|  |  |  |
| --- | --- | --- |
| Министерство науки и высшего образования  Российской Федерации | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования | | |
| «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра прикладной математики | | |
|  | | |
| Лабораторная работа № 1 | | |
| по дисциплине «Объектно-ориентрованное программирование» | | |
|  | | |
| **Программирование графики** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИ-81 |
|  |  |
| Студенты: | Воронкина Дарья |
|  | Кондратьев Игорь |
|  |  |
| Преподаватели: | Лисицин Даниил Валерьевич |
|  | Еланцева Ирина Леонидовна |
|  | | |
| Новосибирск | | |
| 2019 | | |

**1.Условие задачи**

По предложенному преподавателем варианту разработать функции, рисующие следующие геометрические фигуры:

– незакрашенную фигуру (фигуру-контур),

– закрашенную фигуру,

– две вложенных одну в другую фигуры, внешняя фигура закрашена за исключением пространства внутренней фигуры.

Разработать программу, демонстрирующую выполнение указанных функций. Ввод параметров фигур (координат и др.), параметров рисуемых линий и закраски осуществлять из файлов (отдельно для каждого теста). Включить в программу проверки корректности данных (нулевой радиус окружности, нарушение неравенства треугольника и т.д.), включая проверки нахождения фигуры в пределах окна и вложенности двух фигур.

**2.Анализ задачи**

**Дано:** Один или два четырехугольника, координаты которых заданы в двумерной плоскости.

**Результат:** Изображение одного четырехугольника или двух вложенных друг в друга четырехугольников.

**Метод решения:** Решение заключается в выполнении необходимых проверок, вывода ошибок входных данных, в случае если они есть, вывода четырехугольника/четырехугольников на экран консоли.

**Проверка четырехугольника на выпуклость:** Четырехугольник называется выпуклым, если он расположен в одной полуплоскости относительно прямой, содержащей любую его сторону. Таким образом, для того чтобы проверить является ли четырехугольник выпуклым необходимо и достаточно провести прямые через каждые две соседние вершины и определить будут ли остальные вершины лежать в одной из полуплоскостей, образованных этой прямой. Если данное условие не выполняется, то четырехугольник не является выпуклым.

Пусть есть две прямая, проходящая через 2 точки (х1, у1) и (х2, у2), тогда уравнение этой прямой можно представить в виде:

тогда, подставив в это уравнение координаты х двух других точек и сравнив полученные значения с их координатами у можно судить о том, в каких полуплоскостях они находятся. Если рассчитанные величины больше – в левой (нижней) полуплоскости, если меньше – в правой (верхней) полуплоскости, если равны – то точка находится на прямой (некорректное условие для четырехугольника). Частный случай: если прямая имеет вид x=const, то необходимо сравнивать координаты х прямой и точек. Если х точек больше х прямой, то они в правой полуплоскости, если меньше – в левой, если они равны - точка находится на прямой.

**Проверка вложенности двух фигур:** Фигура вложена, если каждая вершина вложенной фигуры находится внутри первой фигуры.

**Проверка нахождения фигуры в окне:** Фигура находится в окне, если ее координаты х и у не превышают максимального значения X, Y, где максимальные значения являются размерами окна, а также являются больше нуля.

**Вывод ошибок:** В соответствии с кодом ошибки выводится сообщение на экран об ошибки.

**3.Структуры данных**

Внутреннее представление:

enum ERROR - поле, хранящее тип ошибки

{

INCORRECT\_DRAW\_TYPE, - ошибка: неправильно задан тип рисовки

INCORRECT\_BRUSH, - ошибка: неправильно задана кисть

INCORRECT\_PEN\_STYLE, - ошибка: неправильно задано перо

OUT\_FRAME, - ошибка: фигура выходит за рамки

NOT\_INCLUDED, - ошибка: фигуры не вложены

NOT\_CONVEX, - ошибка: фигура не является четырехугольником

THREE\_POINTS\_IN\_LINE, - ошибка: координаты фигура лежат на одной прямой

};

enum DRAW\_TYPE - поле, хранящее тип рисовки

{

CONTOUR, - не закрашенная фигура

SHADED, - закрашенная фигура

DONUT - вложенные фигуры

};

struct PEN - структура, хранящая информацию о пере

{

char name[16]; - название пера

int width; - ширина пера

COLORREF color; - цвет пера

};

struct BRUSH - структура, хранящая информацию о кисти

{

char name[16]; - название кисти

COLORREF color; - цвет кисти

};

struct Quadrangle - структура, хранящая информацию о четырехугольнике

{

DRAW\_TYPE type; - тип рисовки четырехугольника

POINT points[4]; - координаты четырехугольника

PEN qd\_pen; - структура, хранящая информацию о пере

BRUSH qd\_brush; - структура, хранящая информацию о кисти

};

