Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра ОТ

ЗВІТ

про виконання комп’ютерного практикуму № 7

з дисципліни

“Об’єктно орієнтоване програмування ”

Тема: «Побудування програмної системи з множини об’єктів, керованих повідомленнями»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Порєв В.М. |  | студент 2-го курсу  гр. ІП-54 ФІОТ  Макаренко Антон Олександрович |

Київ 2016

**Мета роботи:** отримати вміння та навички використовувати засоби обміну інформацією та запрограмувати взаємодію незалежно працюючих програмних компонентів.  
**Завдання:**1. Створити у середовищі MS Visual Studio C++ проект Win32 з ім’ям **Lab7**.  
2. Написати вихідні тексти усіх програм-компонентів згідно варіанту завдання.  
3. Скомпілювати вихідні тексти і отримати виконувані файли програм.  
4. Перевірити роботу програм. Налагодити взаємодію програм.  
5. Проаналізувати та прокоментувати результати та вихідні тексти програм.  
6. Оформити звіт.

|  |  |  |  |
| --- | --- | --- | --- |
| Номер варіанту | Програма Lab7 | Програма Object2 | Програма Object3 |
| 3 | 1. Користувач вводить значення n, Min, Max у діалоговому вікні.  2. Програма ініціює обмін повідомленнями з програмами Object2, 3 | 1. Створює вектор n дробових (double) чисел у діапазоні Min – Max  2. Показує числові значення у декількох стовпчиках та рядках у власному головному вікні  3. Записує дані у Clipboard Windows у текстовому форматі | 1. Зчитує дані з Clipboard Windows  2. Відображає графік y=f(x) у власному головному вікні Значення y – це значення вектора, x – це індекси елементів |

Лістинг

Lab7.cpp

// Lab7.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "Lab7.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

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

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

HWND hWndDest = NULL;

HWND hWndSrc;

// 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\_PTR CALLBACK DlgMsg(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam);

void CallObject3();

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPTSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

MSG msg;

HACCEL hAccelTable;

// Инициализация глобальных строк

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

LoadString(hInstance, IDC\_LAB7, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB7));

// Цикл основного сообщения:

while (GetMessage(&msg, NULL, 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)

{

WNDCLASSEX 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\_LAB7));

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

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

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_LAB7);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassEx(&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)

{

HWND hWnd;

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

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

hWndSrc = hWnd;

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 - обработка меню приложения

// WM\_PAINT -Закрасить главное окно

// WM\_DESTROY - ввести сообщение о выходе и вернуться.

//

//

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

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

switch (message)

{

break;

case WM\_CREATE:

EnableMenuItem(GetMenu(hWnd), IDM\_OBJ3, true);

break;

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_OBJ2:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_DLG), hWnd, DlgMsg);

break;

case IDM\_OBJ3:

CallObject3();

break;

case 500:

if ((int)lParam == 1)

EnableMenuItem(GetMenu(hWnd), IDM\_OBJ3, false);

default:

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

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

// TODO: добавьте любой код отрисовки...

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;

}

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

{

COPYDATASTRUCT cds;

char\* str = new char[7];

int par[3]{0, 0, 0};

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

switch (LOWORD(wParam))

{

case IDOK:

GetDlgItemText(hDlg, IDC\_EDIT\_N, str, 7);

//GetWindowText(GetDlgItem(hDlg, IDC\_EDIT\_N), LPWSTR(str), 7);

par[0] = atoi((char\*)str);

GetDlgItemText(hDlg, IDC\_EDIT\_MIN, str, 7);

//GetWindowText(GetDlgItem(hDlg, IDC\_EDIT\_MIN), LPWSTR(str), 7);

par[1] = atoi((char\*)str);

GetDlgItemText(hDlg, IDC\_EDIT\_MAX, str, 7);

//GetWindowText(GetDlgItem(hDlg, IDC\_EDIT\_MAX), LPWSTR(str), 7);

par[2] = atoi((char\*)str);

if (par[0] == 0 || par[1] > par[2])

{

MessageBox(hWndSrc, "Некоректні дані", "Помилка", 0);

break;

}

hWndDest = FindWindow("OBJECT2", NULL);

if (!hWndDest)

