МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Основы алгоритмизации и программирования»

Лабораторная работа №4

Рекурсия

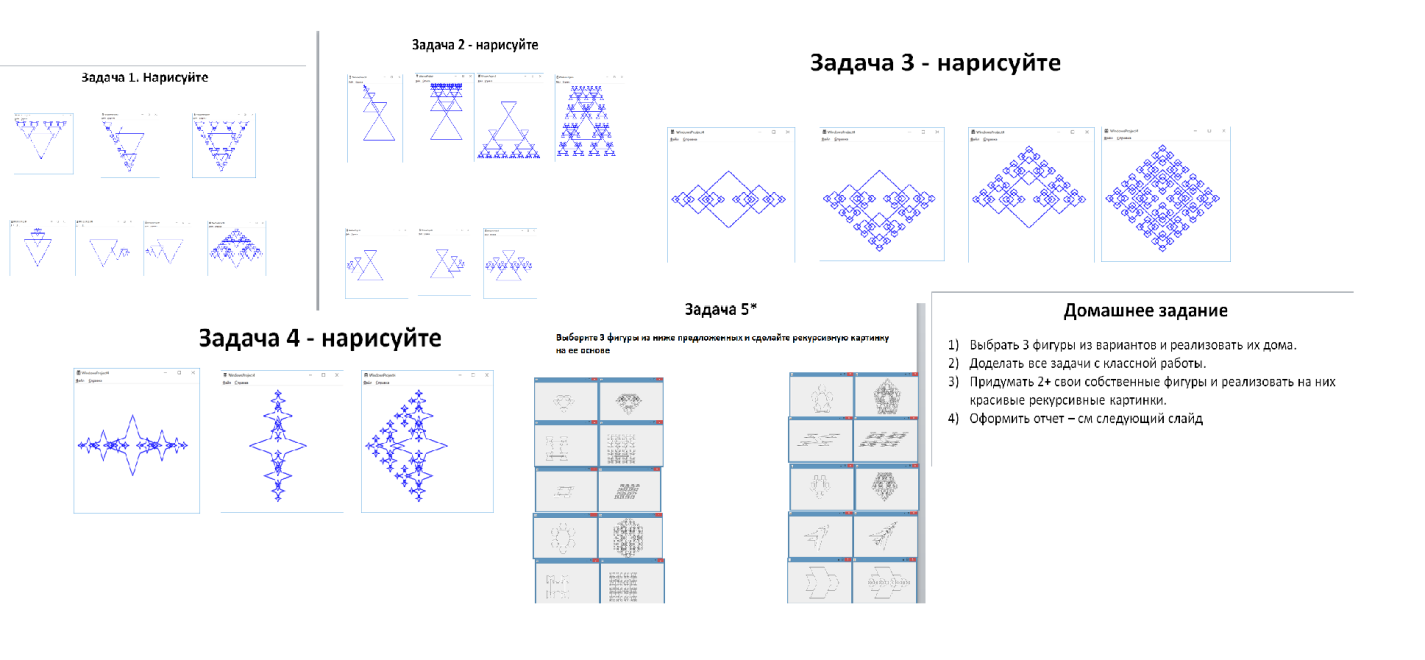
Выполнил:

студент гр. ПИбд-12

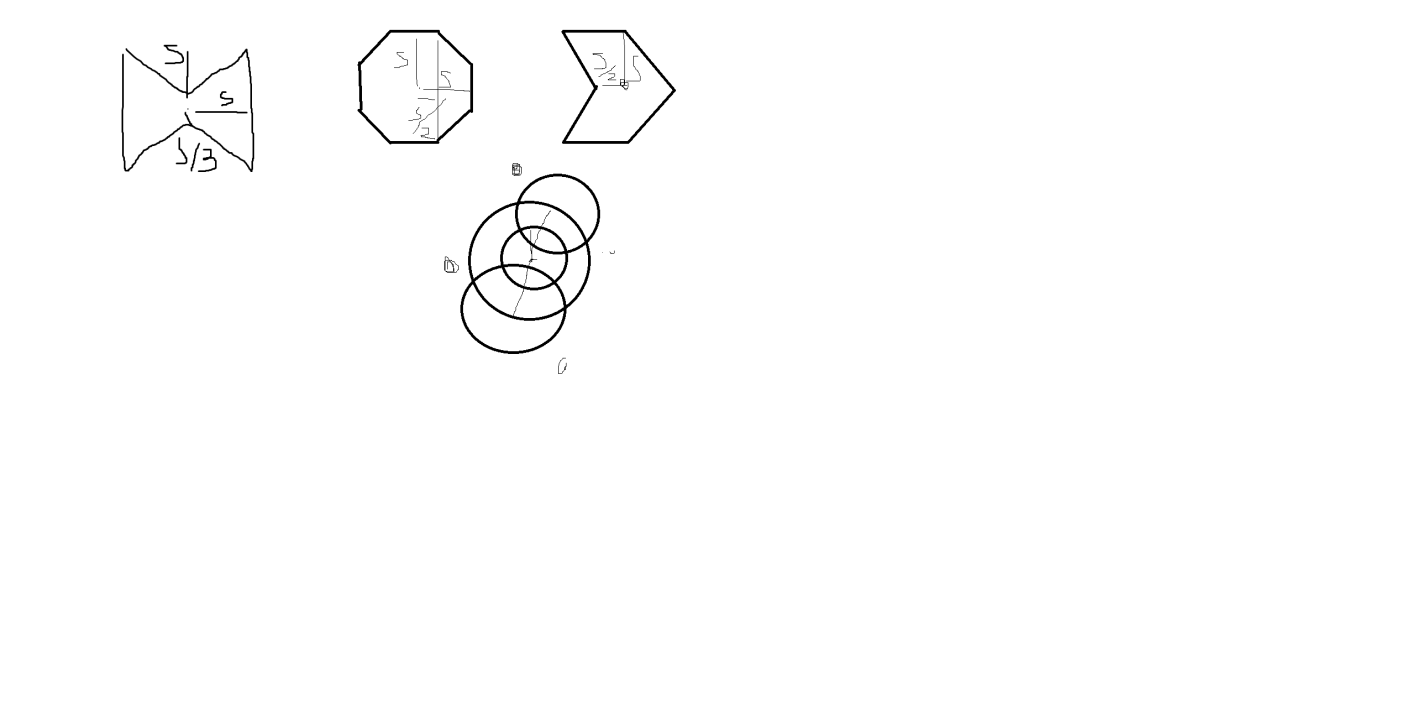
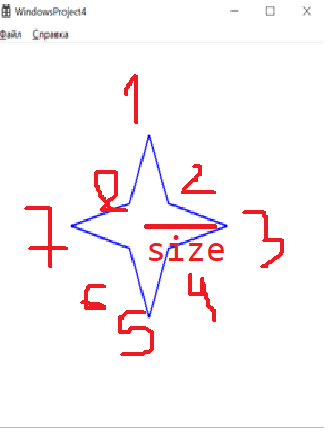
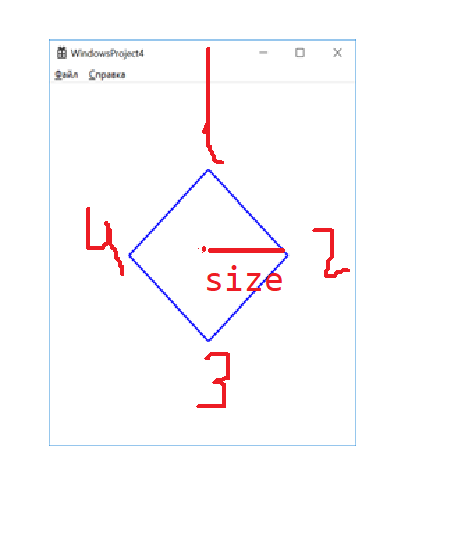
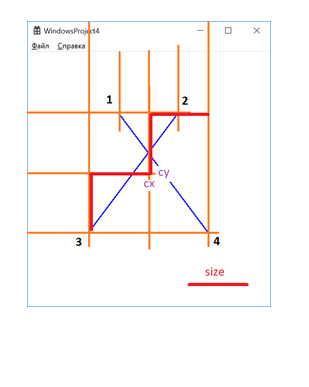
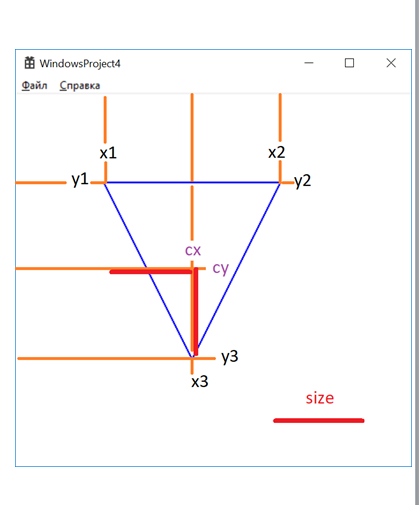
Дозорова Алена