**Внешнее представление:**

**Входные данные:**

**Входные данные находятся в файле “test.txt”**

1. Тип рисовки четырехугольника:

CONTOUR – рисование не закрашенного четырехугольника, с условиями описанными ниже

SHADED – рисование закрашенного четырехугольника, с условиями описанными ниже

DONUT – рисование двух вложенных друг в друга четырехугольников, с закрашенной внешней фигуры, за исключением пространства внутренней фигуры;

1. 4 попарно записанные координаты фигуры, по часовой стрелке;
2. Тип пера, принимающий значения :

|  |  |
| --- | --- |
| PS\_SOLID | Сплошная |
| PS\_DASH | Пунктирная |
| PS\_DOT | Штриховая |
| PS\_DASHDOT | Штрихпунктирная, одна точка на одну линию |
| PS\_DASHDOTDOT | Штрихпунктирная, две точки на одну линию |
| PS\_NULL | Невидимая |
| PS\_INSIDEFRAME | Линия, предназначенная |

Перо не принимает стоковые значения пера, так как достаточно указать тип PS-SOLID и цвет пера;

1. Ширина пера, целое значение;
2. Цвет пера в формате RGB;
3. Тип кисти, принимающий значения :

|  |  |
| --- | --- |
| Значение | Описание |
| BLACK\_BRUSH | Кисть черного цвета |
| WHITE\_BRUSH | Кисть белого цвета |
| GRAY\_BRUSH | Серая кисть |
| LTGRAY\_BRUSH | Светло-серая кисть |
| DKGRAY\_BRUSH | Темно-серая кисть |
| NULL\_BRUSH | Бесцветная кисть, которая ничего не закрашивает |
| HOLLOW\_BRUSH | Синоним ДЛЯ NULL\_BRUSH |
| |  |  | | --- | --- | | Стиль штриховки |  | | HS\_BDIAGONAL | | | HS\_CROSS | | | HS\_DIAGCROSS   |  | | --- | | HS\_FDIAGONAL | | HS\_HORIZONTAL | | HS\_VERTICAL | | | |  |

1. Цвет кисти в формате RGB
2. Пункты со 2 по 5 для второй фигуры, она же вложенная фигура.

Если тип рисовки указан CONTOUR, то программа учитывает входные данные с 1 по 5 пункт, игнорируя остальные входные данные. Если тип рисовки указан SHADED, то программа учитывает входные данные с 1 по 7 пункт, игнорируя остальные входные данные. Если тип рисовки указан DONUT, то программа учитывает входные данные с 1 по 8 пункт.

**Выходные данные:**

Один или два четырехугольника, нарисованных в консоли.

**4.Укрупненный алгоритм решения задачи**

Открыть файл test.txt

Инициализировать четырехугольники по данным из файла

Закрыть файл test.txt

Пока не нажат Esc

Нарисовать фигуру

**5.Структура программы**

Программа разбита на 2 модуля:

1. основной модуль, выполняющий запуск всех остальных.
2. модуль, содержащий все необходимые подпрограммы и типы данных, необходимые для работы программы

1 модуль:

Файлы: main.cpp

Функции: void main (); - основная функция, осуществляющая управление всеми остальными.

2 модуль:

Файлы: QD.h, QD.cpp

Функции:

DRAW\_TYPE StringToEnum(const char string[]); - функция, определяющая тип рисовки

COLORREF RGBToColor(int R, int G, int B); - функция, приводящая целые значения цветов в формат COLORREF

void DrawContour(HDC hdc, HWND hwnd, Quadrangle quad); - функция рисования не закрашенной фигуры

void DrawShaded(HDC hdc, HWND hwnd, Quadrangle quad); - функция рисования закрашенной фигуры

void DrawDonut(HDC hdc, HWND hwnd, Quadrangle quad, Quadrangle quad2); - функция рисования вложенных фигур

void Read\_Characteristics(Quadrangle\* quad, FILE\* file, Quadrangle\* quad2, HDC hdc); - ввод данных из файла file, в структуры четырехугольников quad, quad2, в зависимости от типа рисовки.

