Восьмое занятие

WinApi

WinApi

- Переходник между программой и операционной системой, то есть, теми возможностями, которые она предоставляет.

Основные типы данных

- **BOOL** – этот тип данных аналогичен bool. Он также имеет два значения – 0 или 1. Только при использовании WINAPI принято использовать вместо 0 спецификатор NULL.
- **BYTE** – байт, ну или восьмибитное беззнаковое целое число.
- **DWORD** — 32-битное беззнаковое целое.
- **INT** – 32-битное целое.
- **LONG** – 32-битное целое – аналог всё также long int.
- **NULL** – нулевой указатель. Вот его объявление: void *NULL=0;
- **UINT** – 32-битное беззнаковое целое.

Строковые типы данных

- Есть два вида кодировок символов: ANSI и UNICODE.
- Однобайтные символы относятся к ANSI.
- Двухбайтные — к кодировке UNICODE.
- Каждая функция для работы со строками в WinApi имеет версию для UNICODE, как правила различаются суффиксом (напр. TextOutA(), TextOutW())

```
char ch = 'Q';
```

```
wchar_t wch = 'Л';
```

Стандарт кодирования символов, позволяющий представить знаки почти всех письменных языков

UNICODE

Строковые типы данных

- **LPCSTR** – указатель на константную строку, заканчивающуюся нуль-терминатором.
- **LPCTSTR** – указатель на константную строку, без UNICODE.
- **LPCWSTR** – указатель на константную UNICODE строку.
- **LPSTR** – указатель на строку, заканчивающуюся нуль-терминатором.
- **LPTSTR** – указатель на строку, без UNICODE.
- **LPWSTR** – указатель на UNICODE строку.
- **TCHAR** – символьный тип — аналог char и wchar_t.

Дескрипторные типы данных

- Дескриптор — это идентификатор какого-либо объекта. Для разных типов объектов существуют разные дескрипторы.

```
HANDLE h;
```

- Есть дескрипторы кисти, курсора мыши, шрифта и т.д. С их помощью мы можем при инициализации или в процессе работы приложения поменять какие-нибудь настройки.

`HANDLE` – дескриптор объекта.

`HBITMAP` – дескриптор растрового изображения. От фразы handle bitmap.

`HBRUSH` – дескриптор кисти. От фразы handle brush.

`HCURSOR` – дескриптор курсора. От фразы handle cursor.

`HDC` – дескриптор контекста устройства. От фразы handle device context.

`HFONT` – дескриптор шрифта. От фразы handle font.

`HICONS` – дескриптор криптограммы. От фразы handle icons.

`HINSTANCE` – дескриптор экземпляра приложения. От фразы handle instance.

`HMENU` – дескриптор меню. От фразы handle menu.

`HPEN` – дескриптор пера. От фразы handle pen.

`HWND` – дескриптор окна. От фразы handle window.

Вспомогательные типы

`LPARAM` – тип для описания lParam (long parameter). Используются вместе с wparam в некоторых функциях.

`LRESULT` – значение, возвращаемое оконной процедурой имеет тип `long`.

`WPARAM` – тип для описания wParam (word parameter). Используются вместе с lParam в некоторых функциях.

Наверное стоит посмотреть

- Предыдущие 7 слайдов честно украдены от сюда ->
- http://cppstudio.com/post/9489/

Самое начало

```
// Основная функция - аналог int main() в консольном приложении:
int WINAPI WinMain(HINSTANCE hInstance, // дескриптор экземпляра приложения
                   HINSTANCE hPrevInstance, // в Win32 не используется
                   LPSTR lpCmdLine, // нужен для запуска окна в режиме командной строки
                   int nCmdShow) // режим отображения окна
```

Вывод окна с

```
MessageBox(NULL, "Hello", "world", MB_OKCANCEL);
```

Практика

Сделаем так

Создание более сложного окна

- Для создание сложного окна нужно:
  - Создать и описать класс окна
  - Зарегистрировать класс окна
  - Создать окно и получить его **дескриптор**
  - Вызвать функцию показа окна
  - Запустить цикл обработки сообщений

Пример

Создание класса окна

```
WNDCLASSW wc;
wc.style = CS_HREDRAW | CS_VREDRAW;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.lpszClassName = L"Привет";
wc.hInstance = hInstance;
wc.hbrBackground = GetSysColorBrush(COLOR_3DFACE);
wc.lpszMenuName = NULL;
wc.lpfnWndProc = wWndProc;
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
```

Пример

- Регистрация класса окна

```
RegisterClassW(&wc);
```

Пример

- Создание и получение дескриптора окна

```
hwnd = CreateWindowW(wc.lpszClassName, L"Привет",
                     WS_OVERLAPPEDWINDOW | WS_VISIBLE,
                     100, 100, 350, 250, NULL, NULL, hInstance, NULL);
```

Пример

- Показ окна

```
ShowWindow(hwnd, nCmdShow);
```

Пример

- Запуск цикла обработки сообщений

```
MSG msg;
while (GetMessage(&msg, NULL, 0, 0)) {

    DispatchMessage(&msg);
}
```

Обработка событий

- При создании класса окна, в поле **lpfnWndProc** указывается функции для обработки системных сообщенией (жизненый цико приложения, события пользователя). Как правило в качестве этой функции выступает wndProc

```
LRESULT CALLBACK wWndProc(HWND hwnd, UINT msg,
                          WPARAM wParam, LPARAM lParam)
```

Параметры wndProc

- **HWND hwnd** – дескриптор окна
- **UINT msg** – тип сообщения системы (WM_DESTROY, WM_PAINT)
- **WPARAM wParam, LPARAM lParam** – параметры сообщения

Практика

Выведем Hello world

Практика

А теперь по русски

Практика

Вывести код нажатой пользователем клавиши

Вопросы?