Cursach.h:

#pragma once

#include "resource.h"

#include "shape\_editor.h"

Resource.h:

//{{NO\_DEPENDENCIES}}

// Включаемый файл, созданный в Microsoft Visual C++.

// Используется Cursach.rc

//

#define IDC\_MYICON 2

#define IDD\_CURSACH\_DIALOG 102

#define IDS\_APP\_TITLE 103

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDI\_CURSACH 107

#define IDI\_SMALL 108

#define IDC\_CURSACH 109

#define IDR\_MAINFRAME 128

#define IDB\_BITMAP1 129

#define ID\_32771 32771

#define ID\_32772 32772

#define ID\_32773 32773

#define ID\_32774 32774

#define ID\_32775 32775

#define ID\_32776 32776

#define ID\_32777 32777

#define IDM\_RECT 32778

#define IDM\_ELLIPSE 32779

#define IDM\_CUBE 32780

#define IDM\_LINE 32781

#define IDM\_PUNKT 32782

#define IDM\_ROMB 32783

#define IDM\_CILINDER 32784

#define ID\_32785 32785

#define ID\_32786 32786

#define IDM\_OPENFILE 32787

#define IDM\_SAVEFILE 32788

#define IDM\_STDCOLOR 32789

#define IDM\_STDCOLORM 32790

#define IDM\_PENCOLOR 32792

#define ID\_32793 32793

#define ID\_32794 32794

#define IDM\_NEW 32795

#define IDC\_STATIC -1

// Next default values for new objects

//

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 1

#define \_APS\_NEXT\_RESOURCE\_VALUE 130

#define \_APS\_NEXT\_COMMAND\_VALUE 32796

#define \_APS\_NEXT\_CONTROL\_VALUE 1000

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

shape\_editor:

class ShapeObjectEditor

{

ShapeEditor \*pse = NULL;

bool onLBdown = false,

onLBup = false;

public:

ShapeObjectEditor(void);

~ShapeObjectEditor();

void StartRombEditor(COLORREF,COLORREF);//створює новий об'єкт RombEditor

void StartLineEditor(COLORREF, COLORREF);//створює новий об'єкт LineEditor

void StartRectEditor(COLORREF, COLORREF);//створює новий об'єкт RectEditor

void StartEllipseEditor(COLORREF, COLORREF);//створює новий об'єкт EllipseEditor

void StartPuncktLineEditor(COLORREF, COLORREF);//створює новий об'єкт PuncktLineEditor

void StartCubeEditor(COLORREF, COLORREF);//створює новий об'єкт CubeEditor

void StartCilinderEditor(COLORREF, COLORREF);//створює новий об'єкт CilinderEditor

void StartWriteFile(TCHAR\*);//виконується запис у файл

void StartReadFile(TCHAR\*);//виконується зчитування з файлу

void Reset(HWND);//перевантаження

void OnLBdown(HWND);//функція орбробки повідомлення натиснення лівої клавіши миші

void OnLBup(HWND, int xk, int yk);//функція орбробки повідомлення відпусксання лівої клавіши миші

void OnMosuseMove(HWND);//функція орбробки повідомлення руху миші

void OnPaint(HWND,HDC, int xk, int yk);//функція малювання фігур

void OninitMenuPopup(HWND, WPARAM);//позначає вибраний елемент у меню

void PressButton(HWND);//позначає вибраний елемент на панелі інструментів

void CreateNewScene();//створення нової сцени

};

ToolBar.h:

#pragma once

#if !defined(TOOL\_H)

#define TOOL\_H

#include <CommCtrl.h>

#define IDB\_RECT 1005

#define IDB\_ELLIPSE 1006

#define IDB\_POINT 1007

#define IDB\_LINE 1008

#define IDB\_LINEOO 1009

#define IDB\_CUBE 1010

#define IDB\_PUNCKT 1011

#define IDB\_ROMB 1012

#define IDB\_CILINDER 1013

#define IDB\_CIRCUIT 1014

#define IDB\_FILL 1015

#endif !defined(TOOL\_H)

const TBBUTTON tbb\_1[] = {

{ 0,IDB\_RECT,TBSTATE\_ENABLED,TBSTYLE\_BUTTON,0,0,0,0 },//кнопка прямокутник

{ 1, IDB\_ELLIPSE, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка еліпс

{ 2, IDB\_LINE, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка лінія

{ 3, IDB\_CUBE, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка куб

{ 4, IDB\_PUNCKT, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка пунктирна лінія

{ 5, IDB\_ROMB, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка ромб

{ 6, IDB\_CILINDER, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка циліндр

{ 7, IDB\_FILL, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка зміни кольору заповнення

{ 8, IDB\_CIRCUIT, TBSTATE\_ENABLED, TBSTYLE\_BUTTON, 0, 0, 0, 0 },//кнопка зміни кольору контуру

};

ScrollOperations.h:

#pragma once

#include <windows.h>

class ScrollOperations

{

int xStep; // средняя ширина символа

int yStep; // высота (шаг) строки

int lineLenMax; // максимальная длина строки

SCROLLINFO vsi; // вертикальный скроллинг

int vertRange; // диапазон вертикальной полосы прокрутки

SCROLLINFO hsi; // горизонтальный скроллинг

int horzRange; // диапазон горизонтальной полосы прокрутки

public:

void ScrollSettings(HWND hwnd, int width, int height);//налаштовує скролінг

void UpdateHscroll(HWND hwnd, int xlnc);//оновлює горизонтальний скрол

void UpdateVscroll(HWND hwnd, int ylnc);//оновлює вертикальний скрол

int GetHSIpos();//повертає hsi.nPos

int GetVSIpos();//повертає vsi.nPos

int GetxStep();//повертає xStep

int GetyStep();//повертає yStep

int GetyPage();//повертає vsi.nPage

int GetxPage();//повертає vsi.nPage

ScrollOperations();

~ScrollOperations();

};

Editor.h:

#pragma once

#if !defined(EDITOR\_H)

#define EDITOR\_H

#include "Resource.h"

#include <fstream>

#include <iomanip> // манипуляторы ввода/вывода

#include <iostream>

using namespace std;

class Editor

{

public:

virtual void OnLBdown(HWND) = 0;//функція орбробки повідомлення натиснення лівої клавіши миші

virtual void OnLBup(HWND, int xk, int yk) = 0;//функція орбробки повідомлення відпусксання лівої клавіши миші

virtual void OnMosuseMove(HWND) = 0;//функція орбробки повідомлення руху миші

virtual void OnPaint(HWND,HDC, int xk, int yk) = 0;//функція малювання фігур

};

#endif !defined(EDITOR\_H)

ShapeEditor.h:

#include "PuncktLineShape.h"

#include "RombShape.h"