int StringToBrushHash(const char string[]); - функция, определяющая тип кисти

int StringToPenStyle(const char string[]); -функция, определяющая тип пера

void PrintError(ERROR error); - функция, выводящая ошибки

void CheckConvex(Quadrangle quad); - функция, проверяющая является ли фигура выпуклым четырехугольником

void CheckInFrame(HWND hwnd, Quadrangle qd); - функция, проверяющая лежит ли фигура в окне

void CheckIncluded(Quadrangle quad1, Quadrangle quad2); - функция, проверяющая вложены ли четырехугольники

bool IsInQuadrangle(POINT P1, POINT P2, POINT P3, POINT P4, POINT PTest); - функция проверяющая точки, для того, чтобы проверить вложенность четырехугольников

bool IsPoint(POINT k1, POINT k2, POINT k3, POINT k4); - функция, проверяющая точки, для того чтобы провекрить является четырехугольник выпуклым

**6.Код программы**

//Main.cpp

#include "Qd.h"

void main()

{

setlocale(LC\_ALL, "RUSSIAN");

FILE\* file = fopen("test.txt", "r");

Quadrangle quad, quad2;

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

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

Read\_Characteristics(&quad, file, &quad2, hdc);

do

{

Draw(hdc, hwnd, quad, quad2);

fclose(file);

} while (getch() != 27);

}

//QD.h

#pragma once

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <iostream>

#include <fstream>

#include <string>

#include <stdlib.h>

#include <conio.h>

#undef ERROR

using namespace std;

enum ERROR

{

INCORRECT\_DRAW\_TYPE,

INCORRECT\_BRUSH,

INCORRECT\_PEN\_STYLE,

OUT\_FRAME,

NOT\_INCLUDED,

NOT\_CONVEX,

THREE\_POINTS\_IN\_LINE,

};

enum DRAW\_TYPE

{

CONTOUR,

SHADED,

DONUT

};

struct PEN

{

char name[16];

int width;

COLORREF color;

};

struct BRUSH

{

char name[16];

COLORREF color;

};

struct Quadrangle

{

DRAW\_TYPE type;

POINT points[4];

PEN qd\_pen;

BRUSH qd\_brush;

};

DRAW\_TYPE StringToEnum(const char string[]);

COLORREF RGBToColor(int R, int G, int B);

void DrawContour(HDC hdc, HWND hwnd, Quadrangle quad);

void DrawShaded(HDC hdc, HWND hwnd, Quadrangle quad);

void DrawDonut(HDC hdc, HWND hwnd, Quadrangle quad, Quadrangle quad2);

void Read\_Characteristics(Quadrangle\* quad, FILE\* file, Quadrangle\* quad2, HDC hdc);

int StringToBrushHash(const char string[]);

int StringToPenStyle(const char string[]);

void PrintError(ERROR error);

void CheckConvex(Quadrangle quad);

void CheckInFrame(HWND hwnd, Quadrangle qd);

void CheckIncluded(Quadrangle quad1, Quadrangle quad2);

bool IsInQuadrangle(POINT P1, POINT P2, POINT P3, POINT P4, POINT PTest);

bool IsPoint(POINT k1, POINT k2, POINT k3, POINT k4);

//QD.cpp

#include "Qd.h"

DRAW\_TYPE StringToEnum(const char string[])

{

if (strcmp(string, "CONTOUR") == 0)

{

return CONTOUR;

}

else if (strcmp(string, "SHADED") == 0)

{

return SHADED;

}

else if (strcmp(string, "DONUT") == 0)

{

return DONUT;

}

else

{

throw INCORRECT\_DRAW\_TYPE;

}

}

int StringToBrushHash(const char string[])

{

if (strcmp(string, "HS\_BDIAGONAL") == 0)

{

return HS\_BDIAGONAL;

}

else if (strcmp(string, "HS\_CROSS") == 0)

{

return HS\_CROSS;

}

else if (strcmp(string, "HS\_DIAGCROSS") == 0)

{

return HS\_DIAGCROSS;

}

else if (strcmp(string, "HS\_FDIAGONAL") == 0)

{

return HS\_FDIAGONAL;

}

else if (strcmp(string, "HS\_HORIZONTAL") == 0)

{

return HS\_HORIZONTAL;

}

else if (strcmp(string, "HS\_VERTICAL") == 0)

{

return HS\_VERTICAL;

}

else

{

throw INCORRECT\_BRUSH;

