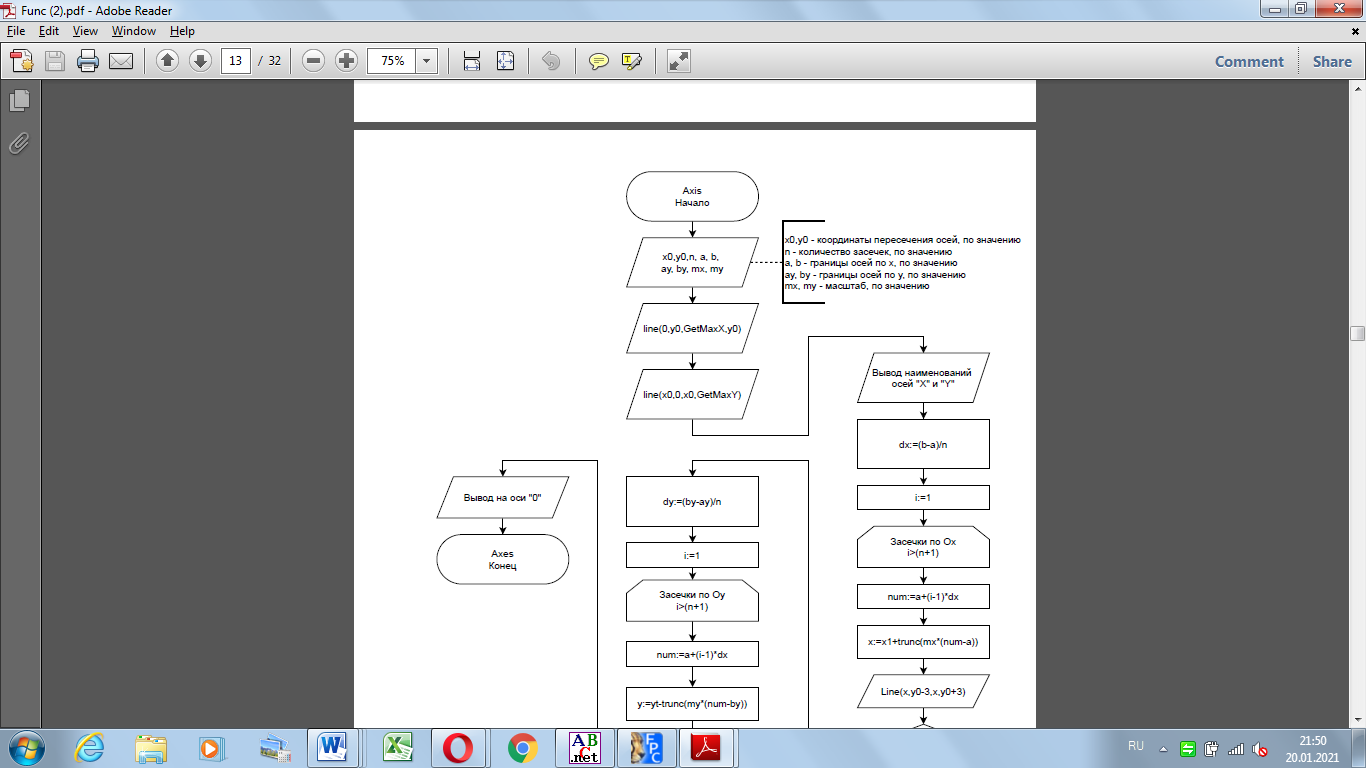


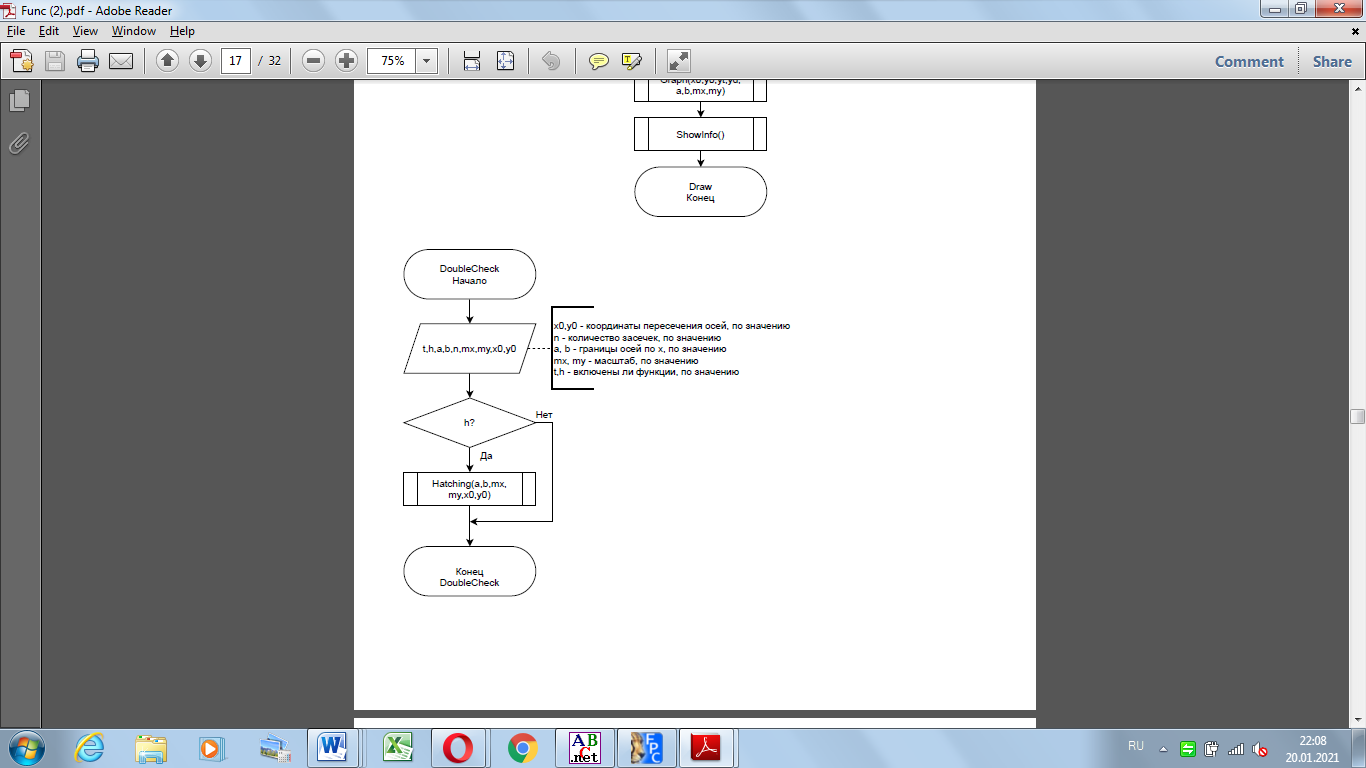
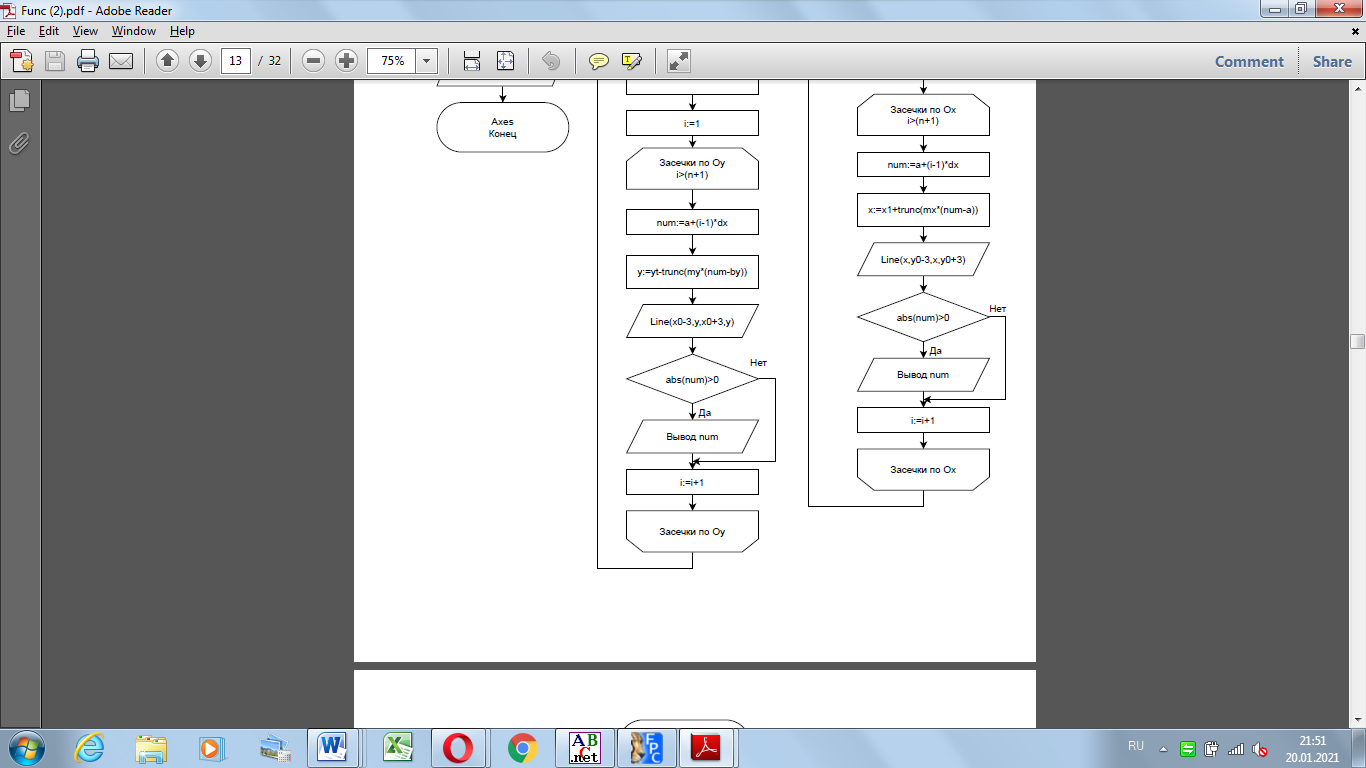
 