#include "CilinderShape.h"

class ShapeEditor :

public Editor

{

protected:

static int curr\_length;//поточна довжина масиву pcshape

static Shape \*pcshape[MY\_SHAPE\_ARRAY\_SIZE];//масив фігур

POINT pt\_old;//початкові координати

POINT pt\_start;//кінцеві координати

COLORREF brColor,//колір заливки

penColor;//колір контуру

public:

ShapeEditor(COLORREF,COLORREF);

ShapeEditor();

void OnLBdown(HWND);

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

void OnPaint(HWND,HDC,int xk, int yk);

void PrintInFile(char \*);//записує у файл масив pcshape

void ReadFromFile(char\*);//зчитує з файлу у масив pcshape

virtual void OnInitMenuPopup(HWND, WPARAM);//позначає вибраний елемент у меню

virtual void PressButton(HWND);//позначає вибраний елемент на панелі інструментів

void OverMeesage();//повідомлення про закінчення місця у масиві pcshape

void CreateNewScene();//створення нової сцени

};

#endif !defined(SHAPEDITOR\_H)

CilinderEditor.h:

#pragma once

#include "ShapeEditor.h"

class CilinderEditor :

public ShapeEditor

{

public:

CilinderEditor(COLORREF,COLORREF);

~CilinderEditor();

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

};

CubeEditor.h:

#pragma once

#include "ShapeEditor.h"

class CubeEditor :

public ShapeEditor

{

public:

CubeEditor(COLORREF,COLORREF);

~CubeEditor();

void OnLBup(HWND hWnd, int xk, int yk);

void OnInitMenuPopup(HWND hWnd, WPARAM wParam);

void PressButton(HWND hWnd);

void OnMosuseMove(HWND hWnd);

};

EllipseEditor.h:

#pragma once

#include "ShapeEditor.h"

class EllipseEditor :

public ShapeEditor

{

public:

EllipseEditor(COLORREF, COLORREF);

~EllipseEditor();

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

};

LineEditor.h:

#pragma once

#include "ShapeEditor.h"

class LineEditor :

public ShapeEditor

{

public:

LineEditor(COLORREF, COLORREF);

~LineEditor();

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

};

PuncktLineEditor.h:

#pragma once

#include "LineEditor.h"

class PuncktLineEditor :

public LineEditor

{

public:

PuncktLineEditor(COLORREF, COLORREF);

~PuncktLineEditor();

void OnLBup(HWND, int xk, int yk);

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

};

RectEditor.h:

#pragma once

#include "ShapeEditor.h"

class RectEditor :

public ShapeEditor

{

public:

RectEditor(COLORREF, COLORREF);

~RectEditor();

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

};

RombEditor.h:

#pragma once

#include "ShapeEditor.h"

class RombEditor :

public ShapeEditor

{

public:

RombEditor(COLORREF, COLORREF);

~RombEditor();

void OnInitMenuPopup(HWND, WPARAM);

void PressButton(HWND);

void OnLBup(HWND, int xk, int yk);

void OnMosuseMove(HWND);

};

Shape.h:

#pragma once

#if !defined(SHAPE\_H)

#define SHAPE\_H

#include <string>

class Shape

{

protected:

long xs1, ys1, xs2, ys2;//координати фігури

std::string type;//тип фігури

COLORREF brColor,//колір заливки

penColor;//колір контуру

public:

void Set(long x1,long y1, long x2, long y2,COLORREF,COLORREF,int xk,int yk);//ініціалізу об'єкт shape

virtual void Show(HDC,int xk, int yk) = 0; //малює об'єкт shape

std::string RetData();//повертає дані про об'єкт shape

};

#endif !defined(SHAPE\_H)

CilinderShape.h:

#pragma once

#include "Shape.h"

class CilinderShape :

public Shape

{

public:

CilinderShape();

~CilinderShape();

void Show(HDC,int xk, int yk);

};

CubeShape.h:

#pragma once

#include "Shape.h"

class CubeShape :

public Shape

{

public:

CubeShape();

~CubeShape();

void Show(HDC, int xk, int yk);

};

EllipseShape.h:

#pragma once

#include "Shape.h"

class EllipseShape :

public Shape

{

public:

EllipseShape();

~EllipseShape();

void Show(HDC, int xk, int yk);

};

LineShape.h:

#pragma once

#include "Shape.h"

class LineShape :

public Shape

{

public:

LineShape();

~LineShape();

void Show(HDC hdc, int xk, int yk);

};

PuncktLineShape.h:

#pragma once

#include "LineShape.h"

class PuncktLineShape :

public Shape

{

public:

PuncktLineShape();

~PuncktLineShape();

void Show(HDC, int xk, int yk);

};

RectShape.h:

#pragma once

#include "Shape.h"

class RectShape :

public Shape

{

public:

RectShape();

~RectShape();

void Show(HDC, int xk, int yk);

};

RombShape.h:

#pragma once

#include "Shape.h"

class RombShape :

public Shape

{

public:

RombShape();

~RombShape();

void Show(HDC, int xk, int yk);

};

Cursach.cpp:

// Cursach.cpp: îïðåäåëÿåò òî÷êó âõîäà äëÿ ïðèëîæåíèÿ.

//

#include "stdafx.h"

#include "Cursach.h"

#include "ToolBar.h"

#include <fstream>

#include <commdlg.h>

#include "ScrollOperations.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 окна

ShapeObjectEditor obj\_editor; // керуючий об'єкт

ScrollOperations doc; // керрування скроллінгом

// 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);

// Инициализация глобальных строк

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

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

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

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

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()

//

// НАЗНАЧЕНИЕ: 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\_CURSACH));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_CURSACH);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// НАЗНАЧЕНИЕ: сохраняет обработку экземпляра и создает главное окно.

//

// КОММЕНТАРИИ:

//

// В данной функции дескриптор экземпляра сохраняется в глобальной переменной, а также

// создается и выводится на экран главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

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

HWND hWnd = CreateWindowW(szWindowClass, LPCWSTR("Курсова робота"), WS\_OVERLAPPEDWINDOW | WS\_CLIPCHILDREN | WS\_VSCROLL | WS\_HSCROLL,

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)

//

// НАЗНАЧЕНИЕ: обрабатывает сообщения в главном окне.

