Министерство науки и образования РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра прикладной математики

Лабораторная работа № 1

по Объектно-ориентированному программированию

на тему:

**Программирование графики**

Факультет: ФПМИ

Группа: ПМИ-82

Студенты: Матюкова У.П., Хайдаев К.Е.

Преподаватель: Еланцева И.Л., Неделько В.М.

Вариант: 7

Новосибирск

2011

1. **Постановка задачи**

Разработать функции, рисующие следующие геометрические фигуры:

– незакрашенный треугольник,

– закрашенную треугольник,

– два вложенных один в другой треугольника, внешняя фигура закрашена за исключением пространства внутренней фигуры.

Разработать программу, демонстрирующую выполнение указанных функций, обеспечить ввод из файлов (отдельно для каждого теста) параметров фигур (координат и др.), параметров рисуемых линий и закраски. Включить в программу проверки нахождения фигуры в пределах окна и вложенности двух фигур.

1. **Алгоритм**

(x1[0],y1[0]), (x1[1],y1[1]) (x1[2],y1[2]) - координаты первого треугольника

(x2[0],y2[0]), (x2[1],y2[1]) (x2[2],y2[2]) - координаты второго треугольника

1. bool CheckTriangle(float x[3], float y[3]) - проверка является ли фигура треугольником
2. bool square(float x1[3], float y1[3], float x2[], float y2[], int n) - проверка вложенности треугольников
3. bool CheckPole(float x[3], float y[3])– проверка принадлежности консольному окну
4. void input(FILE \*f, float x[3], float y[3])
5. void rgb\_color(FILE \*f, int Rgb[3], int Rgb1[3], int width[1])
6. int main()
7. float a, b, c;

bool k;

если(a + b > c && a + c > b && c + b > a)

k = true;

иначе false;

1. float p, a, b, c, O1, O2, O3, S[4];

p = (a + b + c) / 2; //периметр треугольника

p = (O1 + O2 + a) / 2; //периметр треугольника, состоящего из двух точек другого треугольника и стороны а

S[1] = sqrt(p\*(p - O1)\*(p - O2)\*(p - a));

p = (O2 + O3 + b) / 2; //периметр треугольника, состоящего из двух точек другого треугольника и стороны b

S[2] = sqrt(p\*(p - O2)\*(p - O3)\*(p - b));

p = (O3 + O1 + c) / 2; //периметр треугольника, состоящего из двух точек другого треугольника и стороны с

S[3] = sqrt(p\*(p - O3)\*(p - O1)\*(p - c));

если S[0] <= S[1] + S[2] + S[3] + 0.5 и S[0] >= S[1] + S[2] + S[3] - 0.5 и S[1]!= 0 и S[2] != 0 и S[3] != 0;

true;

иначе false;

1. bool k;

от i = 0 до i < 3

если x[i] < rt.right и y[i] < rt.bottom

k = true;

иначе false;

1. если f != 0

от i = 0 до i < 3

считать из f в x[i];

считать из f в y[i];

иначе Ошибка, файл не найден;

1. если f != 0

от i = 0 до i < 3

считать из f в Rgb[i];

от i = 0 до i < 3

считать из f в Rgb1[i];

считать из f в width[0];

иначе Ошибка, файл не найден;

1. если vibor = 1

если фигура является треугольником и принадлежит консольному окну, то

рисуем первый треугольник

иначе выход;

если vibor = 2

если фигура является треугольником и принадлежит консольному окну, то

закрашиваем второй треугольник

иначе выход;

если vibor = 3

если обе фигуры являются треугольниками и принадлежат консольному окну и первый треугольник вложен во второй или второй вложен в первый, то

рисуем два вложенных треугольника, причем область большей фигуры закрашена за исключением области меньшей

иначе выход;

1. **Текст программы**

ProverkaTriangle.h

bool CheckTriangle(float x[3], float y[3])

{

bool k;

float a, b, c;

a = sqrt((x[1] - x[0])\*(x[1] - x[0]) + (y[1] - y[0])\*(y[1] - y[0]));

b = sqrt((x[2] - x[1])\*(x[2] - x[1]) + (y[2] - y[1])\*(y[2] - y[1]));

c = sqrt((x[0] - x[2])\*(x[0] - x[2]) + (y[0] - y[2])\*(y[0] - y[2]));

if (a + b > c && a + c > b && c + b > a)

k = true;

else

return false;

return k;

}

ProverkaVlojenosti.h

bool square(float x1[3], float y1[3], float x2[], float y2[], int n)

{

float p, a, b, c, O1, O2, O3, S[4];

a = sqrt((x1[1] - x1[0])\*(x1[1] - x1[0]) + (y1[1] - y1[0])\*(y1[1] - y1[0]));

b = sqrt((x1[2] - x1[1])\*(x1[2] - x1[1]) + (y1[2] - y1[1])\*(y1[2] - y1[1]));

c = sqrt((x1[0] - x1[2])\*(x1[0] - x1[2]) + (y1[0] - y1[2])\*(y1[0] - y1[2]));

p = (a + b + c) / 2;

