Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №2

з дисципліни
«ООП»

Виконав:

Студент 2-го курсу групи ІМ-13
Нестеров Дмитро Васильович
номер у списку групи: 17

Перевірив:

Порєв Віктор Миколайович

Київ 2022

Мета:

Мета роботи – отримати вміння та навички використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм на основі класів C++, запрограмувавши простий графічний редактор в об’єктно-орієнтованому стилі.

Завдання:

1. Створити у середовищі MS Visual Studio C++ проект типу Windows Desktop Application з ім’ям Lab2.
2. Написати вихідний текст програми згідно варіанту завдання.
3. Скомпілювати вихідний текст і отримати виконуваний файл програми.
4. Перевірити роботу програми. Налагодити програму.
5. Проаналізувати та прокоментувати результати та вихідний текст програми.
6. Оформити звіт.

для 17-го студента у списку (Ж = 17) буде:

статичний масив для Shape (17 mod 3 = 2) обсягом 117 об’єктів

"гумовий" слід (17 mod 4 = 1) – суцільна лінія червоного кольору

прямокутник:

ввід від центру до одного з кутів (17 mod 2 = 1)

чорний контур з кольоровим заповненням (17 mod 5 = 2)

колір заповнення - сірий (17 mod 6 = 5)

еліпс:

по двом протилежним кутам охоплюючого прямокутника (17 mod 2 = 1)

чорний контур еліпсу без заповнення (17 mod 5 = 2)

позначка поточного типу об’єкту: в заголовку вікна (17 mod 2 = 1)

Вихідні тексти файлів:

Lab2.h

```
#pragma once

#include "resource.h"
```

Lab2.cpp

```
#include "framework.h"
#include "Lab2.h"
#include "shape_editor.h"

ShapeObjectsEditor Dima;

#define MAX_LOADSTRING 100

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);

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_LAB2, szWindowClass, MAX_LOADSTRING);
    MyRegisterClass(hInstance);

    if (!InitInstance(hInstance, nCmdShow))
    {
        return FALSE;
    }

    HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC_LAB2));

    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_LAB2));
    wcex.hCursor = LoadCursor(nullptr, IDC_ARROW);
    wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
    wcex.lpszMenuName = MAKEINTRESOURCEW(IDC_LAB2);
    wcex.lpszClassName = szWindowClass;
    wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_SMALL));

    return RegisterClassExW(&wcex);
}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)
{
    hInst = hInstance; // Store instance handle in our global variable

    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_LBUTTONDOWN: //натиснуто ліву кнопку миші у клієнтській частині вікна
        Dima.OnLBdown(hWnd);
        break;
    case WM_LBUTTONUP: //відпущено ліву кнопку миші у клієнтській частині вікна
        Dima.OnLBup(hWnd);
        break;
    case WM_MOUSEMOVE: //пересунуто мишу у клієнтській частині вікна
        Dima.OnMouseMove(hWnd);
        break;
    case WM_PAINT: //потрібно оновлення зображення клієнтської частині вікна
        Dima.OnPaint(hWnd);
        break;

    case WM_COMMAND:
    {
        int wmId = LOWORD(wParam);
        // Parse the menu selections:
        switch (wmId)
        {
```

```
        case ID_32771:
            Dima.StartPointEditor(hWnd); //початок вводу точкових об’єктів
            break;
        case ID_32772:
            Dima.StartLineEditor(hWnd); //початок вводу об’єктів-ліній
            break;
        case ID_32773:
            Dima.StartRectEditor(hWnd); //початок вводу прямокутників
            break;
        case ID_32774:
            Dima.StartEllipseEditor(hWnd); //початок вводу еліпсів
            break;
        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_DESTROY:
        PostQuitMessage(0);
        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;
}
```

editor.h

```
#pragma once
#include "framework.h"


class Editor {
public:
       virtual void OnLBdown(HWND) = 0;
```

```
        virtual void OnLBup(HWND) = 0;
        virtual void OnMouseMove(HWND) = 0;
        virtual void OnPaint(HWND) = 0;
};
```

shape_editor.h

```
#pragma once
class ShapeObjectsEditor
{

public:
        ShapeObjectsEditor(void);
        ~ShapeObjectsEditor();
        void StartPointEditor(HWND);
        void StartLineEditor(HWND);
        void StartRectEditor(HWND);
        void StartEllipseEditor(HWND);
        void OnLBdown(HWND);
        void OnLBup(HWND);
        void OnMouseMove(HWND);
        void OnPaint(HWND);
};
```

