**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»

на тему «Розробка графічного редактора об’єктів на C++»

Виконав: Перевірив:

студент групи ІП-93 Порєв Віктор Миколайович

Завальнюк Максим Євгенович

номер залікової книжки: 9312

Київ 2020

Мета**:** Мета роботи – отримати вміння та навички використовувати інкапсуляцію,

абстракцію типів, успадкування та поліморфізм на основі класів С++, запрограмувавши

простий графічний редактор в об’єктно-орієнтованому стилі.

Зміст

[Мета 2](#_Toc51870973)

[Завдання 3](#_Toc51870974)

[Варіанти завдань 3](#_Toc51870975)

[Вихідні тексти файлів 4](#_Toc51870976)

[Lab2.cpp 4](#_Toc51870977)

[Shape.cpp 7](#_Toc51870978)

[Shape\_editor.cpp 8](#_Toc51870979)

[Shape\_editor.h 12](#_Toc51870980)

[Editor.h 13](#_Toc51870981)

[Shape.h 13](#_Toc51870982)

Завдання**:**

1. Створити у середовищі MS Visual Studio C++ проект Win32 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.

# Варіанти завдань

**10 mod 3 = 1 –** статичний масив об’єктів типу Shape

**10 mod 4 = 2 –** гумовий слід у вигляді суцільної лінії синього кольору

**10 mod 2 = 0 –** ввід прямокутника по двом протилежним кутам

**10 mod 5 = 0 –** відображення прямокутника з чорним контуром з білим заповненням

**10 mod 2 = 0 –** ввід еліпсу від центру до одного з кутів охоплюючого прямокутника

**10 mod 5 = 0 –** відображення еліпсу з чорним контуром без заповнення

**10 mod 2 = 0 –** відображення поточного типу об’єкту позначкою в меню

# Вихідні тексти файлів

## Lab2.cpp

// Lab2.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "Lab2.h"

#include "resource.h"

#include "shape\_editor.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

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

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

ShapeObjectsEditor editorShape;

// 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);

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\_LAB2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB2));

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\_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);

}

//

// 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\_LBUTTONDOWN: //натиснуто ліву кнопку миші у клієнтській частині вікна

editorShape.OnLBdown(hWnd);

break;

case WM\_LBUTTONUP: //відпущено ліву кнопку миші у клієнтській частині вікна

editorShape.OnLBup(hWnd);

break;

case WM\_MOUSEMOVE: //пересунуто мишу у клієнтській частині вікна

editorShape.OnMouseMove(hWnd);

break;

case WM\_INITMENUPOPUP:

editorShape.OnInitMenuPopup(hWnd, wParam);

break;

case WM\_PAINT: //потрібно оновлення зображення клієнтської частині вікна

editorShape.OnPaint(hWnd);

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_POINT:

editorShape.StartPointEditor(); //початок вводу точкових об’єктів

break;

case IDM\_LINE:

editorShape.StartLineEditor(); //початок вводу об’єктів-ліній

break;

case IDM\_RECT:

editorShape.StartRectEditor(); //початок вводу прямокутників

break;

case IDM\_ELLIPSE:

editorShape.StartEllipseEditor(); //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;

}

// 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;

}

## Shape.cpp

#include "framework.h"

#include "shape.h"

void Shape::Set(long x1, long y1, long x2, long y2)

{

xs1 = x1;

ys1 = y1;

xs2 = x2;

ys2 = y2;

}

void PointShape::Show(HDC hdc) {

SetPixel(hdc, xs1, ys1, RGB(0, 255, 0));

}

void LineShape::Show(HDC hdc) {

HPEN hPen, hPenOld;

hPen = CreatePen(PS\_SOLID, 1, RGB(255, 0, 0));

hPenOld = (HPEN)SelectObject(hdc, hPen);

MoveToEx(hdc, xs1, ys1, NULL);

LineTo(hdc, xs2, ys2);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

}

void RectShape::Show(HDC hdc) {

HPEN hPen, hPenOld;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

hPenOld = (HPEN)SelectObject(hdc, hPen);

hBrush = CreateSolidBrush(RGB(255, 255, 255));

hBrushOld = (HBRUSH)SelectObject(hdc, hBrush);

SelectObject(hdc, hBrush);