S[0] = sqrt(p\*(p - a)\*(p - b)\*(p - c));

O1 = sqrt((x2[n] - x1[0])\*(x2[n] - x1[0]) + (y2[n] - y1[0])\*(y2[n] - y1[0]));

O2 = sqrt((x2[n] - x1[1])\*(x2[n] - x1[1]) + (y2[n] - y1[1])\*(y2[n] - y1[1]));

O3 = sqrt((x2[n] - x1[2])\*(x2[n] - x1[2]) + (y2[n] - y1[2])\*(y2[n] - y1[2]));

p = (O1 + O2 + a) / 2;

S[1] = sqrt(p\*(p - O1)\*(p - O2)\*(p - a));

p = (O2 + O3 + b) / 2;

S[2] = sqrt(p\*(p - O2)\*(p - O3)\*(p - b));

p = (O3 + O1 + c) / 2;

S[3] = sqrt(p\*(p - O3)\*(p - O1)\*(p - c));

//cout << S[0] << endl << S[1] << endl << S[2] << endl << S[3] << endl;

if (S[0] <= S[1] + S[2] + S[3] + 0.5 && S[0] >= S[1] + S[2] + S[3] - 0.5 && S[1] != 0 && S[2] != 0 && S[3] != 0)

return true;

else

return false;

}

proverkaPole.h

bool CheckPole(float x[3], float y[3])

{

bool k = false;

GetClientRect(ind, &rt); //функция для определения размеров области окна

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

{

if (x[i] < rt.right && y[i] < rt.bottom)

k = true;

else

return false;

}

return k;

}

#include <windows.h>

#include <windowsx.h>

#include <stdio.h>

#include <conio.h>

#include <iostream>

#include <string>

#include <vector>

#include <cmath>

using namespace std;

HWND ind = GetConsoleWindow();// индентификатор окна

HDC context = GetDC(ind); //контекст отображения

RECT rt; //структура с полями left, top, right, bottom

//#include "proverkaPole.h"

#include "ProverkaTriangle.h"

#include "ProverkaVlojenosti.h"

#include "proverkaPole.h"

FILE \*f1, \*f2, \*f3, \*f4, \*f5;

//чтение координат треугольника из файла

void input(FILE \*f, float x[3], float y[3]) {

char sym;

if (f != NULL) {

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

fscanf\_s(f, "%c", &sym);

fscanf\_s(f, "%f", &x[i]);

fscanf\_s(f, "%c", &sym);

fscanf\_s(f, "%f", &y[i]);

fscanf\_s(f, "%c", &sym);

fscanf\_s(f, "%c", &sym);

}

}

else {

cout << "Ошибка, не удается открыть файл" << endl;

}

}

//чтение цветов и ширины пера из файла

void rgb\_color(FILE \*f, int Rgb[3], int Rgb1[3], int width[1]) {

char sym;

if (f != NULL) {

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

fscanf\_s(f, "%d", &Rgb[i]);

fscanf\_s(f, "%c", &sym);

//cout << Rgb[i]<<' ';

}

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

fscanf\_s(f, "%d", &Rgb1[i]);

fscanf\_s(f, "%c", &sym);

//cout << Rgb1[i] << ' ';

}

fscanf\_s(f, "%d", &width[0]);

}

else {

cout << "Ошибка, не удалось открыть файл" << endl;

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

string s1;

SetBkColor(context, RGB(0, 0, 0));

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

cout << "Graphics in Console Window." << endl;

GetClientRect(ind, &rt);

float x1[3], y1[3], a, b, c, p, x2[3], y2[3], n;

int rgb[3], rgb1[3];

int vibor, i;

int width[1];

do {

HPEN hPen, hOldPen; //Получение индетификатора пера

HBRUSH hBrush, hOldBrush;//Получение индетификатора кисти

cout << "выберете из 4: 1 = не закрашенный треугольник, 2 = закрашенный треугольник, 3 = вложенные фигуры, 4 - завершение программы." << endl;

cin >> vibor;

system("cls");//чистит консоль

switch (vibor)

{

case 1:

{

fopen\_s(&f1, "Koord.txt", "r");

fopen\_s(&f4, "RGB.txt", "r");

input(f1, x1, y1);

rgb\_color(f4, rgb, rgb1, width);

hPen = CreatePen(PS\_SOLID, width[0], RGB(rgb[0], rgb[1], rgb[2]));

hOldPen = SelectPen(context, hPen);

if (CheckPole(x1, y1) == false || CheckTriangle(x1, y1) == false)

{

cout << "Невозможно построить треугольник или он выходит за границы окна, введите данные корректно\n";

}

else {

system("cls");

do {

s1 = "Не закрашенный треугольник, чтобы вернуться к предыдущему выбору нажмите esc";

LPCSTR lpcstr = s1.c\_str();

TextOutA(context, 10, 10, lpcstr, strlen(lpcstr));

POINT treugolnik[4] = { { x1[0],y1[0] }, { x1[1],y1[1] }, { x1[2],y1[2] }, { x1[0],y1[0] } };

Polyline(context, treugolnik, 4);

