МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра информационных систем

Отчет по дисциплине Основы алгоритмизации и программирование

Домашняя работа №1

Выполнил студент гр. ИСЭбд-11

Гапон Николай Иванович

Преподаватель

Власенко Олег Федосович

ССЫЛКА НА GIT <https://github.com/NikGapon/1year/tree/main/lab6>

Домашнее задание

1) Доделать все задачи 1-4, которые не были сделаны в классе

2) Собрать в единой программе (Лабораторная работа №6) ВСЕ рисунки, выполненные на данный момент в классе и дома.

3) Переключение между рисунками обеспечить при помощи подходящих клавиш. Конкретные клавиши выбрать самостоятельно!

4) Нарисовать блок-схемы для всех Switch, найденных в коде программы.

void Cor(HDC hdc, int cx, int cy, int sizeX, int sizeY, COLORREF color) {

POINT p[] = {

cx, cy - sizeY,

cx + sizeX / 2, cy,

cx + sizeX, cy - sizeY,

cx + sizeX, cy + sizeY,

cx - sizeX, cy + sizeY,

cx - sizeX, cy - sizeY,

cx - sizeX / 2, cy,

cx, cy - sizeY

};

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 3, color);

SelectObject(hdc, hPen);

Polyline(hdc, p, 8);

DeleteObject(hPen);

}

void cor\_gor(HDC hdc) {

{

int r = 0;

int x = 50;

int sizex = 10;

while (x < 2100) {

Cor(hdc, x, 50, sizex, 20, RGB(r, 0, 0));

x = x + 100;

sizex = sizex + 3;

r += 20;

}

}

}

void cor\_ver(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

while (y < 2100) {

Cor(hdc, x, y, sizex, sizey, RGB(r, 0, 0));

y = y + 100;

sizey = sizey + 3;

r += 20;

}

}

}

void cor\_riad(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

int b = 0;

while (y < 2100) {

while (x < 2100) {

x = x + 100;

sizex = sizex + 1;

b += 1;

Cor(hdc, x, y, sizex, sizey, RGB(r, 0, b));

}

x = 100;

y = y + 100;

sizey = sizey + 1;

r += 20;

}

}

}

void cor\_diag(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

while (y < 2100) {

Cor(hdc, x, y, sizex, sizey, RGB(r, 0, 0));

y = y + 100;

sizey = sizey + 3;

x = x + 100;

sizex = sizex + 3;

r += 20;

}

}

}

void MyFigure(HDC hdc, int cx, int cy, int sizeX, int sizeY, COLORREF color) {

POINT l[] = {

cx, cy,

cx + sizeX, cy + sizeY,

cx, cy + sizeY \* 2,

cx - sizeX, cy + sizeY,

cx, cy

};

HPEN hPen;

hPen = CreatePen(PS\_SOLID, 3, color);

SelectObject(hdc, hPen);

Polyline(hdc, l, 5);

DeleteObject(hPen);

}

void f\_gor(HDC hdc) {

{

int r = 0;

int x = 50;

int sizex = 10;

while (x < 2100) {

MyFigure(hdc, x, 50, sizex, 20, RGB(r, 0, 0));

x = x + 100;

sizex = sizex + 3;

r += 20;

}

}

}

void f\_riad(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

int b = 0;

while (y < 2100) {

while (x < 2100) {

x = x + 100;

sizex = sizex + 1;

b += 1;

MyFigure(hdc, x, y, sizex, sizey, RGB(r, 0, b));

}

x = 100;

y = y + 100;

sizey = sizey + 1;

r += 20;

}

}

}

void f\_diag(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

while (y < 2100) {

MyFigure(hdc, x, y, sizex, sizey, RGB(r, 0, 0));

y = y + 100;

sizey = sizey + 3;

x = x + 100;

sizex = sizex + 3;

r += 20;

}

}

}

void f\_ver(HDC hdc) {

{

int r = 0;

int x = 100;

int sizex = 10;

int y = 50;

int sizey = 10;

while (y < 2100) {

MyFigure(hdc, x, y, sizex, sizey, RGB(r, 0, 0));

y = y + 100;

sizey = sizey + 3;

r += 20;

}

}

}

void f\_p(HDC hdc) {

{

int r = 0;

int x = 50;

int sizex = 10;

int y = 50;

int sizey = 0;

while (y < 2100) {

MyFigure(hdc, x, y, sizex, sizey, RGB(r, 0, 0));

y = y + 100;

x = x + 100;

sizex = sizex + 1;

sizey = sizey + 1;

r += 20;