Ульяновск, 2021 г

Задание лабораторной работы



Расчеты картинок



Код прграммы

void Image1(HDC hdc, int cx, int cy, int size)

{

int x1 = cx - size;

int x2 = cx + size;

int x3 = cx;

int y1 = cy - size;

int y2 = cy - size;

int y3 = cy+size;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage1\_1(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_1(hdc, cx - size, cy - size, size / 2);

RecursiveImage1\_1(hdc, cx + size, cy - size, size / 2);

}

void RecursiveImage1\_2(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_2(hdc, cx - size, cy - size, size / 2);

RecursiveImage1\_2(hdc, cx, cy + size, size / 2);

}

void RecursiveImage1\_3(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_3(hdc, cx - size, cy - size, size / 2);

RecursiveImage1\_3(hdc, cx, cy + size, size / 2);

RecursiveImage1\_3(hdc, cx + size, cy - size, size / 2);

}

void RecursiveImage1\_4(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_4(hdc, cx, cy - size, size / 2);

}

void RecursiveImage1\_5(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_5(hdc, cx + size, cy, size / 2);

}

void RecursiveImage1\_6(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_6(hdc, cx - size, cy, size / 2);

}

void RecursiveImage1\_7(HDC hdc, int cx, int cy, int size) {

Image1(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage1\_7(hdc, cx, cy-size, size / 2);

RecursiveImage1\_7(hdc, cx+size, cy, size / 2);

RecursiveImage1\_7(hdc, cx-size, cy, size / 2);

}

void Image2(HDC hdc, int cx, int cy, int size)

{

int x1 = cx - size/2;

int x2 = cx + size/2;

int x3 = cx - size;

int x4 = cx + size;

int y1 = cy - size;

int y2 = cy - size;

int y3 = cy + size;

int y4 = cy + size;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage2\_1(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_1(hdc, cx-size, cy + size, size / 2);

}

void RecursiveImage2\_2(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_2(hdc, cx - size/2, cy - size, size / 2);

}

void RecursiveImage2\_3(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_3(hdc, cx - size/2, cy-size, size / 2);

RecursiveImage2\_3(hdc, cx + size/2, cy-size , size / 2);

}

void RecursiveImage2\_4(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_4(hdc, cx - size, cy + size, size / 2);

RecursiveImage2\_4(hdc, cx + size, cy + size, size / 2);

}

void RecursiveImage2\_5(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage2\_5(hdc, cx - size, cy-size, size / 2);

RecursiveImage2\_5(hdc, cx + size, cy - size, size / 2);

RecursiveImage2\_5(hdc, cx - size, cy + size, size / 2);

RecursiveImage2\_5(hdc, cx + size, cy + size, size / 2);

}

void RecursiveImage2\_6(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_6(hdc, cx - size, cy, size / 2);

}

void RecursiveImage2\_7(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_7(hdc, cx + size, cy, size / 2);

}

void RecursiveImage2\_8(HDC hdc, int cx, int cy, int size) {

Image2(hdc, cx, cy, size);

if (size < 5) {

return;

}

RecursiveImage2\_8(hdc, cx - size, cy, size / 2);

RecursiveImage2\_8(hdc, cx + size, cy, size / 2);

}

void Image3(HDC hdc, int cx, int cy, int size)

{

int x1 = cx;

int x2 = cx + size;

int x3 = cx;

int x4 = cx - size;

int y1 = cy - size;

int y2 = cy;

int y3 = cy + size;

int y4 = cy;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage3\_1(HDC hdc, int cx, int cy, int size) {

Image3(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage3\_1(hdc, cx + size, cy, size / 2);

}

void RecursiveImage3\_2(HDC hdc, int cx, int cy, int size) {

Image3(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage3\_2(hdc, cx + size, cy, size / 2);

RecursiveImage3\_2(hdc, cx - size, cy, size / 2);

}

void RecursiveImage3\_3(HDC hdc, int cx, int cy, int size) {

Image3(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage3\_3(hdc, cx + size, cy, size / 2);

RecursiveImage3\_3(hdc, cx - size, cy, size / 2);

RecursiveImage3\_3(hdc, cx, cy+size, size / 2);

}

void RecursiveImage3\_4(HDC hdc, int cx, int cy, int size) {

Image3(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage3\_4(hdc, cx + size, cy, size / 2);

RecursiveImage3\_4(hdc, cx - size, cy, size / 2);

RecursiveImage3\_4(hdc, cx, cy - size, size / 2);

}

void RecursiveImage3\_5(HDC hdc, int cx, int cy, int size) {

Image3(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage3\_5(hdc, cx + size, cy, size / 2);

RecursiveImage3\_5(hdc, cx - size, cy, size / 2);

RecursiveImage3\_5(hdc, cx, cy - size, size / 2);

RecursiveImage3\_5(hdc, cx, cy + size, size / 2);

}

void Image4(HDC hdc, int cx, int cy, int size)

{

int x1 = cx;

int x2 = cx + size/4;

int x3 = cx+size;

int x4 = cx + size/4;

int x5 = cx;

int x6 = cx - size / 4;

int x7 = cx - size;

int x8 = cx - size/4;

int y1 = cy - size;

int y2 = cy-size/4;

int y3 = cy;

int y4 = cy+size/4;

int y5 = cy + size;

int y6 = cy + size / 4;

int y7 = cy;

int y8= cy - size / 4;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x5, y5);

LineTo(hdc, x6, y6);

LineTo(hdc, x7, y7);

LineTo(hdc, x8, y8);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage4\_1(HDC hdc, int cx, int cy, int size) {

Image4(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage4\_1(hdc, cx + size, cy, size / 2);

}

void RecursiveImage4\_2(HDC hdc, int cx, int cy, int size) {

Image4(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage4\_2(hdc, cx - size, cy, size / 2);

}

void RecursiveImage4\_3(HDC hdc, int cx, int cy, int size) {

Image4(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage4\_3(hdc, cx + size, cy, size / 2);

RecursiveImage4\_3(hdc, cx - size, cy, size / 2);

