Министерство Образования и Науки Российской Федерации  
Новосибирский Государственный Технический Университет  
Кафедра Теоретической Прикладной Информатики

**Лабораторная работа № 1**

**по дисциплине «Объектно-ориентированное программирование»**

Факультет: ФПМИ

Группа: ПМ-53

Студент: Тябин Егор Алексеевич

Преподаватель: Еланцева Ирина Леонидовна

### Новосибирск 2016

# Условие задачи

Нарисовать контур ромба, ромб и ромб в ромбе средствами visual studio.

# 2. Анализ задачи

*Дано:*

Ромб – четырехугольник, противоположные стороны которого параллельны, и все стороны равны.  
В файле «ok.txt»  
Последовательность целых чисел, которые являются координатами вершин ромба, всего 8 чисел.  
И последовательность целых чисел, которые являются числами задания цвета контура и заливки фигуры, всего 6 чисел.

*Результат:*   
Нарисованный контур 1-ого указанного ромба/1-й указанный ромб/ромб в ромбе или сообщение об ошибке.

## Метод решения

Для работы с нашей задачей выделим несколько основных подзадач.

## Выделение основных подзадач

1. Рисование контура ромба
2. Рисование ромба
3. Рисование ромба в ромбе
4. Проверка, является ли фигура ромбом
5. Проверка, вложена ли 1-я фигура во 2-ю
6. Проверка, выходит ли фигура за рамки окна

1) Рисуется линия (контур) (функция Polyline) по вершинам ромба, если вершины фигуры не выходят за границы окна и, если фигура является ромбом.

2) Рисуется многоугольник (функция Polygon) по заданным вершинам, внутренняя часть закрашивается, если вершины фигуры не выходят за границы окна и, если фигура является ромбом.

3) Рисуется 1-й ромб, затем второй (функция Polygon), если обе фигуры являются ромбом и, если вершины фигур не выходят за границы окна, также проверяется, вложена ли вторая фигура в первую.

4) Ромб – четырехугольник, в котором все стороны равны и противоположные стороны параллельны. Соответственно проверяется равны ли стороны, и параллельны ли противоположные стороны.

5) Для проверки вложенности первой фигуры во вторую проверяется вложенность каждой стороны в первую фигуру.

6) Проверятся, выходит ли какая-либо вершина за рамки окна.

# 3. Структуры данных, используемые для представления исходных данных и результатов задачи

## Внешнее представление входных данных

Последовательность целых чисел

## Внешнее представление выходных данных

Надпись или графическое изображение

# Внутреннее представление входных данных

POINT ppt[5], ppt2[5] – массивы типа POINT, в которых хранятся координаты 1-й и 2-й фигуры соответственно. Последний элемент массива = первому. (для замыкания контура фигуры)  
int penColour[2][3], brushColour[2][3] – массивы целочисленных элементов, в которых хранятся цвета пера и кисти соответственно, обеих фигур.

## Внутреннее представление данных

struct ahuman{ char number[14], mark[30]; FIO name; } - структура ahuman, состоящая из 3-х полей: массивов number и mark символьного типа и name типа FIO.   
struct FIO{ char F[20], I[20], O[20]; } – структура FIO, состоящая из 3-х полей: массивов F, I и O символьного типа.

char a – элемент типа char, который предназначен для выбора команды рисования. (нарисовать контур/первую фигуру/фигуру в фигуре)

int error – элемент целочисленного типа, в который записывается номер ошибки.

# 4. Укрупненный алгоритм решения задачи

**АЛГ** check\_FinF(POINT m1, POINT m2, POINT \*n) **{**

T=0;  
 Начиная с i=0, Повторять:   
 z = (n[i].x - m1.x)\*(m2.y - m1.y) - (n[i].y - m1.y)\*(m2.x - m1.x);  
 Если z<0 : t=t+1;  
 Иначе: возврат 0;  
 i=i+1;  
 Пока: i<4  
 Возврат t;  
 **}**

