Министерство образования и науки РФ

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра теоретической и прикладной информатики

Практическое задание № 1

по дисциплине «Объектно-ориентированное программирование»

**Факультет:** ПМИ

**Группа:**  ПМ-92

**Студенты:**  Иванов В., Попов А.

**Преподаватель:**  Неделько В.М.

Новосибирск

2020

***Постановка задачи***

**ЛР1**

1. Разработать функции, рисующие следующие геометрические фигуры:

– незакрашенный **треугольник**,

– закрашенный **треугольник**,

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

2. 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.

3. Подготовить текстовый файл с разработанной программой, оттранслировать, собрать и выполнить программу с учетом требований операционных систем и программных оболочек, в которых эта программа выполняется. При необходимости исправить ошибки и вновь повторить технологический процесс решения задачи.

**ЛР2**

Модифицировать программу, разработанную в лабораторной работе №1, так чтобы в ней был определен класс, реализующий понятие геометрической фигуры (**ЗАКРАШЕННЫЙ ТРЕУГОЛЬНИК)** в графической системе.

2.1. Определить ответственность класса. Учесть, что ввод и вывод данных, а также реакцию на возникающие при работе с функциями класса ошибки следует производить вне функций класса.

2.2. Определить атрибуты, необходимые для реализации класса. Поместить атрибуты в закрытую часть описания.

2.3. Определить функции, необходимые для реализации класса. Выделить интерфейс класса и поместить его в открытую часть описания.

Включить в разработанный класс функции

– устанавливающие и изменяющие геометрические и графические характеристики фигуры (set-функции);

– возвращающие геометрические и графические характеристики фигуры (get-функции);

– рисующую фигуру на экране;

– изменяющую положение фигуры на экране;

– обеспечивающие сохраняемость объекта: сохранение набора атрибутов объекта класса в файле и считывание его из файла.

2.4. Разработать функцию, демонстрирующую поведение класса. Поместить реализацию класса в один файл, а демонстрационную функцию – в другой. Обеспечить необходимые межмодульные связи. Поместить все файлы в один проект Visual С++.

2.5. Учесть возможность возникновения ошибок (нулевой радиус, нарушение неравенства треугольника, другие ошибки задания фигуры, ошибки работы с графикой и др.)

***Текст программы***

**main.cpp**

#include "triangle.h"

int main(void) {

triangle tr;

char k[20];

int q = 1;

HWND hwnd = GetConsoleWindow();

HDC hdc = GetDC(hwnd);

printf("Name of the file: ");

scanf("%s", k);

system("cls");

try {

try {

tr.load(k, hwnd);

}

catch (int error) { triangle::processing(error); }

do {

system("cls");

printf("Choose an option: **\n**1. Demonstration **\n**2. Figure editor **\n**3. Load figure **\n**4. Save figure**\n**5. Quit**\n**");

char ch;

int c;

scanf("%d", &c);

switch (c)

{

case 1:

{

ch = '0';

int dir;

system("cls");

printf("Use W, D, S, A to move a figure.**\n**Press Q to quit.**\n**Press any key to continue.**\n**");

\_getch();

system("cls");

while (ch != 'q')

{

int dir;

triangle::fill(hwnd, hdc);

tr.paint(hdc);

ch = \_getch();

switch (ch)

{

case 'w': {

dir = 1;

break; }

case 'd': {

dir = 2;

break; }

case 's': {

dir = 3;

break; }

case 'a': {

dir = 4;

break; }

default: {

dir = 5;

break; }

}

tr.move(hwnd, dir, 5);

triangle::fill(hwnd, hdc);

}

break;

}

case 2: {

int d = 100, c1, c2, c3;

point u;

color g;

system("cls");

while (d != 0) {

printf("Edit:**\n**1. Points of the external triangle**\n**2. Points of the internal triangle**\n**3. Select the type**\n**4. Select the external width**\n**5. Select the internal width**\n**6. Select the color of border**\n**7. Select the color of triangle::fill**\n**0. Quit**\n**");

scanf("%d", &d);

system("cls");

switch (d)