shape_editor.cpp

```
#include "framework.h"
#include "shape_editor.h"
#include "shape.h"
#include "editor.h"
#include "ShapeEditor.h"
#include "point_editor.h"
#include "line_editor.h"
#include "rect_editor.h"
#include "ellipse_editor.h"

ShapeEditor* pse = NULL;

ShapeObjectsEditor::ShapeObjectsEditor(void) {}
ShapeObjectsEditor::~ShapeObjectsEditor() {}
void ShapeObjectsEditor::StartPointEditor(HWND hWnd) {
        if (pse) delete pse;
        pse = new PointEditor;
        SetWindowText(hWnd, L"Режим вводу крапка");
}

void ShapeObjectsEditor::StartLineEditor(HWND hWnd) {
        if (pse) delete pse;
        pse = new LineEditor;
        SetWindowText(hWnd, L"Режим вводу лінія");
}

void ShapeObjectsEditor::StartRectEditor(HWND hWnd) {
        if (pse) delete pse;
        pse = new RectEditor;
        SetWindowText(hWnd, L"Режим вводу прямокутник");
}

void ShapeObjectsEditor::StartEllipseEditor(HWND hWnd) {
```

```
        if (pse) delete pse;
        pse = new EllipseEditor;
        SetWindowText(hWnd, L"Режим вводу елліпс");
}

void ShapeObjectsEditor::OnLBdown(HWND hWnd) {
        if (pse) pse->OnLBdown(hWnd);
}

void ShapeObjectsEditor::OnLBup(HWND hWnd) {
        if (pse) pse->OnLBup(hWnd);
}

void ShapeObjectsEditor::OnMouseMove(HWND hWnd) {
        if (pse) pse->OnMouseMove(hWnd);
}

void ShapeObjectsEditor::OnPaint(HWND hWnd) {
        if (pse) pse->OnPaint(hWnd);
}
```

shape.h

```
#pragma once
class Shape {
protected:
        long xs1, ys1, xs2, ys2;
public:
        void SetStart(long xStart, long yStart);
        void SetEnd(long xEnd, long yEnd);
        long getXStart(), getXEnd(), getYStart(), getYEnd();
        virtual void Show(HDC) = 0;
};
```

shape.cpp

```
#include "framework.h"
#include "shape.h"

void Shape::SetStart(long xStart, long yStart)
{
        xs1 = xStart;
        ys1 = yStart;
}

void Shape::SetEnd(long xEnd, long yEnd)
{
        xs2 = xEnd;
        ys2 = yEnd;
}

long Shape::getXStart()
{
        return xs1;
}

long Shape::getXEnd()
{
        return xs2;
}

long Shape::getYStart()
```

```
{
	return ys1;
}

long Shape::getYEnd()
{
	return ys2;
}
```

point_shape.h

```
#pragma once
#include "shape.h"

class PointShape : public Shape {
public:
	void Show(HDC);
};
```

point_shape.cpp

```
#include "framework.h"
#include "point_shape.h"


void PointShape::Show(HDC hdc) {
	SetPixel(hdc, xs2, ys2, 0x00000000);
}
```

line_shape.h

```
#pragma once
#include "shape.h"

class LineShape : public Shape {
public:
	void Show(HDC);
};
```

line_shape.cpp

```
#include "framework.h"
#include "line_shape.h"

void LineShape::Show(HDC hdc) {
	MoveToEx(hdc, xs1, ys1, NULL);
	LineTo(hdc, xs2, ys2);
}
```

rect_shape.h

```
#pragma once
#include "shape.h"

class RectShape : public Shape {
public:
	void Show(HDC);
};
```

rect_shape.cpp

```
#include "framework.h"
#include "rect_shape.h"

HBRUSH hBrush, hBrushOld;

void RectShape::Show(HDC hdc) {
	hBrush = (HBRUSH)CreateSolidBrush(RGB(192, 192, 192));
	hBrushOld = (HBRUSH)SelectObject(hdc, hBrush);

	long xStart = xs2 - (2 * (xs2 - xs1));	//до будь-якого кута
	long yStart = ys2 - (2 * (ys2 - ys1));

	Rectangle(hdc, xStart, yStart, xs2, ys2);

	SelectObject(hdc, hBrushOld);	DeleteObject(hBrush);
}
```

ellipse_shape.h

```
#pragma once
#include "shape.h"

class EllipseShape : public Shape {
public:
	void Show(HDC);
};
```

ellipse_shape.cpp

```
#include "framework.h"
#include "ellipse_shape.h"

void EllipseShape::Show(HDC hdc) {
	Arc(hdc, xs1, ys1, xs2, ys2, 0, 0, 0, 0);
}
```

ShapeEditor.h

```
#pragma once

#include "editor.h"
#include "shape.h"

#define MY_SHAPE_ARRAY_SIZE 117

extern Shape* pcshape[];

class ShapeEditor : public Editor {
protected:
	POINT pt;
	int index = 0;
	void GetPossibleIndex();
	bool isPainting = false;
	virtual void GetShape() = 0;
public:
	ShapeEditor(void);
	void OnLBdown(HWND); // Я не робив його віртуальним бо він однаковий для всіх 4
нащадків
```

```
	void OnLBup(HWND);  // теж не робив бо зміг зробити однаковим за домпомогою віртуального метода GetShape()
	virtual void OnMouseMove(HWND) = 0;
	void OnPaint(HWND);
};
```

