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

**Лабораторная работа № 2**

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

Факультет: ФПМИ

Группа: ПМ-53

Студент: Тябин Егор Алексеевич

Преподаватель: Еланцева Ирина Леонидовна

### Новосибирск 2016

# Условие задачи

Нарисовать ромб средствами visual studio, чтобы был определён класс, реализующий понятие геометрической фигуры в графической системе. Определить функции, необходимые для реализации класса.

# 2. Анализ задачи

*Дано:*

Ромб – четырехугольник, противоположные стороны которого параллельны, и все стороны равны.  
Последовательность целых чисел, которые являются координатами вершин ромба, всего 8 чисел.  
Далее следуют графические характеристики.   
Формат: <тип пера[0-6]> <толщина пера> (<цвет границы(пера) в формате RGB>) <тип кисти(заливки)> (<цвет заливки в формате RGB>)  
Пример задания фигуры в файле: 200 150 250 100 300 150 250 200; 0 5 (200, 200, 0) -1 (100, 100, 200)  
Также при общении с пользователем просят ввести 2 координаты перемещения фигуры, которые являются целыми числами.  
При загрузке/сохранении в файл – последовательность символов(название файла).

*Результат:*   
Нарисованный ромб в графическом интерфейсе visual studio или сообщение об ошибке.

## Метод решения

Для работы с нашей задачей выделим несколько основных подзадач.

## Выделение основных подзадач

1. Перемещение ромба
2. Установка новых значений вершин ромба
3. Проверка, является ли фигура ромбом
4. Проверка, не выходит ли фигура за границы окна
5. Рисование ромба
6. Загрузка из файла
7. Сохранение в файл

1) Каждая вершина ромба перемещается на указанное количество пикселей-координат по осям Ох и Оу

2) Атрибутам класса, отвечающие за значения координат вершин, присваивается новое значение

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

4) Каждая вершина проверяется на выход за границы консольного окна

5) Рисуется многоугольник по 4 вершинам ромба (функция Polygon)

6) Данные считываются из файла (формат представления данных можно посмотреть в *Дано*), после чего проверяется, является ли фигура ромбом, и не выходит ли она за границы окна

7) Данные сохраняются в указанный файл (формат смотреть в *Дано*)

# 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

*Class romb:*

POINT t[4] (tk[4])– массив типа POINT, в котором хранятся координаты 4 вершин ромба.

color pcolor, bcolor – элементы типа color, которые являются параметрическими характеристиками цветов пера и кисти соответственно; формат RGB

int ptype, btype – целочисленный элемент, в котором записывается тип пера и кисти

int width – целочисленный элемент, являющийся толщиной пера; не может быть меньше 0

## Внутреннее представление данных

struct color { int r, g, b; } – структура color, которая является совокупностью 3-х целочисленных элементов являющимися параметрами цвета в формате RGB

class romb – класс “romb” , реализующий понятие геометрической фигуры – ромба, в графической системе.

int i – целочисленный элемент используемый для перебора

float koef – элемент вещественного типа используемый для перебора

float dl – элемент вещественного типа обозначающий длину стороны

# 4. Укрупненный алгоритм решения задачи

check\_figure(POINT \*ppt) { Проверка, является ли фигура ромбом

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;  
}

check\_borders(POINT \*ppt, long rr, long rb) { Проверка выхода вершин ромба за границы окна

Начиная с i=0, Повторять:

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

Пока i<4;  
  
Возврат 0;  
}

SetPos(int new\_x, int new\_y) {

Начиная с i=0, Повторять:  
 Если координата вершины + длина перемещения выходит за границы окна, то   
 ошибка  
 Пока i<4  
Начиная с i=0, Повторять:  
 t[i].x=new\_x+t[i].x  
 t[i].y=new\_y+t[i].y  
 Пока i<4

}

main() {

Функция-переключатель:

Если ‘1’: Считывание данных из указанного файла, и последующее рисование ромба, если он таковым является, иначе ошибка.  
Если ‘2’: Сохранение данных в файл  
Если ‘3’: Задание координат фигуры, затем задание параметров пера, затем задание параметров кисти  
Если ‘4’: Перемещение фигуры, если не выходит за границы окна, иначе ошибка  
Если ‘5’: Показать фигуру  
Если Esc, то завершение программы

}

# Структура программы

## Взаимосвязь функций

Void main()

## Составные части программы

Наименование функции:   
SetPos  
Прототип функции:  
void romb::SetPos(int new\_x, int new\_y)  
Данная функция перемещает каждую вершину на указанный вектор xy.