{

WinExec("Object2.exe", SW\_SHOW);

hWndDest = FindWindow("OBJECT2", NULL);

}

SetForegroundWindow(hWndDest);

cds.dwData = 1;

cds.cbData = sizeof(par);

cds.lpData = par;

SendMessage(hWndDest, WM\_COPYDATA, (WPARAM)hWndSrc, (LPARAM)&cds);

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

break;

case IDCANCEL:

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

break;

default:

break;

}

break;

}

return (INT\_PTR)FALSE;

}

void CallObject3()

{

hWndDest = FindWindow("OBJECT3", NULL);

if (!hWndDest)

{

WinExec("Object3.exe", SW\_SHOW);

hWndDest = FindWindow("OBJECT3", NULL);

}

SetForegroundWindow(hWndDest);

}

Object2.cpp

// Object2.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "Object2.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

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

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

vector<double> vct;

HWND hWndDest;

HWND hWndL7;

//int numN;

// 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);

void OnCopyData(HWND hWnd, WPARAM wParam, LPARAM lParam);

int ToClipBoard(HWND hWnd);

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPTSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

MSG msg;

HACCEL hAccelTable;

// Инициализация глобальных строк

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

LoadString(hInstance, IDC\_OBJECT2, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_OBJECT2));

// Цикл основного сообщения:

while (GetMessage(&msg, NULL, 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)

{

WNDCLASSEX 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\_OBJECT2));

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

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

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_OBJECT2);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassEx(&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)

{

HWND hWnd;

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

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

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 - обработка меню приложения

// WM\_PAINT -Закрасить главное окно

// WM\_DESTROY - ввести сообщение о выходе и вернуться.

//

//

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

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

RECT rc;

int a = 0;

int k = 0;

char\* buf = new char[20];

switch (message)

{

case WM\_COPYDATA:

hWndL7 = (HWND)wParam;

OnCopyData(hWnd, wParam, lParam);

break;

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_CD:

ToClipBoard(hWnd);

PostMessage(hWndL7, WM\_COMMAND, (WPARAM)500, (LPARAM)1);

//SetForegroundWindow(hWndL7);

break;

default:

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

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

GetClientRect(hWnd, &rc);

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

{

sprintf\_s(buf, 20, "%f", vct[i]);

if (a != ((20 \* i) / rc.bottom))

{

a = ((20 \* i) / rc.bottom);

k = 0;

}

TextOut(hdc, 10 + 100 \* a, 10 + 20 \* k, buf, strlen(buf));

k++;

}

// TODO: добавьте любой код отрисовки...

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;

}

void OnCopyData(HWND hWnd, WPARAM wParam, LPARAM lParam)

{

vct.clear();

COPYDATASTRUCT \*cds;

cds = (COPYDATASTRUCT \*)lParam;

int \*p = (int \*)cds->lpData;

int n = p[0];

int min = p[1];

int max = p[2];

srand(time(0));

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

vct.push\_back((double)(rand()) / RAND\_MAX\*(max - min) + min);

InvalidateRect(hWnd, 0, true);

}

int ToClipBoard(HWND hWnd)

{

HGLOBAL hglbCopy;

BYTE \*pTmp;

long len = sizeof(double)\*vct.size();

if (len == 0)

return 0;

int bufsize = vct.size() \* 20 + 10;

char\* buf = new char[bufsize];

char\* b = new char[20];

buf[0] = '\0';

//\_itoa\_s(vct.size(), b, 20, 10);

//strcat\_s(buf, bufsize, b);

//strcat\_s(buf, bufsize, " ");

////strcat\_s(buf)

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

{

sprintf\_s(b, 20, "%f", vct[i]);

strcat\_s(buf, bufsize, b);

strcat\_s(buf, bufsize, " ");

}

hglbCopy = GlobalAlloc(GHND, bufsize);

if (hglbCopy == NULL)

return FALSE;

pTmp = (BYTE \*)GlobalLock(hglbCopy);

memcpy(pTmp, buf, bufsize);

GlobalUnlock(hglbCopy);

if (!OpenClipboard(hWnd))

{

GlobalFree(hglbCopy);

return 0;

}

EmptyClipboard();

SetClipboardData(CF\_TEXT, hglbCopy);

CloseClipboard();

}

Object3.cpp

// Object3.cpp: определяет точку входа для приложения.