**АЛГ** check\_figure(POINT \*ppt) {

Начиная с i=0, Повторять:  
 koef[i] = slope(ppt[i], ppt[i + 1]);

dl[i] = sqrt((ppt[i].x - ppt[i + 1].x)\*(ppt[i].x - ppt[i + 1].x) + (ppt[i].y - ppt[i + 1].y)\*(ppt[i].y - ppt[i + 1].y));  
i=i+1;

Пока i<4;

Начиная с i=0, Повторять:

Если dl[i] ≠ dl[i + 1]: возврат 2;  
 i=i+1;  
 Пока i<3;

Начиная с i=0, Повторять:

Если koef[i] ≠ koef[i + 2]: возврат 3;  
i=i+1;

Пока i<2;  
  
Возврат 0;  
}

**АЛГ** check\_borders(POINT \*ppt, long rr, long rb) {

Начиная с i=0, Повторять:

Если ppt[i].x >= rr || ppt[i].y >= rb: возврат 1;  
i=i+1;

Пока i<4;  
  
Возврат 0;  
}

**АЛГ** main() {

Считывание координат первой фигуры;  
Проверка фигуры: check\_figure(POINT \*ppt);  
Считывание цветов контура и заливки 1-й фигуры;

Считывание координат второй фигуры;  
Проверка фигуры: check\_figure(POINT \*ppt2);  
Считывание цветов контура и заливки 2-й фигуры;

Функция-переключатель:

Если ‘1’: Если line(hwnd, hdc, ppt) ≠ 0, то Сообщение об ошибке  
Если ‘2’: Если figure(hwnd, hdc, ppt) ≠ 0, то Сообщение об ошибке  
Если ‘3’: error = FinF(hwnd, hdc, ppt, ppt2, hBrush2, hPen2);  
 Если error = 1, то Сообщение об ошибке  
 Если error = 4, то Сообщение об ошибке  
Если Esc, то завершение программы

}

# Структура программы

## Взаимосвязь функций

Void main()

## Составные части программы

Наименование функции:   
line  
Прототип функции:  
int line(HWND hwnd, HDC hdc, POINT \*ppt)  
Данная функция рисует контур ромба, если он не выходит за границы окна.

Наименование функции:  
figure  
Прототип функции:  
int figure(HWND hwnd, HDC hdc, POINT \*ppt)  
Данная функция рисует закрашенный ромб, если он не выходит за границы окна.

Наименование функции:  
FinF  
Прототип функции:  
int FinF(HWND hwnd, HDC hdc, POINT \*ppt, POINT \*ppt2, HBRUSH brush, HPEN pen)  
Данная функция рисует 2-й ромб внутри первого, если второй ромб не выходит за границы первого. И если оба ромба не выходят за границы окна.

Наименование функции:  
check\_borders  
Прототип функции:  
int check\_borders(POINT \*ppt, long rr, long rb)  
Данная функция проверяет нахождение вершин ромба внутри консольного окна. Если одна из вершин за границами окна, возвращается 1.

Наименование функции:  
check\_FinF  
Прототип функции:  
int check\_FinF(POINT m1, POINT m2, POINT \*n)  
Данная функция проверяет нахождение стороны ромба внутри другого ромба. Таким образом, эту функцию надо вызывать 4 раза.

Наименование функции:  
slope  
Прототип функции:  
float slope(POINT a, POINT b)  
Данная функция вычисляет угловой коэффициент прямой. Если угловые коэффициенты двух прямых равны, то они параллельны.

# Текст программы

**/\* Файл main.cpp \*/**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <locale.h>

#include <conio.h>

#include <iostream>

#include <string.h>

//Коды ошибок

#define outside\_borders 1

#define wrong\_dlina 2

#define wrong\_parallel 3

#define outside\_figure 4

int check\_FinF(POINT m1, POINT m2, POINT \*n)

{

int t = 0, z;

for (int i = 0; i < 4; i++)

{

z = (n[i].x - m1.x)\*(m2.y - m1.y) - (n[i].y - m1.y)\*(m2.x - m1.x);

if (z < 0) t++;

else return 0;

}

return t;

}

float slope(POINT a, POINT b)

{

return (b.y - a.y) / (b.x - a.x);