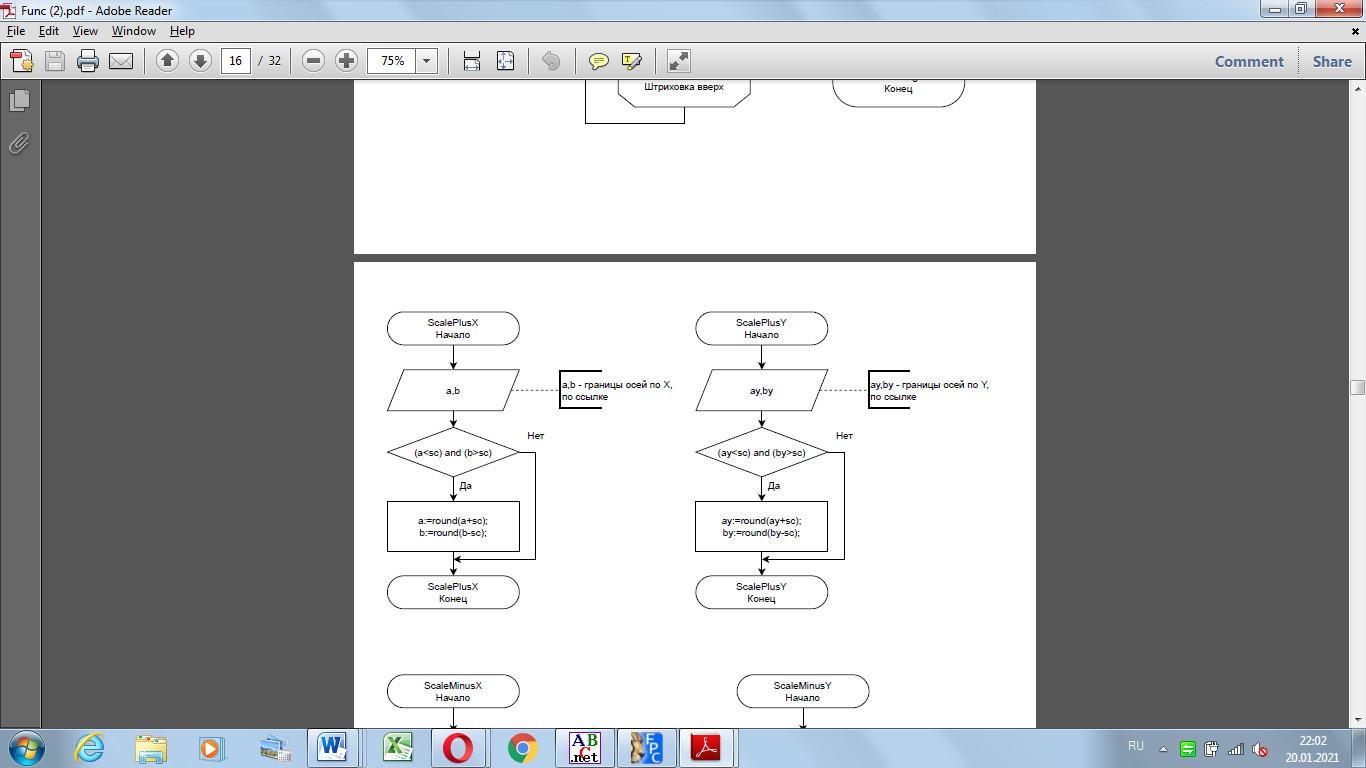
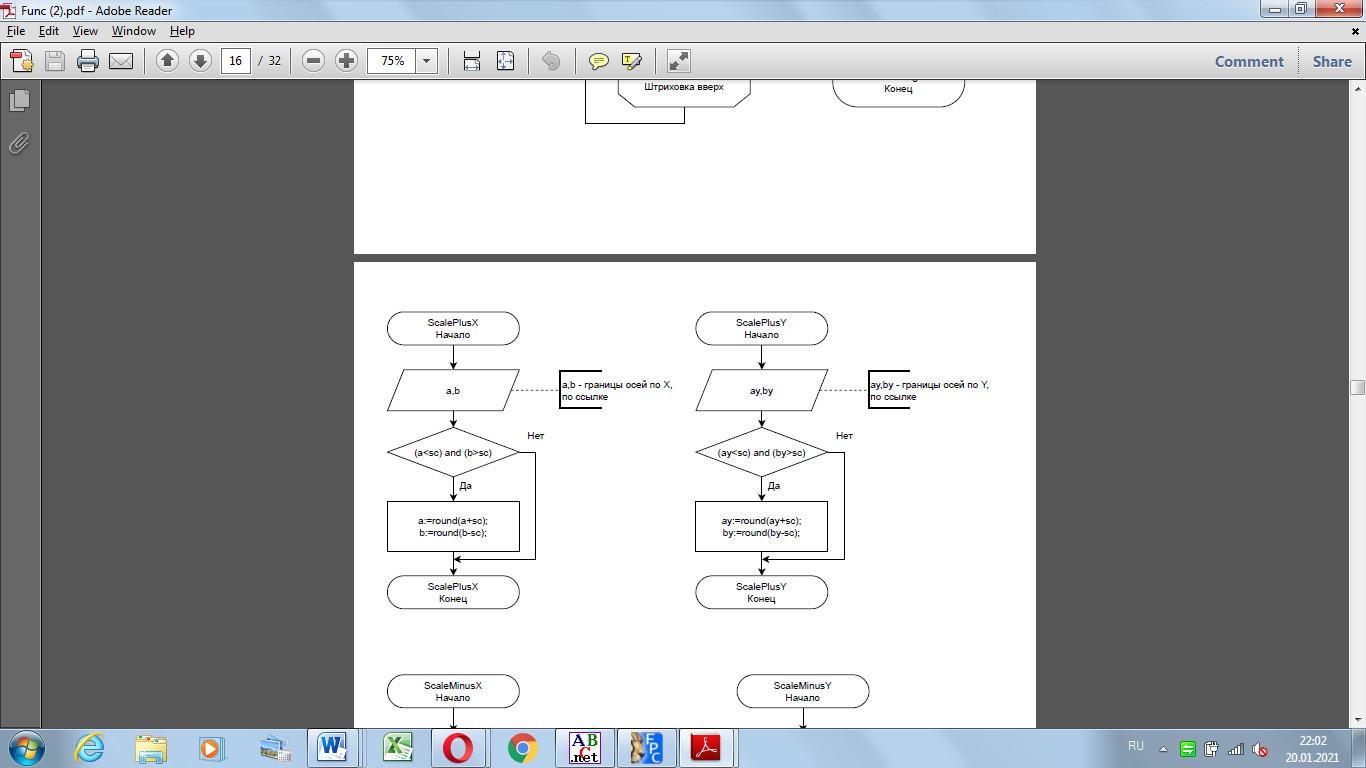


