**Отчёт к лабораторной работе №2**

**Худяков Константин А-13-17  
Обработка сообщений**

**Введение**

1. Тип данных TCHAR, переключение режима кодировки символов, функции для работы со строками (strlen, strcpy, strcat, itoa, atoi).

**#ifdef \_UNICODE**

**typedef** **wchar\_t** TCHAR;

**#else** **typedef** **char** TCHAR;

**#endif**

Макрос \_UNICODE будет включен если вы укажите «Use Unicode Character set» и тогда тип TCHAR будет определен как wchar\_t. Когда же вы укажите «Use Multi-byte character set» TCHAR будет определен как char.

**size\_t** \_tcslen(**const** TCHAR\* );

**#ifdef \_UNICODE** **#define \_tcslen wcslen #else** **#define \_tcslen strlen** **#endif**

\_tcscpy, \_tcscat, \_itot

**#ifdef \_UNICODE** #define \_ttoi       \_wtoi **#else** #define \_ttoi       atoi **#endif**

1. LOWORD, HIWORD

WORD LOWORD(

 DWORD dwValue

); // Получает слово низкого порядка от заданного значения

WORD HIWORD(

 DWORD dwValue

); // Получает слово высокого порядка из указанного 32-битового значения.

1. Функция SendMessage

Например, передать сообщение главному окну SendMessage(hWnd,WM\_LBUTTONDOWN,MK\_LBUTTON,MAKELONG(100,100));

или сообщение для кнопки SendMessage(hBut1,BM\_SETSTATE,TRUE,0);

LRESULT WINAPI SendMessage(

\_In\_ HWND   hWnd,

\_In\_ UINT   Msg,

\_In\_ WPARAM wParam,

\_In\_ LPARAM lParam

);

1. Стиль окна: CS\_DBLCLKS, сообщение WM\_LBUTTONDBLCLK.

Окно будет получать сообщения о двойном щелчке (DBLCLK) только в том случае, если стиль соответствующего класса окна содержит флаг CS\_DBLCLKS.

style = GetClassLong(hWnd. GCL\_STYLE); SetClassLong(hWnd, GCL\_STYLE. style | CS\_DBLCLKS):

Если класс окна определен без флага CS\_DBLCLKS и пользователь делает двойной щелчок левой кнопкой мыши, то оконная процедура последовательно получает сообщения WMJ.BUTT0ND0WN, WM\_LBUTTONUP, WM\_LBUTT0ND0WN и WM\_LBUTTONUP.  
Если класс окна определен с флагом CS\_DBLCLKS, то после двойного щелчка оконная процедура получит сообщения WM\_LBUTT0ND0WN, WM\_LBUTTONUP, WM\_ LBUTTON-DBLCLK и WM\_LBUTTONUP.

1. Сообщение WM\_NCHITTEST, процедура DefWindowProc.

WM\_NCHITTEST Уведомляет окно, пpинимающее ввод от мыши, о том, что мышь пеpеместилась.

lParamLo: x-кооpдината куpсоpа мыши.

lParamHi: y-кооpдината куpсоpа мыши.

Функция **DefWindowProc** вызывается оконной процедурой по умолчанию, чтобы обеспечить обработку по умолчанию любого сообщения окна, которые приложение не обрабатывает. Эта функция гарантирует то, что обрабатывается каждое сообщение. Функция **DefWindowProc** вызывается с теми же самыми параметрами, принятыми оконной процедурой.

LRESULT DefWindowProc  - базовый обработчик окна

(

HWND hWnd, // указатель окна

UINT Msg, // идентификатор сообщения

WPARAM wParam, // первый параметр сообщения

LPARAM lParam // второй параметр сообщения

);

1. SetCapture(hWnd); ReleaseCapture();

Функция **SetCapture** устанавливает захват мыши в заданном окне, принадлежащем текущему потоку. **SetCapture** захватывает ввод данных от мыши или когда мышь находится над захватывающим окном, или когда нажималась кнопка мыши, в то время, когда мышь была над захватывающим окном, а кнопка все еще находилась в нажатом состоянии. Только одно окно одновременно может захватить мышь.

Когда окно больше не требует всего ввода данных от мыши, поток, который создавал окно, должен вызвать функцию [**ReleaseCapture**](http://vsokovikov.narod.ru/New_MSDN_API/Mouse_input/fn_releasecapture.htm), чтобы освободить мышь.

