

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт  информационных технологий** | **Кафедра информационных систем** |

|  |  |  |
| --- | --- | --- |
| **Основная образовательная программа 09.03.02 «Информационные системы и технологии»**  **Отчет по дисциплине «Объектно-ориентированное программирование на языке высокого уровня»** | | |
| **по лабораторной работе №3** | | |
|  | | |
|  | | |
| **Проверил**  **ассистент** |  | **Михайлуца Н.А.** |
|  |  |  |
| **Выполнил**  **студент группы ИДБ-22-06** |  | **Мустафаева П.М.** |

ОГЛАВЛЕНИЕ

[ЗАДАНИЕ 1 3](#_Toc147596591)

[1.1 ОПИСАНИЕ 3](#_Toc147596592)

[1.2 ТАБЛИЦА 1 ИСПОЛЬЗОВАННЫЕ ПЕРЕМЕННЫЕ 3](#_Toc147596593)

[1.3 ПРОГРАММА 3](#_Toc147596594)

[ЗАДАНИЕ 2 9](#_Toc147596595)

[2.1 ОПИСАНИЕ 9](#_Toc147596596)

[2.2 ТАБЛИЦА 2 ИСПОЛЬЗОВАННЫЕ ПЕРЕМЕННЫЕ 9](#_Toc147596597)

[2.3 ПРОГРАММА 9](#_Toc147596598)

[2.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ 15](#_Toc147596599)

[ЗАКЛЮЧЕНИЕ 16](#_Toc147596600)

# **ЗАДАНИЕ 1**

## **1.1 ОПИСАНИЕ**

В программе Win32->Win32Project в функции WndProc вставить обработчик события

case WM\_CREATE: { break;}

В нем создать две фигуры типа окружности и прямоугольника и положить их в общее глобальное хранилище, применяя контейнер vector. (Использовать следует конструктор с необходимыми параметрами для каждого типа)

Далее в функции WndProc, в обработке события WM\_PAINT, между строками

hdc = BeginPaint(hWnd, &ps);

и

EndPaint(hWnd, &ps);

последовательно проходя по хранимым в контейнере vector данными, осуществить рисование окружности и прямоугольника.

## **1.2 ТАБЛИЦА 1 ИСПОЛЬЗОВАННЫЕ ПЕРЕМЕННЫЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя переменной** | **Класс** | **Тип** | **Смысл** |
| i | Локальный | int | Итерации цикла |
| m\_left | Локальный | int | Координаты слева |
| m\_top | Локальный | int | Координаты сверху |
| m\_right | Локальный | int | Координаты справа |
| m\_bottom | Локальный | int | Координаты снизу |
| my\_x | Локальный | int | Координаты по оси x |
| my\_y | Локальный | int | Координаты по оси y |
| my\_radius | Локальный | int | Значение радиуса |

## **1.3 ПРОГРАММА**

#include "framework.h"

#include <vector>

#include "laba32.h"

#define MAX\_LOADSTRING 100

using namespace std;

HINSTANCE hInst;

WCHAR szTitle[MAX\_LOADSTRING];

WCHAR szWindowClass[MAX\_LOADSTRING];

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

class Shape {

public:

Shape() {}

Shape(Shape& obj) {}

virtual Shape\* copy() {

return new Shape(\*this);

}

virtual void Draw(HDC& hdc) {}

virtual ~Shape() {}

};

class MyRectangle : public Shape {

private:

int m\_left;

int m\_top;

int m\_right;

int m\_bottom;

public:

MyRectangle() {}

MyRectangle(int left, int top, int right, int bottom)

{

m\_left = left;

m\_top = top;

m\_right = right;

m\_bottom = bottom;

}

void Draw(HDC& hdc)

{

Rectangle(hdc, m\_left, m\_top, m\_right, m\_bottom);

}

};

class MyCircle : public Shape {

private:

int my\_x;

int my\_y;

int my\_radius;

public:

MyCircle() {}

MyCircle(int x, int y, int radius) {

my\_x = x;

my\_y = y;

my\_radius = radius;

}

void Draw(HDC& hdc)

{

Ellipse(hdc, my\_x - my\_radius, my\_y - my\_radius, my\_x + my\_radius, my\_y + my\_radius);

}

};

vector<Shape\*> Shapes;

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LABA32, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LABA32));

MSG msg;

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LABA32));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LABA32);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance;

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_CREATE:

Shapes.push\_back(new MyRectangle(300, 350, 500, 200));

Shapes.push\_back(new MyCircle(750, 270, 80));

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

for (int i = 0; i < Shapes.size(); ++i)

{

Shapes[i]->Draw(hdc);

}

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

for (int i = 0; i < Shapes.size(); ++i) {

delete Shapes[i];

