1. Создать приложение Win32 Project (в Microsoft Visual Studio).
2. Работа с ресурсами приложения:
   * Изменить заголовок окна (в заголовке указать номер лабораторной работы и фамилию студента);

Изменяем строку в файле параметров:

STRINGTABLE

BEGIN

IDC\_WINDOWSPROJECT1 "WINDOWSPROJECT1"

IDS\_APP\_TITLE "Lab #1 Худяков К."

* + Изменить иконку приложения (создать собственную);

Создаём новую иконку и изменяем соответствующие поля в файле параметров:

IDI\_WINDOWSPROJECT1 ICON "Dapino-Summer-Holiday-Palm-tree.ico"

IDI\_SMALL ICON "Dapino-Summer-Holiday-Palm-tree.ico"

* + Добавить в меню новый пункт.

Изменяем меню в файле параметров:

IDC\_WINDOWSPROJECT1 MENU

BEGIN

POPUP "&File"

BEGIN

MENUITEM "E&xit", IDM\_EXIT

END

POPUP "&Work"

BEGIN

MENUITEM "&DoSomething ...", IDM\_SOMETHING

END

POPUP "&Help"

BEGIN

MENUITEM "&About ...", IDM\_ABOUT

END

END

1. Задать новые координаты (левый верхний угол, например, (100, 100)) и размеры запускаемого приложения (например, 300 и 200). Продумать несколько возможных вариантов изменения.

Изменяем в функции InitInstance параметры создания окна:

mainWindow = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

100, 100, 1080, 720, nullptr, nullptr, hInstance, nullptr);

1. Создать управляющие элементы пользовательского интерфейса: две кнопки (BUTTON), поле ввода (EDIT) и поле вывода (STATIC). Для кнопок задать идентификатор управляющего элемента.

Объявляем константы идентификаторов для строк из ресурсов и команд:

#define IDS\_BUTTON1\_TITLE 110

#define IDS\_BUTTON2\_TITLE 111

#define IDS\_EDIT\_TITLE 112

#define IDS\_STATIC\_TITLE 113

#define IDC\_BUTTON1 114

#define IDC\_BUTTON2 115

#define IDC\_EDIT1 116

#define IDC\_STATIC1 117

Добавляем значения этих строк в файле ресурсов:

STRINGTABLE

BEGIN

IDC\_WINDOWSPROJECT1 "WINDOWSPROJECT1"

IDS\_APP\_TITLE "Lab #1 Худяков К."

IDS\_BUTTON1\_TITLE "Button1"

IDS\_BUTTON2\_TITLE "Button2"

IDS\_EDIT\_TITLE ""

IDS\_STATIC\_TITLE ""

END

Создаём объекты в функции InitInstance и запоминаем их в глобальные переменные:

button1 = CreateWindow(TEXT("BUTTON"), button1Title, WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

250, 250, 170, 90, mainWindow, (HMENU)IDC\_BUTTON1, hInstance, nullptr);

button2 = CreateWindow(TEXT("BUTTON"), button2Title, WS\_CHILD | WS\_VISIBLE | BS\_PUSHBUTTON,

660, 250, 170, 90, mainWindow, (HMENU)IDC\_BUTTON2, hInstance, nullptr);

edit = CreateWindow(TEXT("EDIT"), editTitle, WS\_CHILD | WS\_VISIBLE | ES\_LEFT | WS\_BORDER,

400, 100, 285, 30, mainWindow, (HMENU)IDC\_EDIT1, hInstance, nullptr);

staticOut = CreateWindow(TEXT("STATIC"), staticTitle, WS\_CHILD | WS\_VISIBLE | SS\_LEFT,

400, 450, 285, 30, mainWindow, (HMENU)IDC\_STATIC1, hInstance, nullptr);

1. Написать обработчики событий для кнопок, пункта меню и некоторых событий:
   * События: WM\_CREATE, WM\_DESTROY, нажатие левой кнопки мыши, нажатие правой кнопки мыши, щелчок на добавленном пункте меню должны выводить сообщения о том, какое именно событие произошло;

Добавляем обработку данных событий в switch в функции WndProc:

case WM\_CREATE:

SetWindowText(edit, TEXT("Window created"));

break;

case WM\_DESTROY:

SetWindowText(edit, TEXT("Window destroyed"));

PostQuitMessage(0);

break;

case WM\_LBUTTONDOWN:

SetWindowText(edit, TEXT("Left mouse button pressed"));

break;

case WM\_RBUTTONDOWN:

{

SetWindowText(edit, TEXT("Right mouse button pressed"));

KillTimer(button1, IDT\_BUTTON1\_TIMER);

SetParent(button1, mainWindow);

RECT rect;

GetWindowRect(button1, &rect);

SetWindowPos(button1, nullptr, 250, 250,

rect.right - rect.left, rect.bottom - rect.top, 0);

}

break;

* + Первая кнопка BUTTON: текст введенный в поле EDIT должен появиться в поле STATIC;

Добавляем в фукнции WndProc в блок WM\_COMAND обработку команды нажатия на кнопку:

case IDC\_BUTTON1:

{

int strSize = GetWindowTextLength(edit) + 1;

LPWSTR str = new TCHAR[strSize];

GetWindowText(edit, str, strSize);

SetWindowText(staticOut, str);

}

break;

* + Вторая кнопка BUTTON: в поле STATIC должен быть выведен заголовок окна приложения.

Добавляем в фукнции WndProc в блок WM\_COMAND обработку команды нажатия на кнопку:

case IDC\_BUTTON2:

SetWindowText(staticOut, szTitle);

break;

1. Обработчик пункта меню должен: сменить окно - родителя у первой кнопки, а затем заставить ее медленно перемещаться по экрану. Нажатие правой кнопки мыши должно обеспечить возвращение кнопки в окно программы.

Будем использовать таймер, для этого создадим функцию, которая будет вызываться по тику таймера и сдвигать нашу кнопку:

void CALLBACK move(HWND hwnd, UINT uMsg, UINT timerId, DWORD dwTime)

{

RECT rect;

GetWindowRect(hwnd, &rect);

MoveWindow(hwnd, rect.left + 1, rect.top + 1,

rect.right - rect.left, rect.bottom - rect.top, true);

}

Создадим обработчик нажатия на добавленную кнопку меню, в котором назначим рабочий стол родителем для кнопки и запустим таймер для анимации её движения:

case IDM\_SOMETHING:

{

SetWindowText(edit, TEXT("Menu item \"DoSomething\" clicked"));

RECT rectButton;

GetWindowRect(button1, &rectButton);

SetParent(button1, GetDesktopWindow());

SetWindowPos(button1, nullptr, rectButton.left, rectButton.top,

rectButton.right - rectButton.left, rectButton.bottom - rectButton.top, 0);

SetTimer(button1, IDT\_BUTTON1\_TIMER, 30, move);

}

break;

Добавим в обработчик нажатия правой кнопкой мыши остановку таймера и возвращение кнопки в родное окно:

case WM\_RBUTTONDOWN:

{

SetWindowText(edit, TEXT("Right mouse button pressed"));

KillTimer(button1, IDT\_BUTTON1\_TIMER);

SetParent(button1, mainWindow);

RECT rect;

GetWindowRect(button1, &rect);

SetWindowPos(button1, nullptr, 250, 250,

rect.right - rect.left, rect.bottom - rect.top, 0);

}

break;

1. Оформить окно приложения в форме региона: прямоугольник с эллипсом и вырезанным посередине кругом (расположить все управляющие элементы, чтобы они были видны). Проверить работоспособность приложения.

Для этого в функции InitInstance после создания главного окна добавим построение региона из пересечения прямоугольника, круга и эллипса:

mainWindow = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

100, 100, 1080, 720, nullptr, nullptr, hInstance, nullptr);

HRGN windowRegion = CreateRectRgn(0, 0, 1080, 490);

HRGN ellipseRegion = CreateEllipticRgn(0, 200, 1080, 720);

HRGN circleRegion = CreateEllipticRgn(450, 250, 650, 450);

CombineRgn(windowRegion, windowRegion, ellipseRegion, RGN\_OR);

CombineRgn(windowRegion, windowRegion, circleRegion, RGN\_XOR);

SetWindowRgn(mainWindow, windowRegion, true);