}

}

int StringToPenStyle(const char string[])

{

if (strcmp(string, "PS\_SOLID") == 0)

{

return PS\_SOLID;

}

else if (strcmp(string, "PS\_DASH") == 0)

{

return PS\_DASH;

}

else if (strcmp(string, "PS\_DOT") == 0)

{

return PS\_DOT;

}

else if (strcmp(string, "PS\_DASHDOT") == 0)

{

return PS\_DASHDOT;

}

else if (strcmp(string, "PS\_DASHDOTDOT") == 0)

{

return PS\_DASHDOTDOT;

}

else if (strcmp(string, "PS\_NULL") == 0)

{

return PS\_NULL;

}

else if (strcmp(string, "PS\_INSIDEFRAME") == 0)

{

return PS\_INSIDEFRAME;

}

else

{

throw INCORRECT\_PEN\_STYLE;

}

}

COLORREF RGBToColor(int R, int G, int B)

{

return RGB(R, G, B);

}

void Read\_Characteristics(Quadrangle\* quad, FILE \*file, Quadrangle\* quad2, HDC hdc)

{

char type[15];

fscanf(file, "%s", &type);

try

{

quad->type = StringToEnum(type);

}

catch (ERROR error)

{

return PrintError(error);

}

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

{

fscanf(file, "%d %d", &(quad->points[i].x), &(quad->points[i].y));

}

fscanf(file, "%s", &(quad->qd\_pen.name));

fscanf(file, "%d", &(quad->qd\_pen.width));

int Red, Green, Blue;

fscanf(file, "%i %i %i", &Red, &Green, &Blue);

quad->qd\_pen.color = RGBToColor(Red, Green, Blue);

switch (quad->type)

{

case CONTOUR:

{

memcpy(quad->qd\_brush.name, "SOLID", 11);

quad->qd\_brush.color = GetBkColor(hdc);

break;

}

case SHADED:

{

fscanf(file, "%s", &quad->qd\_brush.name);

int Red, Green, Blue;

fscanf(file, "%i %i %i", &Red, &Green, &Blue);

quad->qd\_brush.color = RGBToColor(Red, Green, Blue);

break;

}

case DONUT:

{

fscanf(file, "%s", &quad->qd\_brush.name);

int Red, Green, Blue;

fscanf(file, "%i %i %i", &Red, &Green, &Blue);

quad->qd\_brush.color = RGBToColor(Red, Green, Blue);

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

fscanf(file, "%d %d", &(quad2->points[i].x), &(quad2->points[i].y));

fscanf(file, "%s", &quad2->qd\_pen.name);

fscanf(file, "%d", &quad2->qd\_pen.width);

int Red2, Green2, Blue2;

fscanf(file, "%i %i %i", &Red2, &Green2, &Blue2);

quad2->qd\_pen.color = RGBToColor(Red2, Green2, Blue2);

memcpy(quad2->qd\_brush.name, "SOLID", 11);

quad2->qd\_brush.color = GetBkColor(hdc);

break;

}

}

}

void DrawContour(HDC hdc, HWND hwnd, Quadrangle quad)

{

try

{

CheckConvex(quad);

CheckInFrame(hwnd, quad);

}

catch (ERROR error)

{

return PrintError(error);

}

//Creating pen

HPEN newPen;

try

{

newPen = CreatePen(StringToPenStyle(quad.qd\_pen.name), quad.qd\_pen.width, quad.qd\_pen.color);

}

catch (ERROR error)

{

return PrintError(error);

}

HPEN oldPen = SelectPen(hdc, newPen);

HBRUSH newBrush;

HBRUSH oldBrush;

//Creating brush

newBrush = GetStockBrush(NULL\_BRUSH);

oldBrush = SelectBrush(hdc, newBrush);

Polygon(hdc, quad.points, 4);

}

void DrawShaded(HDC hdc, HWND hwnd, Quadrangle quad)

{

try

{

CheckConvex(quad);

CheckInFrame(hwnd, quad);

}

catch (ERROR error)

{

return PrintError(error);

}

//Creating pen

HPEN newPen;

try

{

newPen = CreatePen(StringToPenStyle(quad.qd\_pen.name), quad.qd\_pen.width, quad.qd\_pen.color);

}

catch (ERROR error)

{

return PrintError(error);

}

HPEN oldPen = SelectPen(hdc, newPen);

HBRUSH newBrush;

HBRUSH oldBrush;