}

void RecursiveImage4\_4(HDC hdc, int cx, int cy, int size) {

Image4(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage4\_4(hdc, cx - size, cy, size / 2);

RecursiveImage4\_4(hdc, cx, cy-size, size / 2);

RecursiveImage4\_4(hdc, cx, cy+size, size / 2);

}

void Image5(HDC hdc, int cx, int cy, int size)

{

int x1 = cx;

int x2 = cx + size;

int x3 = cx + size;

int x4 = cx;

int x5 = cx-size;

int x6 = cx-size;

int y1 = cy - size/2;

int y2 = cy - size;

int y3 = cy+size;

int y4 = cy + size / 2;

int y5 = cy + size;

int y6 = cy - size;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x5, y5);

LineTo(hdc, x6, y6);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage5\_1(HDC hdc, int cx, int cy, int size) {

Image5(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage5\_1(hdc, cx - size, cy-size, size / 2);

RecursiveImage5\_1(hdc, cx+size, cy - size, size / 2);

RecursiveImage5\_1(hdc, cx-size, cy + size, size / 2);

RecursiveImage5\_1(hdc, cx + size, cy + size, size / 2);

}

void Image6(HDC hdc, int cx, int cy, int size) {

int x1 = cx+size/2;

int x2 = cx + size;

int x3 = cx + size;

int x4 = cx + size / 2;

int x5 = cx - size/2;

int x6 = cx - size;

int x7 = cx - size;

int x8 = cx - size / 2;

int y1 = cy-size;

int y2 = cy -size/2;

int y3 = cy + size/2;

int y4 = cy+size;

int y5 = cy+size;

int y6 = cy + size/2;

int y7 = cy - size / 2;

int y8 = cy - size;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x5, y5);

LineTo(hdc, x6, y6);

LineTo(hdc, x7, y7);

LineTo(hdc, x8, y8);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage6\_1(HDC hdc, int cx, int cy, int size) {

Image6(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage6\_1(hdc, cx, cy - 2\*size, size / 2);

RecursiveImage6\_1(hdc, cx+size, cy - size, size / 2);

RecursiveImage6\_1(hdc, cx+2\*size, cy, size / 2);

RecursiveImage6\_1(hdc, cx+size, cy +size, size / 2);

RecursiveImage6\_1(hdc, cx, cy + 2 \* size, size / 2);

RecursiveImage6\_1(hdc, cx-size, cy +size, size / 2);

RecursiveImage6\_1(hdc, cx-2\*size, cy, size / 2);

RecursiveImage6\_1(hdc, cx-size, cy-size, size / 2);

}

void Image7(HDC hdc, int cx, int cy, int size)

{

int x1 = cx-size;

int x2 = cx;

int x3 = cx + size;

int x4 = cx;

int x5 = cx - size;

int x6 = cx - size/2;

int y1 = cy - size;

int y2 = cy - size;

int y3 = cy;

int y4 = cy + size;

int y5 = cy + size;

int y6 = cy;

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 2, RGB(0, 0, 0));

SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x5, y5);

LineTo(hdc, x6, y6);

LineTo(hdc, x1, y1);

DeleteObject(hPen);

}

void RecursiveImage7\_1(HDC hdc, int cx, int cy, int size) {

Image7(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage7\_1(hdc, cx + 3\*size/2, cy, size / 2);

RecursiveImage7\_1(hdc, cx - size, cy, size / 2);

}

void Image8(HDC hdc, int cx, int cy, int size)

{

Ellipse(hdc, cx - size, cy - size, cx + size, cy + size);

Ellipse(hdc, cx - size / 4, cy - 3\*size / 2, cx + 5\*size / 4, cy);

Ellipse(hdc, cx - size / 2, cy - size / 2, cx + size / 2, cy + size / 2);

}

void RecursiveImage8\_1(HDC hdc, int cx, int cy, int size) {

Image8(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage8\_1(hdc, cx + size, cy, size / 2);

RecursiveImage8\_1(hdc, cx - size, cy, size / 2);

RecursiveImage8\_1(hdc, cx , cy + size, size / 2);

RecursiveImage8\_1(hdc, cx , cy-size, size / 2);

}

void Image9(HDC hdc, int cx, int cy, int size)

{

Image3(hdc, cx, cy, size / 3);

Image3(hdc, cx-size, cy, size / 3);

Image3(hdc, cx+size, cy, size / 3);

}

void RecursiveImage9\_1(HDC hdc, int cx, int cy, int size) {

Image9(hdc, cx, cy, size);

if (size < 20) {

return;

}

RecursiveImage9\_1(hdc, cx + size, cy, size / 2);

RecursiveImage9\_1(hdc, cx - size, cy, size / 2);

RecursiveImage9\_1(hdc, cx, cy + size, size / 2);

RecursiveImage9\_1(hdc, cx, cy - size, size / 2);

}

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

RecursiveImage1\_1(hdc, 200, 160, 80);

RecursiveImage1\_2(hdc, 550, 150, 80);

RecursiveImage1\_3(hdc, 800, 150, 80);

RecursiveImage1\_4(hdc, 80, 400, 80);

RecursiveImage1\_5(hdc, 260, 400, 80);

RecursiveImage1\_6(hdc, 550, 400, 80);

RecursiveImage1\_7(hdc, 800, 500, 80);

RecursiveImage2\_1(hdc, 200, 100, 80);

RecursiveImage2\_2(hdc, 400, 150, 60);

RecursiveImage2\_3(hdc, 600, 150, 60);

RecursiveImage2\_4(hdc, 800, 100, 80);

RecursiveImage2\_5(hdc, 200, 400, 80);

RecursiveImage2\_6(hdc, 600, 400, 80);

RecursiveImage2\_7(hdc, 800, 400, 80);

RecursiveImage2\_8(hdc, 1000, 400, 80);

RecursiveImage3\_1(hdc, 100, 100, 60);

RecursiveImage3\_2(hdc, 500, 200, 60);

RecursiveImage3\_3(hdc, 800, 200, 60);

RecursiveImage3\_4(hdc, 100, 400, 60);

RecursiveImage3\_5(hdc, 500, 400, 60);

RecursiveImage4\_1(hdc, 100, 100, 80);

RecursiveImage4\_2(hdc, 450, 100, 80);

RecursiveImage4\_3(hdc, 800, 100, 80);

RecursiveImage4\_4(hdc, 150, 300, 80);

RecursiveImage5\_1(hdc, 200, 200, 80);

RecursiveImage5\_1(hdc, 500, 100, 20);

RecursiveImage6\_1(hdc, 200, 200, 80);