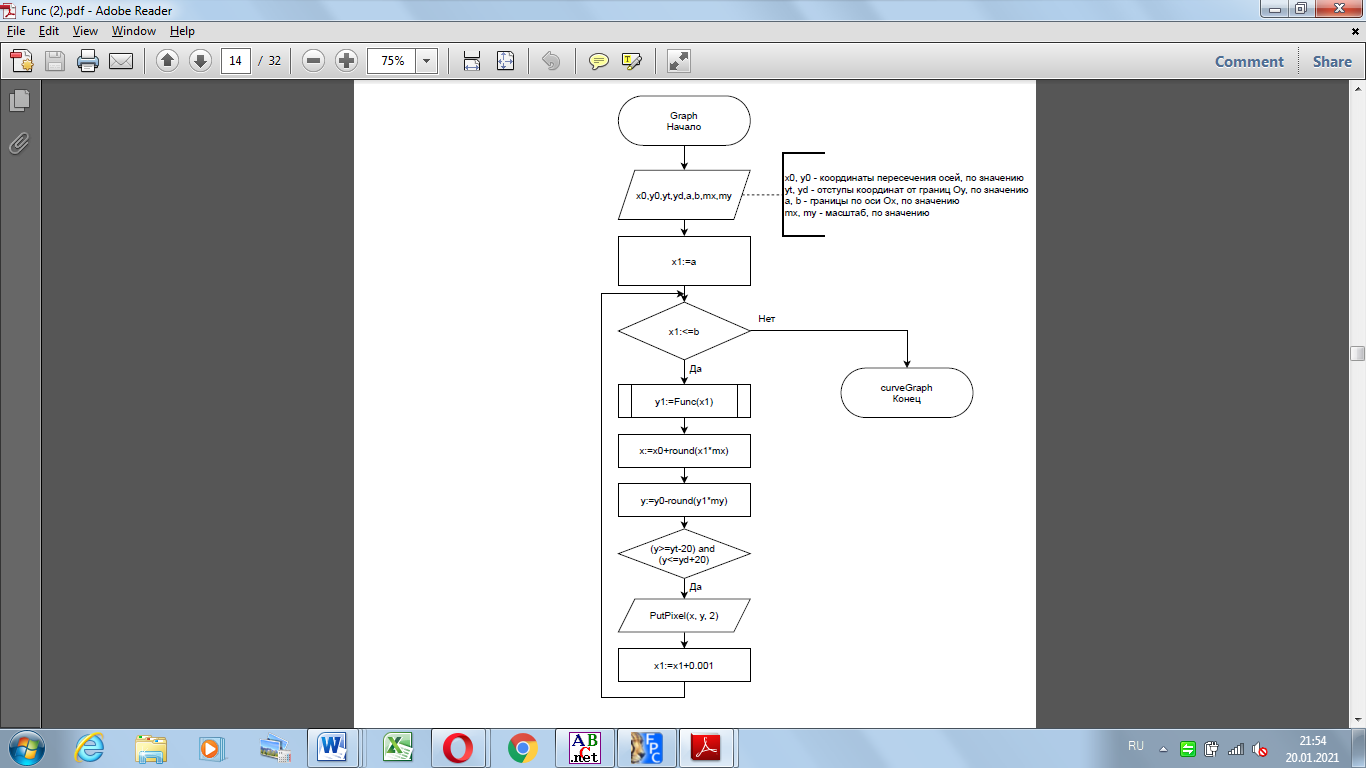
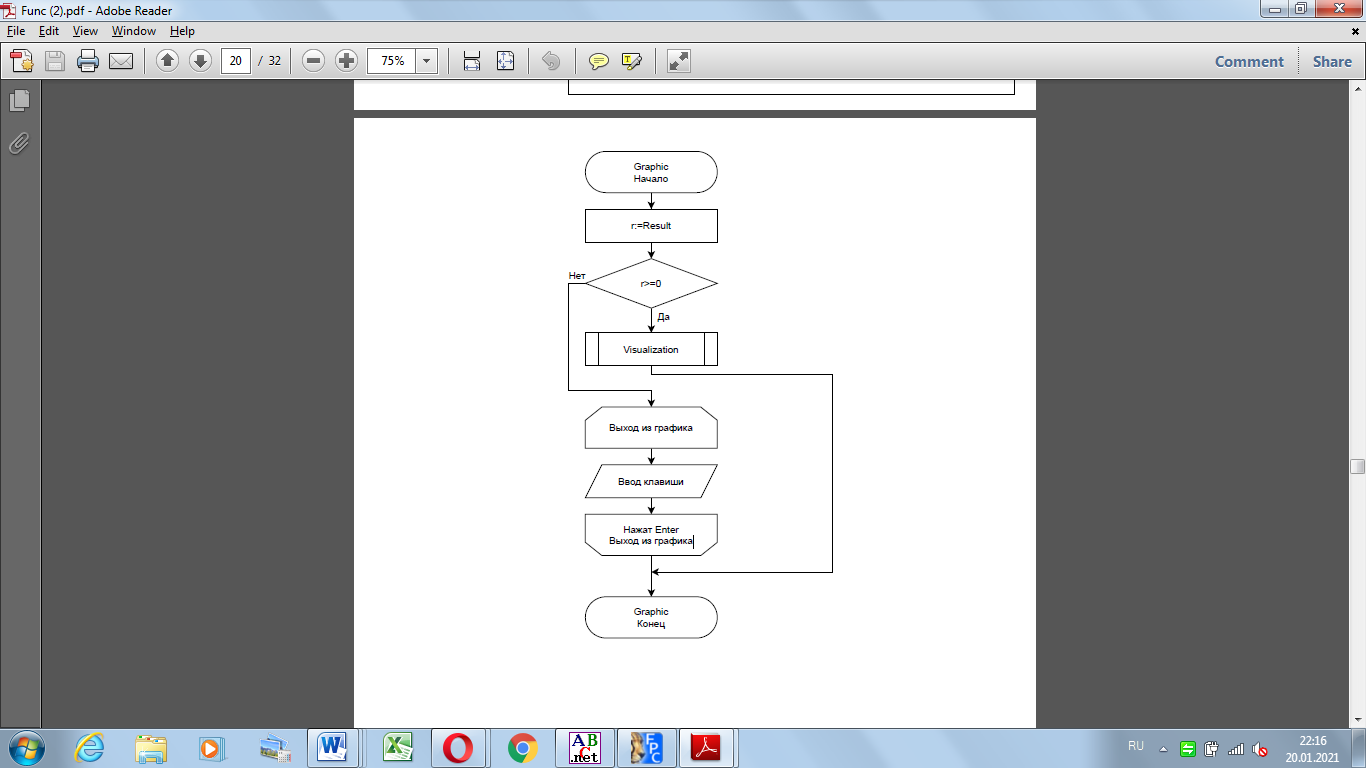