Наименование функции:  
SetFigure  
Прототип функции:  
void romb::SetFigure(POINT\* new\_t)  
Данная функция присваивает атрибутам класса(координаты вершин) новые значения

Наименование функции:  
GetCoord  
Прототип функции:  
POINT\* romb::GetCoord()  
Данная функция возвращает координаты вершин фигуры

Наименование функции:  
GetPchar  
Прототип функции:  
int\* romb::GetPchar()  
Данная функция возвращает координаты параметры пера (тип, ширина, цвет(RGB))

Наименование функции:  
GetBchar  
Прототип функции:  
int\* romb::GetBchar()  
Данная функция возвращает параметры кисти(тип, цвет)

Наименование функции:  
SetPStyle  
Прототип функции:  
void romb::SetPStyle(int new\_ptype, int new\_width, color new\_pcolor)  
Данная функция устанавливает новые параметры-атрибуты пера в классе.

Наименование функции:  
SetBStyle  
Прототип функции:  
void romb::SetBStyle(int new\_btype, color new\_bcolor)  
Данная функция устанавливает новые параметры-атрибуты кисти в классе

Наименование функции:  
draw  
Прототип функции:  
void romb::draw()  
Данная функция рисует фигуру с применением всех атрибутов класса

Наименование функции:  
load  
Прототип функции:  
void romb::load(char \*fname)  
Данная функция загружает данные о фигуре из файла, и проверяет является ли фигура ромбом, и не выходит ли она за границы окна, если какое-либо из этих условий не выполняется, то ошибка.

Наименование функции:  
save  
Прототип функции:  
void romb::save(char \*fname)  
Данная функция загружает данные в файл

Наименование функции:  
check\_borders  
Прототип функции:  
int romb::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\_figure  
Прототип функции:  
void romb::check\_figure(POINT \*ppt)  
Данная функция вычисляет длины всех сторон и коэффициенты. Сравнивает длины, если они не равны, то ошибка. Сравнивает коэффициенты противоположных сторон, если они не равны, то ошибка.

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 romb.h \*/**

#ifndef SQUARE

#define SQUARE

struct color { int r, g, b; };

class romb

{

private:

POINT t[4];

color pcolor, bcolor;

int ptype, btype;

int width;

public:

void SetPos(int new\_x, int new\_y);

void SetFigure(POINT\* new\_t);

void SetPStyle(int new\_ptype, int new\_width, color new\_pcolor);

void SetBStyle(int new\_btype, color new\_bcolor);

POINT\* GetCoord();

int\* GetPchar();

int\* GetBchar();

float slope(POINT a, POINT b);

void check\_figure(POINT \*ppt);

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

void draw();

void load(char \*fname);

void save(char \*fname);

};

#endif

**/\* 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>

#include "user.h"

void main()

{

setlocale(LC\_CTYPE, "Russian");

bool exit = false, showMenu = true;

char fin[30], fout[30], input;

romb sq;

RECT rt;

POINT\* k;

HWND hwnd = GetConsoleWindow();

while (!exit)

{

if (showMenu) { printMenu(); showMenu = false; }

input = \_getch();

switch (input)

{

case '1': //загрузка

{

GetClientRect(hwnd, &rt);

printf\_s("Введите имя файла: ");

scanf("%s", &fin);

try { sq.load(fin); sq.draw(); }

catch (int err) { errMsg(err); }

} break;

case '2': //сохранение

{

printf\_s("Введите имя файла: ");

scanf("%s", &fout);

try { sq.save(fout); }

catch (int err) { errMsg(err); }

} break;

case '3': //задание новой фигуры

{

sq.erase();

POINT tk[5];

int tside, tptype, tbtype, twidth;

color tpcol, tbcol;

bool OK = false;

while (!OK)

{

printf\_s("Задайте параметры фигуры координаты точек:\n");

scanf\_s("%d %d %d %d %d %d %d %d", &tk[0].x, &tk[0].y, &tk[1].x, &tk[1].y, &tk[2].x, &tk[2].y, &tk[3].x, &tk[3].y);

printf\_s("Задайте стиль границы (тип[0-6], толщина, цвет в формате RGB):\n");

scanf\_s("%d %d %d %d %d", &tptype, &twidth, &tpcol.r, &tpcol.g, &tpcol.b);

printf\_s("Задайте стиль заливки (тип[-2 - без заливки; -1 - сплошная][-2 - 5], цвет RGB):\n");

scanf\_s("%d %d %d %d", &tbtype, &tbcol.r, &tbcol.g, &tbcol.b);

try

{

tk[4] = tk[0];