Rectangle(hdc, xs1, ys1, xs2, ys2);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

}

void EllipseShape::Show(HDC hdc) {

HPEN hPen, hPenOld;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 0));

hPenOld = (HPEN)SelectObject(hdc, hPen);

Arc(hdc, xs1, ys1, xs2, ys2, 0, 0, 0, 0);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

};

## Shape\_editor.cpp

#include "framework.h"

#include "shape\_editor.h"

const int MY\_SHAPE\_ARRAY\_SIZE = 110;

Shape\* pcshape[MY\_SHAPE\_ARRAY\_SIZE];

int size = 0;

bool pressed;

ShapeObjectsEditor::ShapeObjectsEditor()

{

pse = new PointEditor;

}

ShapeObjectsEditor::~ShapeObjectsEditor()

{

for (int i = 0; i < size; i++) delete pcshape[i];

}

void ShapeObjectsEditor::StartPointEditor()

{

if (pse) delete pse;

pse = new PointEditor;

}

void ShapeObjectsEditor::StartLineEditor()

{

if (pse) delete pse;

pse = new LineEditor;

}

void ShapeObjectsEditor::StartRectEditor()

{

if (pse) delete pse;

pse = new RectEditor;

}

void ShapeObjectsEditor::StartEllipseEditor()

{

if (pse) delete pse;

pse = new EllipseEditor;

}

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 && pressed) pse->OnMouseMove(hWnd);

}

void ShapeObjectsEditor::OnPaint(HWND hWnd) {

ShapeEditor \*draw = new ShapeEditor;

draw->OnPaint(hWnd);

}

void ShapeObjectsEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {

if (pse)

pse->OnInitMenuPopup(hWnd, wParams);

}

// ---------------------------------- ShapeEditor Functions

void ShapeEditor::OnLBdown(HWND hWnd)

{

pressed = true;

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

x1 = x2 = pt.x;

y1 = y2 = pt.y;

}

void ShapeEditor::OnLBup(HWND hWnd)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

x2 = pt.x;

y2 = pt.y;

pressed = false;

}

void ShapeEditor::OnMouseMove(HWND hWnd) {}

void ShapeEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {};

void ShapeEditor::OnPaint(HWND hWnd) {

PAINTSTRUCT ps;

HDC hdc;

hdc = BeginPaint(hWnd, &ps);

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

if (pcshape[i])

pcshape[i]->Show(hdc);

EndPaint(hWnd, &ps);

}

// Point:

void PointEditor::OnLBdown(HWND hWnd) {

\_\_super::OnLBdown(hWnd);

}

void PointEditor::OnLBup(HWND hWnd) {

\_\_super::OnLBup(hWnd);

PointShape\* Point = new PointShape;

Point->Set(x1, y1, x2, y2);

pcshape[size] = Point;

size++;

InvalidateRect(hWnd, NULL, TRUE);

}

void PointEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1);

if ((HMENU)wParams == hSubMenu) {

CheckMenuItem(hSubMenu, IDM\_POINT, MF\_CHECKED);

CheckMenuItem(hSubMenu, IDM\_LINE, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_RECT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED);

}

}

// Line:

void LineEditor::OnLBdown(HWND hWnd) {

\_\_super::OnLBdown(hWnd);

}

void LineEditor::OnLBup(HWND hWnd) {

\_\_super::OnLBup(hWnd);

LineShape\* Line = new LineShape;

Line->Set(x1, y1, x2, y2);

pcshape[size] = Line;

size++;

InvalidateRect(hWnd, NULL, TRUE);

}

void LineEditor::OnMouseMove(HWND hWnd) {

POINT pt;

HPEN hPen, hPenOld;

HDC hdc = GetDC(hWnd);

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 255));

hPenOld = (HPEN)SelectObject(hdc, hPen);

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

x2 = pt.x;

y2 = pt.y;

MoveToEx(hdc, x1, y1, NULL);

LineTo(hdc, x2, y2);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

void LineEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1);

if ((HMENU)wParams == hSubMenu) {

CheckMenuItem(hSubMenu, IDM\_POINT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_LINE, MF\_CHECKED);

CheckMenuItem(hSubMenu, IDM\_RECT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED);

}

}

// Rect:

void RectEditor::OnLBdown(HWND hWnd) {

\_\_super::OnLBdown(hWnd);