//

#include "stdafx.h"

#include "Object3.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

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

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

vector<double> vct;

// 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);

void OnPaint(HWND hWnd, HDC hdc);

bool GetDataFromClipboar(HWND hWnd);

int APIENTRY \_tWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPTSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: разместите код здесь.

MSG msg;

HACCEL hAccelTable;

// Инициализация глобальных строк

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

LoadString(hInstance, IDC\_OBJECT3, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance(hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_OBJECT3));

// Цикл основного сообщения:

while (GetMessage(&msg, NULL, 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)

{

WNDCLASSEX 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\_OBJECT3));

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

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

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_OBJECT3);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassEx(&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)

{

HWND hWnd;

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

hWnd = CreateWindow(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

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

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 - обработка меню приложения

// WM\_PAINT -Закрасить главное окно

// WM\_DESTROY - ввести сообщение о выходе и вернуться.

//

//

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

{

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

switch (message)

{

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_GRAPH:

GetDataFromClipboar(hWnd);

InvalidateRect(hWnd, NULL, true);

break;

default:

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

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

OnPaint(hWnd, hdc);

// TODO: добавьте любой код отрисовки...

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;

}

bool GetDataFromClipboar(HWND hWnd)

{

vct.clear();

int maxsize = 2000;

char\* dest = new char[maxsize];

HGLOBAL hglb;

LPTSTR lptstr;

long size, res;

res = 0;

if (!IsClipboardFormatAvailable(CF\_TEXT)) return 0;

if (!OpenClipboard(hWnd)) return 0;

hglb = GetClipboardData(CF\_TEXT);

if (hglb != NULL)

{

lptstr = (char \*)GlobalLock(hglb);

if (lptstr != NULL)

{

size = strlen(lptstr);

if (size > maxsize)

{

lptstr[maxsize] = 0;

size = strlen(lptstr);

}

strcpy\_s(dest, maxsize, lptstr);

res = size;

GlobalUnlock(hglb);

}

}

CloseClipboard();

char\* context = new char[30];

char\* buf = strtok\_s(dest, " ", &context);

try

{

while (buf)

{

vct.push\_back(atof(buf));

buf = strtok\_s(context, " ", &context);

}

}

catch (exception e)

{

MessageBox(hWnd, "Некоректні дані", "Помилка", 0);

return false;

}

}

void OnPaint(HWND hWnd, HDC hdc)

{

if (vct.size() == 0)

return;

RECT rc;

HPEN hPenOld, hPen;

COLORREF color;

char\* buf = new char[20];

int otsY = 30;

int otsX = 70;

GetClientRect(hWnd, &rc);

rc.left = otsX;

rc.top = otsY;

rc.bottom = rc.bottom - rc.top \* 2;

rc.right = rc.right - rc.left;

int dX = rc.right / vct.size();

double Ymin = vct[0];

double Ymax = vct[0];

double Ylen = 0;

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

{

Ymin = min(Ymin, vct[i]);

Ymax = max(Ymax, vct[i]);

}

Ylen = Ymax - Ymin;

double dY = (double)(rc.bottom - rc.top) / Ylen;

// Побудова системи кординат

//=============================

MoveToEx(hdc, rc.left, rc.top - 20, NULL);

LineTo(hdc, otsX, rc.bottom);

LineTo(hdc, rc.right + rc.top, rc.bottom);

LineTo(hdc, rc.right + rc.top - 10, rc.bottom - 10);

MoveToEx(hdc, rc.right + rc.top, rc.bottom, NULL);

LineTo(hdc, rc.right + rc.top - 10, rc.bottom + 10);

MoveToEx(hdc, otsX, otsY - 20, NULL);

LineTo(hdc, otsX - 10, otsY - 10);

MoveToEx(hdc, otsX, otsY - 20, NULL);

LineTo(hdc, otsX + 10, otsY - 10);

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

{

//Вісь X

MoveToEx(hdc, rc.left + dX\*i, rc.bottom - 5, NULL);

LineTo(hdc, otsX + dX\*i, rc.bottom + 5);