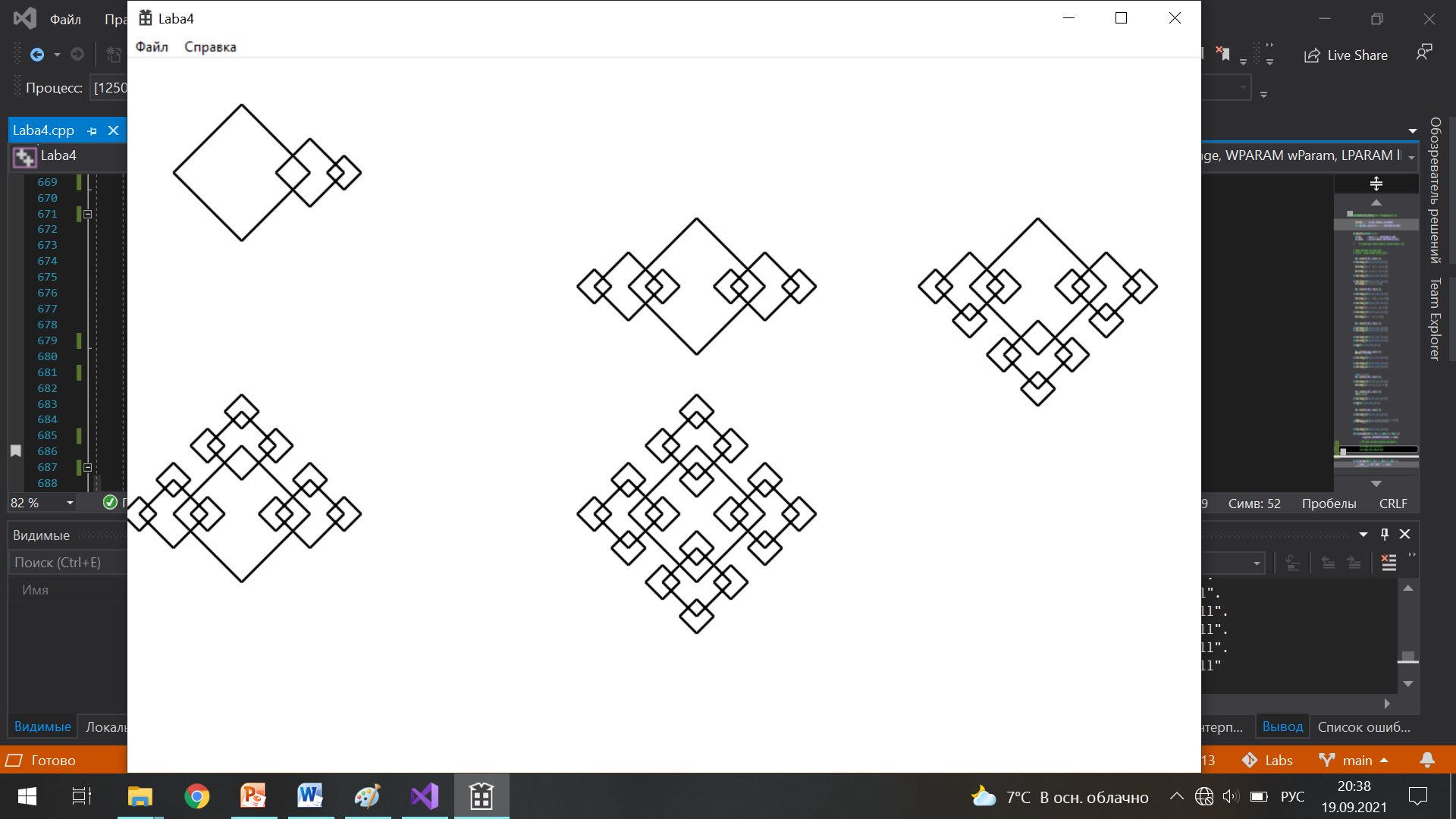
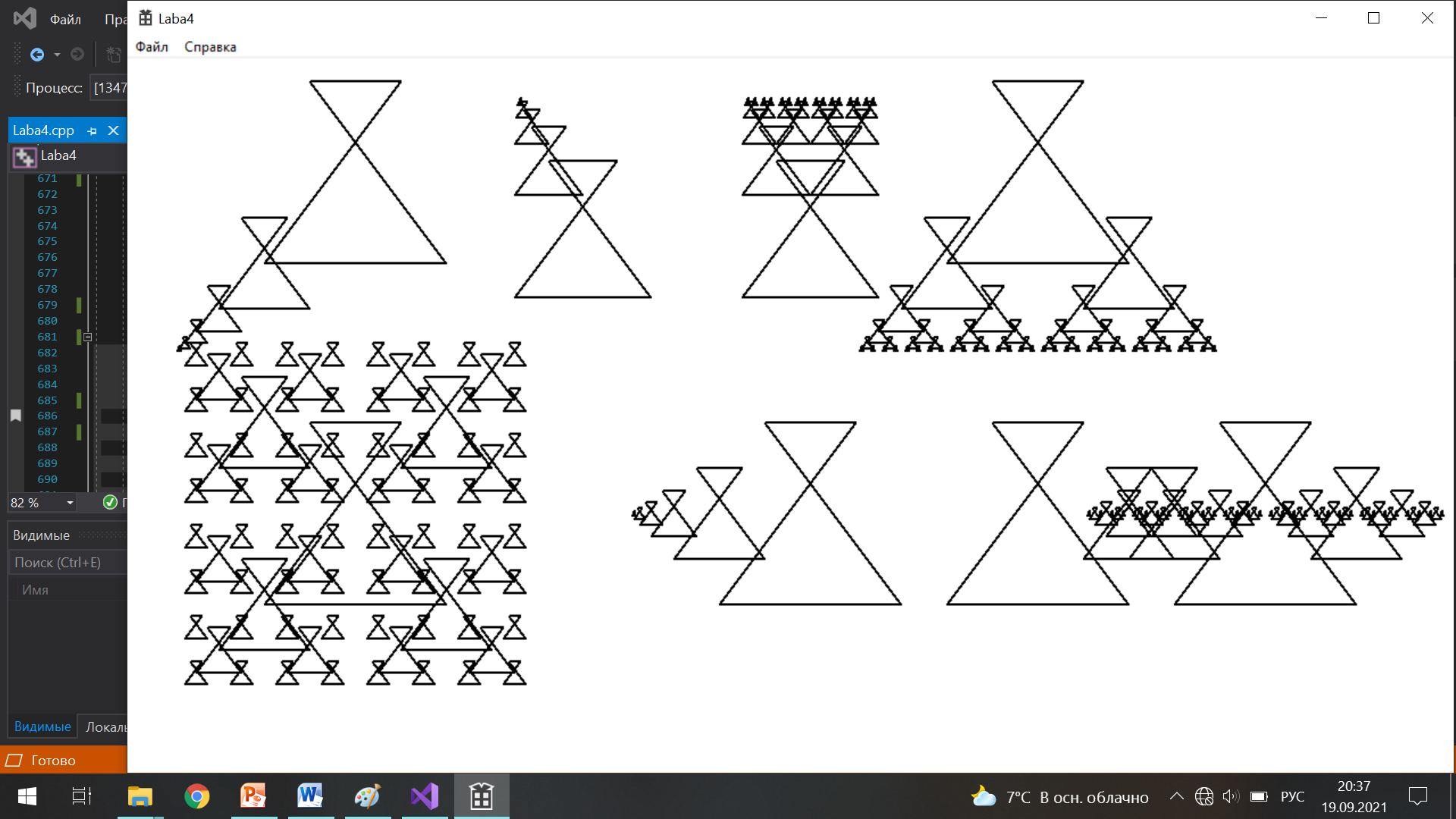
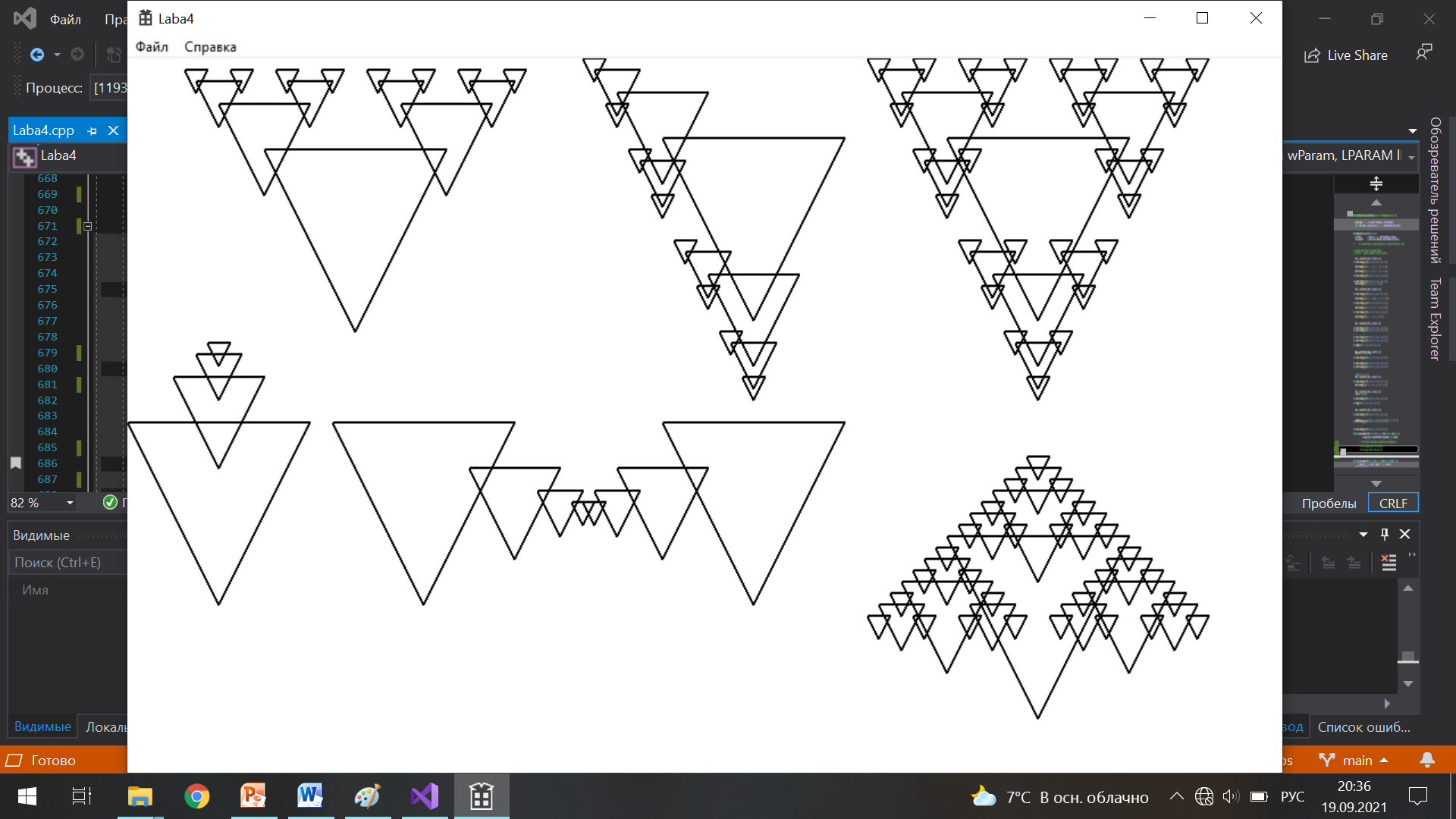
RecursiveImage7\_1(hdc, 200, 200, 80);