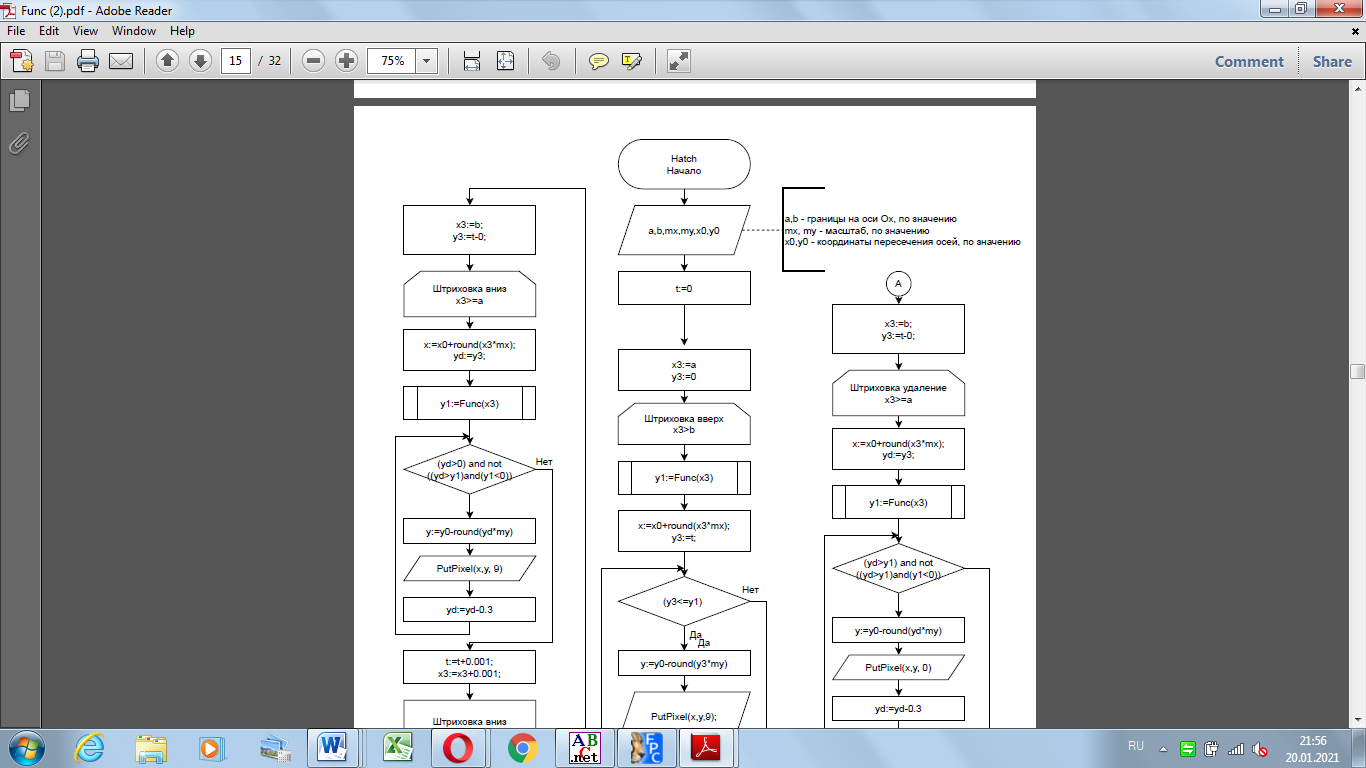


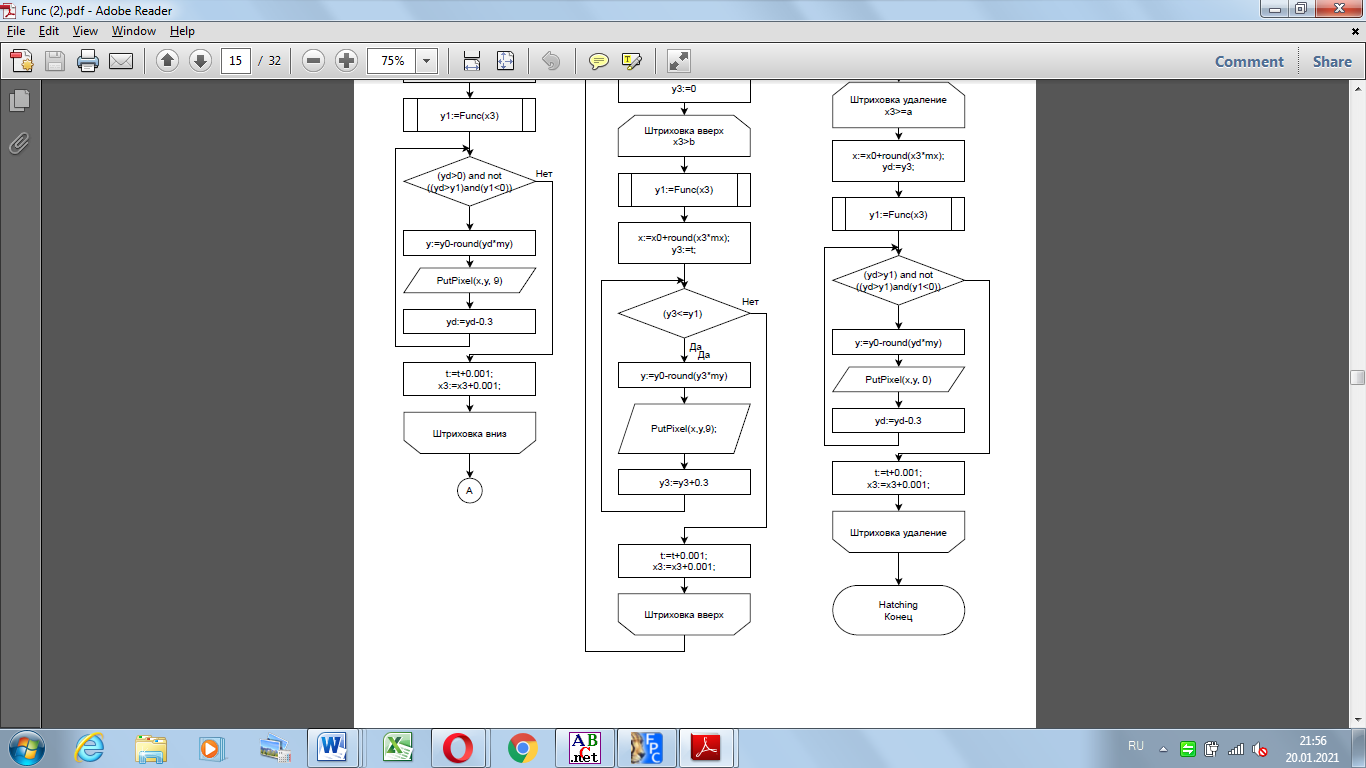


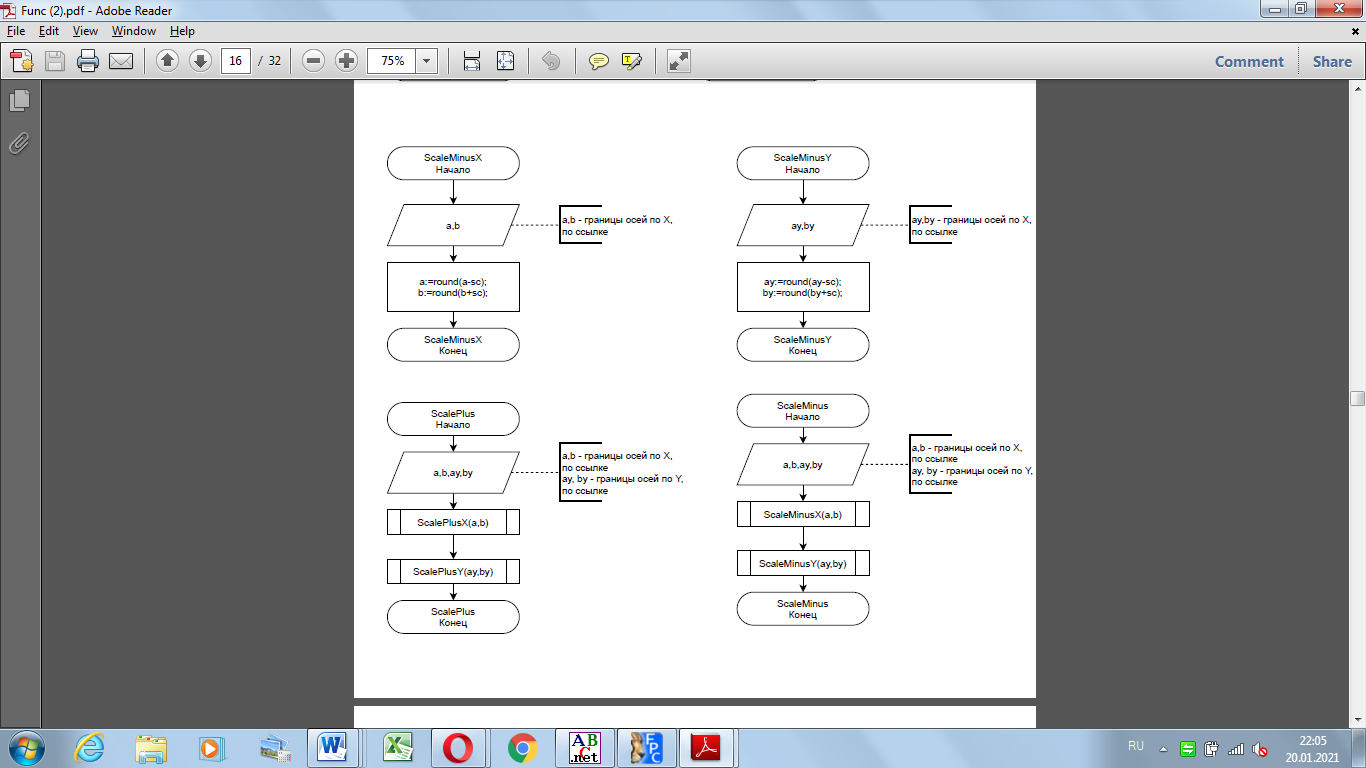


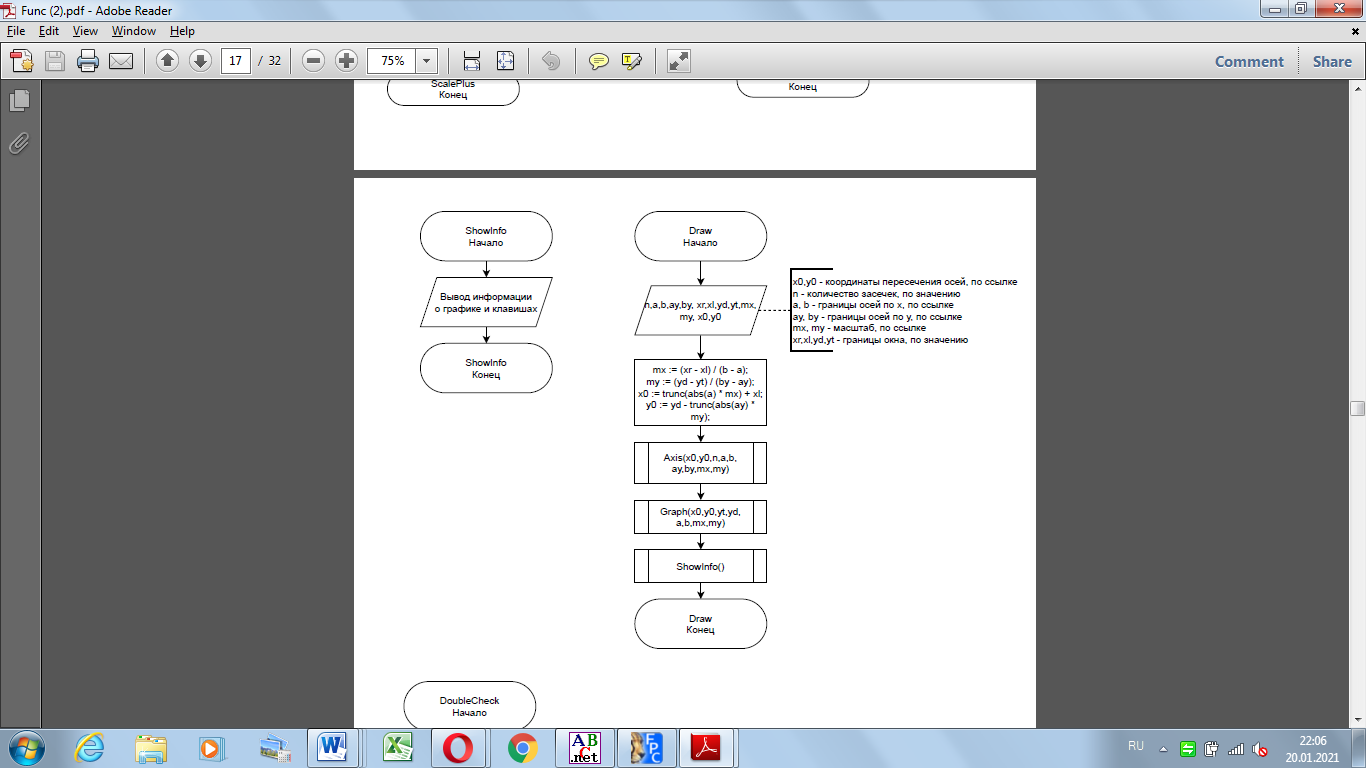