ShapeEditor.cpp

```
#include "ShapeEditor.h"

Shape* pcshape[MY_SHAPE_ARRAY_SIZE];


ShapeEditor::ShapeEditor(void) { };

void ShapeEditor::OnLBdown(HWND hWnd) {
	isPainting = true;
	GetCursorPos(&pt);
	ScreenToClient(hWnd, &pt);
	GetPossibleIndex();
	GetShape();
	pcshape[index]->SetEnd(pt.x, pt.y);
	pcshape[index]->SetStart(pt.x, pt.y);
};

void ShapeEditor::OnLBup(HWND hWnd) {
	GetCursorPos(&pt);
	ScreenToClient(hWnd, &pt);
	pcshape[index]->SetEnd(pt.x, pt.y);
	InvalidateRect(hWnd, NULL, TRUE);
	isPainting = false;
};

void ShapeEditor::GetPossibleIndex() {
	for (int i = 0; i < MY_SHAPE_ARRAY_SIZE; i++)
		if (!pcshape[i]) {
			index = i;
			break;
		}
};

void ShapeEditor::OnPaint(HWND hWnd)
{
	PAINTSTRUCT ps;
	HDC hdc;
	hdc = BeginPaint(hWnd, &ps);
	for (int i = 0; i < MY_SHAPE_ARRAY_SIZE; i++)
		if (pcshape[i])
			pcshape[i]->Show(hdc);
		else break;
	EndPaint(hWnd, &ps);
}
```

point_editor.h

```
#pragma once
class PointEditor : public ShapeEditor {
	void GetShape();
	void OnMouseMove(HWND);
};
```

point_editor.cpp

```
#include "ShapeEditor.h"
#include "point_shape.h"
#include "point_editor.h"


void PointEditor::OnMouseMove(HWND hWnd) {
}

void PointEditor::GetShape() {
	pcshape[index] = new PointShape;
};
```

line_editor.h

```
#pragma once

class LineEditor : public ShapeEditor {
	void GetShape();
	void OnMouseMove(HWND);
};
```

line_editor.cpp

```
#include "ShapeEditor.h"
#include "line_shape.h"
#include "line_editor.h"


void LineEditor::OnMouseMove(HWND hWnd) {
	if (!isPainting) return;
	HPEN hPenOld, hPen;
	HDC hdc;
	hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання
	SetROP2(hdc, R2_NOTXORPEN);
	hPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 0));
	hPenOld = (HPEN)SelectObject(hdc, hPen); //Малюються лінії "гумового" сліду
попереднього розташування курсору

	MoveToEx(hdc, pcshape[index]->getXStart(), pcshape[index]->getYStart(), NULL);
	LineTo(hdc, pcshape[index]->getXEnd(), pcshape[index]->getYEnd());

	GetCursorPos(&pt);
	ScreenToClient(hWnd, &pt);
	pcshape[index]->SetEnd(pt.x, pt.y); //координати поточної точки курсору
	MoveToEx(hdc, pcshape[index]->getXStart(), pcshape[index]->getYStart(), NULL);
	LineTo(hdc, pcshape[index]->getXEnd(), pcshape[index]->getYEnd()); //Малюються лінії
"гумового" сліду для поточного розташування курсору

	SelectObject(hdc, hPenOld);
	DeleteObject(hPen);
	ReleaseDC(hWnd, hdc); //закриваємо контекст вікна
}

void LineEditor::GetShape() {
	pcshape[index] = new LineShape;
};
```

rect_editor.h

```
#pragma once

class RectEditor : public ShapeEditor {
	void OnMouseMove(HWND);
	void GetShape();
};
```

rect_editor.cpp

```
#include "ShapeEditor.h"
#include "rect_shape.h"
#include "rect_editor.h"

void RectEditor::OnMouseMove(HWND hWnd) {
	if (!isPainting) return;
	HPEN hPenOld, hPen;
	HDC hdc;
	hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання
	SetROP2(hdc, R2_NOTXORPEN);
	hPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 0));
	hPenOld = (HPEN)SelectObject(hdc, hPen); //Малюються лінії "гумового" сліду
попереднього розташування курсору
	long xStart = pcshape[index]->getXEnd() - (2 * (pcshape[index]->getXEnd() -
pcshape[index]->getXStart()));
	long yStart = pcshape[index]->getYEnd() - (2 * (pcshape[index]->getYEnd() -
pcshape[index]->getYStart()));
	Rectangle(hdc, xStart, yStart, pcshape[index]->getXEnd(), pcshape[index]->getYEnd());

	GetCursorPos(&pt);
	ScreenToClient(hWnd, &pt);
	pcshape[index]->SetEnd(pt.x, pt.y); //координати поточної точки курсору
	xStart = pcshape[index]->getXEnd() - (2 * (pcshape[index]->getXEnd() -
pcshape[index]->getXStart()));	//до правого верхнього кута
	yStart = pcshape[index]->getYEnd() - (2 * (pcshape[index]->getYEnd() -
pcshape[index]->getYStart()));
	Rectangle(hdc, xStart, yStart, pcshape[index]->getXEnd(), pcshape[index]->getYEnd());
	//Малюються лінії "гумового" сліду для поточного розташування курсору

	SelectObject(hdc, hPenOld);
	DeleteObject(hPen);
	ReleaseDC(hWnd, hdc); //закриваємо контекст вікна
}

void RectEditor::GetShape() {
	pcshape[index] = new RectShape;
};
```