//

// WM\_COMMAND — обработать меню приложения

// WM\_PAINT — отрисовать главное окно

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

//

//

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

{

static HWND hWndToolBar;// панель інструментів

static TCHAR name[256] = \_T(""); //ім'я файлу для запису та зберігання

static OPENFILENAME file;

static CHOOSECOLOR ccs,//заливка

penCCS;//контур

static COLORREF acrCustClr[16];

static HBRUSH hBrush;

int xClient = 0,

yClient = 0;

static int xInc, yInc,

currX = 0,//х позиція скролл бару

currY = 0;//у позиція скролл бару

static COLORREF stdColor = RGB(255, 255, 255);//початковий колір заливки

static COLORREF penColor = RGB(0, 0, 0);//початковий колір контуру

HDC hDc;

switch (message)

{

case WM\_CREATE:

hWndToolBar = CreateToolbarEx(hWnd, TBSTYLE\_TOOLTIPS | WS\_CHILD | WS\_VISIBLE | WS\_BORDER | WS\_CLIPSIBLINGS , 1, 9,hInst, IDB\_BITMAP1, tbb\_1, 9, 25, 25, 25, 25, sizeof(TBBUTTON));//створення панелі інструментів

//ініціалізація file

file.lStructSize = sizeof(OPENFILENAME);

file.hInstance = hInst;

file.lpstrFilter = \_T("Text\0\*.txt");

file.lpstrFile = name;

file.nMaxFile = 256;

file.lpstrInitialDir = \_T(".\\");

file.lpstrDefExt = \_T("txt");

//ініціалізація ccs

ccs.lStructSize = sizeof(CHOOSECOLOR);

ccs.hwndOwner = hWnd;

ccs.rgbResult = stdColor;

ccs.Flags = CC\_RGBINIT | CC\_FULLOPEN;

ccs.lpCustColors = (LPDWORD)acrCustClr;

//ініціалізація penCCS

penCCS.lStructSize = sizeof(CHOOSECOLOR);

penCCS.hwndOwner = hWnd;

penCCS.rgbResult = penColor;

penCCS.Flags = CC\_RGBINIT | CC\_FULLOPEN;

penCCS.lpCustColors = (LPDWORD)acrCustClr;

break;

case WM\_SIZE:

SendMessage(hWndToolBar,TB\_AUTOSIZE, 0,0);

hDc = GetDC(hWnd);

xClient = LOWORD(lParam);

yClient = HIWORD(lParam);

if (xClient > 0)

doc.ScrollSettings(hWnd, xClient, yClient);

ReleaseDC(hWnd, hDc);

break;

case WM\_VSCROLL:

//обробник повідомлень вертикального скролінгу

switch (LOWORD(wParam)) {

case SB\_LINEUP:

yInc = -1;

break;

case SB\_LINEDOWN:

yInc = 1;

break;

case SB\_PAGEUP:

yInc = -(int)doc.GetyPage();

break;

case SB\_PAGEDOWN:

yInc = (int)doc.GetyPage();

break;

case SB\_THUMBTRACK:

yInc = HIWORD(wParam) - doc.GetVSIpos();

break;

default: yInc = 0;

}

doc.UpdateVscroll(hWnd, yInc);

currY = -doc.GetVSIpos() \* doc.GetyStep();

SendMessage(hWndToolBar, TB\_AUTOSIZE, 0, 0);

break;

case WM\_HSCROLL:

//обробник повідомлень горизонтального скролінгу

switch (LOWORD(wParam))

{

case SB\_LINELEFT:

xInc = -1; break;

case SB\_LINERIGHT:

xInc = 1;

break;

case SB\_PAGELEFT:

xInc = -doc.GetxPage();

break;

case SB\_PAGERIGHT:

xInc = doc.GetxPage();

break;

case SB\_THUMBTRACK:

xInc = HIWORD(wParam) - doc.GetHSIpos();

break;

default: xInc = 0;

}

doc.UpdateHscroll(hWnd, xInc);

currX = -doc.GetHSIpos() \* doc.GetxStep();

SendMessage(hWndToolBar, TB\_AUTOSIZE, 0, 0);

break;

case WM\_NOTIFY:

{

//TOOLTIPS

LPNMHDR pnmh = (LPNMHDR)lParam;

LPSTR pText;

if (pnmh->code == TTN\_NEEDTEXT)

{

LPTOOLTIPTEXT lpttt = (LPTOOLTIPTEXT)lParam;

switch (lpttt->hdr.idFrom)

{

case IDB\_ROMB:

pText = "Ромб";

break;

case IDB\_LINE:

pText = "Лінія";

break;

case IDB\_RECT:

pText = "Прямокутник";

break;

case IDB\_ELLIPSE:

pText = "Еліпс";

break;

case IDB\_CUBE:

pText = "Куб";

break;

case IDB\_CILINDER:

pText = "Циліндр";

break;

case IDB\_PUNCKT:

pText = "Пуктирна лінія";

break;

case IDB\_FILL:

pText = "Колір заливки";

break;

case IDB\_CIRCUIT:

pText = "Колір контуру";

break;

default:

pText = "ToolBar";

break;

}

lstrcpy(lpttt->szText, pText);

}

break;

}

case WM\_LBUTTONDOWN:

obj\_editor.OnLBdown(hWnd);//функція орбробки повідомлення натиснення лівої клавіши миші

break;

case WM\_LBUTTONUP:

obj\_editor.OnLBup(hWnd, currX, currY);//функція орбробки повідомлення відпусксання лівої клавіши миші

break;

case WM\_MOUSEMOVE:

obj\_editor.OnMosuseMove(hWnd);//функція орбробки повідомлення руху миші

break;

case WM\_INITMENUPOPUP:

obj\_editor.OninitMenuPopup(hWnd, wParam);//позначає вибраний елемент у меню

break;

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDB\_ELLIPSE:

obj\_editor.StartEllipseEditor(stdColor,penColor);//створює новий об'єкт EllipseEditor

obj\_editor.PressButton(hWndToolBar);//позначає вибраний елемент на панелі інструментів

break;

case IDB\_RECT:

obj\_editor.StartRectEditor(stdColor, penColor);//створює новий об'єкт RectEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDB\_LINE:

obj\_editor.StartLineEditor(stdColor, penColor);//створює новий об'єкт LineEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDB\_CUBE:

obj\_editor.StartCubeEditor(stdColor, penColor);//створює новий об'єкт CubeEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDB\_PUNCKT:

obj\_editor.StartPuncktLineEditor(stdColor, penColor);//створює новий об'єкт PuncktLineEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDB\_ROMB:

obj\_editor.StartRombEditor(stdColor, penColor);//створює новий об'єкт RombEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDB\_CILINDER:

obj\_editor.StartCilinderEditor(stdColor, penColor);//створює новий об'єкт CilinderEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_LINE:

obj\_editor.StartLineEditor(stdColor, penColor);//створює новий об'єкт LineEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_RECT:

obj\_editor.StartRectEditor(stdColor, penColor);//створює новий об'єкт RectEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_ELLIPSE:

obj\_editor.StartEllipseEditor(stdColor, penColor);//створює новий об'єкт EllipseEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_CUBE:

obj\_editor.StartCubeEditor(stdColor, penColor);//створює новий об'єкт CubeEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_PUNKT:

obj\_editor.StartPuncktLineEditor(stdColor, penColor);//створює новий об'єкт PuncktLineEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_ROMB:

obj\_editor.StartRombEditor(stdColor, penColor);//створює новий об'єкт RombEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_CILINDER:

obj\_editor.StartCilinderEditor(stdColor, penColor);//створює новий об'єкт CilinderEditor

obj\_editor.PressButton(hWndToolBar);

break;

case IDM\_ABOUT:

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

break;

case IDM\_SAVEFILE:

//викликає діалогове вікно запису у файл

file.lpstrTitle = \_T("Открыть файл для записи");

file.Flags = OFN\_NOTESTFILECREATE;

if (!GetSaveFileName(&file)) return 1;

obj\_editor.StartWriteFile(name);

break;

case IDM\_OPENFILE:

//викликає дфалогове вікно зчитування з файлу

file.lpstrTitle = \_T("Открыть файл для чтения");

file.Flags = OFN\_HIDEREADONLY;

if (!GetOpenFileName(&file)) return 1;

obj\_editor.StartReadFile(name);

InvalidateRect(hWnd, NULL, 1);

break;

case IDB\_FILL:

case IDM\_STDCOLOR:

//викликає діалогове вікно вибіру кольору заливки

if (ChooseColor(&ccs))

{

stdColor = ccs.rgbResult;

obj\_editor.Reset(hWndToolBar);

}

break;

case IDB\_CIRCUIT:

case IDM\_PENCOLOR:

//викликає діалогове вікно вибіру кольору контуру

if (ChooseColor(&penCCS))

{

penColor = penCCS.rgbResult;

obj\_editor.Reset(hWndToolBar);

}

break;

case IDM\_NEW:

obj\_editor.CreateNewScene();

obj\_editor.Reset(hWndToolBar);

InvalidateRect(hWnd, NULL, TRUE);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

//малює фігури

PAINTSTRUCT ps;

HDC hdc;

hdc = BeginPaint(hWnd, &ps);

obj\_editor.OnPaint(hWnd,hdc,currX,currY);

EndPaint(hWnd, &ps);

}

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\_editor.cpp:

#include "stdafx.h"

#include "shape\_editor.h"

ShapeObjectEditor::ShapeObjectEditor(void)

{

}

ShapeObjectEditor:: ~ShapeObjectEditor()

{

}

/\*

\* створює новий об'єкт RombEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor:: StartRombEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new RombEditor(brushColor,peColor);

}

/\*

\* створює новий об'єкт LineEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartLineEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new LineEditor(brushColor, peColor);

}

/\*

\* створює новий об'єкт PuncktLineEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartPuncktLineEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new PuncktLineEditor(brushColor, peColor);

}

/\*

\* створює новий об'єкт CilinderEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartCilinderEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new CilinderEditor(brushColor, peColor);

}

/\*

\* створює новий об'єкт CubeEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartCubeEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new CubeEditor(brushColor, peColor);

}

/\*

\* створює новий об'єкт RectEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartRectEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new RectEditor(brushColor, peColor);

}

/\*

\* створює новий об'єкт EllipseEditor

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

void ShapeObjectEditor::StartEllipseEditor(COLORREF brushColor, COLORREF peColor)

{

if (pse)

pse = NULL;

pse = new EllipseEditor(brushColor, peColor);

}

/\*

\* виконується запис у файл

\* name - ім'я

\*/

void ShapeObjectEditor::StartWriteFile(TCHAR\* name)

{

if (pse)

pse->PrintInFile(name);

}

/\*

\* виконується зчитування з файлу

\* name - ім'я

\*/

void ShapeObjectEditor::StartReadFile(TCHAR\* name)

{

if (pse)

pse = NULL;

pse = new ShapeEditor();

pse->ReadFromFile(name);

}

//функція орбробки повідомлення натиснення лівої клавіши миші

void ShapeObjectEditor::OnLBdown(HWND hWnd)

{

if (pse)

{

pse->OnLBdown(hWnd);

onLBdown = true;

onLBup = false;

}

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void ShapeObjectEditor::OnLBup(HWND hWnd, int xk, int yk)

{

if (pse)

{

pse->OnLBup(hWnd,xk,yk);

onLBup = true;

onLBdown = false;

}

}

//функція орбробки повідомлення руху миші

void ShapeObjectEditor::OnMosuseMove(HWND hWnd)

{

if (pse && !onLBup && onLBdown)

pse->OnMosuseMove(hWnd);

}

/\*

\* функція малювання фігур

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void ShapeObjectEditor::OnPaint(HWND hWnd,HDC hdc,int xk, int yk)

{

if (pse)

pse->OnPaint(hWnd,hdc,xk,yk);

}

//позначає вибраний елемент у меню

void ShapeObjectEditor::OninitMenuPopup(HWND hWnd, WPARAM hWparam)

{

if (pse)

pse->OnInitMenuPopup(hWnd, hWparam);

}

//позначає вибраний елемент на панелі інструментів

void ShapeObjectEditor::PressButton(HWND hWnd)

{

if (pse)

pse->PressButton(hWnd);

}

//перевантаження

void ShapeObjectEditor::Reset(HWND hWndTool)

{

if (pse)

pse = NULL;

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWndTool, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

//створення нової сцени

void ShapeObjectEditor::CreateNewScene()

{

if (!pse)

pse = new ShapeEditor();

pse->CreateNewScene();

}

ScrollOperations.cpp:

#include "stdafx.h"

#include "ScrollOperations.h"

ScrollOperations::ScrollOperations()

{

xStep = 10;

yStep = 10;

vsi.nMin = 0;

hsi.nMin = 0;

}

ScrollOperations::~ScrollOperations()

{

}

//налаштовує скролінг

void ScrollOperations::ScrollSettings (HWND hwnd, int width, int height)

{

// Вертикальный скроллинг

vsi.cbSize = sizeof(vsi);

vsi.fMask = SIF\_RANGE | SIF\_PAGE | SIF\_POS;

vsi.nPage = height / yStep - 1;

vsi.nMax = 199;

if (vsi.nPage > vsi.nMax)

vsi.nPos = vsi.nMin;

vertRange = vsi.nMax - vsi.nMin + 1;

SetScrollInfo(hwnd, SB\_VERT, &vsi, TRUE);

// Горизонтальный скроллинг

hsi.cbSize = sizeof(SCROLLINFO);

hsi.fMask = SIF\_RANGE | SIF\_PAGE | SIF\_POS;

hsi.nPage = width / xStep - 2;

hsi.nMax = 198;

if (hsi.nPage > hsi.nMax)

hsi.nPos = hsi.nMin;

horzRange = hsi.nMax - hsi.nMin + 1;

SetScrollInfo(hwnd, SB\_HORZ, &hsi, TRUE);

}

//оновлює вертикальний скрол

void ScrollOperations::UpdateVscroll(HWND hwnd, int yInc)

{

// ограничение на положительное приращение

yInc = min(yInc, vertRange - (int)vsi.nPage - vsi.nPos);

// ограничение на отрицательное приращение

yInc = max(yInc,vsi.nMin - vsi.nPos);

if (yInc)

{

ScrollWindow(hwnd, 0, -yStep \* yInc, NULL, NULL);

vsi.nPos += yInc;

