Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут”

Кафедра ОТ

**Лабораторна робота №12**

з дисципліни “СП”

Тема: «Прийоми поглибленого програмування. Формування віконного інтерфейсу»

|  |  |  |
| --- | --- | --- |
| Прийняв: |  | Виконав: |
| Павлов В. Г. |  | студент 2-го курсу  гр. ІП-53 ФІОТ  Сулима Олександр Сергійович |

Київ – 2016

**Мета роботи**

Вивчення прийомів поглибленого програмування по розробці віконного інтерфейсу.

**Порядок виконання роботи**

1. Вивчити методи розробки віконного інтерфейсу на мові Асемблер [1 - 4].

2. Розробити програму\* побудови вікна з наступними елементами інтерфейсу:

* текстове **popup**-меню, що складається з трьох пунктів, перший з яких дозволяє вибрати одну з лабораторних робіт першого циклу, другий – одну з лабораторних робіт другого циклу, третій, – вивести інформацію про програму і про її автора;
* кнопкове меню, кожна кнопка якого дозволяє вибрати одну з лабораторних робіт або інформацію про програму та про її автора, повторне натиснення тієї ж кнопки відміняє вибір і очищує вікно;
* рядок стану, в якому виводиться назва вибраного в даний момент пункту меню, при закритому меню або відсутності вибору рядок стану очищується;
* стандартні кнопки мінімізації, розгортання і закриття вікна.

3. При виборі за допомогою меню або кнопкою однією з лабораторних робіт повинен відкриватися відповідний файл, що містить текст основної програми вибраної лабораторної роботи, з виведенням цього тексту в полі вікна в режимі перегляду без можливості редагування.

4. Підготувати за допомогою графічного редактора піктограму розміром 32 х 32 пікселів, яку використовувати як іконку даної програми, причому вона повинна також відображуватися у верхньому лівому кутку вікна програми, що відкривається. Цю іконку розташувати у файлі ресурсів проекту.

5. Виконати компіляцію і перевірити виконання всіх описаних вище режимів роботи програми..

6. Привести лістинг програми, її окремих модулів і розроблених бібліотек (при їх наявності) в звіті по лабораторній роботі

7. У звіті по лабораторній роботі виконати опис послідовності розробки програми із посиланнями на відповідні фрагменти її тексту.

8. Привести в звіті по лабораторній роботі скріншот основного вікна програми.

9. Зробити висновки по лабораторній роботі.