{

case 1: {

printf("First point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_a1(hwnd, u);

printf("**\n**Second point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_b1(hwnd, u);

printf("**\n**Third point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_c1(hwnd, u);

system("cls");

break; }

case 2: {

printf("First point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_a2(hwnd, u);

printf("**\n**Second point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_b2(hwnd, u);

printf("**\n**Third point coodinates: ");

scanf("%i %i", &c1, &c2);

u.x = c1;

u.y = c2;

tr.set\_c2(hwnd, u);

system("cls");

break; }

case 3: {

printf("**\n**Select type: ");

scanf("%i", &c1);

tr.set\_type(c1);

system("cls");

break; }

case 4: {

printf("**\n**Select width: ");

scanf("%i", &c1);

tr.set\_l\_width1(c1);

system("cls");

break; }

case 5: {

printf("**\n**Select width: ");

scanf("%i", &c1);

tr.set\_l\_width2(c1);

system("cls");

break; }

case 6: {

scanf("%i %i %i %i", &c1, &c2, &c3);

g.r = c1;

g.g = c2;

g.b = c3;

tr.set\_linecol(g);

system("cls");

break; }

case 7: {

scanf("%i %i %i %i", &c1, &c2, &c3);

g.r = c1;

g.g = c2;

g.b = c3;

tr.set\_introcol(g);

system("cls");

break; }

case 0: {

d = 0;

system("cls");

break; }

}

}

system("cls");

break;

}

case 3: {

char k1[20];

system("cls");

printf("Insert the name of a source file: ");

scanf("%s", k1);

tr.load(k1, hwnd);

system("cls");

break; }

case 4: {

char k2[20];

system("cls");

printf("Insert the name of a source file: ");

scanf("%s", k2);

try {

tr.save(k2);

system("cls");

}

catch (int error) { triangle::processing(error); }

break; }

case 5: {

q = 0;

break; }

}

} while (q != 0);

}

catch (int error) { triangle::processing(error); }

}

**triangle.cpp**

#include "triangle.h"

void triangle::set\_a1(HWND hwnd, const point &n\_a1) {

if (!triangle::reconwindow(hwnd, n\_a1)) throw 1;

else {

a1.x = n\_a1.x;

a1.y = n\_a1.y;

}

}

void triangle::set\_b1(HWND hwnd, const point &n\_b1) {

if (!triangle::reconwindow(hwnd, n\_b1)) throw 1;

else {

b1.x = n\_b1.x;

b1.y = n\_b1.y;

}

}

void triangle::set\_c1(HWND hwnd, const point &n\_c1) {

point pt[3];

pt[0].x = a1.x;

pt[0].y = a1.y;

pt[1].x = b1.x;

pt[1].y = b1.y;

pt[2].x = n\_c1.x;

pt[2].y = n\_c1.y;

if (!triangle::reconwindow(hwnd, n\_c1)) throw 1;

else {

if (!triangle::checkpoint(pt)) throw 2;

else {

c1.x = n\_c1.x;

c1.y = n\_c1.y;

}

}

}

void triangle::set\_a2(HWND hwnd, const point &n\_a2) {

if (!triangle::reconwindow(hwnd, n\_a2)) throw 1;

else {

a2.x = n\_a2.x;

a2.y = n\_a2.y;

}

}

void triangle::set\_b2(HWND hwnd, const point &n\_b2) {

if (!triangle::reconwindow(hwnd, n\_b2)) throw 1;

else {

b2.x = n\_b2.x;

b2.y = n\_b2.y;