}

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

**1.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

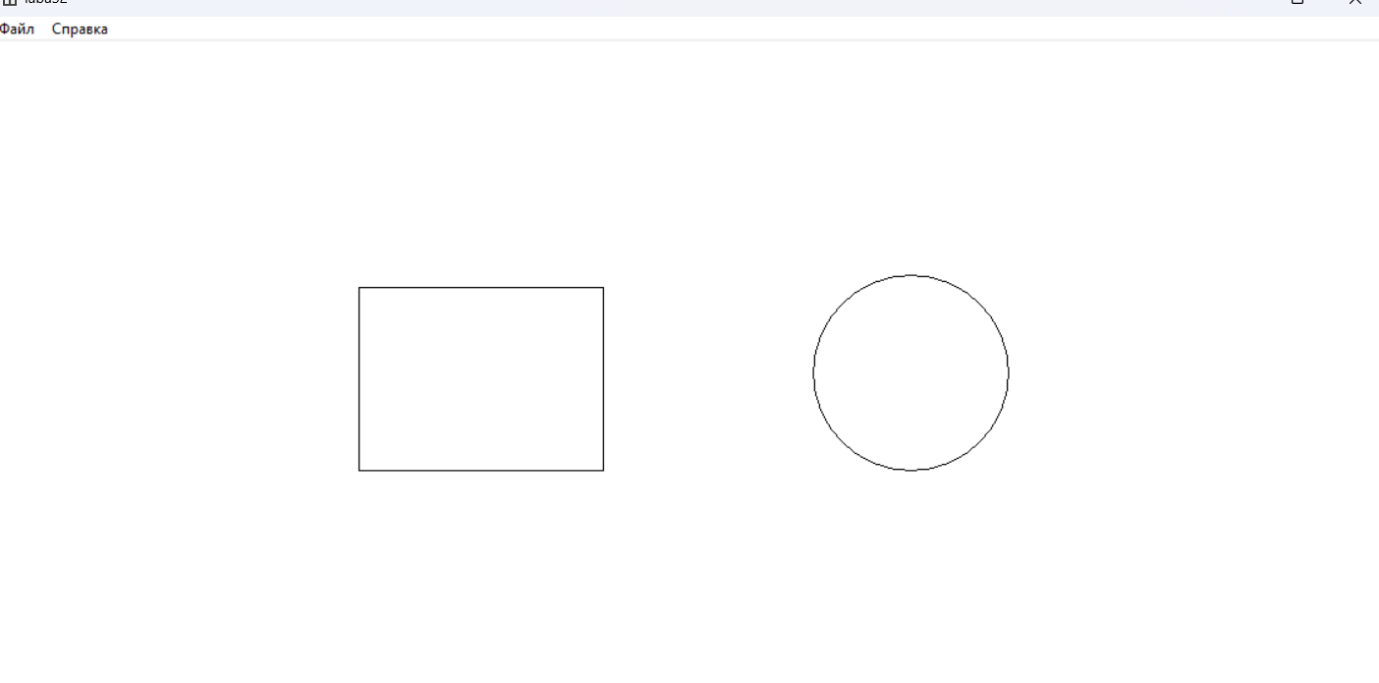


Рис. 1. Результат выполнения программы 1

# **ЗАДАНИЕ 2**

## **2.1 ОПИСАНИЕ**

Создать Win32Project (не консольное приложение).

Задать класс Vector2D для работы с плоскими векторами и точками. Реализовать в этом классе алгебраические операции для полноценного использования этого класса при визуализации плоских геометрических фигур и их преобразований. (использовать дружественные операторные функции, например: Vector2D operator+(const Vector2D v1, const Vector2D v2); не забыть реализовать оператор присваивания).

По обработке сообщения WM\_PAINT осуществить построение 9 окружностей (Ellipse) (3 на 3), используя для задания сдвига класс Vector2D (т. е. функция Ellipse должна будет принимать параметры, задаваемые через объекты типа Vector2D). (Например, исходная точка для рисования окружности Vector2D p(10,20); первый сдвиг Vector2D p(30,0); следующая точка p1=p+v; и т. д. в цикле)

## **2.2 ТАБЛИЦА 2 ИСПОЛЬЗОВАННЫЕ ПЕРЕМЕННЫЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя переменной** | **Класс** | **Тип** | **Смысл** |
| i | Локальный | int | Итерации цикла |

## **2.3 ПРОГРАММА**

#include "framework.h"

#include <vector>

#include "laba32.h"

using namespace std;

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

class Vector2D

{

private:

int\* x;

int\* y;

public:

Vector2D(int x, int y) {

this->x = new int(x);

this->y = new int(y);

}

Vector2D& operator=(const Vector2D& obj) {

if (this == &obj) return \*this;

CopyData(obj);

return \*this;

}

friend Vector2D operator+(const Vector2D& obj1, const Vector2D& obj2) {

return Vector2D((\*obj2.x + \*obj1.x), (\*obj2.y + \*obj1.y));

}

friend Vector2D operator-(const Vector2D& obj1, const Vector2D& obj2) {

return Vector2D((\*obj2.x - \*obj1.x), (\*obj2.y - \*obj1.y));

}

void CopyData(const Vector2D& obj) {

this->x = new int(\*obj.x);

this->y = new int(\*obj.y);

}

int getx() {

return \*x;

}

int gety() {

return \*y;

}

~Vector2D() {

delete x;

delete y;

}

};

vector < Vector2D\*> points;

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

LoadStringW(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadStringW(hInstance, IDC\_LABA32, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LABA32));