SetScrollInfo(hwnd, SB\_VERT, &vsi, TRUE);

InvalidateRect(hwnd, NULL, TRUE);

UpdateWindow(hwnd);

}

}

//оновлює горизонтальний скрол

void ScrollOperations::UpdateHscroll(HWND hwnd, int xInc)

{

// ограничение на положительное приращение

xInc = min(xInc,horzRange - (int)hsi.nPage - hsi.nPos);

// ограничение на отрицательное приращение

xInc = max(xInc, hsi.nMin - hsi.nPos);

if (xInc) {

ScrollWindow(hwnd, -xStep \* xInc, 0, NULL, NULL);

hsi.nPos += xInc;

SetScrollInfo(hwnd, SB\_HORZ, &hsi, TRUE);

InvalidateRect(hwnd, NULL, TRUE);

UpdateWindow(hwnd);

}

}

//повертає hsi.nPos

int ScrollOperations::GetHSIpos()

{

return hsi.nPos;

}

//повертає vsi.nPos

int ScrollOperations::GetVSIpos()

{

return vsi.nPos;

}

//повертає xStep

int ScrollOperations::GetxStep()

{

return xStep;

}

//повертає yStep

int ScrollOperations::GetyStep()

{

return yStep;

}

//повертає vsi.nPage

int ScrollOperations::GetyPage()

{

return vsi.nPage;

}

//повертає vsi.nPage

int ScrollOperations::GetxPage()

{

return hsi.nPage;

}

ShapeEditor.cpp:

#include "stdafx.h"

#include "ShapeEditor.h"

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

int ShapeEditor::curr\_length = 0;

/\*

\* brushColor - колір заливки

\* peColor - колір контуру

\*/

ShapeEditor::ShapeEditor(COLORREF brushColor,COLORREF peColor)

{

brColor = brushColor;

penColor = peColor;

}

ShapeEditor::ShapeEditor()

{

}

/\*

\* функція малювання фігур

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void ShapeEditor::OnPaint(HWND hWnd,HDC hdc,int xk, int yk)

{

for (int i = 0; i < curr\_length; i++)

{

if (pcshape[i])

pcshape[i]->Show(hdc,xk,yk);

}

}

//функція орбробки повідомлення натиснення лівої клавіши миші

void ShapeEditor::OnLBdown(HWND hWnd)

{

GetCursorPos(&pt\_start);

ScreenToClient(hWnd, &pt\_start);

pt\_old = pt\_start; //записуємо координати початкової точки

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void ShapeEditor::OnLBup(HWND hWnd, int xk, int yk)

{

}

//функція орбробки повідомлення руху миші

void ShapeEditor::OnMosuseMove(HWND)

{

}

/\*

\* записує у файл масив pcshape

\* name - ім'я файлу

\*/

void ShapeEditor::PrintInFile(char \*name)

{

ofstream fout(name, ios\_base::out);//відкриття потоку

for (int i = 0; i < curr\_length; i++)

{

if (pcshape[i])

{

fout << pcshape[i]->RetData().c\_str() <<endl;//запис у файл

}

}

fout.close();//закриття потоку

}

/\*

\* зчитує з файлу у масив pcshape

\* name - ім'я файлу

\*/

void ShapeEditor::ReadFromFile(char\* name)

{

ifstream file;

file.open(name, ios\_base::in);

if (file) // якщо файл існує

{

int n = 0,

c\_dots = 0;

int dots[6];

char str[350],

\*next\_token = NULL,

\*p,

\*delimiter = "\t";

bool fl = true;

//видалення елементів з pcshape

for (int i = 0; i < curr\_length;i++)

{

pcshape[i] = NULL;

}

//цикл зчитування з файлу

while (!file.eof() && n <= MY\_SHAPE\_ARRAY\_SIZE)

{

file.getline(str, 350);

p = strtok\_s(str, delimiter, &next\_token);

if (p != NULL)

{

//визначення типу об'єкта

if (!strcmp(p, "PuncktLine"))

{

pcshape[n] = new PuncktLineShape;

}

else if (!strcmp(p, "Cube"))

{

pcshape[n] = new CubeShape;

}

else if (!strcmp(p, "Line"))

{

pcshape[n] = new LineShape;

}

else if (!strcmp(p, "Ellipse"))

{

pcshape[n] = new EllipseShape;

}

else if (!strcmp(p, "Cilinder"))

{

pcshape[n] = new CilinderShape;

}

else if (!strcmp(p, "Rect"))

{

pcshape[n] = new RectShape;

}

else if (!strcmp(p, "Romb"))

{

pcshape[n] = new RombShape;

}

else

{

fl = false;

}

p = strtok\_s(NULL, delimiter, &next\_token);

if (fl)

{

//зчитування координат та зміщення

while (p != NULL && c\_dots < 6)

{

dots[c\_dots] = atoi(p);

c\_dots++;

p = strtok\_s(NULL, delimiter, &next\_token);

}

if (c\_dots == 6)

{

pcshape[n]->Set(dots[0], dots[1], dots[2], dots[3], dots[4], dots[5], 0, 0);

c\_dots = 0;

n++;

}

}

else

fl = true;

}

}

curr\_length = n;

file.close();

}

else

{

MessageBox(0, "Не вдалося відкрити файл", "Курсова робота", 0);

}

}

//позначає вибраний елемент у меню

void ShapeEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

}

//позначає вибраний елемент на панелі інструментів

void ShapeEditor::PressButton(HWND hWnd)

{

}

void ShapeEditor::OverMeesage()

{

MessageBox(0, "Закінчилось місце для фігур", "Курсова робота", 0);

}

//створення нової сцени

void ShapeEditor::CreateNewScene()

{

//видалення елементів з pcshape

for (int i = 0; i < curr\_length;i++)

{

pcshape[i] = NULL;

}

curr\_length = 0;

}

CilinderEditor.cpp:

#include "stdafx.h"

#include "CilinderEditor.h"

CilinderEditor::CilinderEditor(COLORREF brushColor,COLORREF peColor):ShapeEditor(brushColor,peColor)

{

}

CilinderEditor::~CilinderEditor()

{

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void CilinderEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pcshape[curr\_length] = new CilinderShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt\_old.x, pt\_old.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//функція орбробки повідомлення руху миші

void CilinderEditor::OnMosuseMove(HWND hWnd)

{

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

//прорахунок координат фігури, яку треба замалювати

int xs\_A = pt\_old.x,

xs\_B = pt\_start.x,

ys\_A = pt\_start.y + (pt\_old.y - pt\_start.y) / 6,

ys\_B = pt\_old.y - (pt\_old.y - pt\_start.y) / 6,

xs\_C = pt\_start.x,

xs\_D = pt\_old.x,

ys\_C = pt\_start.y + (pt\_old.y - pt\_start.y) / 12,

ys\_D = pt\_old.y - (pt\_old.y - pt\_start.y) / 12;