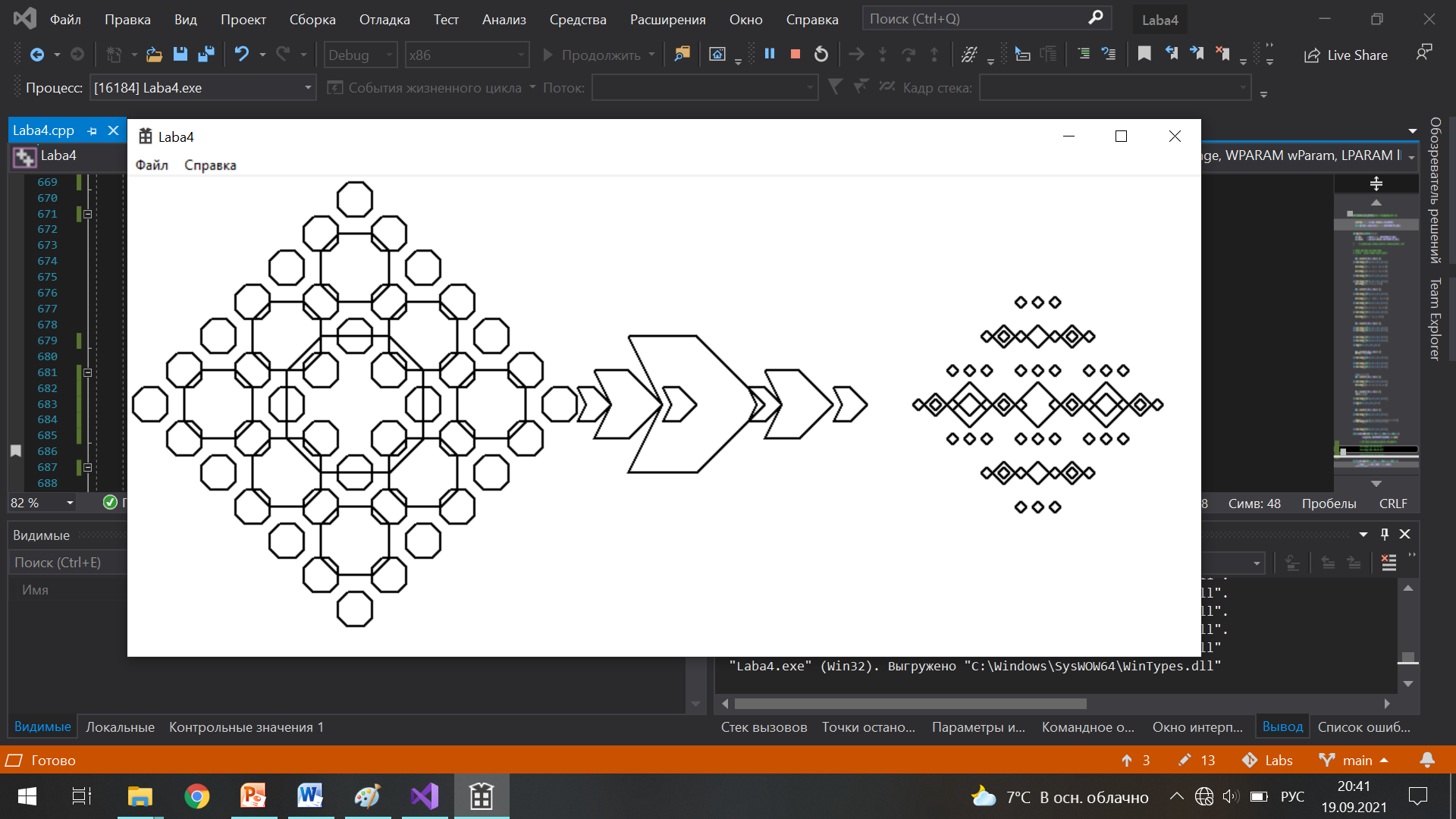
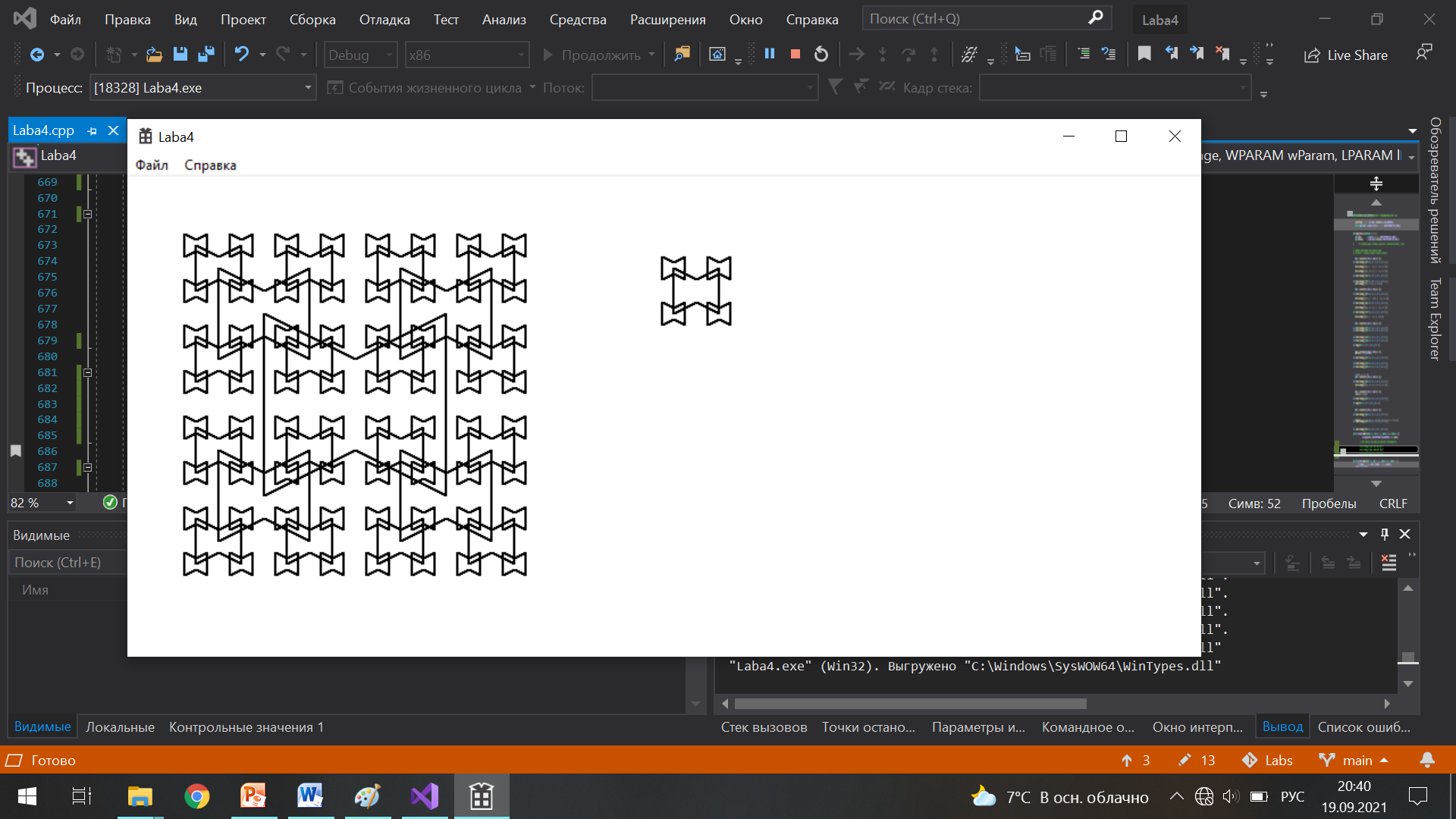