Лістинг Sulyma-12.asm

.386

.model flat,stdcall

option casemap:none

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\user32.inc

include \masm32\include\Comctl32.inc

includelib \masm32\lib\Comctl32.lib

includelib \masm32\lib\user32.lib

includelib \masm32\lib\kernel32.lib

WinMain proto :DWORD

FileOpen proto :DWORD, :DWORD

MEMSIZE equ 65535

bHeigth equ 17

bWigth equ 70

.data

info db "Author: Sulyma Aleksandr", 10, "IP-53", 10, "5318",0

about db "About",0

ClassName db "Class",0

AppName db "Sulyma-12",0

MenuName db "Menu",0

EditClass db "Edit",0

BtnClass db "Button",0

lab1 db "1",0

lab2 db "2",0

lab3 db "3",0

lab4 db "4",0

lab5 db "5",0

lab6 db "6",0

lab7 db "7",0

lab8 db "8",0

lab9 db "9",0

lab10 db "10",0

lab11 db "11",0

lab12 db "12",0

File1 db "labs/Sulyma-1.asm",0

ttp1 db "Sulyma-1.asm",0

File2 db "labs/Sulyma-2.asm",0

ttp2 db "Sulyma-2.asm",0

File3 db "labs/Sulyma-3.asm",0

ttp3 db "Sulyma-3.asm",0

File4 db "labs/Sulyma-4.asm",0

ttp4 db "Sulyma-4.asm",0

File5 db "labs/Sulyma-5.asm",0

ttp5 db "Sulyma-5.asm",0

File6 db "labs/Sulyma-6.asm",0

ttp6 db "Sulyma-6.asm",0

File7 db "labs/Sulyma-7.asm",0

ttp7 db "Sulyma-7.asm",0

File8 db "labs/Sulyma-8.asm",0

ttp8 db "Sulyma-8.asm",0

File9 db "labs/Sulyma-9.asm",0

ttp9 db "Sulyma-9.asm",0

File10 db "labs/Sulyma-10.asm",0

ttp10 db "Sulyma-10.asm",0

File11 db "labs/Sulyma-11.asm",0

ttp11 db "Sulyma-11.asm",0

File12 db "labs/Sulyma-12.asm",0

ttp12 db "Sulyma-12.asm",0

check dw 0

hInstance HINSTANCE ?

hStatus HWND ?

hEdit HWND ?

hFile HANDLE ?

hMemory HANDLE ?

pMemory DWORD ?

SizeReadWrite DWORD ?

hlab1 HWND ?

hlab2 HWND ?

hlab3 HWND ?

hlab4 HWND ?

hlab5 HWND ?

hlab6 HWND ?

hlab7 HWND ?

hlab8 HWND ?

hlab9 HWND ?

hlab10 HWND ?

hlab11 HWND ?

hlab12 HWND ?

.code

start:

invoke GetModuleHandle, NULL

mov hInstance,eax

invoke WinMain, hInstance

invoke ExitProcess, eax

WinMain proc hInst:HINSTANCE

LOCAL wc:WNDCLASSEX

LOCAL msg:MSG

mov wc.cbSize,sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, offset WndProc

mov wc.cbClsExtra,NULL

mov wc.cbWndExtra,NULL

push hInstance

pop wc.hInstance

mov wc.hbrBackground,COLOR\_WINDOW

mov wc.lpszMenuName,offset MenuName

mov wc.lpszClassName,offset ClassName

invoke LoadIcon,hInstance,100

mov wc.hIcon,eax

mov wc.hIconSm,eax

invoke LoadCursor,NULL,IDC\_ARROW

mov wc.hCursor,eax

invoke RegisterClassEx, addr wc

invoke CreateWindowEx,0, ADDR ClassName, ADDR AppName, WS\_OVERLAPPEDWINDOW or WS\_VISIBLE, 100, 100, 755, 600, NULL, NULL, hInst, NULL

.WHILE TRUE

invoke GetMessage, ADDR msg,NULL,0,0

.BREAK .IF (!eax)

invoke DispatchMessage, ADDR msg

.ENDW

mov eax,msg.wParam

ret

WinMain endp

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

local pos

.IF uMsg==WM\_CREATE

invoke CreateStatusWindow,WS\_CHILD or WS\_VISIBLE,NULL, hWnd, 200

mov hStatus, eax

INVOKE CreateWindowEx,NULL, ADDR EditClass, NULL,\

WS\_VISIBLE or WS\_CHILD or ES\_MULTILINE or WS\_VSCROLL or ES\_AUTOVSCROLL or WS\_HSCROLL or ES\_AUTOHSCROLL,\

0,\

0,\

0,\

0,\

hWnd,\

75,\

hInstance,\

NULL

mov ecx, 0

mov hEdit,eax

invoke CreateWindowEx,0, ADDR BtnClass,ADDR lab1,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,1,hInstance,0

mov hlab1, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab2,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,2,hInstance,0

mov hlab2, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab3,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,3,hInstance,0

mov hlab3, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab4,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,4,hInstance,0

mov hlab4, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab5,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,5,hInstance,0

mov hlab5, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab6,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,6,hInstance,0

mov hlab6, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab7,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,7,hInstance,0

mov hlab7, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab8,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,8,hInstance,0

mov hlab8, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab9,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,9,hInstance,0

mov hlab9, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab10,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,10,hInstance,0

mov hlab10, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab11,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,11,hInstance,0