\_itoa\_s(i, buf, 20, 10);

TextOut(hdc, rc.left + dX\*i - 2, rc.bottom + 7, buf, strlen(buf));

//Вісь Y

MoveToEx(hdc, rc.left - 5, rc.bottom - (double)(vct[i] - Ymin) \* dY, NULL);

LineTo(hdc, rc.left + 5, rc.bottom - (double)(vct[i] - Ymin) \* dY);

sprintf\_s(buf, 20, "%f", vct[i]);

TextOut(hdc, 2, rc.bottom - (double)(vct[i] - Ymin) \* dY - 8, buf, 8);

if (i == 0)

continue;

// Графік

color = RGB(255, 0, 0);

hPen = CreatePen(0, 1, color);

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

MoveToEx(hdc, rc.left + dX\*(i - 1), rc.bottom - (double)(vct[i - 1] - Ymin) \* dY, NULL);

LineTo(hdc, rc.left + dX\*(i), rc.bottom - (double)(vct[i] - Ymin) \* dY);

if (vct[i] == Ymin)

{

SelectObject(hdc, hPenOld);

DeleteObject(hPen);

continue;

}

// Пунктир

color = RGB(127, 127, 127);

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

SelectObject(hdc, hPen);

MoveToEx(hdc, rc.left - 5, rc.bottom - (double)(vct[i] - Ymin) \* dY, NULL);

LineTo(hdc, rc.left + dX\*(i), rc.bottom - (double)(vct[i] - Ymin) \* dY);

LineTo(hdc, rc.left + dX\*(i), rc.bottom);

SelectObject(hdc, hPenOld);

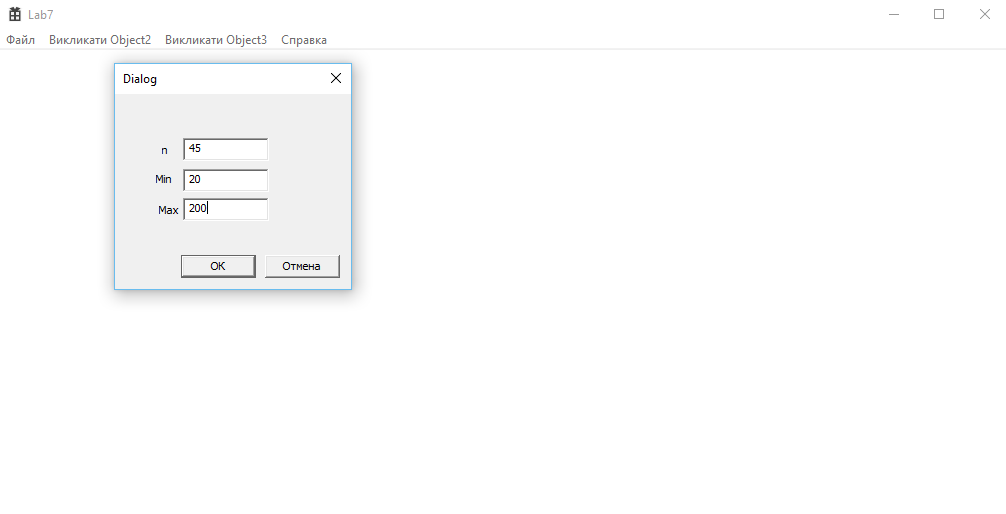
DeleteObject(hPen);