} while (\_getch() != 27);

}

fclose(f1);

fclose(f4);

SelectPen(context, hOldPen);

DeletePen(hPen);

break;

}

case 2:

{

fopen\_s(&f1, "Koord.txt", "r");

fopen\_s(&f4, "RGB.txt", "r");

input(f1, x1, y1);

rgb\_color(f4, rgb, rgb1, width);

hPen = CreatePen(PS\_SOLID, width[0], RGB(rgb[0], rgb[1], rgb[2]));

hOldPen = SelectPen(context, hPen);

hBrush = CreateSolidBrush(RGB(rgb1[0], rgb1[1], rgb1[2]));

hOldBrush = SelectBrush(context, hBrush);

if (CheckPole(x1, y1) == false || CheckTriangle(x1, y1) == false)

{

cout << "Невозможно построить треугольник или он выходит за границы окна, введите данные корректно\n";

}

else {

system("cls");

do {

s1 = "Закрашенный треугольник, чтобы вернуться к предыдущему выбору нажмите esc";

LPCSTR lpcstr = s1.c\_str();

TextOutA(context, 10, 10, lpcstr, strlen(lpcstr));

POINT treugolnik[4] = { {x1[0],y1[0]},{x1[1],y1[1]},{x1[2],y1[2]},{x1[0],y1[0]} };

Polygon(context, treugolnik, 4);

} while (\_getch() != 27);

}

fclose(f1);

fclose(f4);

SelectPen(context, hOldPen);

SelectBrush(context, hOldBrush);

DeletePen(hPen);

DeleteBrush(hBrush);

break;

}

case 3:

{

fopen\_s(&f2, "Text.txt", "r");

fopen\_s(&f3, "Text1.txt", "r");

fopen\_s(&f5, "RGB1.txt", "r");

input(f2, x1, y1);

input(f3, x2, y2);

rgb\_color(f5, rgb, rgb1, width);

hPen = CreatePen(PS\_SOLID, width[0], RGB(rgb[0], rgb[1], rgb[2]));

hOldPen = SelectPen(context, hPen);

hBrush = CreateSolidBrush(RGB(rgb1[0], rgb1[1], rgb1[2]));

hOldBrush = SelectBrush(context, hBrush);

if (CheckPole(x1, y1) == false || CheckTriangle(x1, y1) == false || CheckPole(x2, y2) == false || CheckTriangle(x2, y2) == false)

{

cout << "Невозможно построить треугольник или он выходит за границы окна\n";

}

else {

if (square(x1, y1, x2, y2, 0) == false) {

cout << "Точка (x2[0], y2[0]) не лежит внутри треугольника" << endl;

}

else if (square(x1, y1, x2, y2, 1) == false && x2[0] != x2[1] && y2[0] != y2[1]) {

cout << "Точка (x2[1], y2[1]) не лежит внутри треугольника" << endl;

}

else if (square(x1, y1, x2, y2, 2) == false) {

cout << "Точка (x2[2], y2[2]) не лежит внутри треугольника или невозможно построить треугольник" << endl;

}

else {

s1 = "вложенные треугольники:";

system("cls");

do

{

POINT pt;

MoveToEx(context, x1[0], y1[0], &pt);

LineTo(context, x1[1], y1[1]);

LineTo(context, x1[2], y1[2]);

LineTo(context, x1[0], y1[0]);

MoveToEx(context, x2[0], y2[0], &pt);

LineTo(context, x2[1], y2[1]);

LineTo(context, x2[2], y2[2]);

LineTo(context, x2[0], y2[0]);

FloodFill(context, x2[2] + 5, y2[2] + 5, RGB(rgb[0], rgb[1], rgb[2]));

} while (\_getch() != 27);

}

}

fclose(f2);

fclose(f3);

fclose(f5);

SelectPen(context, hOldPen);

SelectBrush(context, hOldBrush);

DeletePen(hPen);

DeleteBrush(hBrush);

break;

}

case 4:

{

ReleaseDC(ind,context);

exit(0);// завершение программы

}

}

} while (vibor != 1 || vibor != 2 || vibor != 3);

system("pause");

}

1. **Результаты тестирования**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | Результат | Назначение |
| 1 | (100 100)(200 200)(300 300) | Фигура не треугольник. Не рисуем | Проверка является ли фигура треугольником |
| 2 | (10 10)(10 20)(10 30) | Фигура не треугольник. Не закрашиваем | Проверка является ли фигура треугольником |
| 3 | (500;50)(60;80)(50;50)  (10;10)(10;20)(50;300) | Фигура не принадлежит консольному окну | Проверка принадлежности консольному окну |
| 4 | {10 10} {120 150} {250 130} |  | Рисование  фигуры |
| 5 | {10 10} {120 150} {250 130} |  | Закрашивание фигуры |
| 6 | {10 10} {620 280} {10 280}  {15 15} {350 200} {15 260} |  | Проверка вложенности |
| 7 | {10 10} {620 280} {10 280}  {10 15} {330 100} {10 250} | Фигуры не вложены | Проверка вложенности |