//Creating brush

newBrush;;

if (strcmp(quad.qd\_brush.name, "SOLID") == 0)

{

newBrush = CreateSolidBrush(quad.qd\_brush.color);

}

else

{

try

{

newBrush = CreateHatchBrush(StringToBrushHash(quad.qd\_brush.name), quad.qd\_brush.color);

}

catch (ERROR error)

{

return PrintError(error);

}

}

oldBrush = SelectBrush(hdc, newBrush);

Polygon(hdc, quad.points, 4);

}

void DrawDonut(HDC hdc, HWND hwnd, Quadrangle quad, Quadrangle quad2)

{

try

{

CheckConvex(quad);

CheckInFrame(hwnd, quad);

CheckConvex(quad2);

CheckIncluded(quad, quad2);

}

catch (ERROR error)

{

return PrintError(error);

}

//Creating pen

HPEN newPen;

try

{

newPen = CreatePen(StringToPenStyle(quad.qd\_pen.name), quad.qd\_pen.width, quad.qd\_pen.color);

}

catch (ERROR error)

{

return PrintError(error);

}

HPEN oldPen = SelectPen(hdc, newPen);

HBRUSH newBrush;

HBRUSH oldBrush;

//Creating brush

if (strcmp(quad.qd\_brush.name, "SOLID") == 0)

{

newBrush = CreateSolidBrush(quad.qd\_brush.color);

}

else

{

try

{

newBrush = CreateHatchBrush(StringToBrushHash(quad.qd\_brush.name), quad.qd\_brush.color);

}

catch (ERROR error)

{

return PrintError(error);

}

}

oldBrush = SelectBrush(hdc, newBrush);

Polygon(hdc, quad.points, 4);

//Creating pen

try

{

newPen = CreatePen(StringToPenStyle(quad2.qd\_pen.name), quad2.qd\_pen.width, quad2.qd\_pen.color);

}

catch (ERROR error)

{

return PrintError(error);

}

oldPen = SelectPen(hdc, newPen);

//Creating brush

newBrush = CreateSolidBrush(GetBkColor(hdc));

oldBrush = SelectBrush(hdc, newBrush);

Polygon(hdc, quad2.points, 4);

}

void PrintError(ERROR error)

{

switch (error)

{

case INCORRECT\_DRAW\_TYPE:

{

printf("Неверно задан тип фигуры\n");

break;

}

case INCORRECT\_BRUSH:

{

printf("Неверно задан тип кисти\n");

break;

}

case INCORRECT\_PEN\_STYLE:

{

printf("Неверно задан тип пера\n");

break;

}

case OUT\_FRAME:

{

printf("Координаты фигуры не входят в рамки окна\n");

break;

}

case NOT\_INCLUDED:

{

printf("Второй четырехугольник не вложен в первый\n");

break;

}

case NOT\_CONVEX:

{

printf("Координаты не удовлетворют условию выпуклого четырехугольника\n");

break;

}

case THREE\_POINTS\_IN\_LINE:

{

printf("В заданных кооринатах три точки лежат на одной прямой\n");

break;

}

}

\_getch();

}

void CheckConvex(Quadrangle quad)

{

if (!IsPoint(quad.points[0], quad.points[1], quad.points[2], quad.points[3])

|| !IsPoint(quad.points[1], quad.points[2], quad.points[3], quad.points[0])

|| !IsPoint(quad.points[2], quad.points[3], quad.points[0], quad.points[1])

|| !IsPoint(quad.points[3], quad.points[0], quad.points[1], quad.points[2]))

{

throw NOT\_CONVEX;

}

}

void CheckInFrame(HWND hwnd, Quadrangle qd)

{

RECT rt;

GetClientRect(hwnd, &rt);

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

{

if (qd.points[i].x > rt.right || qd.points[i].y > rt.bottom || qd.points[i].x < 0 || qd.points[i].y < 0)

{

throw OUT\_FRAME;

}

}

}

void CheckIncluded(Quadrangle quad1, Quadrangle quad2)

{

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

{

if (!IsInQuadrangle(quad1.points[0], quad1.points[1], quad1.points[2], quad1.points[3], quad2.points[i]))

{

throw NOT\_INCLUDED;

}

}

}

bool IsInQuadrangle(POINT P1, POINT P2, POINT P3, POINT P4, POINT PTest)