}

}

}

void Claus(HDC hdc, int x, int y) {

MoveToEx(hdc, 20 + x, 0 + y, NULL);

LineTo(hdc, 30 + x, 20 + y);

LineTo(hdc, 10 + x, 20 + y);

LineTo(hdc, 20 + x, 0 + y);

MoveToEx(hdc, 20 + x, 10 + y, NULL);

LineTo(hdc, 30 + x, 40 + y);

LineTo(hdc, 10 + x, 40 + y);

LineTo(hdc, 20 + x, 10 + y);

MoveToEx(hdc, 20 + x, 30 + y, NULL);

LineTo(hdc, 40 + x, 80 + y);

LineTo(hdc, 0 + x, 80 + y);

LineTo(hdc, 20 + x, 30 + y);

}

void el(HDC hdc) {

{

int y = 0;

int x = 0;

while (y < 1100) {

Claus(hdc, x, y);

x = x + 50;

y = y + 80;

}

}

}

void was(HDC hdc) {

{

int y = 0;

int x = 500;

int p = 0;

int k = 1;

int l = 0;

while (y < 500) {

p = (x - 50 \* (k / 3));

while (l < k) {

Claus(hdc, p, y);

p = p + 50;

l = l + 1;

}

l = 0;

k = k \* 3;

y = y + 80;

x = 500;

}

}

}

void centr(HDC hdc) {

Claus(hdc, 0, 0);

Claus(hdc, 1380, 0);

Claus(hdc, 0, 620);

Claus(hdc, 1380, 620);

Claus(hdc, 700, 310);

}

void line(HDC hdc) {

{

int x = 0;

while (x < 1100) {

Claus(hdc, x, 0);

x = x + 50;

}

}

}

void linever(HDC hdc) {

{

int y = 0;

while (y < 1100) {

Claus(hdc, 0, y);

y = y + 80;

}

}

}

void diag(HDC hdc) {

{

int y = 0;

int x = 0;

while (y < 1100) {

Claus(hdc, x, y);

x = x + 50;

y = y + 80;

}

}

}

void riad(HDC hdc) {

{

int y = 0;

int x = 0;

while (y < 1100) {

while (x < 1000) {

Claus(hdc, x, y);

x = x + 50;

}

x = 0;

y = y + 80;

}

}

}

void Sneg(HDC hdc, int x, int y) {

Ellipse(hdc, 20 + x, 0 + y, 40 + x, 20 + y);

Ellipse(hdc, 10 + x, 20 + y, 50 + x, 60 + y);

Ellipse(hdc, 0 + x, 60 + y, 60 + x, 120 + y);

MoveToEx(hdc, 20 + x, 20 + y, NULL);

LineTo(hdc, 0 + x, 30 + y);

MoveToEx(hdc, 38 + x, 20 + y, NULL);

LineTo(hdc, 58 + x, 30 + y);

MoveToEx(hdc, 35 + x, 120 + y, NULL);

LineTo(hdc, 60 + x, 10 + y);

LineTo(hdc, 58 + x, 0 + y);

MoveToEx(hdc, 60 + x, 10 + y, NULL);

LineTo(hdc, 64 + x, 0 + y);

MoveToEx(hdc, 60 + x, 10 + y, NULL);

LineTo(hdc, 70 + x, 0 + y);

MoveToEx(hdc, 60 + x, 10 + y, NULL);

}

void Logo(HDC hdc, int x, int y) {

MoveToEx(hdc, 25 + x, 0 + y, NULL);

LineTo(hdc, 0 + x, 50 + y);

LineTo(hdc, 25 + x, 100 + y);

LineTo(hdc, 50 + x, 50 + y);

LineTo(hdc, 25 + x, 0 + y);

}

void centr\_logo(HDC hdc) {

Logo(hdc, 0, 0);

Logo(hdc, 1380, 0);

Logo(hdc, 0, 620);

Logo(hdc, 1380, 620);

Logo(hdc, 700, 310);

}

void line\_logo(HDC hdc) {

{

int x = 0;

while (x < 1100) {

Logo(hdc, x, 0);

x = x + 70;

}

}

}

void linever\_Logo(HDC hdc) {

{

int y = 0;

while (y < 1100) {

Logo(hdc, 0, y);

y = y + 120;