ellipse_editor.h

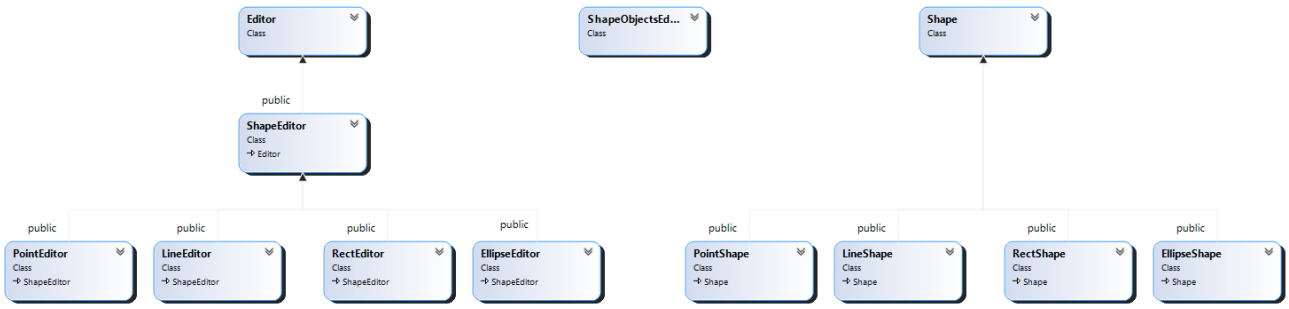
```
#pragma once

class EllipseEditor : public ShapeEditor {
	void GetShape();
	void OnMouseMove(HWND);
};
```

ellipse _editor.cpp

```
#include "ShapeEditor.h"
#include "ellipse_shape.h"
#include "ellipse_editor.h"
```

```
void EllipseEditor::OnMouseMove(HWND hWnd) {
	if (!isPainting) return;
	HPEN hPenOld, hPen;
	HDC hdc;
	hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання
	SetROP2(hdc, R2_NOTXORPEN);
	hPen = CreatePen(PS_SOLID, 1, RGB(255, 0, 0));
	hPenOld = (HPEN)SelectObject(hdc, hPen); //Малюються лінії "гумового" сліду
попереднього розташування курсору
	Ellipse(hdc, pcshape[index]->getXStart(), pcshape[index]->getYStart(),
pcshape[index]->getXEnd(), pcshape[index]->getYEnd());

	GetCursorPos(&pt);
	ScreenToClient(hWnd, &pt);
	pcshape[index]->SetEnd(pt.x, pt.y); //координати поточної точки курсору

	Ellipse(hdc, pcshape[index]->getXStart(), pcshape[index]->getYStart(),
pcshape[index]->getXEnd(), pcshape[index]->getYEnd());
	//Малюються лінії "гумового" сліду для поточного розташування курсору

	SelectObject(hdc, hPenOld);
	DeleteObject(hPen);
	ReleaseDC(hWnd, hdc); //закриваємо контекст вікна
}

void EllipseEditor::GetShape() {
	pcshape[index] = new EllipseShape;
};
```