}

void RectEditor::OnLBup(HWND hWnd) {

\_\_super::OnLBup(hWnd);

RectShape\* Rect = new RectShape;

Rect->Set(x1, y1, x2, y2);

pcshape[size] = Rect;

size++;

InvalidateRect(hWnd, NULL, TRUE);

}

void RectEditor::OnMouseMove(HWND hWnd) {

POINT pt;

HPEN hPen, hPenOld;

HDC hdc = GetDC(hWnd);

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 255));

hPenOld = (HPEN)SelectObject(hdc, hPen);

Rectangle(hdc, x1, y1, x2, y2);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

x2 = pt.x;

y2 = pt.y;

Rectangle(hdc, x1, y1, x2, y2);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

void RectEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1);

if ((HMENU)wParams == hSubMenu) {

CheckMenuItem(hSubMenu, IDM\_POINT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_LINE, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_RECT, MF\_CHECKED);

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED);

}

}

// Ellipse:

void EllipseEditor::OnLBdown(HWND hWnd) {

\_\_super::OnLBdown(hWnd);

}

void EllipseEditor::OnLBup(HWND hWnd) {

\_\_super::OnLBup(hWnd);

EllipseShape\* Ellipse = new EllipseShape;

Ellipse->Set(2 \* x1 - x2, 2 \* y1 - y2, x2, y2);

pcshape[size] = Ellipse;

size++;

InvalidateRect(hWnd, NULL, TRUE);

}

void EllipseEditor::OnMouseMove(HWND hWnd) {

POINT pt;

HPEN hPen, hPenOld;

HDC hdc = GetDC(hWnd);

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_SOLID, 1, RGB(0, 0, 255));

hPenOld = (HPEN)SelectObject(hdc, hPen);

Arc(hdc, 2 \* x1 - x2, 2 \* y1 - y2, x2, y2, 0, 0, 0, 0);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

x2 = pt.x;

y2 = pt.y;

Arc(hdc, 2 \* x1 - x2, 2 \* y1 - y2, x2, y2, 0, 0, 0, 0);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

void EllipseEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParams) {

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1);

if ((HMENU)wParams == hSubMenu) {

CheckMenuItem(hSubMenu, IDM\_POINT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_LINE, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_RECT, MF\_UNCHECKED);

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_CHECKED);

}

}

## Shape\_editor.h

#pragma once

#include "editor.h"

#include "shape.h"

#include "resource.h"

class ShapeEditor : public Editor

{

protected:

int x1, x2, y1, y2;

public:

void OnLBdown(HWND);

void OnLBup(HWND);

void OnMouseMove(HWND);

void OnPaint(HWND);

virtual void OnInitMenuPopup(HWND, WPARAM);

};

class ShapeObjectsEditor

{

private:

ShapeEditor \*pse;

public:

ShapeObjectsEditor(void);

~ShapeObjectsEditor();

void StartPointEditor();

void StartLineEditor();

void StartRectEditor();

void StartEllipseEditor();

void OnLBdown(HWND);

void OnLBup(HWND);

void OnMouseMove(HWND);

void OnPaint(HWND);

void OnInitMenuPopup(HWND, WPARAM);

};

class PointEditor : public ShapeEditor

{

public:

void OnLBdown(HWND);

void OnLBup(HWND);

void OnInitMenuPopup(HWND, WPARAM);

};

class LineEditor : public ShapeEditor

{

public:

void OnLBdown(HWND);

void OnLBup(HWND);

void OnMouseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

};

class RectEditor : public ShapeEditor

{

public:

void OnLBdown(HWND);

void OnLBup(HWND);

void OnMouseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

};

class EllipseEditor : public ShapeEditor

{

public:

void OnLBdown(HWND);

void OnLBup(HWND);

void OnMouseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

};

## Editor.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.h

class Shape

{

protected:

long xs1, ys1, xs2, ys2;

public:

void Set(long x1, long y1, long x2, long y2);

virtual void Show(HDC) = 0;

};

class PointShape : public Shape

{

public:

void Show(HDC);

};

class LineShape : public Shape

{

public:

void Show(HDC);

};

class RectShape : public Shape

{

public:

void Show(HDC);

};

class EllipseShape : public Shape

{

public:

void Show(HDC);

};