}

}

}

void diag\_Logo(HDC hdc) {

{

int y = 0;

int x = 0;

while (y < 1100) {

Logo(hdc, x, y);

x = x + 50;

y = y + 80;

}

}

}

void riad\_Logo(HDC hdc) {

{

int y = 0;

int x = 0;

while (y < 1100) {

while (x < 1000) {

Logo(hdc, x, y);

x = x + 70;

}

x = 0;

y = y + 120;

}

}

}

void im1(HDC hdc, int cx, int cy, int size) {

int x1 = cx - size;

int y1 = cy - size;

int x2 = cx + size;

int y2 = cy - size;

int x3 = cx;

int y3 = cy + size;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x1, y1);

}

void recurklim1\_1(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_1(hdc, cx - size, cy - size, size / 2);

}

void recurklim1\_2(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_2(hdc, cx - size, cy - size, size / 2);

recurklim1\_2(hdc, cx + size, cy - size, size / 2);

}

void recurklim1\_3(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_3(hdc, cx - size, cy - size, size / 2);

recurklim1\_3(hdc, cx, cy + size, size / 2);

}

void recurklim1\_4(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_4(hdc, cx - size, cy - size, size / 2);

recurklim1\_4(hdc, cx + size, cy - size, size / 2);

recurklim1\_4(hdc, cx, cy + size, size / 2);

}

void recurklim1\_5(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_5(hdc, cx, cy - size, size / 2);

}

void recurklim1\_6(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_6(hdc, cx + size, cy, size / 2);

}

void recurklim1\_7(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_7(hdc, cx - size, cy, size / 2);

}

void recurklim1\_8(HDC hdc, int cx, int cy, int size) {

im1(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim1\_8(hdc, cx - size, cy, size / 2);

recurklim1\_8(hdc, cx + size, cy, size / 2);

recurklim1\_8(hdc, cx, cy - size, size / 2);

}

void im2(HDC hdc, int cx, int cy, int size) {

int x1 = cx - size / 2;

int y1 = cy - size;

int x2 = cx + size / 2;

int y2 = cy - size;

int x3 = cx - size;

int y3 = cy + size;

int x4 = cx + size;

int y4 = cy + size;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

}

void recurklim2\_1(HDC hdc, int cx, int cy, int size) {

im2(hdc, cx, cy, size);

if (size < 5) {

return;

}

recurklim2\_1(hdc, cx - size, cy + size, size / 2);

}

void recurklim2\_2(HDC hdc, int cx, int cy, int size) {

im2(hdc, cx, cy, size);

if (size < 5) {

return;

}

recurklim2\_2(hdc, cx - size / 2, cy - size, size / 2);

}

void recurklim2\_3(HDC hdc, int cx, int cy, int size) {

im2(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim2\_3(hdc, cx - size / 2, cy - size, size / 2);

recurklim2\_3(hdc, cx + size / 2, cy - size, size / 2);

}

void recurklim2\_4(HDC hdc, int cx, int cy, int size) {

im2(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim2\_4(hdc, cx - size, cy + size, size / 2);

recurklim2\_4(hdc, cx + size, cy + size, size / 2);

}

void recurklim2\_5(HDC hdc, int cx, int cy, int size) {

im2(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurklim2\_3(hdc, cx - size / 2, cy - size, size / 2);

recurklim2\_3(hdc, cx + size / 2, cy - size, size / 2);

recurklim2\_5(hdc, cx - size, cy + size, size / 2);

recurklim2\_5(hdc, cx + size, cy + size, size / 2);

}

void im3(HDC hdc, int cx, int cy, int size) {

int x1 = cx;

int y1 = cy - size;

int x2 = cx + size;

int y2 = cy;

int x3 = cx;

int y3 = cy + size;

int x4 = cx - size;

int y4 = cy;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

}

void recurs\_troi(HDC hdc, int cx, int cy, int size) {

im3(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_troi(hdc, cx + size, cy, size / 2);

}

void recurs\_troi2(HDC hdc, int cx, int cy, int size) {

im3(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_troi2(hdc, cx + size, cy, size / 2);

recurs\_troi2(hdc, cx - size, cy, size / 2);