GetClientRect(hwnd, &rt);

sq.check\_figure(tk);

sq.check\_borders(tk, rt.right, rt.bottom);

sq.SetFigure(tk);

sq.SetPStyle(tptype, twidth, tpcol);

sq.SetBStyle(tbtype, tbcol);

sq.draw();

OK = true;

}

catch (int err) { errMsg(err); }

}

} break;

case '4': //перемещение фигуры

{

int new\_x, new\_y;

printf\_s("Введите координаты перемещения(по Ох и по Оу):\n");

scanf\_s("%d%d", &new\_x, &new\_y);

try {

sq.SetPos(new\_x, new\_y);

sq.draw();

}

catch (int err) { errMsg(err); }

} break;

case '5': //показать фигуру

{ sq.draw();} break;

case 'm': { showMenu = true; } break; //показать меню

case 27: { exit = true; } break; //выход

}

}

**/\* Файл romb.cpp \*/**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <windowsx.h>

#include "romb.h"

#include <iostream>

//коды ошибок

#define OUTSIDE\_BORDERS 1

#define NEG\_VALUE 2

#define WRONG\_PARALLEL 3

#define IO\_ERR 5

#define TYPE\_ERR 6

#define RGB\_ERR 7

#define WRONG\_DLINA 8

HWND hwnd = GetConsoleWindow();

RECT rt;

//функция перемещения фигуры

void romb::SetPos(int new\_x, int new\_y)

{

int i;

GetClientRect(hwnd, &rt);

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

{

if (t[i].x + new\_x < 0 || t[i].x + new\_x > rt.right || t[i].y + new\_y < 0 || t[i].y + new\_y > rt.bottom) throw OUTSIDE\_BORDERS;

}

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

{

t[i].x += new\_x;

t[i].y += new\_y;

}

}

//функция присваивания значения новых координат вершин

void romb::SetFigure(POINT\* new\_t)

{

int i;

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

t[i].x = new\_t[i].x; t[i].y = new\_t[i].y;

}

}

//функция вычисляющая коэффициент наклона прямой

float romb::slope(POINT a, POINT b)

{

if ((b.x - a.x) != 0)

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

else return 3000;

}

//функция проверки фигуры(является ли ромбом)

void romb::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] || dl[i] == 0) throw WRONG\_DLINA;

}

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

{

if (koef[i] != koef[i + 2]) throw WRONG\_PARALLEL;

}

}

//функция проверки выхода фигуры за границы окна

void romb::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 || ppt[i].x<0 || ppt[i].y<0) throw OUTSIDE\_BORDERS;

}

}

//Получение координат фигуры

POINT\* romb::GetCoord()

{

return t;

}

//характеристик пера

int\* romb::GetPchar()

{

int a[5];

a[0] = pcolor.r;

a[1] = pcolor.g;

a[2] = pcolor.b;

a[3] = ptype;

a[4] = width;

return a;

}

//характеристик кисти

int\* romb::GetBchar()

{

int a[4];

a[0] = bcolor.r;

a[1] = bcolor.g;

a[2] = bcolor.b;

a[3] = btype;

return a;

}

//установка стиля пера

void romb::SetPStyle(int new\_ptype, int new\_width, color new\_pcolor)

{

if (new\_ptype < 0 || new\_ptype > 6) throw TYPE\_ERR;

if (new\_width < 0) throw NEG\_VALUE;

if (new\_pcolor.r < 0 || new\_pcolor.r > 255) throw RGB\_ERR;

if (new\_pcolor.g < 0 || new\_pcolor.g > 255) throw RGB\_ERR;

if (new\_pcolor.b < 0 || new\_pcolor.b > 255) throw RGB\_ERR;

ptype = new\_ptype;

width = new\_width;

pcolor = new\_pcolor;

}

//установка стиля кисти

void romb::SetBStyle(int new\_btype, color new\_bcolor)

{

if (new\_btype < -2 || new\_btype > 5) throw TYPE\_ERR;

if (new\_bcolor.r < 0 || new\_bcolor.r > 255) throw RGB\_ERR;

if (new\_bcolor.g < 0 || new\_bcolor.g > 255) throw RGB\_ERR;

if (new\_bcolor.b < 0 || new\_bcolor.b > 255) throw RGB\_ERR;

btype = new\_btype;

bcolor = new\_bcolor;

}

//рисование ромба

void romb::draw()

{

HDC hdc = GetDC(hwnd);

COLORREF BG = GetPixel(hdc, 10, 10);