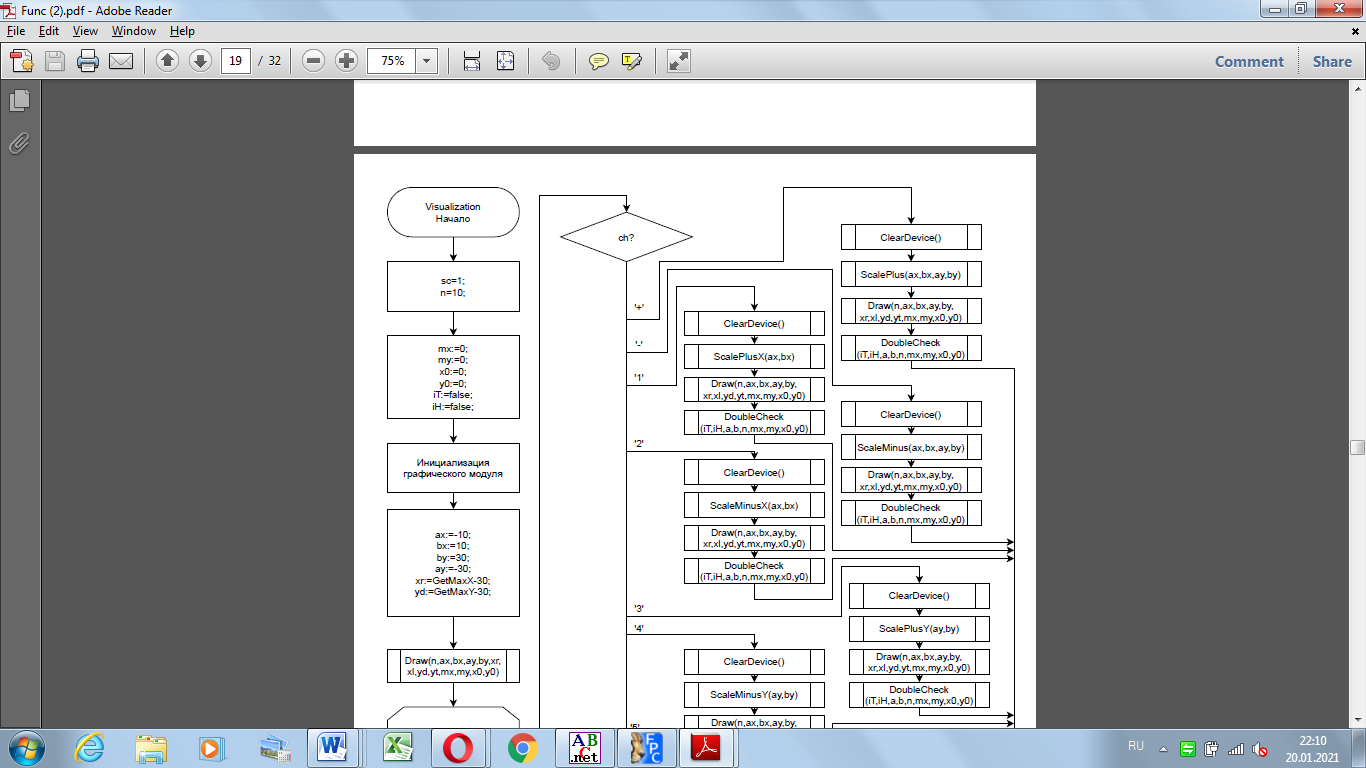


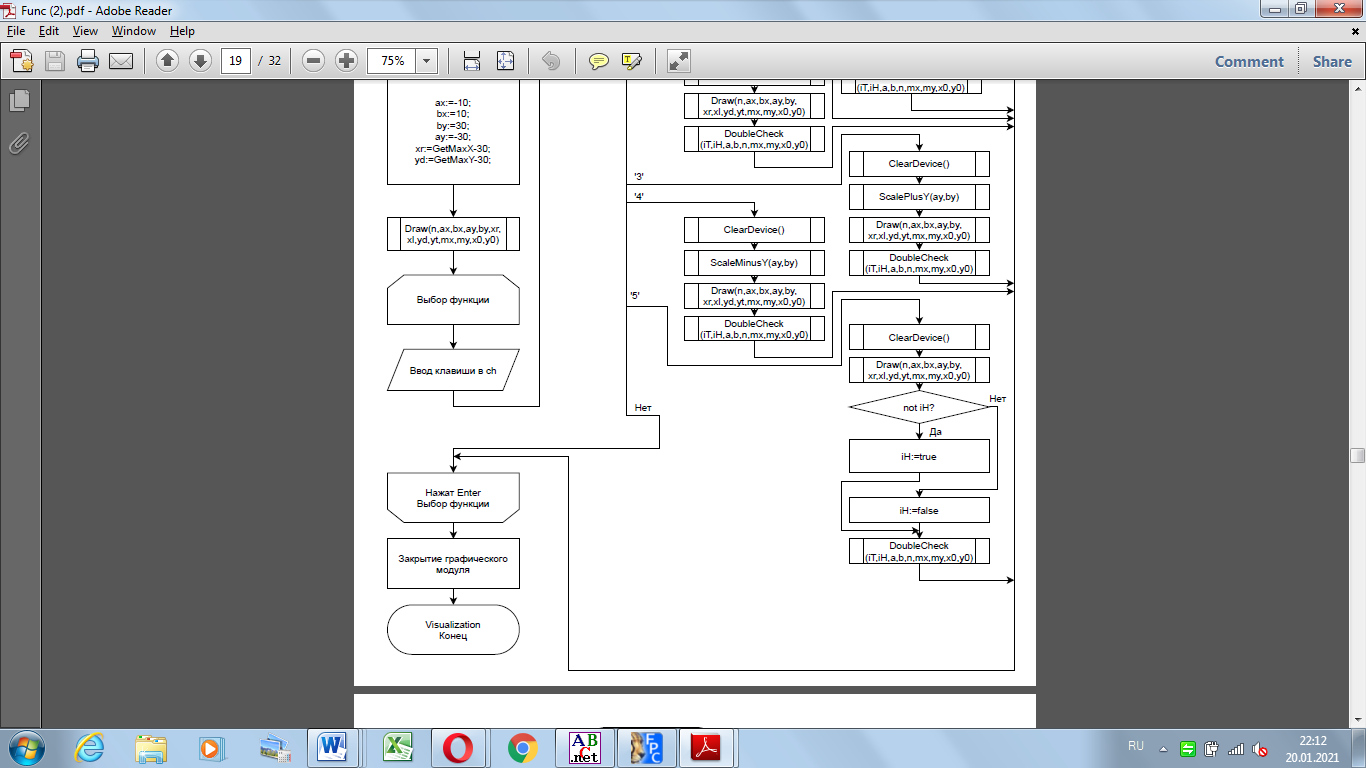












Листинг программы

program lab3;