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовуємо стару фігуру

Ellipse(hdc, xs\_B, ys\_B, pt\_old.x, pt\_old.y);

Rectangle(hdc, xs\_C, ys\_C, xs\_D, ys\_D);

Ellipse(hdc, pt\_start.x, pt\_start.y, xs\_A, ys\_A);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//прорахунок координат фігури, яку треба намалювати

xs\_A = pt\_old.x,

xs\_B = pt\_start.x,

ys\_A = pt\_start.y + (pt\_old.y - pt\_start.y) / 6,

ys\_B = pt\_old.y - (pt\_old.y - pt\_start.y) / 6,

xs\_C = pt\_start.x,

xs\_D = pt\_old.x,

ys\_C = pt\_start.y + (pt\_old.y - pt\_start.y) / 12,

ys\_D = pt\_old.y - (pt\_old.y - pt\_start.y) / 12;

//малюється новий гумовий слід

Ellipse(hdc, xs\_B, ys\_B, pt\_old.x, pt\_old.y);//нижний еллипс

Rectangle(hdc, xs\_C, ys\_C, xs\_D, ys\_D);

Ellipse(hdc, pt\_start.x, pt\_start.y, xs\_A, ys\_A);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc); //закриваємо контекст вікна

}

//позначає вибраний елемент у меню

void CilinderEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

CheckMenuItem(hSubMenu, IDM\_ROMB, MF\_UNCHECKED); //зняти позначку

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

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

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

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

CheckMenuItem(hSubMenu, IDM\_CILINDER, MF\_CHECKED);//позначити цей пункт

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

}

}

//позначає вибраний елемент на панелі інструментів

void CilinderEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

CubeEditor.cpp:

#include "stdafx.h"

#include "CubeEditor.h"

CubeEditor::CubeEditor(COLORREF brushColor, COLORREF peColor) :ShapeEditor(brushColor, peColor)

{

}

CubeEditor::~CubeEditor()

{

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void CubeEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pcshape[curr\_length] = new CubeShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt\_old.x, pt\_old.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//функція орбробки повідомлення руху миші

void CubeEditor::OnMosuseMove(HWND hWnd)

{

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

//прорахунок координат фігури, яку треба замалювати

int xs\_B = pt\_old.x + (pt\_start.x - pt\_old.x) / 3,

ys\_B = pt\_old.y - (pt\_old.y - pt\_start.y) / 3,

xs\_D = pt\_start.x - (pt\_start.x - pt\_old.x) / 3,

ys\_D = pt\_start.y + (pt\_old.y - pt\_start.y) / 3;

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовуємо стару фігуру

Rectangle(hdc, xs\_B, ys\_B, pt\_start.x, pt\_start.y);

Rectangle(hdc, xs\_D, ys\_D, pt\_old.x, pt\_old.y);

MoveToEx(hdc, pt\_old.x, pt\_old.y, NULL);

LineTo(hdc, xs\_B, ys\_B);

MoveToEx(hdc, pt\_old.x, ys\_D, NULL);

LineTo(hdc, xs\_B, pt\_start.y);

MoveToEx(hdc, xs\_D, pt\_old.y, NULL);

LineTo(hdc, pt\_start.x, ys\_B);

MoveToEx(hdc, xs\_D, ys\_D, NULL);

LineTo(hdc, pt\_start.x, pt\_start.y);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//прорахунок координат фігури, яку треба намалювати

xs\_B = pt\_old.x + (pt\_start.x - pt\_old.x) / 3;

ys\_B = pt\_old.y - (pt\_old.y - pt\_start.y) / 3;

xs\_D = pt\_start.x - (pt\_start.x - pt\_old.x) / 3;

ys\_D = pt\_start.y + (pt\_old.y - pt\_start.y) / 3;

//малюється новий гумовий слід

Rectangle(hdc, xs\_B, ys\_B, pt\_start.x, pt\_start.y);

Rectangle(hdc, xs\_D, ys\_D, pt\_old.x, pt\_old.y);

MoveToEx(hdc, pt\_old.x, pt\_old.y, NULL);

LineTo(hdc, xs\_B, ys\_B);

MoveToEx(hdc, pt\_old.x, ys\_D, NULL);

LineTo(hdc, xs\_B, pt\_start.y);

MoveToEx(hdc, xs\_D, pt\_old.y, NULL);

LineTo(hdc, pt\_start.x, ys\_B);

MoveToEx(hdc, xs\_D, ys\_D, NULL);

LineTo(hdc, pt\_start.x, pt\_start.y);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc); //закриваємо контекст вікна

}

//позначає вибраний елемент у меню

void CubeEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

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

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

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

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

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void CubeEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

EllipseEditor.cpp:

#include "stdafx.h"

#include "EllipseEditor.h"

EllipseEditor::EllipseEditor(COLORREF brushColor, COLORREF peColor) :ShapeEditor(brushColor, peColor)

{

}

EllipseEditor::~EllipseEditor()

{

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void EllipseEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pcshape[curr\_length] = new EllipseShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt.x, pt.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//функція орбробки повідомлення руху миші

void EllipseEditor::OnMosuseMove(HWND hWnd)

{

//Ellipse(hdc, xstart - (pt.x - xstart), ystart - (pt.y - ystart), pt.x, pt.y);

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовуємо стару фігуру

Ellipse(hdc, pt\_start.x, pt\_start.y, pt\_old.x, pt\_old.y);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//малюється новий гумовий слід

Ellipse(hdc, pt\_start.x, pt\_start.y, pt\_old.x, pt\_old.y);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc); //закриваємо контекст вікна

}

//позначає вибраний елемент у меню

void EllipseEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

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

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

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

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

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void EllipseEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

LineEditor.cpp:

#include "stdafx.h"

#include "LineEditor.h"

LineEditor::LineEditor(COLORREF brushColor, COLORREF peColor) :ShapeEditor(brushColor, peColor)

{

}

LineEditor::~LineEditor()

{

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void LineEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

//PrintInFile("Лінія", pt\_start.x, pt\_start.y, pt.x, pt.y);

pcshape[curr\_length] = new LineShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt.x, pt.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//функція орбробки повідомлення руху миші

void LineEditor::OnMosuseMove(HWND hWnd)

{

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовується стара фігура

MoveToEx(hdc, pt\_start.x, pt\_start.y, NULL);

LineTo(hdc, pt\_old.x, pt\_old.y);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//малюється новий гумовий слід

MoveToEx(hdc, pt\_start.x, pt\_start.y, NULL);

LineTo(hdc, pt.x, pt.y);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

//позначає вибраний елемент у меню

void LineEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

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

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

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

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED); //позначити цей пункт

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void LineEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

PuncktLineEditor.cpp:

#include "stdafx.h"

#include "PuncktLineEditor.h"

PuncktLineEditor::PuncktLineEditor(COLORREF brushColor, COLORREF peColor) :LineEditor(brushColor, peColor)