mov hlab11, eax

add ecx, bWigth

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab12,WS\_CHILD or WS\_VISIBLE,ecx,0,bWigth,bHeigth,hWnd,12,hInstance,0

mov hlab12, eax

.ELSEIF uMsg==WM\_COMMAND

mov eax,wParam

.IF ax==1

invoke FileOpen, offset File1,offset ttp1

.ELSEIF ax==2

invoke FileOpen, offset File2,offset ttp2

.ELSEIF ax==3

invoke FileOpen, offset File3,offset ttp3

.ELSEIF ax==4

invoke FileOpen, offset File4,offset ttp4

.ELSEIF ax==5

invoke FileOpen, offset File5,offset ttp5

.ELSEIF ax==6

invoke FileOpen, offset File6,offset ttp6

.ELSEIF ax==7

invoke FileOpen, offset File7,offset ttp7

.ELSEIF ax==8

invoke FileOpen, offset File8,offset ttp8

.ELSEIF ax==9

invoke FileOpen, offset File9,offset ttp9

.ELSEIF ax==10

invoke FileOpen, offset File10,offset ttp10

.ELSEIF ax==11

invoke FileOpen, offset File11,offset ttp11

.ELSEIF ax==12

invoke FileOpen, offset File12,offset ttp12

.ELSEIF ax==13

invoke MessageBoxA,0, offset info, offset about,0

.ELSEIF ax==90

invoke DestroyWindow,hWnd

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

.ELSEIF uMsg == WM\_SIZE

mov eax,lParam

mov edx,eax

shr edx,16

and eax,0ffffh

sub edx, 40

sub eax, 0

mov pos, 0

invoke MoveWindow,hEdit,0,20,eax,edx,TRUE

mov pos, 0

invoke MoveWindow,hlab1,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab2,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab3,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab4,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab5,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab6,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab7,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab8,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab9,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab10,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab11,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hlab12,pos,0,bWigth,bHeigth,TRUE

add pos, bWigth

invoke MoveWindow,hStatus,0,0,0,0,TRUE

.ELSEIF uMsg==WM\_DESTROY

invoke PostQuitMessage,NULL

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

xor eax,eax

ret

WndProc endp

FileOpen proc filename:DWORD, message:DWORD

.IF check==ax

invoke SendMessage,hEdit,WM\_SETTEXT,NULL,NULL

invoke SendMessage,hStatus,WM\_SETTEXT,NULL, NULL

mov check, 0

.ELSE

mov check, ax

invoke CreateFile,filename,GENERIC\_READ,FILE\_SHARE\_READ,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_READONLY,NULL

mov hFile,eax

invoke GlobalAlloc,GMEM\_MOVEABLE or GMEM\_ZEROINIT,MEMSIZE

mov hMemory,eax

invoke GlobalLock,hMemory

mov pMemory,eax

invoke ReadFile,hFile,pMemory,MEMSIZE-1,ADDR SizeReadWrite,NULL

invoke SendMessage,hEdit,WM\_SETTEXT,NULL,pMemory

invoke SendMessage,hStatus,WM\_SETTEXT,NULL, message

invoke CloseHandle,hFile

invoke GlobalUnlock,pMemory

invoke GlobalFree,hMemory

.ENDIF

ret 8

FileOpen endp

end start

Лістинг Sulyma.rc

100 ICON "logo.ico"

Menu MENU

{

POPUP "File"

{

MENUITEM "Exit",90

}

POPUP "First part"

{

MENUITEM "1",1

MENUITEM "2",2

MENUITEM "3",3

MENUITEM "4",4

MENUITEM "5",5

MENUITEM "6",6

}

POPUP "Second part"

{

MENUITEM "7",7

MENUITEM "8",8

MENUITEM "9",9

MENUITEM "10",10

MENUITEM "11",11

MENUITEM "12",12

}

POPUP "Help"