{

int a = (P1.x - PTest.x) \* (P2.y - P1.y) - (P2.x - P1.x) \* (P1.y - PTest.y);

int b = (P2.x - PTest.x) \* (P3.y - P2.y) - (P3.x - P2.x) \* (P2.y - PTest.y);

int c = (P3.x - PTest.x) \* (P4.y - P3.y) - (P4.x - P3.x) \* (P3.y - PTest.y);

int d = (P4.x - PTest.x) \* (P1.y - P4.y) - (P1.x - P4.x) \* (P4.y - PTest.y);

if ((a >= 0 && b >= 0 && c >= 0 && d >= 0) || (a <= 0 && b <= 0 && c <= 0 && d <= 0))

{

return true;

}

else

{

return false;

}

}

bool IsPoint(POINT k1, POINT k2, POINT k3, POINT k4)

{

if (k1.x - k2.x)

{

float k;

k = (float)(k1.y - k2.y) / (float)(k1.x - k2.x);

float b = k1.y - k \* k1.x;

float y1 = k \* k3.x + b;

float y2 = k \* k4.x + b;

if (y1 == k3.y || y2 == k4.y)

{

throw THREE\_POINTS\_IN\_LINE;

}

if ((y1 > (float)k3.y && y2 > (float)k4.y) || (y1 < (float)k3.y && y2 < (float)k4.y))

{

return true;

}

else

{

throw NOT\_CONVEX;

}

}

else

{

if (k3.x == k1.x || k4.x == k1.x)

{

throw THREE\_POINTS\_IN\_LINE;

}

if ((k1.x > k3.x && k2.x > k4.x) || (k1.x < k3.x && k2.x < k4.x))

{

return true;

}

else

{

throw NOT\_CONVEX;

}

}

}

**7. Тесты**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результаты | Примечания |
| 1 | CONTOUR  25 25  200 100  250 250  25 400  PS\_SOLID  5  255 155 0  HS\_CROSS  0 155 0 |  | Незакрашенная фигура выпуклого четырехугольника |
| 2 | SHADED  25 25  200 100  250 250  25 275  PS\_SOLID  5  0 0 255  SOLID  255 155 0 |  | Закрашенная фигура выпуклого четырехугольника |
| 3 | SHADED  25 25  200 100  250 250  25 275  PS\_DASHDOTDOT  1  255 255 255  HS\_CROSS  0 155 0 |  | Закрашенная фигура с штрихпунктирным контуром и закраской в клеточку |
| 4 | DONUT  25 25  250 100  250 250  25 455  PS\_SOLID  4  255 0 0  SOLID  0 155 0  75 100  150 150  150 200  50 200  PS\_SOLID  7  255 255 0 |  | Вложенные фигуры,внешняя фигура закрашена, за исключением внутренностей вложенной фигуры |
| 5 | DONUT  25 25  250 100  250 400  25 455  PS\_SOLID  4  255 0 0  HS\_DIAGCROSS  0 155 0  75 100  150 150  150 200  50 200  PS\_SOLID  7  255 255 0 |  | Вложенные фигуры, внешняя фигура закрашена штриховкой |
| 6 | DON  25 25  250 100  250 400  25 455  PS\_SOLID  4  255 0 0 | Неверно задан тип фигуры | Ошибка в задании типа рисовки |
| 7 | CONTOUR  25 25  25 100  25 400  25 455  PS\_SOLID  4  255 0 0 | В заданных координатах три точки лежат на одной прямой | Ошибка в задании фигуры |
| 8 | SHADED  25 25  200 100  250 100  25 275  PS\_DASH  1  255 255 255  SOLID  255 155 0 | Координаты не удовлетворяют условию выпуклого четырехугольника | Ошибка в задании фигуры |
| 9 | SHADED  25 25  200 100  250 250  25 500  PS\_DASH  1  255 255 255  SOLID  255 155 0 | Координаты фигуры не входят в рамки окна | Фигура не входит в рамки |
| 10 | DONUT  25 25  200 100  250 250  25 275  PS\_DASH  1  255 255 255  SOLID  255 155 0  50 150  150 150  150 200  50 400  PS\_SOLID  7  255 255 0 | Второй четырехугольник не вложен в первый | Не выполняются условия вложенности |
| 11 | DONUT  25 25  200 100  250 250  25 275  PS\_DASHASASA  1  255 255 255  SOLID  255 155 0  50 150  150 150  150 200  50 400  PS\_SOLID  7  255 255 0 | Неверно задан тип пера |  |

**8.Вывод**

Программа работает исправно на всех тестах.