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/lab4>

Домашнее задание

1) Выбрать 3 фигуры из вариантов и реализовать их дома.

2) Доделать все задачи с классной работы.

3) Придумать 2+ свои собственные фигуры и реализовать на них красивые рекурсивные картинки.

4) Оформить отчет – см следующий слайд

Всё делал снова в функциях:

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);

}