{

MENUITEM "About",13

}}

Послідовність розробки

1. Ініціалізуємо клас вікна, де якому задаємо: HINSTANCE процеса, вікна WndProc (функцію обробки повідомлень), колір вікна, назву меню, назву класа, стиль, іконку та курсор.

mov wc.cbSize,sizeof WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, offset WndProc

mov wc.cbClsExtra,NULL

mov wc.cbWndExtra,NULL

push hInstance

pop wc.hInstance

mov wc.hbrBackground,COLOR\_WINDOW

mov wc.lpszMenuName,offset MenuName

mov wc.lpszClassName,offset ClassName

invoke LoadIcon,hInstance,100

mov wc.hIcon,eax

mov wc.hIconSm,eax

invoke LoadCursor,NULL,IDC\_ARROW

mov wc.hCursor,eax

invoke RegisterClassEx, addr wc

1. Створену іконку задаєм в файлі ресурсів (Sulyma.rc), присвоюємо їй ID (100). Тут же створюємо меню, де задаєм пункти з їх ID.

100 ICON "ico.ico"

Menu MENU

{

POPUP "File"

{

MENUITEM "Exit",100

}

POPUP "Labs"

{

POPUP "First part"

{

MENUITEM "1",1

MENUITEM "2",2

MENUITEM "3",3

MENUITEM "4",4

MENUITEM "5",5

MENUITEM "6",6

}

MENUITEM SEPARATOR

POPUP "Second part"

{

MENUITEM "7",7

MENUITEM "8",8

MENUITEM "9",9

MENUITEM "10",10

MENUITEM "11",11

MENUITEM "12",12

}

MENUITEM SEPARATOR

MENUITEM "About",13

}

POPUP "Help"

{

MENUITEM "About",13

}}

1. Створюємо вікно. В якості параметрів задаєм: назву класа вікна, заголовок, стилі вікна, координати х, у, довжину, ширину. Створюємо цикл, в котрому будуть отримуватися всі повідомлення вікна і передавати їх процедурі обробки повідомлень, а вихід з циклу відбувається при повідомленні WM\_QUIT.

invoke CreateWindowEx,0, ADDR ClassName, ADDR AppName, WS\_OVERLAPPEDWINDOW or WS\_VISIBLE, 100, 100, 755, 600, NULL, NULL, hInst, NULL

.WHILE TRUE

invoke GetMessage, ADDR msg,NULL,0,0

.BREAK .IF (!eax)

invoke DispatchMessage, ADDR msg

.ENDW

1. Описуємо функцію обробки повідомлень вікна.

WndProc proc hWnd:HWND, uMsg:UINT, wParam:WPARAM, lParam:LPARAM

1. Після створення вікна передається повідомлення WM\_CREATE, в якому створюємо рядок стану (StatusBar), Edit control, для виведення тексту та кнопки.

.IF uMsg==WM\_CREATE

invoke CreateStatusWindow,WS\_CHILD or WS\_VISIBLE,NULL, hWnd, 200

mov hStatus, eax

INVOKE CreateWindowEx,NULL, ADDR EditClass, NULL,\

WS\_VISIBLE or WS\_CHILD or ES\_MULTILINE or WS\_VSCROLL or ES\_AUTOVSCROLL or WS\_HSCROLL or ES\_AUTOHSCROLL,\

0,\

0,\

0,\

0,\

hWnd,\

75,\

hInstance,\

NULL

mov hEdit,eax

invoke CreateWindowEx,0, ADDR BtnClass,ADDR lab1,WS\_CHILD or WS\_VISIBLE,0,0,60,20,hWnd,1,hInstance,0

mov hlab1, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab2,WS\_CHILD or WS\_VISIBLE,60,0,60,20,hWnd,2,hInstance,0

mov hlab2, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab3,WS\_CHILD or WS\_VISIBLE,120,0,60,20,hWnd,3,hInstance,0

mov hlab3, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab4,WS\_CHILD or WS\_VISIBLE,180,0,60,20,hWnd,4,hInstance,0