{

}

PuncktLineEditor::~PuncktLineEditor()

{

}

//позначає вибраний елемент у меню

void PuncktLineEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

CheckMenuItem(hSubMenu, IDM\_ROMB, MF\_UNCHECKED); //зняти позначку

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

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

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED); //позначити цей пункт

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void PuncktLineEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, true);

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void PuncktLineEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

//PrintInFile("Пунктирна лінія", pt\_start.x, pt\_start.y, pt.x, pt.y);

pcshape[curr\_length] = new PuncktLineShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt.x, pt.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

RectEditor.cpp:

#include "stdafx.h"

#include "RectEditor.h"

RectEditor::RectEditor(COLORREF brushColor, COLORREF peColor) :ShapeEditor(brushColor, peColor)

{

}

RectEditor::~RectEditor()

{

}

//функція орбробки повідомлення руху миші

void RectEditor::OnMosuseMove(HWND hWnd)

{

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовується стара фігура

Rectangle(hdc, pt\_start.x , pt\_start.y, pt\_old.x, pt\_old.y);

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//малюється новий гумовий слід

Rectangle(hdc, pt\_start.x , pt\_start.y, pt\_old.x, pt\_old.y);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc); //закриваємо контекст вікна

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void RectEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

//PrintInFile("Прямокутник", pt\_start.x, pt\_start.y, pt.x, pt.y);

pcshape[curr\_length] = new RectShape();

pcshape[curr\_length]->Set(pt\_start.x , pt\_start.y , pt\_old.x, pt\_old.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//позначає вибраний елемент у меню

void RectEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

CheckMenuItem(hSubMenu, IDM\_ROMB, MF\_UNCHECKED); //зняти позначку

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

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

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED); //позначити цей пункт

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void RectEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

RombEditor.cpp:

#include "stdafx.h"

#include "RombEditor.h"

RombEditor::RombEditor(COLORREF brushColor, COLORREF peColor) :ShapeEditor(brushColor, peColor)

{

}

RombEditor::~RombEditor()

{

}

//позначає вибраний елемент у меню

void RombEditor::OnInitMenuPopup(HWND hWnd, WPARAM wParam)

{

HMENU hMenu, hSubMenu;

hMenu = GetMenu(hWnd);

hSubMenu = GetSubMenu(hMenu, 1); //POPUP-меню Об'єкти

if ((HMENU)wParam == hSubMenu)

{

CheckMenuItem(hSubMenu, IDM\_ROMB, MF\_CHECKED); //зняти позначку

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

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

CheckMenuItem(hSubMenu, IDM\_ELLIPSE, MF\_UNCHECKED); //позначити цей пункт

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

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

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

}

}

//позначає вибраний елемент на панелі інструментів

void RombEditor::PressButton(HWND hWnd)

{

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ROMB, true);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_LINE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_RECT, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_ELLIPSE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CUBE, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_CILINDER, false);

SendMessage(hWnd, TB\_PRESSBUTTON, IDB\_PUNCKT, false);

}

/\*

\* функція орбробки повідомлення натиснення лівої клавіши миші

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void RombEditor::OnLBup(HWND hWnd, int xk, int yk)

{

//перевірка наявності місця у масиві

if (curr\_length < MY\_SHAPE\_ARRAY\_SIZE)

{

POINT pt;

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

//PrintInFile("Ромб", pt\_start.x, pt\_start.y, pt.x, pt.y);

pcshape[curr\_length] = new RombShape();

pcshape[curr\_length]->Set(pt\_start.x, pt\_start.y, pt\_old.x, pt\_old.y, brColor, penColor, xk, yk);//ініціалізація змінної

curr\_length++;//збільшення лічильника елементів у массиві pcshape

}

else

OverMeesage();

InvalidateRect(hWnd, NULL, TRUE);//вікликається WM\_PAINT

}

//функція орбробки повідомлення руху миші

void RombEditor::OnMosuseMove(HWND hWnd)

{

POINT pt;

HPEN hPenOld, hPen;

HDC hdc;

//прорахунок координат фігури, яку треба замалювати

int xs\_A = pt\_start.x - (pt\_old.x - pt\_start.x),

xs\_B = pt\_start.x,

ys\_A = pt\_old.y,

ys\_B = pt\_start.y + ((pt\_old.y - pt\_start.y) \* 2);

hdc = GetDC(hWnd); //отримуємо контекст вікна для малювання

SetROP2(hdc, R2\_NOTXORPEN);

hPen = CreatePen(PS\_DOT, 1, 0);

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

//замальовується стара фігура

MoveToEx(hdc, pt\_start.x, pt\_start.y, NULL);

LineTo(hdc, xs\_A, ys\_A);

LineTo(hdc, xs\_B, ys\_B);

LineTo(hdc, pt\_old.x, pt\_old.y);

LineTo(hdc, pt\_start.x, pt\_start.y);

//Малюються лінії "гумового" сліду попереднього розташування курсору

GetCursorPos(&pt);

ScreenToClient(hWnd, &pt);

pt\_old = pt;

//прорахунок координат фігури, яку треба намалювати

xs\_A = pt\_start.x - (pt\_old.x - pt\_start.x);

xs\_B = pt\_start.x;

ys\_A = pt\_old.y;

ys\_B = pt\_start.y + ((pt\_old.y - pt\_start.y) \* 2);

//малюється новий гумовий слід

MoveToEx(hdc, pt\_start.x, pt\_start.y, NULL);

LineTo(hdc, xs\_A, ys\_A);

LineTo(hdc, xs\_B, ys\_B);

LineTo(hdc, pt.x, pt.y);

LineTo(hdc, pt\_start.x, pt\_start.y);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

ReleaseDC(hWnd, hdc);

}

Shape.cpp:

#include "stdafx.h"

#include "Shape.h"

/\*

\* ініціалізу об'єкт shape

\* x1 - х координата початкової точки

\* у1 - у координата початкової точки

\* x2 - х координата кінцевої точки

\* у2 - у координата кінцевої точки

\* brushColor - колір заливки

\* peColor - колір контуру

\* xk - х зміщення

\* yk - у зміщення

\*/

void Shape::Set(long x1, long y1, long x2, long y2,COLORREF brushColor,COLORREF peColor, int xk, int yk)

{

xs1 = x1 - xk;

ys1 = y1 - yk;

xs2 = x2 - xk;

ys2 = y2 - yk;

brColor = brushColor;

penColor = peColor;

}

//повертає дані про об'єкт shape

std::string Shape::RetData()

{

std::string value = "";

value = type + "\t" + std::to\_string(xs1) + "\t" + std::to\_string(ys1) + "\t" + std::to\_string(xs2) + "\t" + std::to\_string(ys2) + "\t" + std::to\_string(brColor) + "\t" + std::to\_string(penColor);

return value;

}

CilinderShape.cpp:

#include "stdafx.h"

#include "CilinderShape.h"

CilinderShape::CilinderShape()