}

}

void triangle::set\_c2(HWND hwnd, const point &n\_c2) {

point pt[3];

pt[0].x = a2.x;

pt[0].y = a2.y;

pt[1].x = b2.x;

pt[1].y = b2.y;

pt[2].x = n\_c2.x;

pt[2].y = n\_c2.y;

point ptt[6];

ptt[0].x = a1.x;

ptt[0].y = a1.y;

ptt[1].x = b1.x;

ptt[1].y = b1.y;

ptt[2].x = c1.x;

ptt[2].y = c1.y;

ptt[3].x = a2.x;

ptt[3].y = a2.y;

ptt[4].x = b2.x;

ptt[4].y = b2.y;

ptt[5].x = n\_c2.x;

ptt[5].y = n\_c2.y;

if (!triangle::reconwindow(hwnd, n\_c2)) throw 1;

else {

if (!triangle::checkpoint(pt)) throw 2;

else {

if (!triangle::checkencl(ptt)) throw 6;

else {

c2.x = n\_c2.x;

c2.y = n\_c2.y;

}

}

}

}

void triangle::set\_type(const int &n\_type) {

if ((n\_type > 3) || (n\_type < 1)) throw 3;

else {

type = n\_type;

}

}

void triangle::set\_l\_width1(const int &n\_l\_width1) {

if ((n\_l\_width1 < 1) || (n\_l\_width1 > 10)) throw 4;

else l\_width1 = n\_l\_width1;

}

void triangle::set\_l\_width2(const int &n\_l\_width2) {

if ((n\_l\_width2 < 1) || (n\_l\_width2 > 10)) throw 4;

else l\_width2 = n\_l\_width2;

}

void triangle::set\_linecol(const color &n\_linecol) {

if ((n\_linecol.r > 255) || (n\_linecol.r < 0) ||

(n\_linecol.g > 255) || (n\_linecol.g < 0) ||

(n\_linecol.b > 255) || (n\_linecol.b < 0)) throw 5;

else {

linecol.r = n\_linecol.r;

linecol.g = n\_linecol.g;

linecol.b = n\_linecol.b;

}

}

void triangle::set\_introcol(const color &n\_introcol) {

if ((n\_introcol.r > 255) || (n\_introcol.r < 0) ||

(n\_introcol.g > 255) || (n\_introcol.g < 0) ||

(n\_introcol.b > 255) || (n\_introcol.b < 0)) throw 5;

else {

introcol.r = n\_introcol.r;

introcol.g = n\_introcol.g;

introcol.b = n\_introcol.b;

}

}

void triangle::load(const char\* fname, HWND hwnd) {

FILE\* F;

F = fopen(fname, "r");

if (F == NULL)

throw 0;

else {

int a1x, a1y, b1x, b1y, c1x, c1y, a2x, a2y, b2x, b2y, c2x, c2y,

t, lw1, lw2,

lr, lg, lb, ir, ig, ib;

point n\_a1, n\_b1, n\_c1, n\_a2, n\_b2, n\_c2;

color line, intro;

fscanf(F, "%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i", &a1x, &a1y, &b1x, &b1y, &c1x, &c1y, &a2x, &a2y, &b2x, &b2y, &c2x, &c2y, &t, &lw1, &lw2, &lr, &lg, &lb, &ir, &ig, &ib);

fclose(F);

n\_a1.x = a1x;

n\_a1.y = a1y;

n\_b1.x = b1x;

n\_b1.y = b1y;

n\_c1.x = c1x;

n\_c1.y = c1y;

n\_a2.x = a2x;

n\_a2.y = a2y;

n\_b2.x = b2x;

n\_b2.y = b2y;

n\_c2.x = c2x;

n\_c2.y = c2y;

line.r = lr;

line.g = lg;

line.b = lb;

intro.r = ir;

intro.g = ig;

intro.b = ib;

set\_a1(hwnd, n\_a1);

set\_b1(hwnd, n\_b1);

set\_c1(hwnd, n\_c1);

set\_a2(hwnd, n\_a2);

set\_b2(hwnd, n\_b2);

set\_c2(hwnd, n\_c2);

set\_linecol(line);

set\_introcol(intro);

set\_type(t);

set\_l\_width1(lw1);

set\_l\_width2(lw2);

}

fclose(F);

}

void triangle::save(const char\* fname) {

FILE\* F;