mov hlab4, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab5,WS\_CHILD or WS\_VISIBLE,240,0,60,20,hWnd,5,hInstance,0

mov hlab5, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab6,WS\_CHILD or WS\_VISIBLE,300,0,60,20,hWnd,6,hInstance,0

mov hlab6, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab7,WS\_CHILD or WS\_VISIBLE,360,0,60,20,hWnd,7,hInstance,0

mov hlab7, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab8,WS\_CHILD or WS\_VISIBLE,420,0,60,20,hWnd,8,hInstance,0

mov hlab8, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab9,WS\_CHILD or WS\_VISIBLE,480,0,60,20,hWnd,9,hInstance,0

mov hlab9, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab10,WS\_CHILD or WS\_VISIBLE,540,0,60,20,hWnd,10,hInstance,0

mov hlab10, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab11,WS\_CHILD or WS\_VISIBLE,600,0,60,20,hWnd,11,hInstance,0

mov hlab11, eax

invoke CreateWindowEx,0,ADDR BtnClass,ADDR lab12,WS\_CHILD or WS\_VISIBLE,660,0,60,20,hWnd,12,hInstance,0

mov hlab12, eax

1. Змінюючи розмір вікна передається повідомлення WM\_SIZE, в якому описуємо змінене розташування всіх елементів у вікні.

.ELSEIF uMsg == WM\_SIZE

mov eax,lParam

mov edx,eax

shr edx,16

and eax,0ffffh

sub edx, 40

sub eax, 0

11 invoke MoveWindow,hEdit,0,20,eax,edx,TRUE

invoke MoveWindow,hlab1,0,0,60,20,TRUE

invoke MoveWindow,hlab2,60,0,60,20,TRUE

invoke MoveWindow,hlab3,120,0,60,20,TRUE

invoke MoveWindow,hlab4,180,0,60,20,TRUE

invoke MoveWindow,hlab5,240,0,60,20,TRUE

invoke MoveWindow,hlab6,300,0,60,20,TRUE

invoke MoveWindow,hlab7,360,0,60,20,TRUE

invoke MoveWindow,hlab8,420,0,60,20,TRUE

invoke MoveWindow,hlab9,480,0,60,20,TRUE

invoke MoveWindow,hlab10,540,0,60,20,TRUE

invoke MoveWindow,hlab11,600,0,60,20,TRUE

invoke MoveWindow,hlab12,660,0,60,20,TRUE

invoke MoveWindow,hStatus,0,0,0,0,TRUE

1. Закривши вікно передається повідомлення WM\_DESTROY. Функція PostQuitMessage генерує повідомлення WM\_QUIT.

.ELSEIF uMsg==WM\_DESTROY

invoke PostQuitMessage,NULL

1. WM\_COMMAND передається при натиснені кнопки або вибору меню. Там перевіряється, що було натиснуто і викликаєм функцію (FileOpen), в яку передаєм відповідні параметри.

.ELSEIF uMsg==WM\_COMMAND

mov eax,wParam

.IF ax==1

invoke FileOpen, offset File1,offset ttp1

.ELSEIF ax==2

invoke FileOpen, offset File2,offset ttp2

.ELSEIF ax==3

invoke FileOpen, offset File3,offset ttp3

.ELSEIF ax==4

invoke FileOpen, offset File4,offset ttp4

.ELSEIF ax==5

invoke FileOpen, offset File5,offset ttp5

.ELSEIF ax==6

invoke FileOpen, offset File6,offset ttp6

.ELSEIF ax==7

invoke FileOpen, offset File7,offset ttp7

.ELSEIF ax==8

invoke FileOpen, offset File8,offset ttp8

.ELSEIF ax==9

invoke FileOpen, offset File9,offset ttp9

.ELSEIF ax==10

invoke FileOpen, offset File10,offset ttp10

.ELSEIF ax==11

invoke FileOpen, offset File11,offset ttp11

.ELSEIF ax==12

invoke FileOpen, offset File12,offset ttp12

.ELSEIF ax==13

invoke MessageBoxA,0, offset info, offset about,0

.ELSEIF ax==100

invoke DestroyWindow,hWnd

1. При передачі повідомлення для якого в програмі не передбачена обробка, то викликається функція DefWindowProc, яка обробляє ці повідомлення.