MSG msg;

// Цикл основного сообщения:

while (GetMessage(&msg, nullptr, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int)msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LABA32));

wcex.hCursor = LoadCursor(nullptr, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW + 1);

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LABA32);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI\_SMALL));

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

//

// ФУНКЦИЯ: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

HBRUSH hb = CreateSolidBrush(RGB(255, 80, 100));

static HPEN hpen = CreatePen(PS\_SOLID, 3, RGB(0, 0, 0));

HGDIOBJ oldb = SelectObject(hdc, hb);

SelectObject(hdc, hpen);

Vector2D p = Vector2D(0, 0);

for (int i = 0; i < points.size(); i++) {

p = p + \*points[i];

Ellipse(hdc, p.getx() - 45, p.gety() - 45, p.getx() + 45, p.gety() + 45);

}

EndPaint(hWnd, &ps);

}

break;

case WM\_CREATE:

{

points.push\_back(new Vector2D(100, 100));

points.push\_back(new Vector2D(100, 0));

points.push\_back(new Vector2D(100, 000));

points.push\_back(new Vector2D(-200, 100));

points.push\_back(new Vector2D(100, 0));

points.push\_back(new Vector2D(100, 0));

points.push\_back(new Vector2D(-200, 100));

points.push\_back(new Vector2D(100, 0));

points.push\_back(new Vector2D(100, 0));

break;

}

case WM\_DESTROY:

for (int i = 0; i < points.size(); i++) {

delete points[i];

}

points.~vector();

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Обработчик сообщений для окна "О программе".

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}

## **2.4 РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

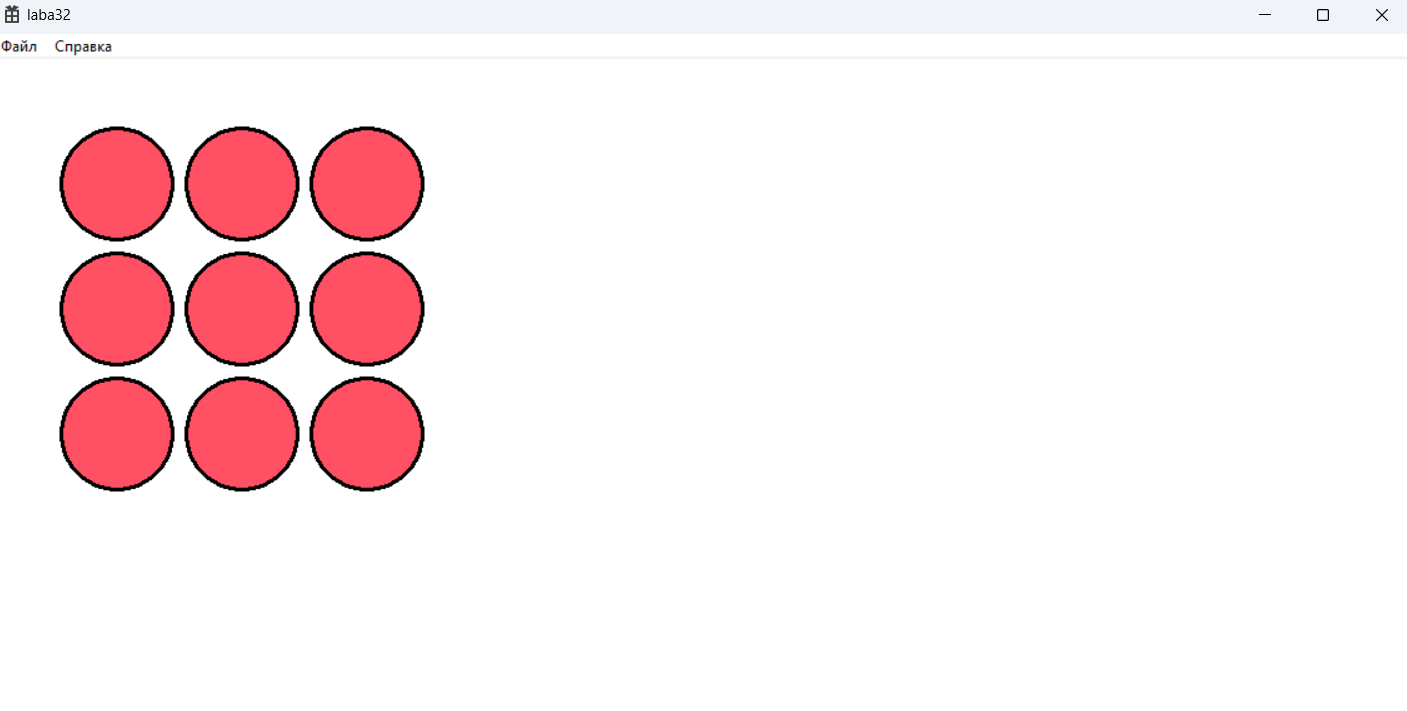


Рис. 2. Результат выполнения кода 2

# **ЗАКЛЮЧЕНИЕ**

В ходе лабораторной работы были реализованы возможности создания нескольких геометрических фигур и их визуализацию с использованием класса Vector2D.