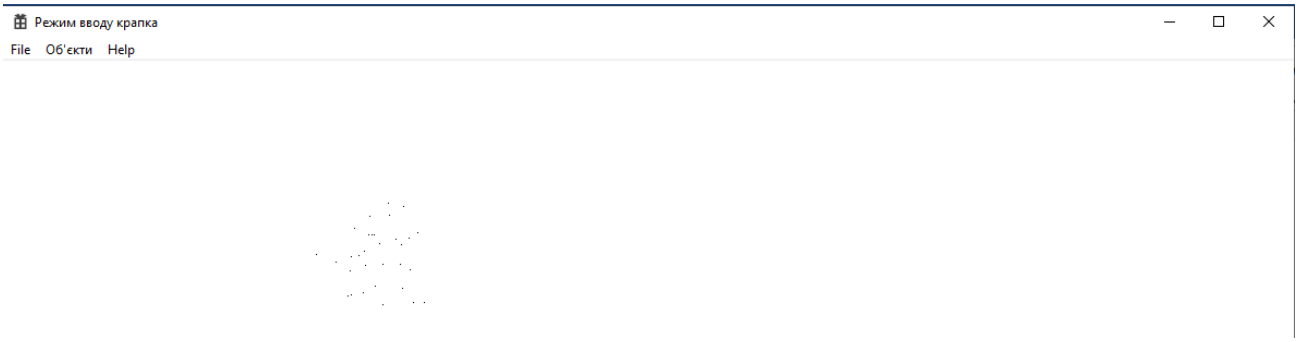
Діаграма класів



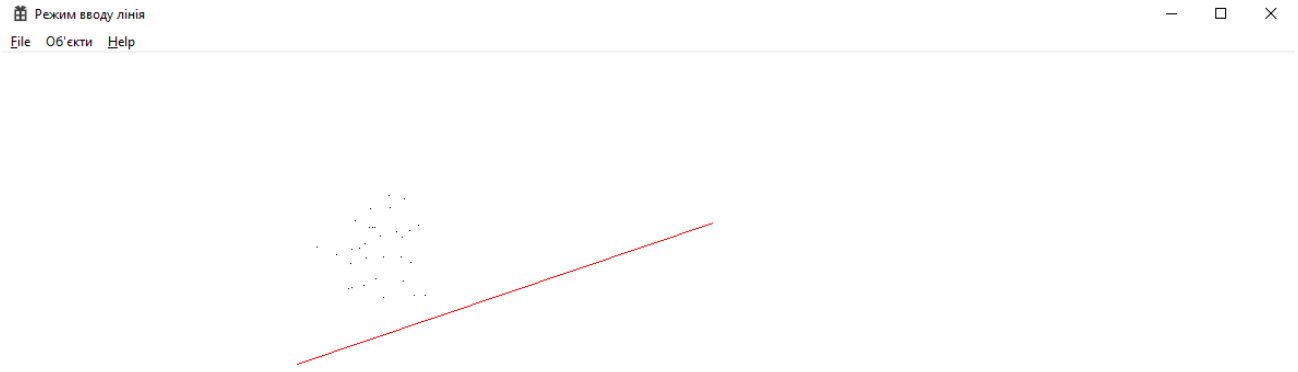
Скріншоти

При натисканні на “об’єкти” бачимо 4 пункти меню – крапка, лінія, прямокутник, еліпс

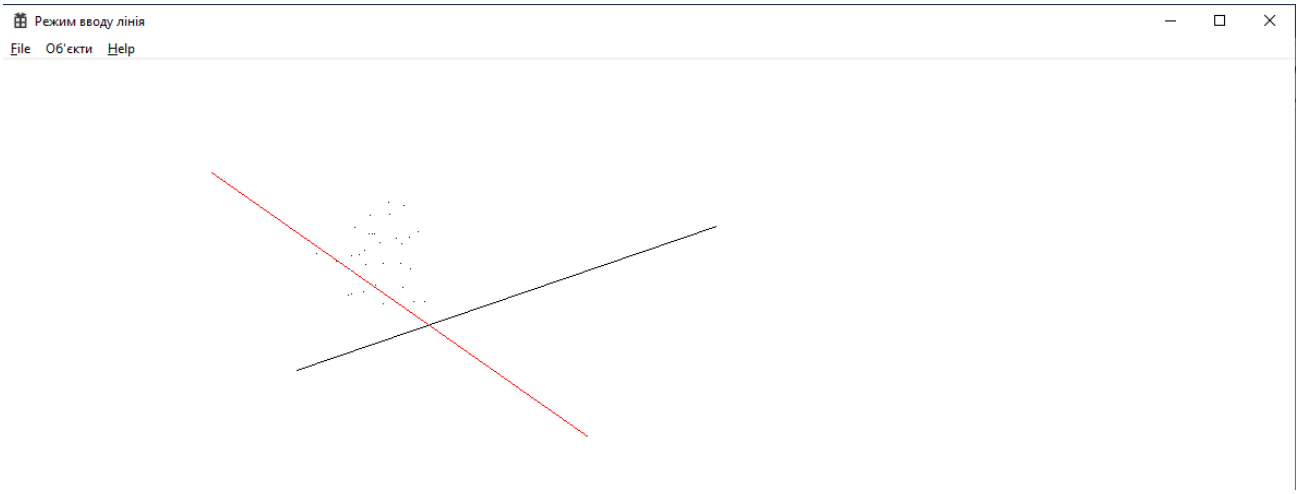
Крапки ставляться після відпускання миші, поставимо декілька.

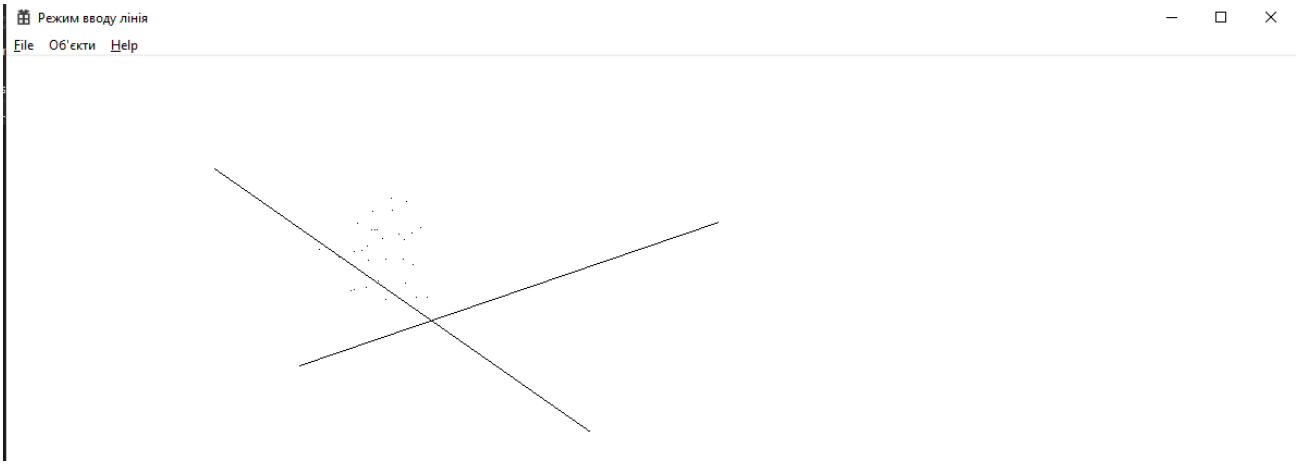


Оберемо лінію у меню і почнемо її малювати, поки не відпустимо ліву клавішу миші, будемо бачити гумовий слід, за варіантом лінія червона та суцільна.