F = fopen(fname, "w");

if (F == NULL)

throw 0;

else {

fprintf(F, "%i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i %i", a1.x, a1.y, b1.x, b1.y, c1.x, c1.y, a2.x, a2.y, b2.x, b2.y, c2.x, c2.y, type, l\_width1, l\_width2, linecol.r, linecol.g, linecol.b, introcol.r, introcol.g, introcol.r);

}

fclose(F);

}

void triangle::move(HWND hwnd, const int &dir, const int &l) {

point x1, y1, z1, x2, y2, z2;

switch (dir) {

case 1: {

x1.y = a1.y - l;

x1.x = a1.x;

y1.y = b1.y - l;

y1.x = b1.x;

z1.y = c1.y - l;

z1.x = c1.x;

x2.y = a2.y - l;

x2.x = a2.x;

y2.y = b2.y - l;

y2.x = b2.x;

z2.y = c2.y - l;

z2.x = c2.x;

break; }

case 2: {

x1.y = a1.y;

x1.x = a1.x + l;

y1.y = b1.y;

y1.x = b1.x + l;

z1.y = c1.y;

z1.x = c1.x + l;

x2.y = a2.y;

x2.x = a2.x + l;

y2.y = b2.y;

y2.x = b2.x + l;

z2.y = c2.y;

z2.x = c2.x + l;

break; }

case 3: {

x1.y = a1.y + l;

x1.x = a1.x;

y1.y = b1.y + l;

y1.x = b1.x;

z1.y = c1.y + l;

z1.x = c1.x;

x2.y = a2.y + l;

x2.x = a2.x;

y2.y = b2.y + l;

y2.x = b2.x;

z2.y = c2.y + l;

z2.x = c2.x;

break; }

case 4: {

x1.y = a1.y;

x1.x = a1.x - l;

y1.y = b1.y;

y1.x = b1.x - l;

z1.y = c1.y;

z1.x = c1.x - l;

x2.y = a2.y;

x2.x = a2.x - l;

y2.y = b2.y;

y2.x = b2.x - l;

z2.y = c2.y;

z2.x = c2.x - l;

break; }

case 5: {

x1.y = a1.y;

x1.x = a1.x;

y1.y = b1.y;

y1.x = b1.x;

z1.y = c1.y;

z1.x = c1.x;

x2.y = a2.y;

x2.x = a2.x;

y2.y = b2.y;

y2.x = b2.x;

z2.y = c2.y;

z2.x = c2.x;

break; }

}

if (triangle::reconwindow(hwnd, x1) && triangle::reconwindow(hwnd, y1) && triangle::reconwindow(hwnd, z1) && triangle::reconwindow(hwnd, x2) && triangle::reconwindow(hwnd, y2) && triangle::reconwindow(hwnd, z2))

{

set\_a1(hwnd, x1);

set\_b1(hwnd, y1);

set\_c1(hwnd, z1);

set\_a2(hwnd, x2);

set\_b2(hwnd, y2);

set\_c2(hwnd, z2);

}

}

void triangle::paint(HDC hdc) {

switch (type) {

case 1: {

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN hOldPen = SelectPen(hdc, Pen1);

POINT ppt[4] = { {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

Polyline(hdc, ppt, 4);

SelectPen(hdc, hOldPen);

DeletePen(Pen1);

break; }

case 2: {

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN hPen = SelectPen(hdc, Pen1);

HBRUSH Brush1 = CreateSolidBrush(RGB(introcol.r, introcol.g, introcol.b));

HBRUSH hBrush = SelectBrush(hdc, Brush1);

POINT ppt1[4] = { {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

Polygon(hdc, ppt1, 4);

SelectBrush(hdc, hBrush);

DeleteBrush(Brush1);

SelectPen(hdc, hPen);

DeletePen(Pen1); }

case 3: {

HPEN Pen1 = CreatePen(0, l\_width1, RGB(linecol.r, linecol.g, linecol.b));