}

void recurs\_troi3(HDC hdc, int cx, int cy, int size) {

im3(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_troi3(hdc, cx + size, cy, size / 2);

recurs\_troi3(hdc, cx - size, cy, size / 2);

recurs\_troi3(hdc, cx, cy - size, size / 2);

recurs\_troi3(hdc, cx, cy + size, size / 2);

}

void im4(HDC hdc, int cx, int cy, int size) {

int x1 = cx;

int y1 = cy - size;

int x2 = cx + size / 4;

int y2 = cy - size / 4;

int x3 = cx + size;

int y3 = cy;

int x4 = cx + size / 4;

int y4 = cy + size / 4;

int x5 = cx;

int y5 = cy + size;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

}

void im5(HDC hdc, int cx, int cy, int size) {

int x1 = cx - size / 2;

int y1 = cy;

int x2 = cx - size;

int y2 = cy + size / 2;

int x3 = cx + size / 2;

int y3 = cy + size / 2;

int x4 = cx + size;

int y4 = cy;

int x5 = cx;

int y5 = cy + size;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

}

void recurs\_kvadr(HDC hdc, int cx, int cy, int size) {

im5(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_kvadr(hdc, cx + size, cy, size / 2);

}

void recurs\_kvadr2(HDC hdc, int cx, int cy, int size) {

im5(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_kvadr2(hdc, cx + size / 2, cy + size / 2, size / 2);

recurs\_kvadr2(hdc, cx - size / 2, cy + size / 2, size / 2);

recurs\_kvadr2(hdc, cx + size / 2, cy - size / 2, size / 2);

recurs\_kvadr2(hdc, cx - size / 2, cy - size / 2, size / 2);

}

void im6(HDC hdc, int cx, int cy, int size) {

int x1 = cx - size;

int y1 = cy - size;

int x2 = cx + size;

int y2 = cy + size;

Ellipse(hdc, x1, y1, x2, y2);

}

void recurs\_krug(HDC hdc, int cx, int cy, int size) {

im6(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_krug(hdc, cx + size, cy, size / 2);

}

void recurs\_krug2(HDC hdc, int cx, int cy, int size) {

im6(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_krug2(hdc, cx + size, cy, size / 2);

recurs\_krug2(hdc, cx - size, cy, size / 2);

recurs\_krug2(hdc, cx, cy - size, size / 2);

recurs\_krug2(hdc, cx, cy + size, size / 2);

}

void recurs\_krug3(HDC hdc, int cx, int cy, int size) {

im6(hdc, cx, cy, size);

if (size < 20) {

return;

}

recurs\_krug3(hdc, cx + size, cy, size / 2);

recurs\_krug3(hdc, cx - size, cy, size / 2);

recurs\_krug3(hdc, cx, cy - size, size / 2);

recurs\_krug3(hdc, cx, cy + size, size / 2);

recurs\_krug3(hdc, cx + size / 2, cy + size / 2, size / 2);

recurs\_krug3(hdc, cx - size / 2, cy + size / 2, size / 2);

recurs\_krug3(hdc, cx + size / 2, cy - size / 2, size / 2);

recurs\_krug3(hdc, cx - size / 2, cy - size / 2, size / 2);

}

void im7(HDC hdc, int cx, int cy, int size) {

int x1 = cx - size / 2;

int y1 = cy + size / 2;

int x2 = cx - size;

int y2 = cy;

int x3 = cx;

int y3 = cy;

int x4 = cx;

int y4 = cy + size;

int x5 = cx;

int y5 = cy + size;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

LineTo(hdc, x3, y3);

LineTo(hdc, x4, y4);

LineTo(hdc, x1, y1);

}

void j(HDC hdc) {

if (KArtin == 0) {

f\_gor(hdc);

}

if (KArtin == 2) {

f\_ver(hdc);

}

if (KArtin == 3) {

f\_diag(hdc);

}

if (KArtin == 4) {

f\_riad(hdc);

}

if (KArtin == 5) {

f\_p(hdc);

}

if (KArtin == 6) {

el(hdc);

}

if (KArtin == 7) {

was(hdc);

}

if (KArtin == 8) {

centr(hdc);

}

if (KArtin == 9) {

line(hdc);

}

if (KArtin == 10) {

linever(hdc);

}

if (KArtin == 11) {

diag(hdc);

}

if (KArtin == 12) {

riad(hdc);

}

if (KArtin == 13) {

centr\_logo(hdc);

}

if (KArtin == 14) {

line\_logo(hdc);