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

1. В функції FileOpen спочатку відбувається перевірка: якщо вибраний текст вже відображається в вікні, то очистищаємо (вікно то рядок стану).

.IF check==ax

invoke SendMessage,hEdit,WM\_SETTEXT,NULL,NULL

invoke SendMessage,hStatus,WM\_SETTEXT,NULL, NULL

mov check, 0

.ELSE

mov check, ax

1. Інакше відкриваємо відповідний файл, назва якого передається в процедуру як параметр, в режимі читання і отримуємо його HANDLE. Отримуємо HANDLE кучі і виділяємо та обнуляємо блок пам’яті. Після отримання покажчик на блок пам’яті записуємо в нього текст програми з файлу через функцію ReadFile. Врешті записуємо текст програми в edit control і назву файла в рядок стану, закриваємо файл і звільняємо виділений блок пам'яті.

invoke CreateFile,filename,GENERIC\_READ,FILE\_SHARE\_READ,NULL,OPEN\_EXISTING,FILE\_ATTRIBUTE\_READONLY,NULL

mov hFile,eax

invoke GlobalAlloc,GMEM\_MOVEABLE or GMEM\_ZEROINIT,MEMSIZE

mov hMemory,eax

invoke GlobalLock,hMemory

mov pMemory,eax

invoke ReadFile,hFile,pMemory,MEMSIZE-1,ADDR SizeReadWrite,NULL

invoke SendMessage,hEdit,WM\_SETTEXT,NULL,pMemory

invoke SendMessage,hStatus,WM\_SETTEXT,NULL, message

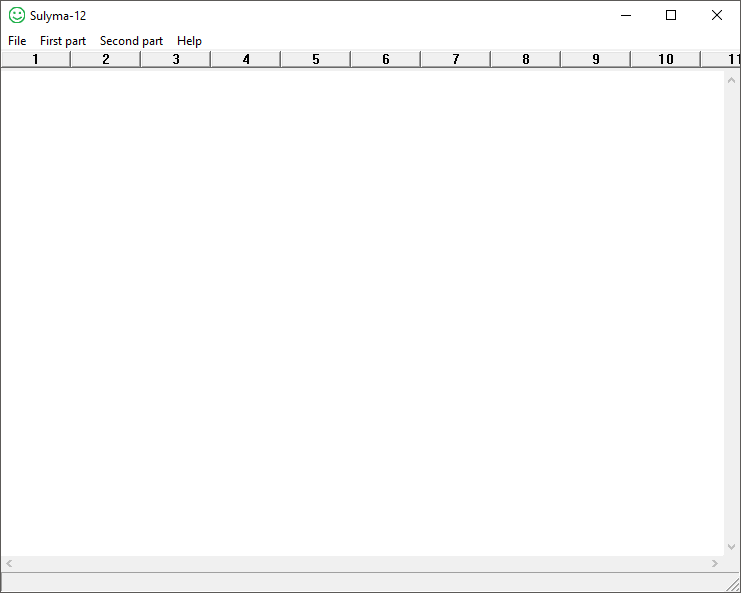
invoke CloseHandle,hFile

invoke GlobalUnlock,pMemory

invoke GlobalFree,hMemory

.ENDIF

Результат



Висновок

Метою даної лабораторної роботи було вивчити прийоми поглибленого програмування по розробці віконного інтерфейсу. Дана програма створює вікно, в якому є меню і кнопки, які дозволяють вибрати будь-яку лабораторну роботу і вивести текст програми, без можливості редагування, вивести інформацію про цю лабораторну роботу, а вибір відображається в рядку стану, що розміщений знизу вікна. Також дана програма має свою власну іконку, яка розміщена в файлі ресурсів.