HPEN pen = CreatePen(ptype, width, RGB(pcolor.r, pcolor.g, pcolor.b));

HBRUSH brush;

if (btype == -2) brush = GetStockBrush(NULL\_BRUSH);

else

if (btype == -1) brush = CreateSolidBrush(RGB(bcolor.r, bcolor.g, bcolor.b));

else brush = CreateHatchBrush(btype, RGB(bcolor.r, bcolor.g, bcolor.b));

SetBkColor(hdc, BG);

SelectPen(hdc, pen);

SelectBrush(hdc, brush);

Polygon(hdc, t, 4);

ReleaseDC(hwnd, hdc);

}

//загрузка фигуры из файла

void romb::load(char \*fname)

{

GetClientRect(hwnd, &rt);

POINT tk[5];

FILE \*fptr = fopen(fname, "r");

if (!fptr) throw IO\_ERR;

if (!fscanf(fptr, "%d %d %d %d %d %d %d %d; ", &tk[0].x, &tk[0].y, &tk[1].x, &tk[1].y, &tk[2].x, &tk[2].y, &tk[3].x, &tk[3].y)) throw IO\_ERR;

SetFigure(tk);

check\_borders(t, rt.right, rt.bottom);

tk[4] = tk[0];

check\_figure(tk);

int type, twidth;

color tcol;

if (!fscanf(fptr, "%d %d (%d, %d, %d) ", &type, &twidth, &tcol.r, &tcol.g, &tcol.b)) throw IO\_ERR;

SetPStyle(type, twidth, tcol);

if (!feof(fptr))

{

if (!fscanf(fptr, "%d (%d, %d, %d)", &type, &tcol.r, &tcol.g, &tcol.b)) throw IO\_ERR;

SetBStyle(type, tcol);

}

else SetBStyle(-2, tcol);

fclose(fptr);

}

//сохранение фигуры в файл

void romb::save(char \*fname)

{

FILE \*fptr = fopen(fname, "w");

if (!fptr) throw IO\_ERR;

fprintf(fptr, "%d %d %d %d %d %d %d %d; ", t[0].x, t[0].y, t[1].x, t[1].y, t[2].x, t[2].y, t[3].x, t[3].y);

fprintf(fptr, "%d %d (%d, %d, %d) ", ptype, width, pcolor.r, pcolor.g, pcolor.b);

if (btype != -2) fprintf(fptr, "%d (%d, %d, %d)", btype, bcolor.r, bcolor.g, bcolor.b);

fclose(fptr);

}

**/\* Файл user.h \*/**

#include <stdio.h>

#include "romb.h"

void printMenu()

{

printf\_s("Выберите действие:\n");

printf\_s("[1] - загрузить данные из файла\n");

printf\_s("[2] - сохранить данные в файл\n");

printf\_s("[3] - задать новые параметры фигуры\n");

printf\_s("[4] - переместить фигуру\n");

printf\_s("[5] - показать фигуру\n");

printf\_s("[m] - показать меню\n");

printf\_s("[esc] - выход из программы\n");

}

void errMsg(int err)

{

switch (err)

{

case 1: printf\_s("Ошибка: фигура за границами окна.\n"); break;

case 2: printf\_s("Ошибка: неверное значение толщины пера\n"); break;

case 3: printf\_s("Ошибка: противоположные стороны не параллельны.\n"); break;

case 4: printf\_s("Ошибка: файл не существует или содержит неверный формат данных.\n"); break;

case 5: printf\_s("Ошибка: неверно указан тип границы или заливки. \n"); break;

case 6: printf\_s("Ошибка: неверно задан цвет границы или заливки. \n"); break;

case 7: printf\_s("Ошибка: длины сторон не равны \n"); break;

}

}

# 7.Тесты

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результат | Примечания |
| 1 | 200 150 250 100 300 150 250 200; 0 5 (200, 200, 0) -1 (100, 100, 200) |  | Рисование ромба Загрузка из файла |
| 2 | 200 100 250 100 250 150 200 150 3 3 183 110 60 3 255 125 0 |  | Рисование ромба Непосредственное задание фигуры |
| 3 | -300 0 |  | Перемещение предыдущей фигуры влево на 300 пиксилей |
| 4 | 200 150 250 100 300 150 300 200 2 2 138 133 40 1 255 125 0 |  | Задание фигуры не являющейся ромбом |
| 5 | 200 150 250 100 300 150 300 200 2 2 138 133 40 1 -255 125 0 |  | Отрицательное значение в параметре цвета |

# 8. Результат работы программы

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