}

int check\_figure(POINT \*ppt)

{

float koef[4];

float dl[4];

int i;

for (i = 0; i < 4; i++)

{

koef[i] = slope(ppt[i], ppt[i + 1]);

dl[i] = sqrt((ppt[i].x - ppt[i + 1].x)\*(ppt[i].x - ppt[i + 1].x) + (ppt[i].y - ppt[i + 1].y)\*(ppt[i].y - ppt[i + 1].y));

}

for (i = 0; i < 3; i++)

{

if (dl[i] != dl[i + 1]) return wrong\_dlina;

}

for (i = 0; i < 2; i++)

{

if (koef[i] != koef[i + 2]) return wrong\_parallel;

}

return 0;

}

int check\_borders(POINT \*ppt, long rr, long rb)

{

int i;

for (i = 0; i < 4; i++)

{

if (ppt[i].x >= rr || ppt[i].y >= rb) return outside\_borders;

}

return 0;

}

int line(HWND hwnd, HDC hdc, POINT \*ppt)

{

int a;

RECT rt;

char buf[100];

GetClientRect(hwnd, &rt);

sprintf\_s(buf, "Размер окна %d на %d пикселей", rt.right, rt.bottom);

TextOutA(hdc, 10, 10, buf, strlen(buf));

a = check\_borders(ppt, rt.right, rt.bottom);

if (a != 0) return a;

Polyline(hdc, ppt, 5);

return 0;

}

int figure(HWND hwnd, HDC hdc, POINT \*ppt)

{

int a;

RECT rt;

char buf[100];

GetClientRect(hwnd, &rt);

sprintf\_s(buf, "Размер окна %d на %d пикселей", rt.right, rt.bottom);

TextOutA(hdc, 10, 10, buf, strlen(buf));

a = check\_borders(ppt, rt.right, rt.bottom);

if (a != 0) return a;

Polygon(hdc, ppt, 4);

return 0;

}

int FinF(HWND hwnd, HDC hdc, POINT \*ppt, POINT \*ppt2, HBRUSH brush, HPEN pen)

{

int a, i;

RECT rt;

char buf[100];

GetClientRect(hwnd, &rt);

sprintf\_s(buf, "Размер окна %d на %d пикселей", rt.right, rt.bottom);

TextOutA(hdc, 10, 10, buf, strlen(buf));

a = check\_borders(ppt, rt.right, rt.bottom);

if (a != 0) return a;

a = check\_borders(ppt2, rt.right, rt.bottom);

if (a != 0) return a;

for (i = 0; i < 4; i++)

{

if (check\_FinF(ppt[i], ppt[i + 1], ppt2) == 0) return outside\_figure;

}

Polygon(hdc, ppt, 4);

SelectBrush(hdc, brush);

SelectPen(hdc, pen);

Polygon(hdc, ppt2, 4);

return 0;

}

void main()

{

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

char a;

int i, exitFlag = 0, error, penColour[2][3], brushColour[2][3];

POINT ppt[5], ppt2[5];

setlocale(LC\_CTYPE, "Russian");

FILE \*fp;

fp = fopen("ok.txt", "r");

for (i = 0; i < 4; i++)

{

fscanf\_s(fp, "%d %d", &ppt[i].x, &ppt[i].y);

}

ppt[4].x = ppt[0].x;

ppt[4].y = ppt[0].y;

error = check\_figure(ppt);

if (error == 0)

{

for (i = 0; i < 3; i++)

{

fscanf\_s(fp, "%d", &penColour[0][i]);

}

for (i = 0; i < 3; i++)

{

fscanf\_s(fp, "%d", &brushColour[0][i]);

}

for (i = 0; i < 4; i++)

{

fscanf\_s(fp, "%d %d", &ppt2[i].x, &ppt2[i].y);

}

ppt2[4].x = ppt2[0].x;

ppt2[4].y = ppt2[0].y;

error = check\_figure(ppt2);

if (error == 0)

{

for (i = 0; i < 3; i++)

{

fscanf\_s(fp, "%d", &penColour[1][i]);