HPEN Pen2 = CreatePen(0, l\_width2, RGB(linecol.r, linecol.g, linecol.b));

HPEN hPen = SelectPen(hdc, Pen1);

HBRUSH Brush1 = CreateSolidBrush(RGB(introcol.r, introcol.g, introcol.b));

HBRUSH Brush2 = GetStockBrush(BLACK\_BRUSH);

HBRUSH hBrush = SelectBrush(hdc, Brush1);

POINT ppt1[4] = { {a1.x, a1.y}, {b1.x, b1.y}, {c1.x , c1.y}, {a1.x, a1.y} };

POINT ppt2[4] = { {a2.x, a2.y}, {b2.x, b2.y}, {c2.x , c2.y}, {a2.x, a2.y} };

Polygon(hdc, ppt1, 4);

SelectBrush(hdc, hBrush);

DeleteBrush(Brush1);

SelectPen(hdc, hPen);

DeletePen(Pen1);

hPen = SelectPen(hdc, Pen2);

hBrush = SelectBrush(hdc, Brush2);

Polygon(hdc, ppt2, 4);

SelectPen(hdc, hPen);

DeletePen(Pen1); }

}

}

**triangle.h**

#ifndef TRIANGLE\_H\_

#define TRIANGLE\_H\_

#include <windows.h>

#include <windowsx.h>

#include <iostream>

#include <stdio.h>

#include <conio.h>

struct point { int x; int y; };

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

class triangle {

private:

point a1, b1, c1, a2, b2, c2;

int type, l\_width1, l\_width2;

color linecol, introcol;

public:

//triangle (point a,point b) {

//

//

//}

void load(const char\* fname, HWND hwnd);

void save(const char\* fname);

void move(HWND hwnd, const int &dir, const int &l);

void paint(HDC hdc);

inline point get\_a1() const{ return a1; }

inline point get\_b1() const{ return b1; }

inline point get\_c1() const{ return c1; }

inline point get\_a2() const{ return a2; }

inline point get\_b2() const{ return b2; }

inline point get\_c2() const{ return c2; }

inline int get\_type() const{ return type; }

inline int get\_l\_width1() const{ return l\_width1; }

inline int get\_l\_width2() const{ return l\_width2; }

inline color get\_linecol() const{ return linecol; }

inline color get\_introcol() const{ return introcol; }

void set\_a1(HWND hwnd, const point &n\_a1);

void set\_b1(HWND hwnd, const point &n\_b1);

void set\_c1(HWND hwnd, const point &n\_c1);

void set\_a2(HWND hwnd, const point &n\_a2);

void set\_b2(HWND hwnd, const point &n\_b2);

void set\_c2(HWND hwnd, const point &n\_c2);

void set\_type(const int &n\_type);

void set\_l\_width1(const int &n\_l\_width1);

void set\_l\_width2(const int &n\_l\_width2);

void set\_linecol(const color &n\_linecol);

void set\_introcol(const color &n\_introcol);

static int reconwindow(HWND hwnd, const point &a);

static float init\_point(const point to[], const int &t1, const int &t2, const int &t3);

static int checkencl(const point to[6]);

static int checkpoint(const point to[3]);

static void processing(const int &error);

static void fill(HWND hwnd, HDC hdc);

};

#endif

**external.cpp**

#include "triangle.h"

int triangle::reconwindow(HWND hwnd, const point &a) {

RECT rect;

GetClientRect(hwnd, &rect);

if ((a.x > 0) && (a.y > 0) && (a.x < rect.right) && (a.y < rect.bottom)) return 1;

else return 0;

}

float triangle::init\_point(const point to[], const int &t1, const int &t2, const int &t3) {

float res;

res = (to[t3].x - to[t1].x) \* (to[t2].y - to[t1].y) - (to[t3].y - to[t1].y) \* (to[t2].x - to[t1].x);

return res;

}

int triangle::checkencl(const point to[6]) {

if (((triangle::init\_point(to, 0, 1, 3) < 0) && (triangle::init\_point(to, 0, 1, 4) < 0) && (triangle::init\_point(to, 0, 1, 5) < 0) &&