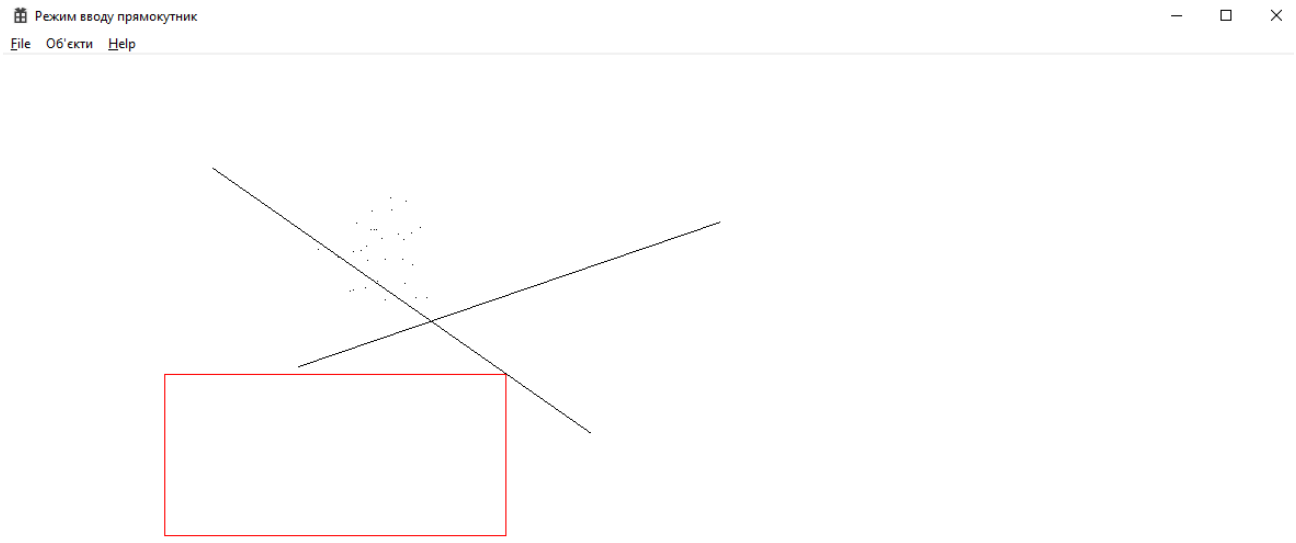


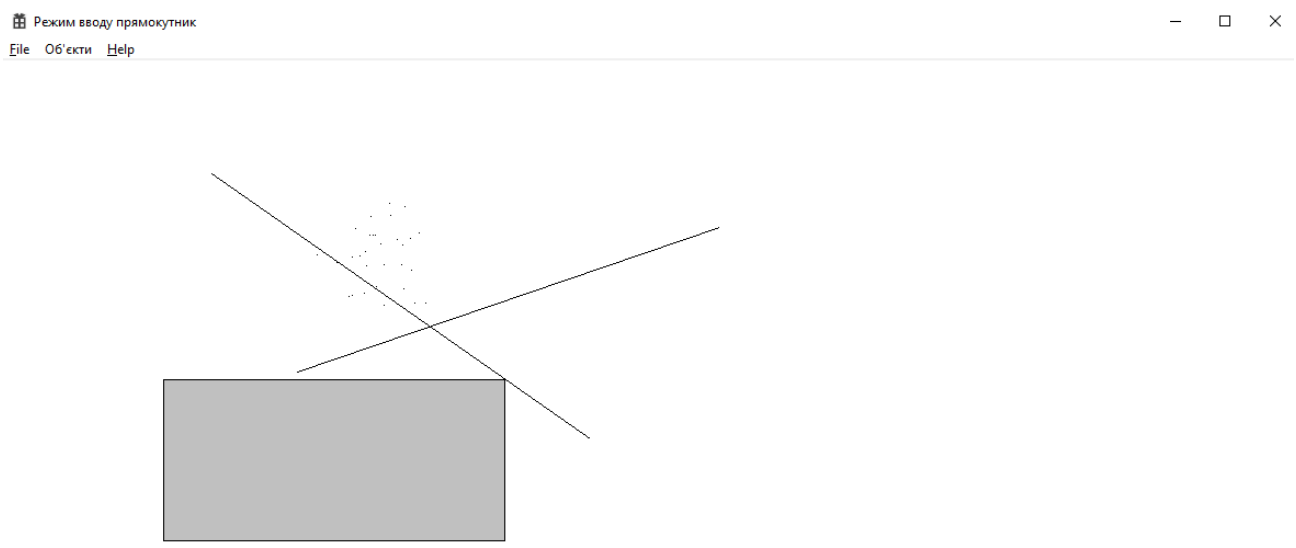
Після відпускання побачимо вже намальовану лінію.