uses graph,Crt,wincrt, math;

const

NORM=LightGray;

SEL=Green;

ERR=Red;

N=8;

x1=-1.247;

t1=-1.247;

t2=1.802;

t3=0.445;

eps=1e-6;

sc=1;

xl=30;

yt=30;

var

menu:array[1..N] of string;

punkt:integer;

ch:char;

Gd,Gm:integer;

my,mx:real;

ay,by:integer;

x0,y0:integer;

s:string;

x,y:integer;

a: real=0;

b: real=0;

c: real=0;

d: real=0;

i: integer;

steps: integer=0;

xr,yd:integer;

Procedure WriteError(str:string);

begin

crt.TextAttr:=ERR;

writeln(str);

crt.TextAttr:=NORM;

end;

Procedure WriteSuccess(str:string);

begin

crt.TextAttr:=SEL;

writeln(str);

crt.TextAttr:=NORM;

end;

Function Func(x: real):real;

begin

Func:=x\*x\*x-x\*x+(-2)\*x+1;

end;

Function FFunc(x: real):real;

begin

FFunc:=1/4\*x\*x\*x\*x-1/3\*x\*x\*x-x\*x+x;

end;

//show information

Procedure Information;

begin

crt.ClrScr;

writeln('The task:');

writeln('1. Implement a program for calculating the area of a figure bounded by a curve');

writeln('x^3+(2)-x^2+(-2)\*x+(1) and an x-axis(in the positive y-axis).');

writeln('2. The calculation of the definite integral must be performed numerivally, using the');

writeln('right rectangle method.');

writeln('3. Integration limirs are entered by the user.');

writeln('4. Interaction with the user should be done through the case-menu.');

writeln('5. It is required to realize the possibility of evaluating the аccuracy of the obtained result.');

writeln('6. Procedures and functions should be used where appropriate.');

WriteSuccess('Enter <Enter> for continue');

repeat

ch:= crt.readkey;

until ch=#13;

end;

//enter limit of integration

Procedure Limit;

begin

crt.ClrScr;

write('Enter the start of integration: ');

Readln(a);

repeat

write('Enter the end of integration: ');

Readln(b);

if b <= a then

WriteError('The number must be greater than the beginning!');

until (b>a);

WriteSuccess('Done! Enter <Enter> for continue');

repeat

ch:= crt.readkey;

until ch=#13;

end;

//enter step of integration

Procedure Step;

begin

ClrScr;

Write('Enter number of steps: ');

repeat

Readln(steps);

if steps <= 0 then begin

WriteError('The number steps must be greater than 0! Try again.');

end;

until (steps > 0);

WriteSuccess('Done! Enter <Enter> for continue');

repeat

ch:= crt.readkey;

until ch=#13;

end;

Function Result:real;

var

r:real = 0;

h:real;

i:integer;

begin

crt.ClrScr;

if (a=0) and (b=0) then begin

WriteError('You have not entered the limits of integration! Enter <Enter> for continue');

end

else

begin

if steps <= 0 then begin

WriteError('You did not set the number of steps! Enter <Enter> for continue');

end

else begin

h:=(b-a)/steps;

r:=Func(b);

for i:=1 to steps-1 do

if ((a+i\*h)>t1) and ((a+i\*h)<t3) then r:=r+Func(a+i\*h)

else if (a+i\*h)>t2 then r:=r+Func(a+i\*h)

else r:=0;

r:=r\*h;

Exit(r);

end;

end;

Exit(-1);

repeat

ch:=crt.readkey;

until ch=#13;

end;

procedure ShowResult;

var r:real;

var h:real;

begin

r:=Result;

if r >= 0 then begin

h:=(b-a)/steps;

write('The area on the segment [',a:5:2,';',b:5:2,'] with a step ',h:5:2,' is equal to ');

writeln(r:5:2);

WriteSuccess('Enter <Enter> for continue.');

end;

repeat

ch:= crt.readkey;

until ch=#13;

end;

procedure AbsAcc;

var r,l,z,h:real;

begin

crt.ClrScr;

r:=Result;

h:=(b-a)/steps;

if r >= 0 then begin

if (b<t1) then l:=0;

if (b>t1) and (b<t3) then begin

if (a<=t1) then begin

a:=t2+h;

l:=FFunc(b)- FFunc(a);

end;

if (a>t1) then l:=FFunc(b)- FFunc(a);

end;

if (b>t3) and (b<t2) then begin

if (a<=t2) then begin

a:=t2+h;

l:=FFunc(b) - FFunc(a);

end;

if (a>t2) and (a<t3) then l:=FFunc(b)-FFunc(a);

if (a>t3) then begin

a:=t3+h;

l:=FFunc(b)-FFunc(a);

end;

end;

if (b>t1) then begin

if (a<t2) then begin

a:=t2+h;

l:=FFunc(b) - FFunc(a);

z:=FFunc(t2) -FFunc(t3);

l:=l-z;

end;

if (a>t2) and (a<t3) then begin

l:=FFunc(b)-FFunc(a);

z:=FFunc(t2)-FFunc(t3);

l:=l-z;

end;

if (a>t3) and (a<t1) then begin

a:=t3;

l:=FFunc(b)-FFunc(a);

z:=FFunc(t2)-FFunc(t3);

l:=l-z;

end;

end;

writeln(abs(l-r):10:5);

WriteSuccess('Enter <Enter> for continue.');

end;

repeat

ch:= crt.readkey;

until ch=#13;

end;

procedure RelAcc;

var r,l,h,z:real;

begin

crt.ClrScr;

r:=Result;

h:=(b-a)/steps;

if r >= 0 then begin

if (b<t1) then l:=0;

if (b>t1) and (b<t3) then begin

if (a<=t1) then begin

a:=t2+h;

l:=FFunc(b)- FFunc(a);

end;

if (a>t1) then l:=FFunc(b)- FFunc(a);

end;

if (b>t3) and (b<t2) then begin

if (a<=t2) then begin

a:=t2+h;

l:=FFunc(b) - FFunc(a);

end;

if (a>t2) and (a<t3) then l:=FFunc(b)-FFunc(a);

if (a>t3) then begin

a:=t3+h;

l:=FFunc(b)-FFunc(a);

end;

end;

if (b>t1) then begin

if (a<t2) then begin

a:=t2+h;

l:=FFunc(b) - FFunc(a);

z:=FFunc(t2) -FFunc(t3);

l:=l-z;

end;

if (a>t2) and (a<t3) then begin

l:=FFunc(b)-FFunc(a);

z:=FFunc(t2)-FFunc(t3);

l:=l-z;

end;

if (a>t3) and (a<t1) then begin

a:=t3;

l:=FFunc(b)-FFunc(a);

z:=FFunc(t2)-FFunc(t3);

l:=l-z;

end;

end;

writeln(abs(l-r)/abs(l):10:5);

WriteSuccess('Enter <Enter> for continue.');

end;

repeat

ch:= crt.readkey;

until ch=#13;

end;

Procedure MenuToScr;

var i:integer;

begin

crt.ClrScr;

for i:=1 to N do

begin

GoToXY(x,y+i-1);

write(menu[i]);

end;

writeln();

writeln();

writeln('[',a:5:2,';',b:5:2,'], number of steps: ',steps);

crt.TextAttr:=SEL;

GoToXY(x,y+punkt-1);

write(menu[punkt]);

crt.TextAttr:=NORM;

end;