}

for (i = 0; i < 3; i++)

{

fscanf\_s(fp, "%d", &brushColour[1][i]);

}

SetBkColor(hdc, RGB(0, 0, 0));

SetTextColor(hdc, RGB(255, 0, 0));

HPEN hPen = CreatePen(PS\_SOLID, 3, RGB(penColour[0][0], penColour[0][1], penColour[0][2]));

HPEN hPen2 = CreatePen(PS\_SOLID, 3, RGB(penColour[1][0], penColour[1][1], penColour[1][2]));

HPEN hOldPen = SelectPen(hdc, hPen);

HBRUSH hBrush = CreateSolidBrush(RGB(brushColour[0][0], brushColour[0][1], brushColour[0][2]));

HBRUSH hBrush2 = CreateSolidBrush(RGB(brushColour[1][0], brushColour[1][1], brushColour[1][2]));

HBRUSH hOldBrush = SelectBrush(hdc, hBrush);

do {

printf("<1> - построить контур 1-й фигуры\n");

printf("<2> - построить 1-ую фигуру\n");

printf("<3> - построить две фигуры\n");

printf("<ESC> - завершение программы\n\n");

int repeatFlag = 0;

do {

a = \_getch();

switch (a)

{

case '1':

SelectBrush(hdc, hBrush);

SelectPen(hdc, hPen);

if (line(hwnd, hdc, ppt) != 0)

printf("Часть фигуры лежит за границами окна\n");

break;

case '2':

SelectBrush(hdc, hBrush);

SelectPen(hdc, hPen);

if (figure(hwnd, hdc, ppt) != 0)

printf("Часть фигуры лежит за границами окна\n");

break;

case '3':

SelectBrush(hdc, hBrush);

SelectPen(hdc, hPen);

error = FinF(hwnd, hdc, ppt, ppt2, hBrush2, hPen2);

if (error == 1)

printf("Часть фигуры лежит за границами окна\n");

if (error == 4)

printf("Одна фигура не входит в другую");

break;

case 27:

exitFlag = 1;

break;

default: printf("ОШИБКА: неверный номер команды\n");

repeatFlag = 1;

}

} while (!repeatFlag && !exitFlag);

} while (!exitFlag);

ReleaseDC(hwnd, hdc);

}

else

{

printf("Вторая фигура не является ромбом.");

\_getch();

}

}

else

{

printf("Первая фигура не является ромбом.");

\_getch();

}

}

# 7.Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результат | Примечания |
| 1 | 200 150 250 100 300 150 250 200 255 0 0 0 255 0  220 150 250 120 280 150 250 180 255 255 255 0 0 255 |  | Рисование не закрашенного ромба Не выходит за границы окна. |
| 2 | 150 100 200 100 200 150 150 150 255 180 180 0 255 100   220 150 250 120 280 150 250 180 255 255 255 0 0 255 |  | Рисование закрашенного ромба |
| 3 | 200 150 250 100 300 150 250 200 255 0 0 0 255 0  220 150 250 120 280 150 250 180 255 255 255 0 0 255 |  | Рисование вложенных ромбов |
| 4 | 200 150 250 100 300 150 250 200 255 0 0 0 255 0  200 150 240 110 280 150 240 190 255 255 255 0 0 255 | Часть фигуры лежит за границами окна | Координаты точек ромба не входят в размер консольного окна |
| 5 | 220 150 250 100 300 150 250 200 255 0 0 0 255 0  220 150 250 120 280 150 250 180 255 255 255 0 0 255 | Первая фигура не является ромбом | Проверка на ромб |
|  | 200 150 250 100 300 150 250 200 255 0 0 0 255 0  200 150 240 110 280 150 240 190 255 255 255 0 0 255 | Одна фигура не входит в другую | Треугольники имеют общую точку |
| 8 | -200 150 250 100 300 150 250 200 255 0 0 0 255 0  220 150 250 120 280 150 250 180 255 255 255 0 0 255 | Неверные координаты | Отрицательная координата |

# 8. Результат работы программы

-Программа выдала верное решение на всех тестах и, следовательно, правильно работает.