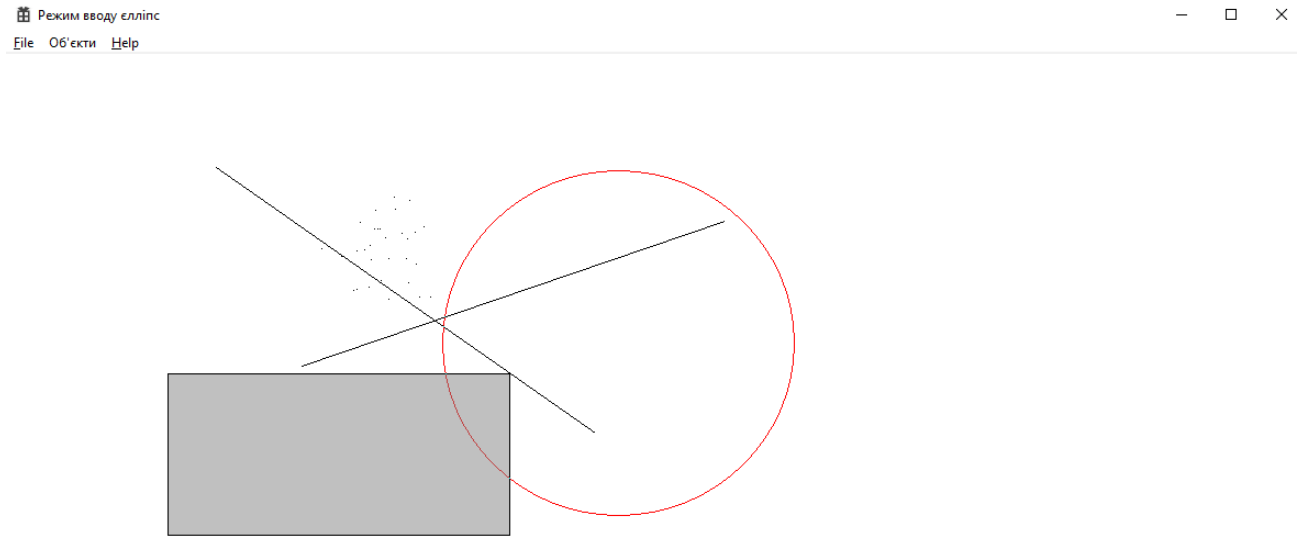


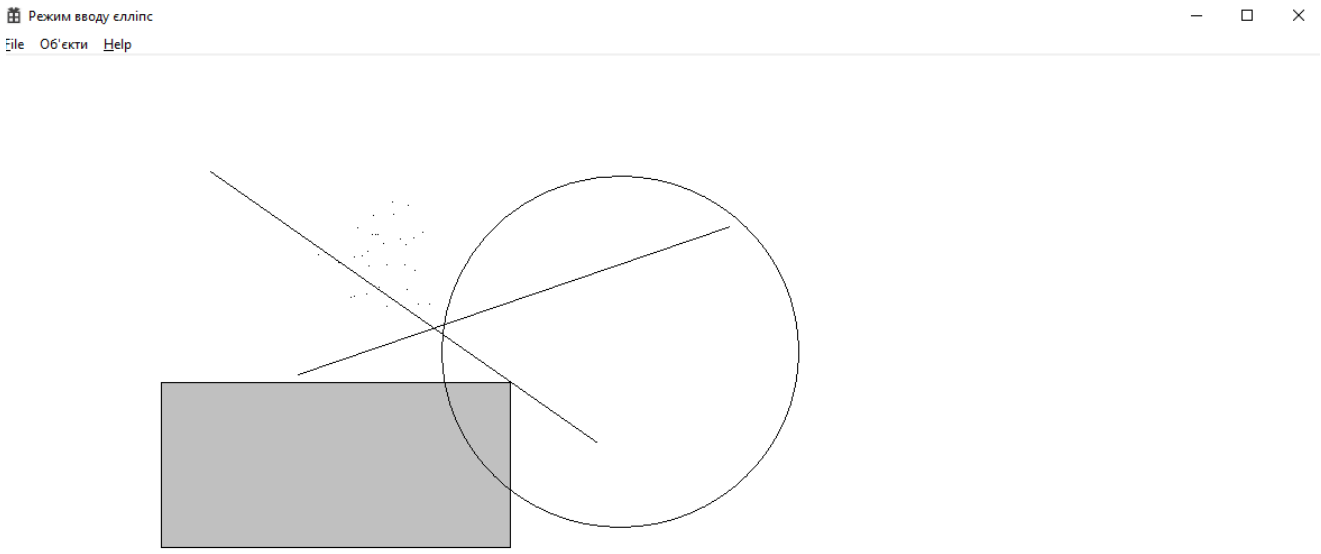
Аналогічно для прямокутника. У мене він із сірим заповненням і малюється із центру до будь-якого кута.