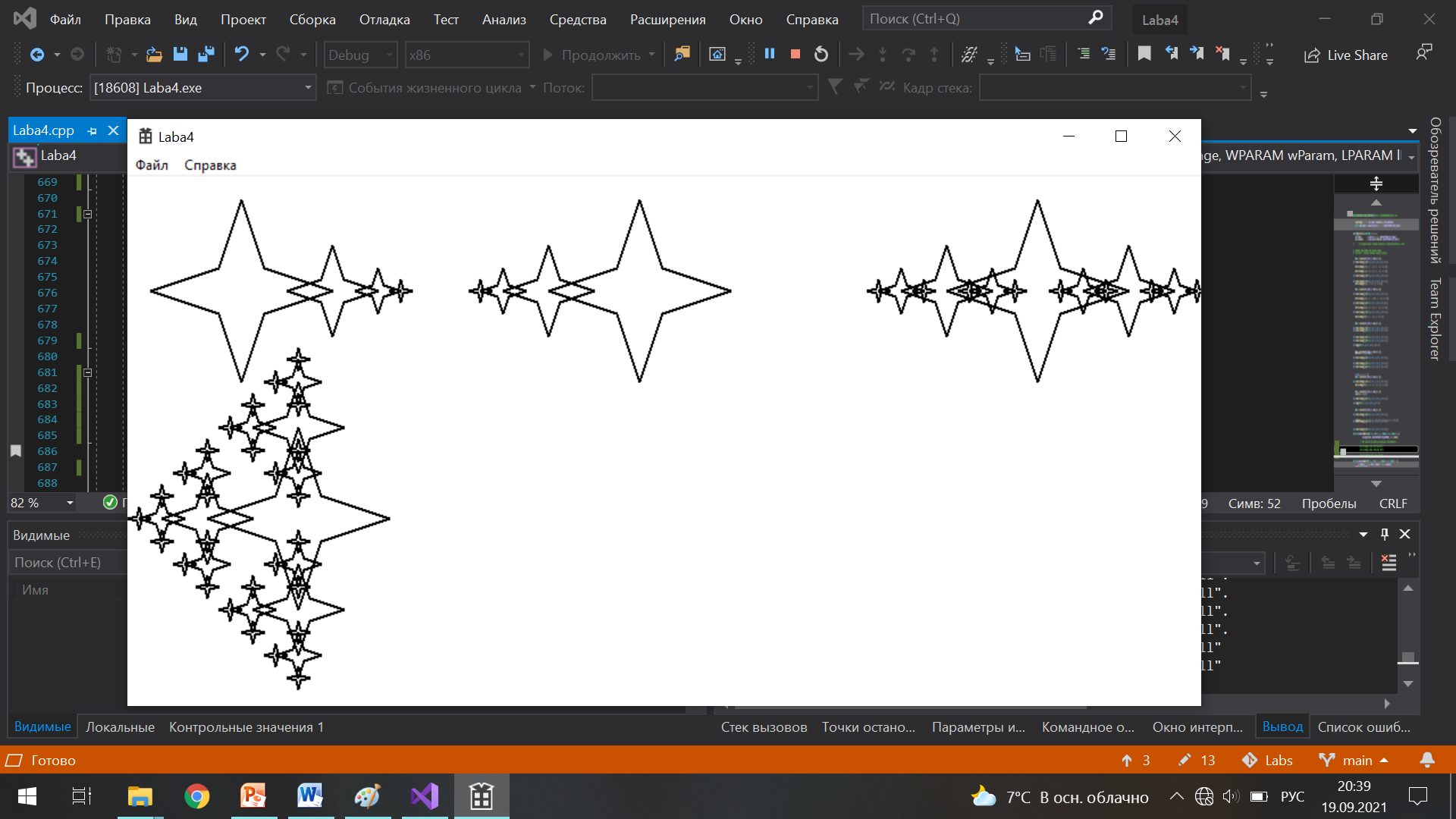
RecursiveImage9\_1(hdc, 200, 200, 80);