### Задание

1. Создать приложение Win32 Project (в Microsoft Visual Studio). Запустив приложение, убедиться в том, что оно работает, а окно с помощью мыши перемещается по экрану (курсор мыши находится в заголовке окна).
2. В окно приложения добавить две кнопки, поля ввода и вывода. Первая кнопка должна возводить в квадрат целое число введенное пользователем, вторая кнопка должна заставить первую кнопку “нажаться” и выполнить код

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

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

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

100, 100, 100, 40, mainWindow, (HMENU)IDC\_BUTTON1, hInstance, nullptr);

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

300, 100, 100, 40, mainWindow, (HMENU)IDC\_BUTTON2, hInstance, nullptr);

edit = CreateWindow(L"EDIT", L"", WS\_CHILD | WS\_VISIBLE | ES\_LEFT | WS\_BORDER,

100, 200, 100, 40, mainWindow, (HMENU)IDC\_EDIT, hInstance, nullptr);

staticOut = CreateWindow(L"STATIC", L"", WS\_CHILD | WS\_VISIBLE | SS\_LEFT,

300, 200, 100, 40, mainWindow, (HMENU)IDC\_STATIC\_OUT, hInstance, nullptr);

staticLog = CreateWindow(L"STATIC", L"", WS\_CHILD | WS\_VISIBLE | SS\_LEFT, 300, 300, 1000, 240, mainWindow, (HMENU)IDC\_STATIC\_LOG, hInstance, nullptr);

case IDC\_BUTTON1:

{

int strSize = GetWindowTextLength(edit) + 1;

LPWSTR str = new TCHAR[strSize];

GetWindowText(edit, str, strSize);

if (isNumber(str)) {

int num = \_wtoi(str);

num \*= num;

LPWSTR newStr = numToStr(num);

SetWindowText(staticOut, newStr);

}

}

break;

case IDC\_BUTTON2:

{

SendMessage(button1, BM\_CLICK, 0, 0);

}

1. По щелчку на правой кнопке мыши определить и вывести координаты курсора.

case WM\_RBUTTONDOWN:

{

POINT p;

GetCursorPos(&p);

ScreenToClient(mainWindow, &p);

rButtonDownX = p.x;

rButtonDownY = p.y;

}

break;

case WM\_RBUTTONUP:

{

POINT p;

GetCursorPos(&p);

ScreenToClient(mainWindow, &p);

if (p.x == rButtonDownX && p.y == rButtonDownY) {

int xNumLen = getLengthOfNum(p.x);

int yNumLen = getLengthOfNum(p.y);

int len = 8 + xNumLen + yNumLen;

LPWSTR str = new TCHAR[100];

str[0] = '\0';

wcscat\_s(str, len, L"X: ");

wcscat\_s(str, len, numToStr(p.x));

wcscat\_s(str, len, L" Y: ");

wcscat\_s(str, len, numToStr(p.y));

SetWindowText(staticOut, str);

}

}

break;

1. Проделать программные эксперименты и проследить за работой событий (для эксперимента можно записывать информацию о событии в строковую переменную, а затем вывести ее в окне сообщений):

* WM\_LBUTTONDBLCLK: какие события и сколько раз срабатывают при двойном щелчке

case WM\_LBUTTONDOWN:

{

wcscat\_s(log, 10000, L"WM\_LBUTTONDOWN ");

SetWindowText(staticLog, log);

lButtonDown = true;

POINT p;

GetCursorPos(&p);

lButtonDownX = p.x;

lButtonDownY = p.y;

GetWindowRect(mainWindow, &mainRect);

}

break;

case WM\_LBUTTONUP:

{

wcscat\_s(log, 10000, L"WM\_LBUTTONUP ");

SetWindowText(staticLog, log);

lButtonDown = false;

ReleaseCapture();

}

break;

case WM\_LBUTTONDBLCLK:

{

wcscat\_s(log, 10000, L"WM\_LBUTTONDBLCLK ");

SetWindowText(staticLog, log);

}

break;

* WM\_NCHITTEST: сколько раз возникает данное событие при работе с мышью.

case WM\_NCHITTEST:

{

wcscat\_s(log, 10000, L"WM\_NCHITTEST ");

SetWindowText(staticLog, log);

}

break;

1. Разработать программный код, который позволит пользователю с помощью мыши перемещать окно, но мышь при этом нажимается и перемещается в клиентской области окна.

case WM\_MOUSEMOVE:

if (lButtonDown) {

SetCapture(mainWindow);

POINT p;

GetCursorPos(&p);

int dx = p.x - lButtonDownX;

int dy = p.y - lButtonDownY;

SetWindowPos(mainWindow, nullptr, mainRect.left + dx, mainRect.top + dy, mainRect.right - mainRect.left, mainRect.bottom - mainRect.top, 0);

}

break;

1. Сохранить разработанное приложение так, чтобы его можно было использовать в программных экспериментах следующей лабораторной работы.