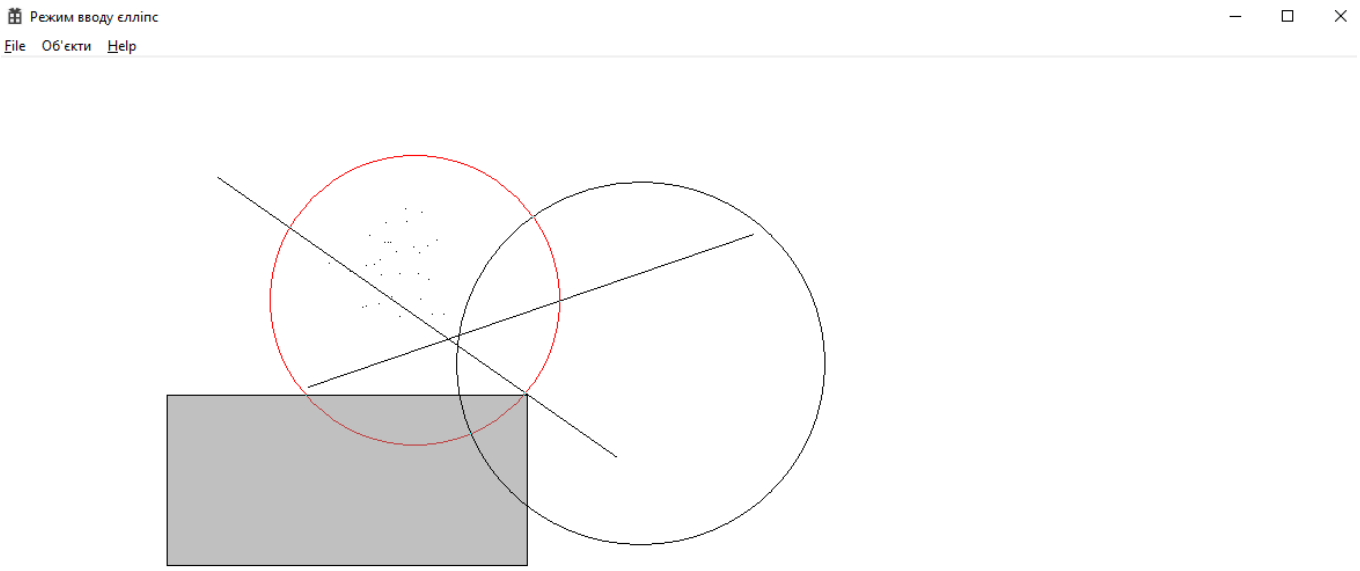


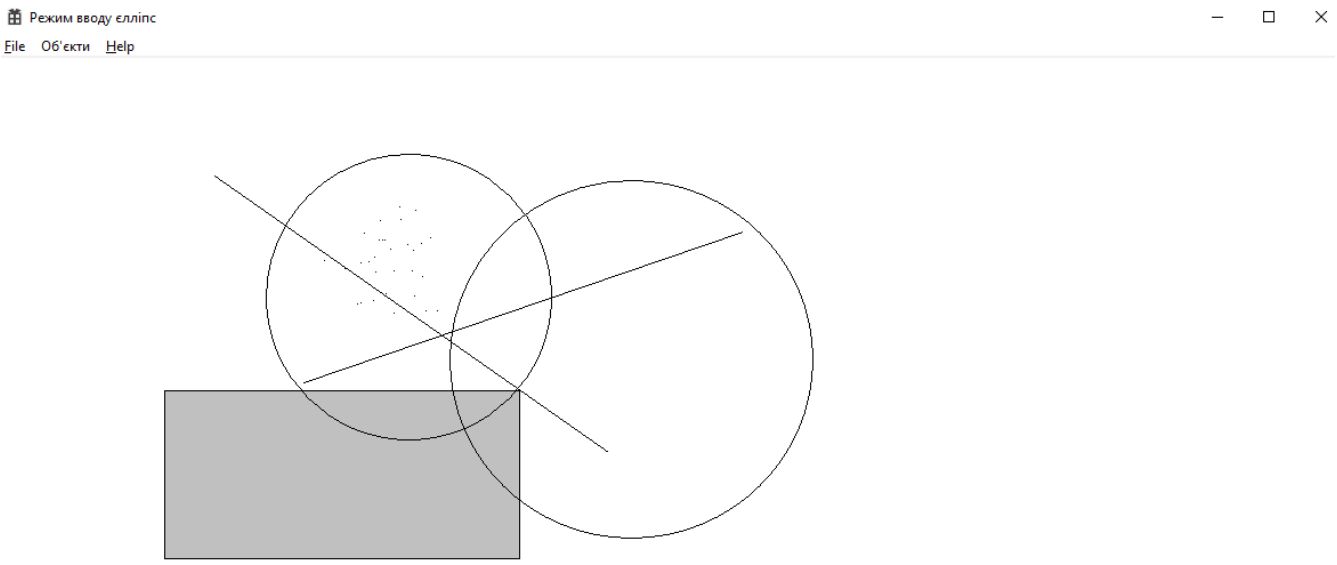


Аналогічно для еліпсу. Він у мене за варіантом без заповнення.


Режим вводу єллінс
File Об'єкти Help


Режим вводу єллінс
File Об'єкти Help





Як видно з скріншотів вище, при виборі іншого пункту меню у головному вікні програми змінюється заголовок, що відображає поточний режим.

Висновки. Під час виконання лабораторної роботи №2 я навчився створювати класи, описувати та реалізовувати їхні методи. Я ознайомився та попрацював із віртуальними методами у класах, навчився створювати похідні класи. На практиці познайомився та навчився використовувати інкапсуляцію, абстракцію типів, успадкування та поліморфізм при поботі із класами у C++. В результаті роботи я запрограмував графічний редактор, за допомогою якого можна малювати деякі фігури. Крім того, засобами Visual Studio C++ я створив діаграму класів, яку було додано до звіту.