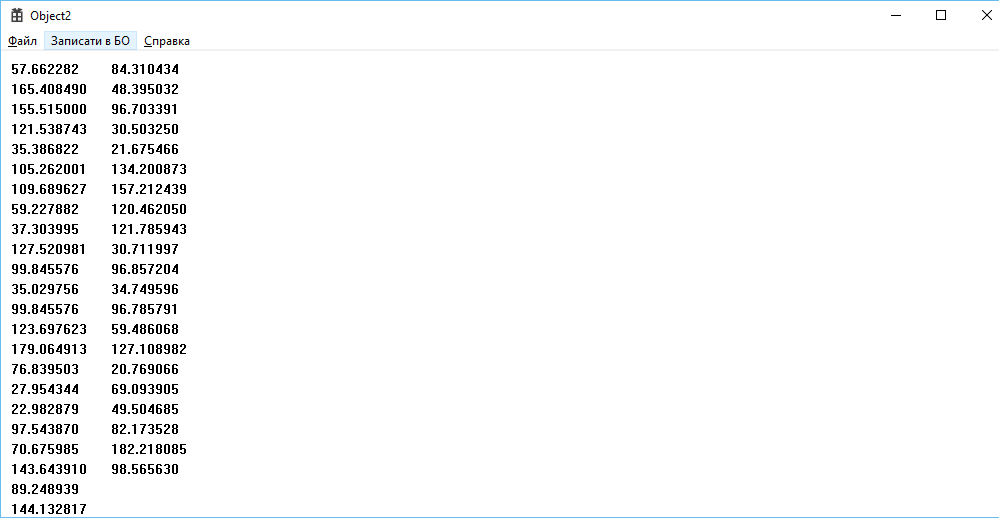
}

//=============================

}

Відеокопія результату





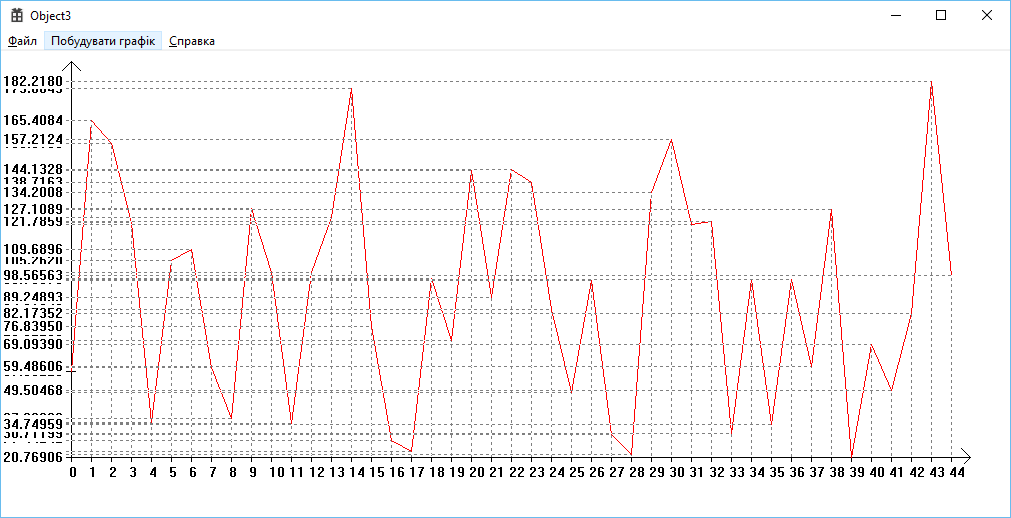
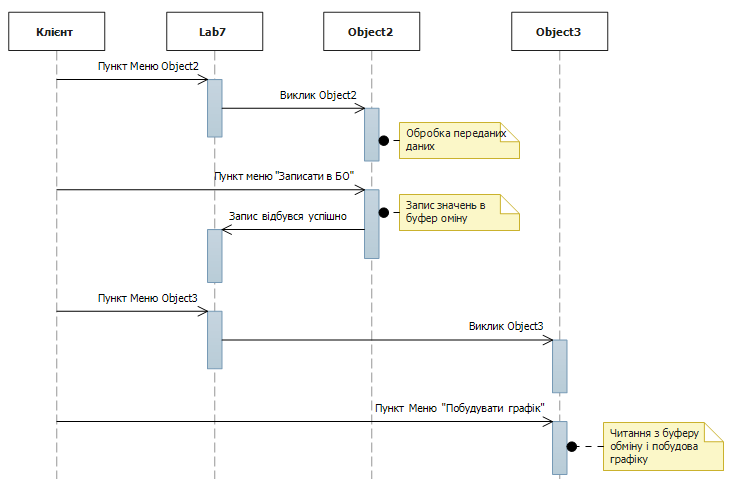


Схема послідовності надсилання-обробки повідомлень 

Висновок

В ході даної лабораторної роботи я отримав вміння та навички використовувати засоби обміну інформацією та запрограмував взаємодію незалежно працюючих програмних компонентів.