(triangle::init\_point(to, 1, 2, 3) < 0) && (triangle::init\_point(to, 1, 2, 4) < 0) && (triangle::init\_point(to, 1, 2, 5) < 0) &&

(triangle::init\_point(to, 2, 0, 3) < 0) && (triangle::init\_point(to, 2, 0, 4) < 0) && (triangle::init\_point(to, 2, 0, 5) < 0)) ||

((triangle::init\_point(to, 0, 1, 3) > 0) && (triangle::init\_point(to, 0, 1, 4) > 0) && (triangle::init\_point(to, 0, 1, 5) > 0) &&

(triangle::init\_point(to, 1, 2, 3) > 0) && (triangle::init\_point(to, 1, 2, 4) > 0) && (triangle::init\_point(to, 1, 2, 5) > 0) &&

(triangle::init\_point(to, 2, 0, 3) > 0) && (triangle::init\_point(to, 2, 0, 4) > 0) && (triangle::init\_point(to, 2, 0, 5) > 0)))

return 1;

else return 0;

}

int triangle::checkpoint(const point to[3]) {

if (((to[1].x - to[0].x) \* (to[2].y - to[0].y)) == ((to[1].y - to[0].y) \* (to[2].x - to[0].x)))return 0;

else return 1;

}

void triangle::processing(const int &error) {

switch (error) {

case 0: {

printf("**\n**Reading/writing error!");

\_getch();

break; }

case 1: {

printf("**\n**Window is too small to contain the triangle!");

\_getch();

break; }

case 2: {

printf("**\n**Not a triangle!");

\_getch();

break; }

case 3: {

printf("**\n**Wrong type!");

\_getch();

break; }

case 4: {

printf("**\n**Wrong width of a line!");

\_getch();

break; }

case 5: {

printf("**\n**Color error!");

\_getch();

break; }

case 6: {

printf("**\n**Figures are'nt nested!");

\_getch();

break; }

}

}

void triangle::fill(HWND hwnd, HDC hdc) {

HPEN bp = CreatePen(1, 1, RGB(0, 0, 0));

HBRUSH bb = GetStockBrush(BLACK\_BRUSH);

HPEN pen = SelectPen(hdc, bp);

HBRUSH brush = SelectBrush(hdc, bb);

RECT rect;

GetClientRect(hwnd, &rect);

Rectangle(hdc, -1, -1, rect.right + 1, rect.bottom + 1);

SelectPen(hdc, bp);

DeletePen(pen);

}

***Тестирование***

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Назначение |
| 1 | 20 5 300 15 150 200 25 10 270 20 150 100 1 2 1 255 0 0 0 255 0 | Проверка на корректность перемещения незакрашенной фигуры |
| 2? | 20 5 300 15 150 200 25 10 270 20 150 100 2 2 1 255 0 0 0 255 0 | Проверка на корректность перемещения закрашенной фигуры |
| 3 | 20 5 300 15 150 200 25 10 270 20 150 100 3 2 1 255 0 0 0 255 0 | Проверка на корректность перемещения вложенных фигур |
| 4? | 400 5 **700** 15 150 200 25 10 270 20 150 100 3 2 1 255 0 0 0 255 0 | Фигура превышает размеры окна |
| 5 | **100 100 50 50 200 200** 25 10 270 20 150 100 3 2 1 255 0 0 0 255 0 | Фигура не является треугольником |
| 6 | 20 5 300 15 150 200 25 10 270 20 150 100 **4** 2 1 255 0 0 0 255 0 | Недопустимый тип фигуры |
| 7? | 220 5 300 150 150 200 25 10 270 20 150 100 3 2 1 255 0 0 0 255 0 | Фигуры не вложенные |
| 8 | 20 5 300 15 150 200 25 10 270 20 150 100 3 2 1 **320 400 0** 0 255 0 | Недопустимый цветовой диапазон |