Procedure ChangeMenu(d:boolean);

begin

crt.GoToXY(x,y+punkt-1);

write(menu[punkt]);

if d then punkt:=punkt+1

else punkt:=punkt-1;

crt.TextAttr:=SEl;

crt.GoToXY(x,y+punkt-1);

write(menu[punkt]);

crt.TextAttr:=NORM;

end;

//Graphic

procedure Axis(x0,y0,n:integer;a,b,ay,by,mx,my:real);

var

//n:integer;

num:real;

x,y,i:integer;

s:string;

dx,dy:real;

begin

line(0,y0,GetMaxX,y0);//Ox

line(x0,0,x0,GetMaxY);//Oy

SetColor(15);

SetTextStyle(1, 0, 2);

OutTextXY(xr - 20, y0 + 20 , 'X');

OutTextXY(x0 - 20, yt + 10, 'Y');

SetColor(15);

SetTextStyle(1, 0, 1);

dx:=(b-a)/n;

for i := 1 to n+1 do

begin

num := a + (i - 1) \* dx;

x := xl + trunc(mx \* (num - a));

Line(x, y0 - 3, x, y0 + 3);

str(num:0:1, s);

if abs(num) > 1e-10 then

OutTextXY(x - TextWidth(s) div 2, y0 + 10, s);

end;

dy:=(by-ay)/n;

for i := 1 to n+1 do

begin

num := ay + (i - 1) \* dy;

y := yt - trunc(my \* (num - by));

Line(x0 - 3, y, x0 + 3, y);

str(num:0:1, s);

if abs(num) > 1e-10 then

OutTextXY(x0 + 7, y - TextHeight(s) div 2, s);

end;

OutTextXY(x0 - 10, y0 + 10, '0');

end;

procedure Graph(x0,y0,yt,yd:integer;a,b:real;mx,my:real);

var

x1:real;

x:integer;

y:Int64;

y1:extended;

begin

x1 := a;

while x1 <= b do

begin

y1 := Func(x1);

x := x0 + round(x1 \* mx);

y := y0 - round(y1 \* my);

if (y >= yt-20) and (y <= yd+20) then PutPixel(x, y, 2);

x1 := x1 + 0.001;

end;

end;

procedure Hatch(a,b,mx,my:real;x0,y0:integer);

var

x1,x2:longint;

y1,y2:extended;

x,y:longint;

x3,y3,yd:extended;

t:real =0;

begin

SetColor(9);

{ x1:=round(x0 + round(a \* mx));

y1:=Func(x1);

y:=round(y0 - round(y1 \* my));

if y < 0 then y:=0;

line(x1,y0,x1,y);}

{x2:=round(x0 + round(b \* mx));

y2:=Func(x2);

y:=round(y0 - round(y2 \* my));

if y < 0 then y:=0;

line(x2,y0,x2,y); }

x3:=a;

y3:=0;

while x3 <= b do

begin

y1:=Func(x3);

x := x0 + round(x3 \* mx);

y3:=0+t;

{y:=1;}

while (y3<=y1){ and (y>=0)} do begin

y := y0 - round(y3\*my);

PutPixel(x, y, 14);

y3:=y3+0.3;

end;

t:=t+0.001;

x3 := x3 + 0.001;

end;

x3:=b;

y3:=t-0;

while x3 >= a do begin

x := x0 + round(x3 \* mx);

y1:=Func(x3);

yd:=y3;

while (yd>0) and not ((yd>y1)and(y1<0))

do begin

y := y0 - round(yd\*my);

PutPixel(x, y, 9);

yd:=yd-0.3;

end;

y3:=y3-0.001;

x3 := x3 - 0.001;

end;

x3:=b;

y3:=t-0;

while x3 >= a do begin

x := x0 + round(x3 \* mx);

y1:=Func(x3);

yd:=y3;

while

(yd>y1)and not ((yd>y1)and(y1<0))

do begin

y := y0 - round(yd\*my);

PutPixel(x, y, 0);

yd:=yd-0.3;

end;

y3:=y3-0.001;

x3 := x3 - 0.001;

end;

SetColor(15);

end;

procedure ScalePlusX(var a,b:real);

begin

if (a<sc+1e-6) and (b>sc+1e-6) then begin

a:=round(a+sc);

b:=round(b-sc);

end;

end;

procedure ScalePlusY(var ay,by:real);

begin

if (ay<sc+1e-6) and (by>sc+1e-6) then begin

ay:=round(ay+sc);

by:=round(by-sc);

end;

end;

procedure ScaleMinusX(var a,b:real);

begin

a:=round(a-sc);

b:=round(b+sc);

end;

procedure ScaleMinusY(var ay,by:real);

begin

ay:=round(ay-sc);

by:=round(by+sc);

end;

procedure ScalePlus(var a,b,ay,by:real);

begin

ScalePlusX(a,b);

ScalePlusY(ay,by);

end;

procedure ScaleMinus(var a,b,ay,by:real);

begin

ScaleMinusX(a,b);

ScaleMinusY(ay,by);

end;

procedure ShowInfo();

var st,s:string;

begin

OutTextXY(10,110,'Task: x^3-x^2+(-2)\*x+(1)');

OutTextXY(10,120,'<+> - The approach');

OutTextXY(10,130,'<-> - The distance');

OutTextXY(10,140,'<1> - The approach of the x-axis');

OutTextXY(10,150,'<2> - The distance of the x-axis');

OutTextXY(10,160,'<3> - The approach of the y-axis');

OutTextXY(10,170,'<4> - The distance of the y-axis');

OutTextXY(10,180,'<5> - Show graphic solution');

OutTextXY(10,190,'<6> - Hatching');

OutTextXY(10,200,'<Enter> - Exit');

Str(a:5:2,s);

st:='a = ' +s + '; ';

Str(b:5:2,s);

st:=st+'b = ' +s+'; ';

Str(steps,s);

st:=st+'steps = '+s+'; ';

OutTextXY(10,210, st);

end;

procedure Draw(n:integer;var a,b,ay,by: real;xr,xl,yd,yt:integer; var mx,my:real;var x0,y0:integer);

begin

mx := (xr - xl) / (b - a);

my := (yd - yt) / (by - ay);

x0 := trunc(abs(a) \* mx) + xl;

y0 := yd - trunc(abs(ay) \* my);

Axis(x0,y0,n,a,b,ay,by,mx,my);

Graph(x0,y0,yt,yd,a,b,mx,my);

ShowInfo();

end;

{procedure Task(a,b,n,mx,my:real;x0,y0:integer);

var

x1,x2:longint;

y1,y2:longint;

x,y:extended;

step:real;

begin

SetColor(13);

step:=(b-a)/n;

x := a;

y := Func(x);

x1 := x0 + round(x \* mx);

y1 := y0 - round(y \* my);

if y1 < 0 then y1:=0;

x:=x+step;

while x <= b do

begin

y := Func(x);

x2 := x0 + round(x \* mx);

y2 := y0 - round(y \* my);

if (y2 < 0) and (y1 <= 0)then y2:=0;

line(x1,y0,x1,y1);

if(y1 > 0) then line(x1,y1,x2,y2);

x1:=x2;

y1:=y2;

x := x + step;

end;

line(x1,y0,x1,y1);

SetColor(15);

end;}

procedure DoubleCheck(t,h:boolean;a,b,n,mx,my:real;x0,y0:integer);

begin

{ if t then Task(a,b,steps,mx,my,x0,y0);}

if h then Hatch(a,b,mx,my,x0,y0);

end;

procedure Visualization;

const

sc=1;

n=10;

var

Gd,Gm: integer;

ch: char;

mx: real = 0;

my: real = 0;

ax,bx:real;

ay,by:real;

x0:integer=0;

y0:integer=0;

iT:boolean = false;

iH: boolean = false;

begin

Gd:=detect;

InitGraph(Gd,Gm,'');

if GraphResult<>0 then Exit();

ax:=-10;

bx:=10;

by:=30;

ay:=-30;

xr:=GetMaxX-30;

yd:=GetMaxY-30;

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

repeat

ch:= wincrt.readkey;

case ch of

'+': begin

ClearDevice();

ScalePlus(ax,bx,ay,by);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'-': begin

ClearDevice();

ScaleMinus(ax,bx,ay,by);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'1': begin

ClearDevice();

ScalePlusX(ax,bx);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'2':begin

ClearDevice();

ScaleMinusX(ax,bx);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'3': begin

ClearDevice();

ScalePlusY(ay,by);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'4': begin

ClearDevice();

ScaleMinusY(ay,by);

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

'5': begin

ClearDevice();