{

type = "Cilinder";

}

CilinderShape::~CilinderShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void CilinderShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

hBrush = (HBRUSH)CreateSolidBrush(brColor);

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

//прорахунок координат фігури

int xs\_1 = xs1 + xk,

xs\_2 = xs2 + xk,

ys\_1 = ys1 + yk,

ys\_2 = ys2 + yk,

xs\_A = xs\_2,

xs\_B = xs\_1,

ys\_A = ys\_1 + (ys\_2 - ys\_1) / 6,

ys\_B = ys\_2 - (ys\_2 - ys\_1) / 6,

xs\_C = xs\_1,

xs\_D = xs\_2,

ys\_C = ys\_1 + (ys\_2 - ys\_1) / 12,

ys\_D = ys\_2 - (ys\_2 - ys\_1) / 12,

xs\_E = xs\_B,

ys\_E = ys\_D,

koef = -2;

Ellipse(hdc, xs\_B, ys\_B, xs\_2, ys\_2);

Rectangle(hdc, xs\_C, ys\_C, xs\_D, ys\_D);

Ellipse(hdc, xs\_1, ys\_1, xs\_A, ys\_A);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

hPen = CreatePen(PS\_SOLID, 2, brColor);

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

if (xs\_1 > xs\_2)

koef = 2;

MoveToEx(hdc, xs\_E + koef / (-2), ys\_E, NULL);

LineTo(hdc, xs\_D + koef, ys\_D);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}

CubeShape.cpp:

#include "stdafx.h"

#include "CubeShape.h"

CubeShape::CubeShape()

{

type = "Cube";

}

CubeShape::~CubeShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void CubeShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

hBrush = (HBRUSH)CreateSolidBrush(brColor);

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

//прорахунок координат фігури

int xs\_1 = xs1 + xk,

xs\_2 = xs2 + xk,

ys\_1 = ys1 + yk,

ys\_2 = ys2 + yk,

xs\_B = xs\_2 + (xs\_1 - xs\_2) / 3,

ys\_B = ys\_2 - (ys\_2 - ys\_1) / 3,

xs\_D = xs\_1 - (xs\_1 - xs\_2) / 3,

ys\_D = ys\_1 + (ys\_2 - ys\_1) / 3,

xs\_A = xs\_2 + (xs\_1 - xs\_2) / 3,

ys\_A = ys\_1,

xs\_E = xs\_2,

ys\_E = ys\_D,

xs\_C = xs\_D,

ys\_C = ys\_2,

xs\_F = xs\_1,

ys\_F = ys\_B;

//паралелограм верхний

POINT pt\_re[7];

pt\_re[0].x = xs\_1;

pt\_re[1].x = xs\_A;

pt\_re[2].x = xs\_E;

pt\_re[3].x = xs\_D;

pt\_re[0].y = ys\_1;

pt\_re[1].y = ys\_A;

pt\_re[2].y = ys\_E;

pt\_re[3].y = ys\_D;

Polygon(hdc, pt\_re, 4);

//паралелограм боковой

pt\_re[0].x = xs\_1;

pt\_re[1].x = xs\_F;

pt\_re[2].x = xs\_C;

pt\_re[3].x = xs\_D;

pt\_re[0].y = ys\_1;

pt\_re[1].y = ys\_F;

pt\_re[2].y = ys\_C;

pt\_re[3].y = ys\_D;

Polygon(hdc, pt\_re, 4);

//прямоугольник

pt\_re[0].x = xs\_E;

pt\_re[1].x = xs\_2;

pt\_re[2].x = xs\_C;

pt\_re[3].x = xs\_D;

pt\_re[0].y = ys\_E;

pt\_re[1].y = ys\_2;

pt\_re[2].y = ys\_C;

pt\_re[3].y = ys\_D;

Polygon(hdc, pt\_re, 4);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}

EllipseShape.cpp:

#include "stdafx.h"

#include "EllipseShape.h"

EllipseShape::EllipseShape()

{

type = "Ellipse";

}

EllipseShape::~EllipseShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void EllipseShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

hBrush = (HBRUSH)CreateSolidBrush(brColor);

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

Ellipse(hdc, xs1 + xk, ys1 + yk, xs2 + xk, ys2 + yk);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}

LineShape.cpp:

#include "stdafx.h"

#include "LineShape.h"

LineShape::LineShape()

{

type = "Line";

}

LineShape::~LineShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void LineShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

MoveToEx(hdc,xs1 + xk,ys1 + yk, NULL);

LineTo(hdc, xs2 + xk, ys2 + yk);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

}

PuncktLineShape.cpp:

#include "stdafx.h"

#include "PuncktLineShape.h"

PuncktLineShape::PuncktLineShape()

{

type = "PuncktLine";

}

PuncktLineShape::~PuncktLineShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void PuncktLineShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

hPen = CreatePen(PS\_DOT, 1, penColor);

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

MoveToEx(hdc, xs1 + xk, ys1 + yk, NULL);

LineTo(hdc, xs2 + xk, ys2 + yk);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

}

RectShape.cpp:

#include "stdafx.h"

#include "RectShape.h"

RectShape::RectShape()

{

type = "Rect";

}

RectShape::~RectShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void RectShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

hBrush = (HBRUSH)CreateSolidBrush(brColor);

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

Rectangle(hdc,xs1 + xk, ys1 + yk, xs2 + xk, ys2+ yk);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}

RombShape.cpp:

#include "stdafx.h"

#include "RombShape.h"

RombShape::RombShape()

{

type = "Romb";

}

RombShape::~RombShape()

{

}

/\*

\* малює об'єкт shape

\* xk - зміщення по х

\* yk - зміщення по у

\*/

void RombShape::Show(HDC hdc, int xk, int yk)

{

HPEN hPenOld, hPen;

HBRUSH hBrush, hBrushOld;

hPen = CreatePen(PS\_SOLID, 1, penColor);

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

hBrush = (HBRUSH)CreateSolidBrush(brColor);

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

POINT pt\_re[4];

//прорахунок координат фігури

int xs\_1 = xs1 + xk,

xs\_2 = xs2 + xk,

ys\_1 = ys1 + yk,

ys\_2 = ys2 + yk,

xs\_A = xs\_1 - (xs\_2 - xs\_1),

ys\_A = ys\_2,

xs\_B = xs\_1,

ys\_B = ys\_1 + ((ys\_2 - ys\_1)\* 2);

pt\_re[0].x = xs\_A;

pt\_re[1].x = xs\_B;

pt\_re[2].x = xs\_2;

pt\_re[3].x = xs\_1;

pt\_re[0].y = ys\_A;

pt\_re[1].y = ys\_B;

pt\_re[2].y = ys\_2;

pt\_re[3].y = ys\_1;

Polygon(hdc, pt\_re, 4);

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

SelectObject(hdc, hBrushOld);

DeleteObject(hBrush);

}