}

if (KArtin == 15) {

linever\_Logo(hdc);

}

if (KArtin == 16) {

diag\_Logo(hdc);

}

if (KArtin == 17) {

riad\_Logo(hdc);

}

if (KArtin == 18) {

recurklim1\_1(hdc, 200, 160, 80);

}

if (KArtin == 19) {

recurklim1\_1(hdc, 200, 160, 80);

}

if (KArtin == 20) {

recurklim1\_2(hdc, 200, 160, 80);

}

if (KArtin == 21) {

recurklim1\_3(hdc, 200, 160, 80);

}

if (KArtin == 22) {

recurklim1\_4(hdc, 200, 160, 80);

}

if (KArtin == 23) {

recurklim1\_5(hdc, 200, 160, 80);

}

if (KArtin == 24) {

recurklim1\_6(hdc, 200, 160, 80);

}

if (KArtin == 25) {

recurklim1\_7(hdc, 200, 160, 80);

}

if (KArtin == 26) {

recurklim1\_8(hdc, 200, 160, 80);

}

if (KArtin == 27) {

im7(hdc, 200, 200, 100);

}

if (KArtin == 28) {

recurs\_krug3(hdc, 200, 200, 100);

}

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_KEYDOWN:

switch (wParam)

{

case VK\_DOWN:

KArtin += 1;

InvalidateRect(hWnd, NULL, TRUE);

break;

case VK\_UP:

KArtin -= 1;

InvalidateRect(hWnd, NULL, TRUE);

break;

case VK\_RETURN:

if (KArtin == 5) {

}

break;

}

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

//Cor(hdc, 400, 300, 200, 100, RGB(0, 255, 255));

//

//cor\_gor(hdc);

//cor\_ver(hdc);

//cor\_diag(hdc);

//cor\_riad(hdc);

//MyFigure(hdc, 400, 400, 100, 100, RGB(0, 0, 0));

//

//f\_gor(hdc);

//f\_ver(hdc);

//f\_diag(hdc);

//f\_riad(hdc);

j(hdc);

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

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

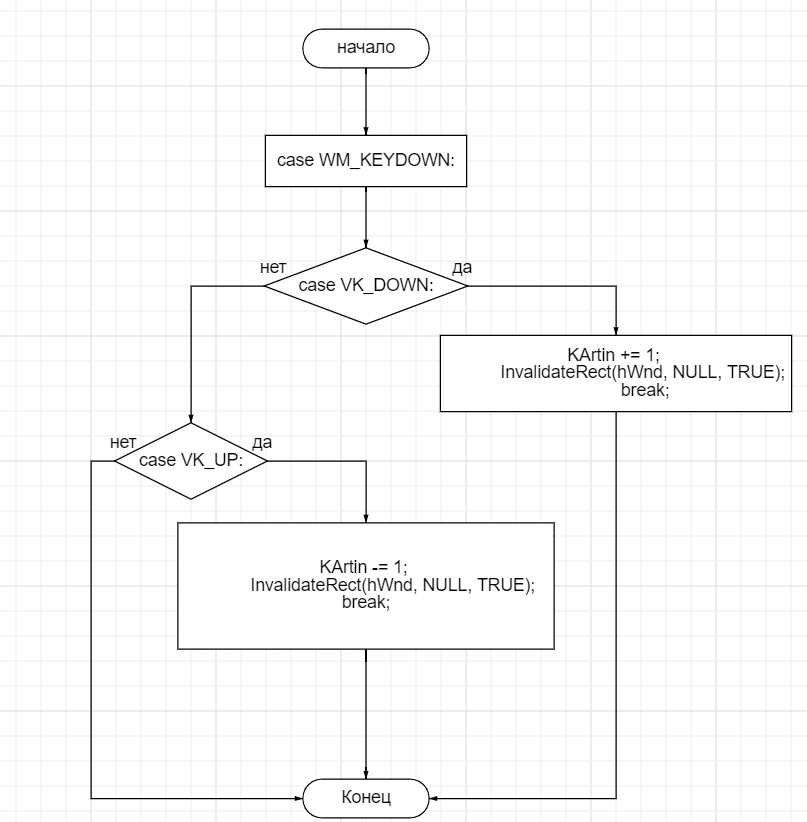
default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}



ВЫВОД

В ЭТОЙ лабораторной работе я научился использовать функцию KEYDOWN