Draw(n,ax,bx,ay,by,xr,xl,yd,yt,mx,my,x0,y0);

if not iH then

begin

iH:=true;

end

else iH:=false;

DoubleCheck(iT,iH,a,b,n,mx,my,x0,y0);

end;

end;

until ch=#13;

CloseGraph;

end;

procedure Graphic;

var r:real;

begin

r:=Result;

if r >= 0 then begin

Visualization;

exit;

end;

repeat

ch:= crt.readkey;

until ch=#13;

end;

begin

menu[1]:='Information on the program';

menu[2]:='Enter limit of integration';

menu[3]:='Enter the number of steps';

menu[4]:='Result';

menu[5]:='Absolute accuracy';

menu[6]:='Relative accuracy';

menu[7]:='Visualization';

menu[8]:='Exit';

punkt:=1;

x:=1;

y:=1;

crt.TextAttr:=NORM;

MenuToScr;

repeat

ch:=crt.ReadKey;

if ch=#0 then begin

ch:=crt.ReadKey;

case ch of

#80:

if punkt<N then

begin

ChangeMenu(true);

end;

#72:

if punkt>1 then

begin

ChangeMenu(false);

end;

end;

end

else if ch=#13 then begin

case punkt of

1:Information;

2:Limit;

3:Step;

4:ShowResult;

5:AbsAcc;

6:RelAcc;

7:Graphic;

8:ch:=#27;

end;

MenuToScr;

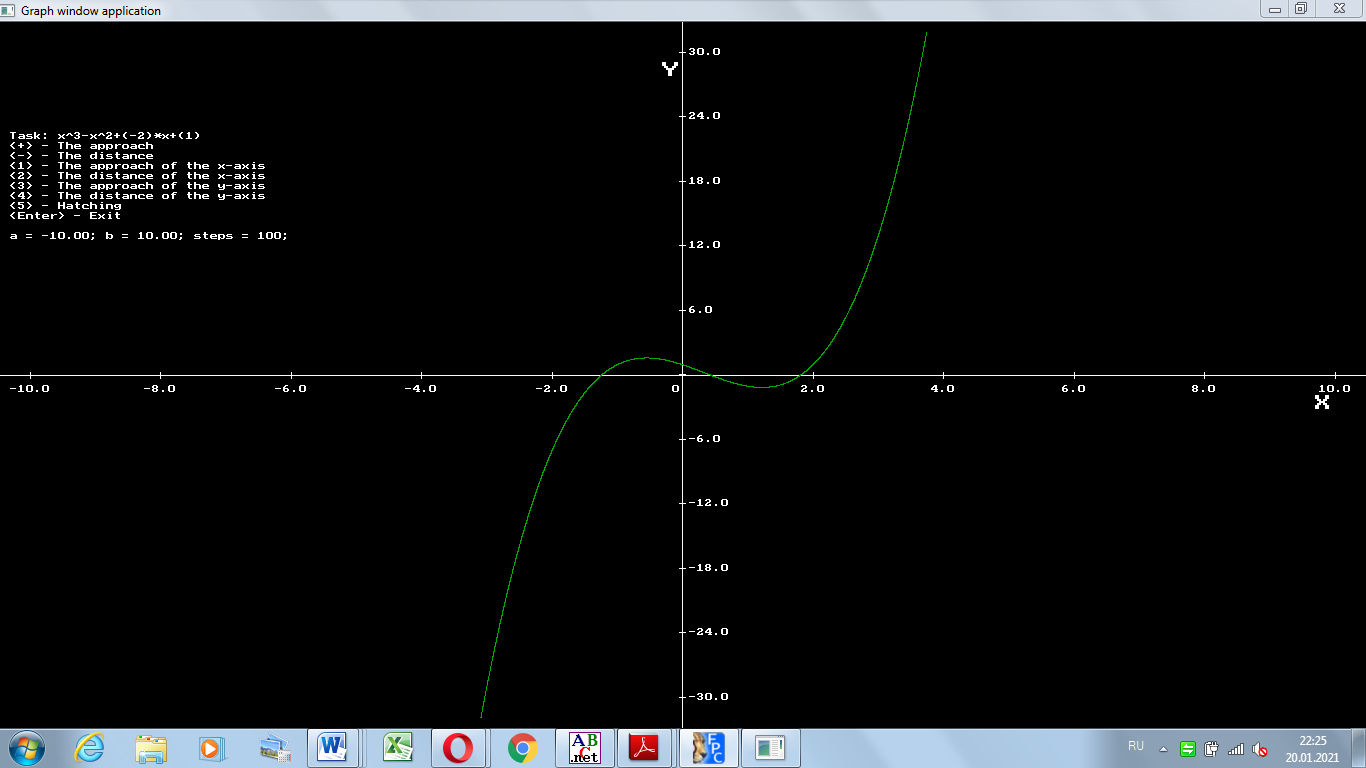
end;

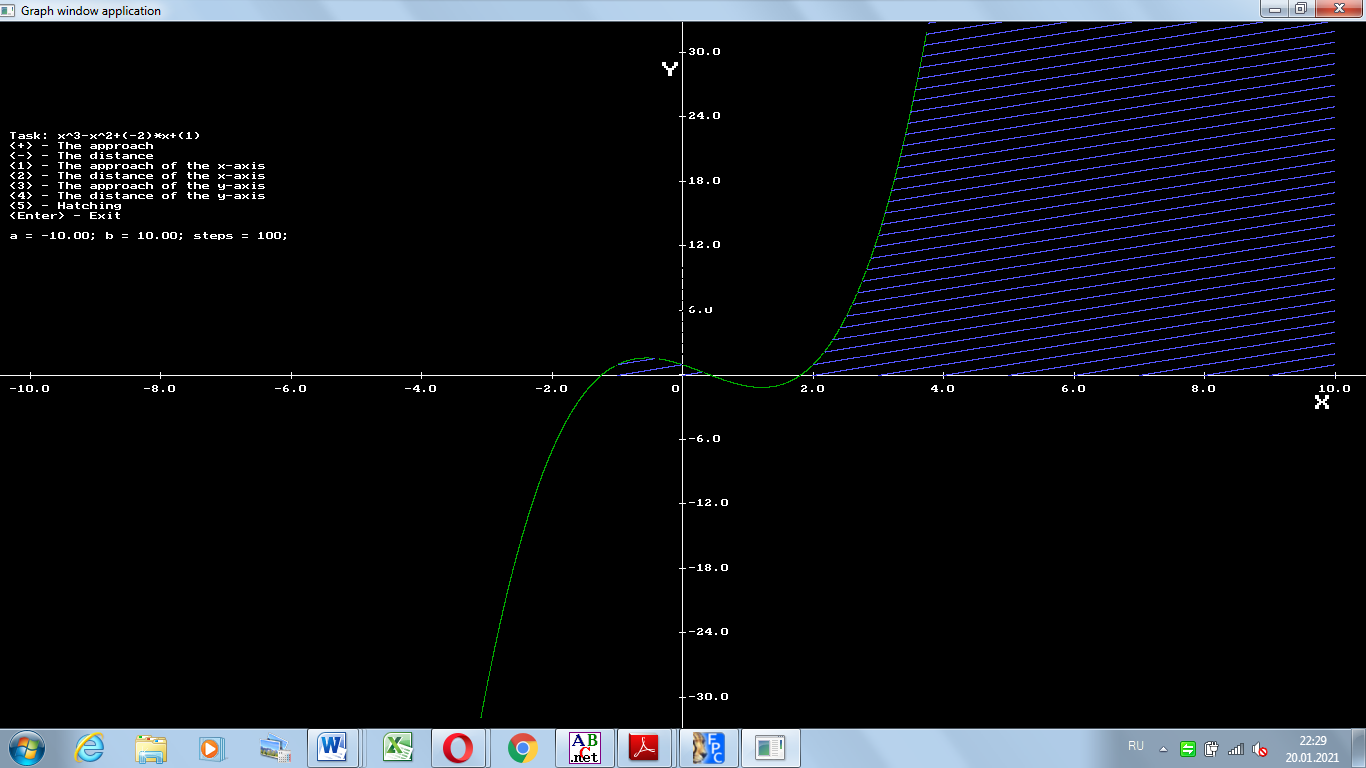
until ch=#27;

crt.ClrScr;

end.

Экранная форма





**Вывод:**  В ходе данной лабораторной работы были освоены принципы работы в графическом режиме; получены базовые навыки взаимодействия с графическими примитивами. Также были изучены модули Graph, который используется для отображения графики в Free Pascal, и wincrt для обработки клавиш в графическом режиме. Научились выводить оси и график функции в графическом режиме, масштабировать этот график