EndPaint(hWnd, &ps);

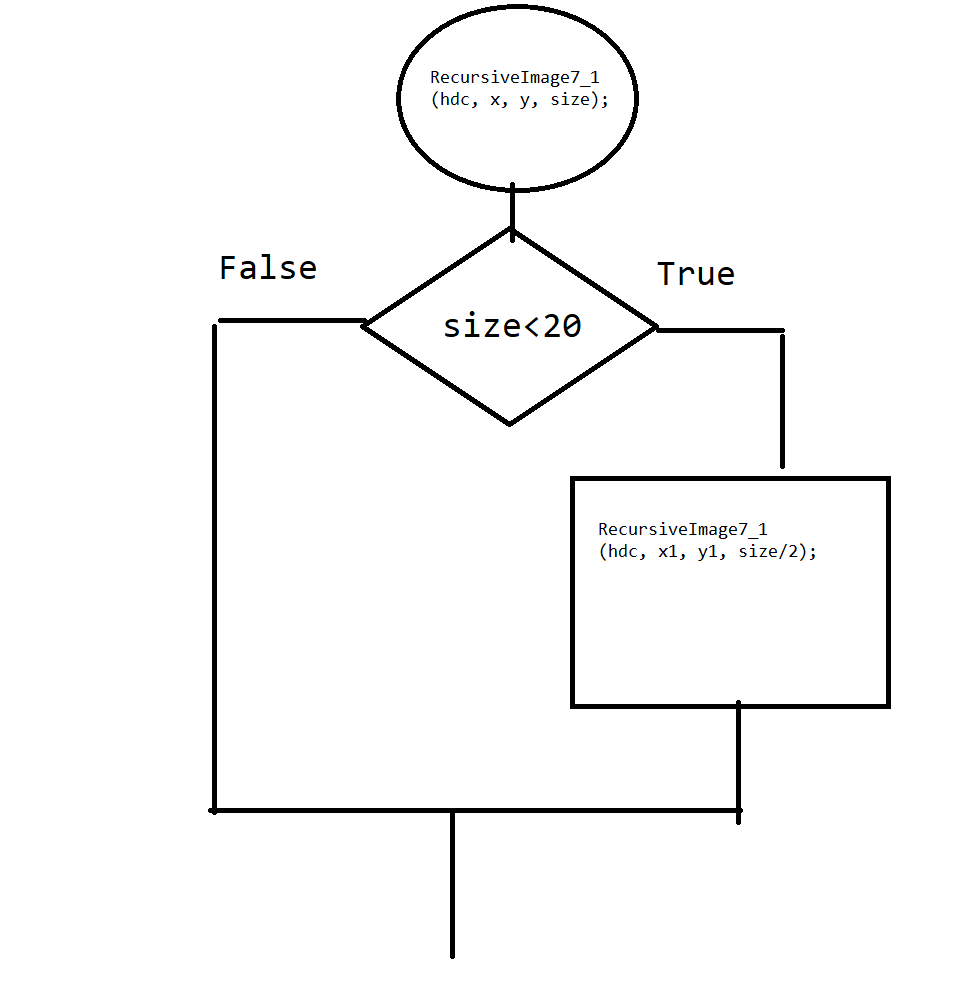
}

Скриншот программы

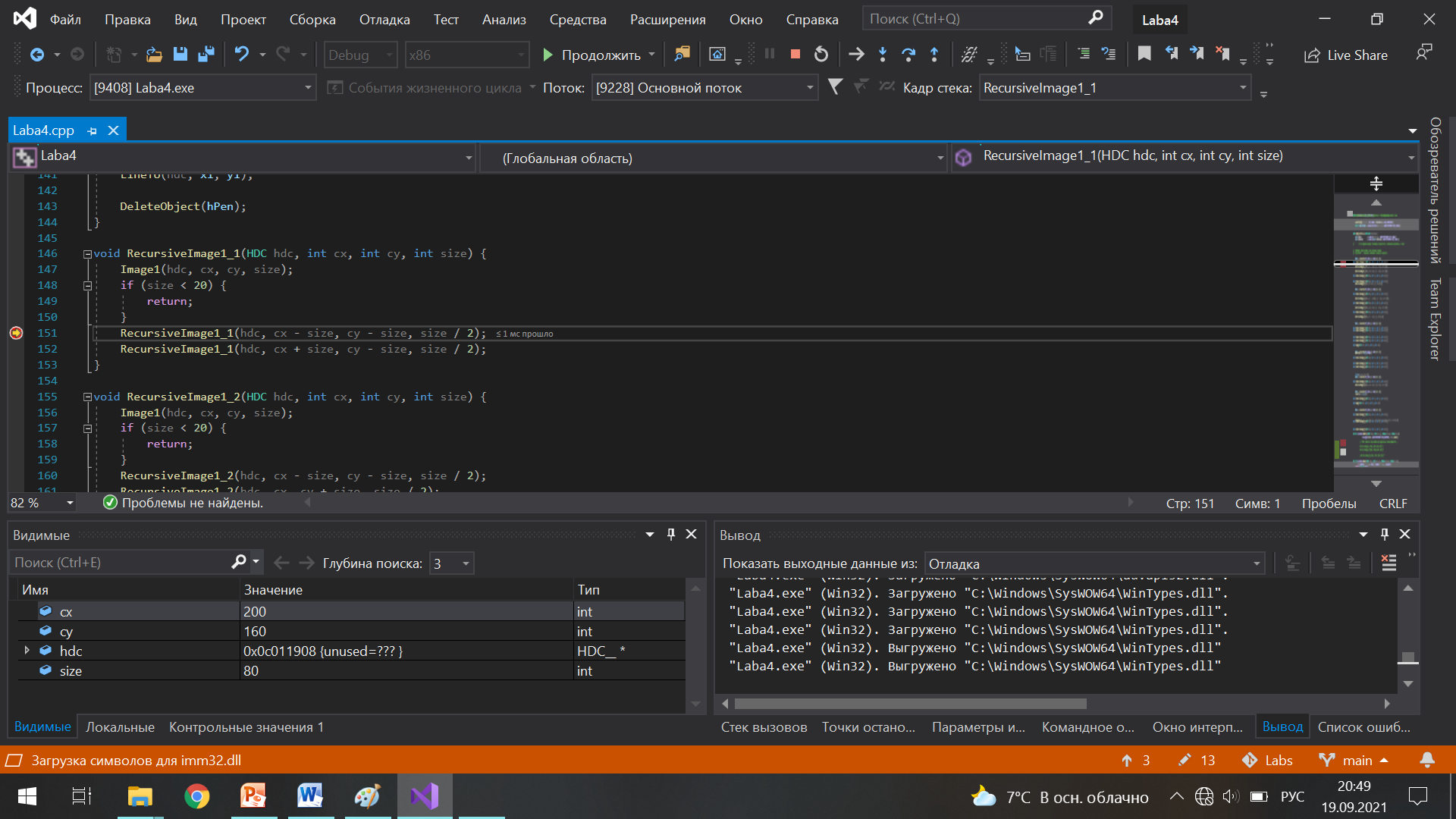
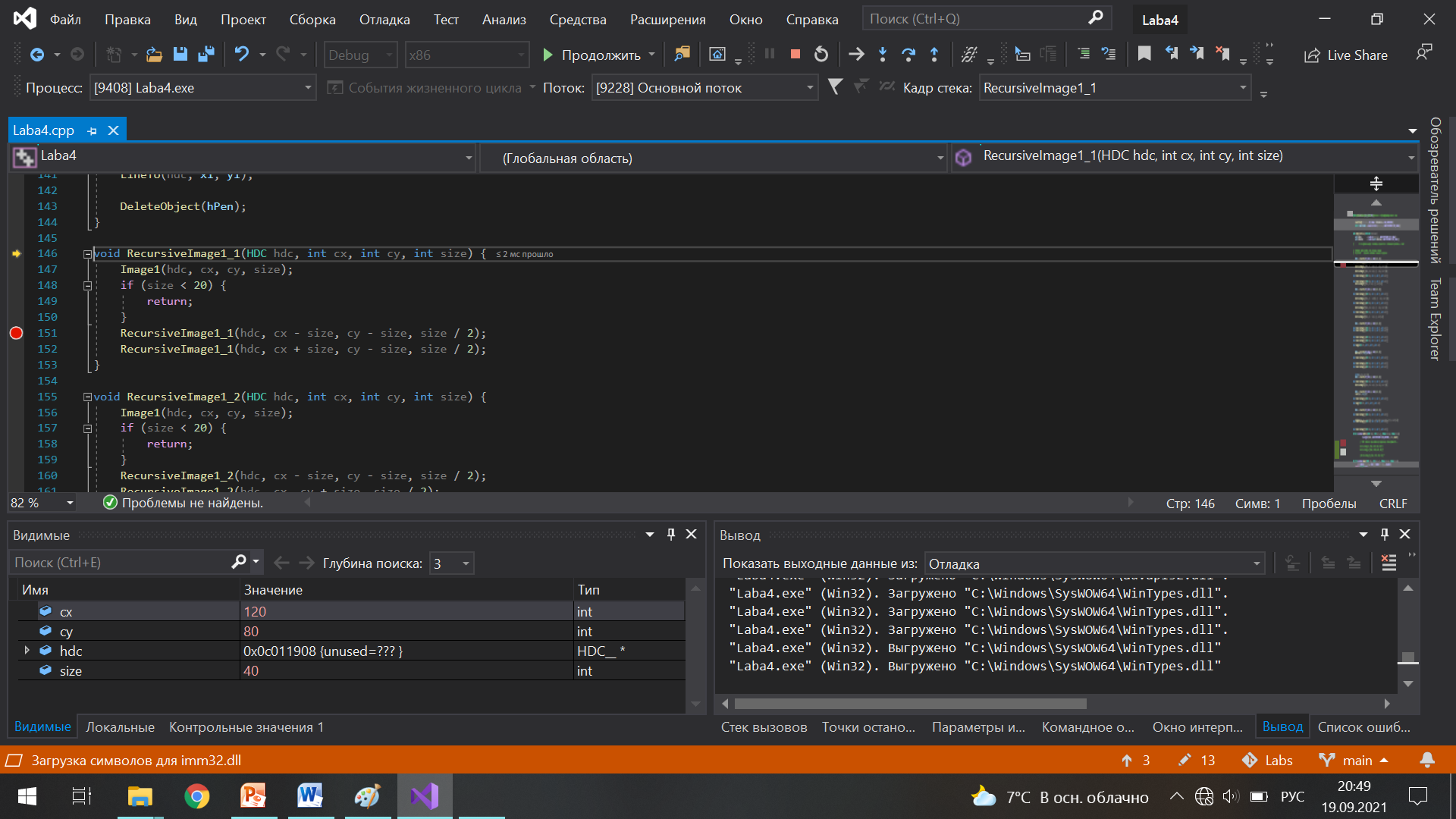
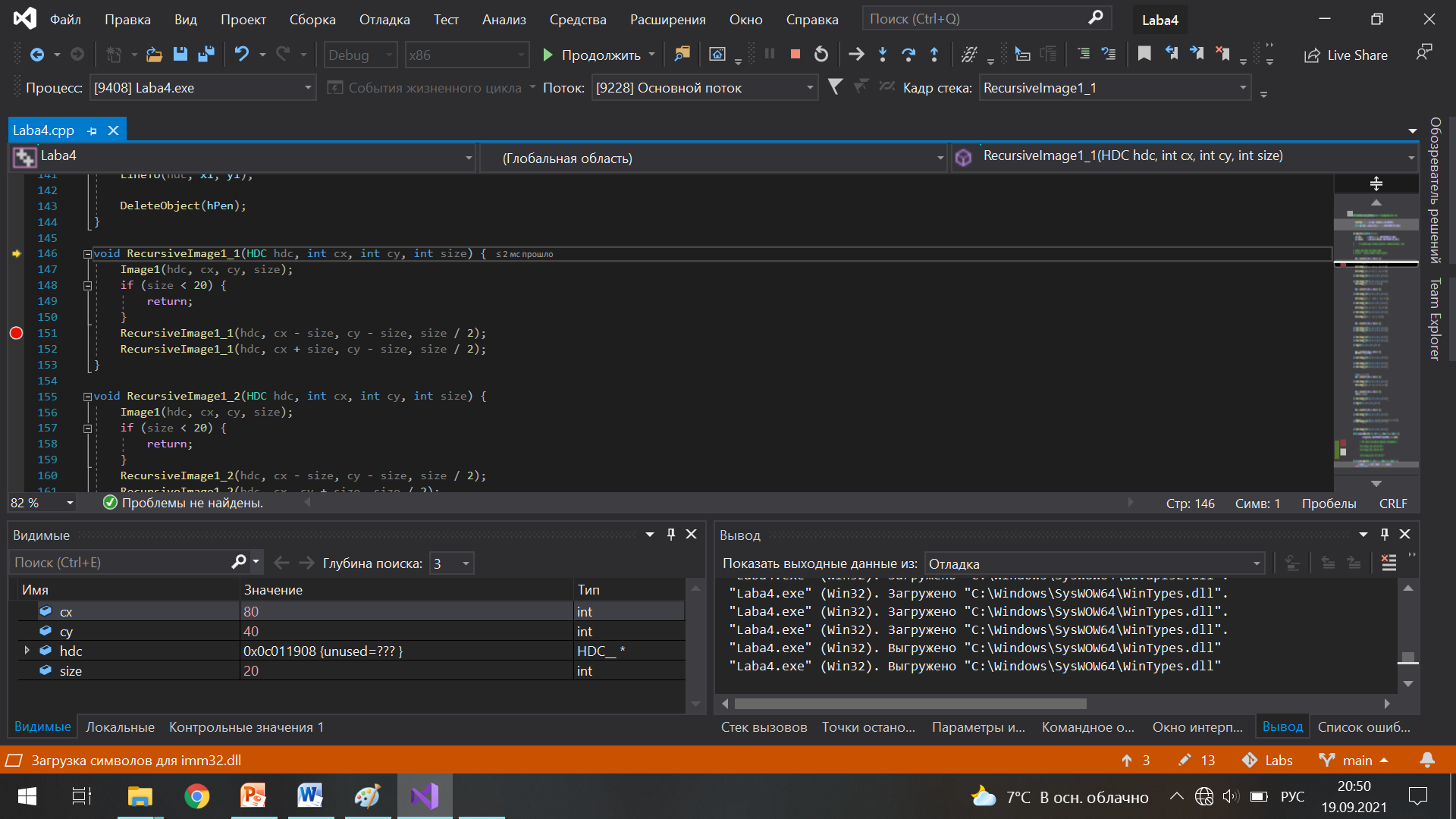




Блоксхема



Ручная трассировка на глубину три первого рисунка

Заключение

Я научилась использовать рекурсию для создания сложных и красивых рисунков из простых элементов.