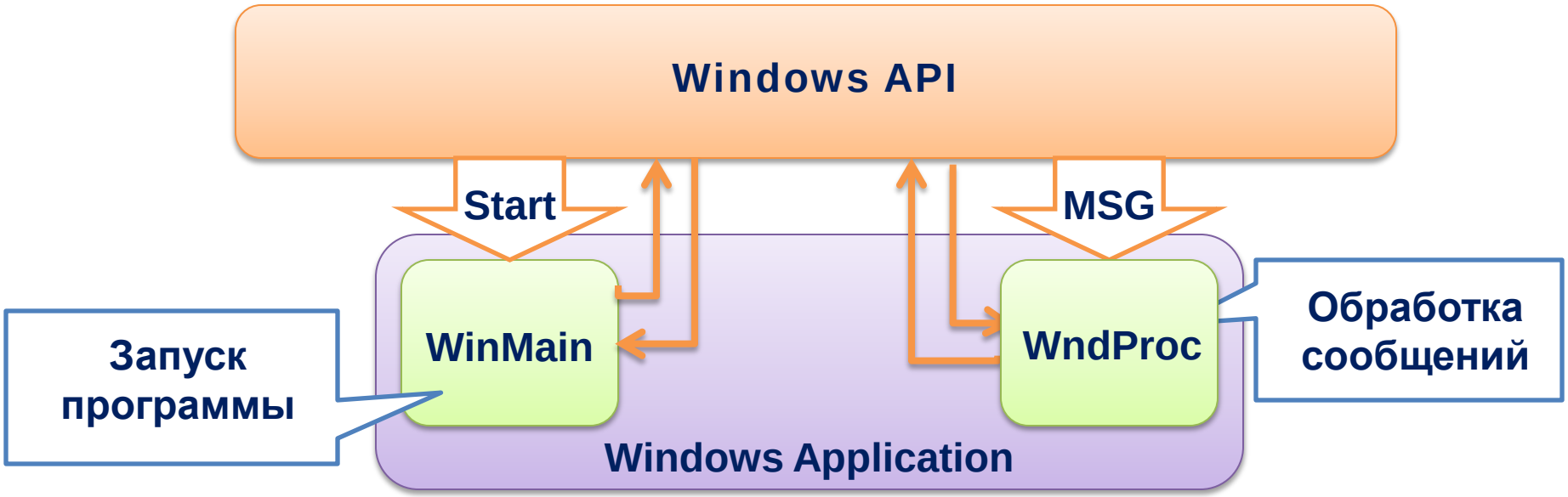
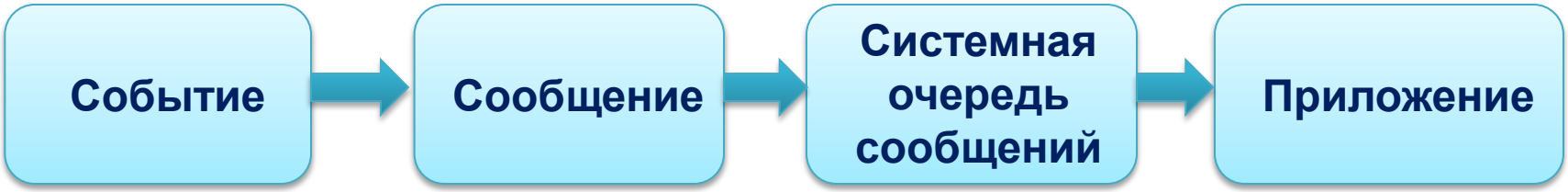
Введение в Windows-программирование

API (Application Program Interface)

- набор констант, классов, структур и функций для разработки приложений
- главный элемент приложения – окно



События и сообщения



Источники событий

- внешние устройства (клавиатура, мышь, таймер)
- другие приложения
- операционная система

Окно

Объект, над которым выполняют действие

- у каждого окна есть родитель (для окон верхнего уровня – окно рабочего стола)
- родитель дочернего окна – окно верхнего уровня или другое дочернее окно
- владельцем окна верхнего уровня может быть другое окно того же уровня (тогда оно отображается поверх окна-владельца)
- кнопки – это тоже окна

Виды окон

окно приложения	• главное окно приложения • появляется первым при запуске • закрывается последним
диалоговое окно	• обеспечивает обмен данными между пользователем и приложением

Дескриптор

- число
- все объекты в Windows описываются через дескрипторы
- дескриптор окна однозначно определяет окно в системе
- тип и имя дескриптора начинается с префикса H (HINSTANCE, HWND, HICON, HFONT, hWnd, hInst и т. п.)

Сообщения окнам

WM_CREATE (создать окно)	**WM_ENABLE** (сделать окно доступным или нет)
WM_DESTROY (разрушить окно)	**WM_MOVE** (положение окна изменилось)
WM_CLOSE (закрыть окно)	**WM_SIZE** (размер окна изменился)
WM_QUIT (завершить приложение)	**WM_SETFOCUS** (получить фокус ввода)
WM_ACTIVATE (активировать окно)	**WM_KILLFOCUS** (потерять фокус ввода)
WM_SHOWWINDOW (скрыть или отобразить окно)	**WM_QUERYENDSESSION** (сообщает о конце сеанса Windows)

Структура сообщения

- каждое сообщение имеет свой код
- каждому коду сопоставлена символическая константа (код 0x200 – WM_MOUSEMOVE и т.п.)
- сообщение представляется как структура

```
struct MSG {
    HWND        hwnd;       // дескриптор окна
    UINT        message;    // код сообщения
    WPARAM      wParam;     // доп. информация
    LPARAM      lParam;     // доп. информация
    DWORD       time;       // время отправки
    POINT       pt;         // позиция курсора
};
```

Функции для работы с окнами

CreateWindow	**FindWindow** (поиск окна верхнего уровня)
ShowWindow	**GetParent** (получить дескриптор родителя)
CloseWindow (закрывает, но не разрушает окно!)	**GetWindow** (получить дескриптор окна)
IsWindowVisible (возвращает состояние окна)	**IsChild** (дочернее ли окно)
MoveWindow (изменяет расположение и размеры)	**IsWindow** (соответствует ли дескриптор окну)
GetClientRect (получить координаты клиентской области)	**OpenIcon** (восстанавливает свернутое окно)
GetWindowPlacement (данные о расположении окна)	**SetWindowPlacement** (свертывает и развертывает окно)

Минимальное Win32-приложение

- включает две функции:
  - WinMain (инициализация приложения в ОС)
    - инициализация переменных
    - регистрация оконного класса
    - создание окна
    - отображение окна
    - создание цикла обработки сообщений
  - WndProc (обработка сообщений; может быть вызвана только ОС)

```
int WINAPI WinMain(HINSTANCE,  HINSTANCE,  LPSTR, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);
```

Структура WNDCLASSEX (WinUser.h)

```
struct WNDCLASSEX {
    UINT        cbSize; // размер структуры
    UINT        style; // стиль класса окна
    WNDPROC     lpfnWndProc; // указатель на оконную функцию
    int         cbClsExtra; // к-во байт доп. инф. о классе
    int         cbWndExtra; // к-во байт доп. инф. об окне
    HINSTANCE   hInstance; // дескриптор приложения
    HICON       hIcon; // дескриптор пиктограммы
    HCURSOR     hCursor; // дескриптор курсора мыши
    HBRUSH      hbrBackground; // дескриптор кисти для фона
    LPCWSTR     lpszMenuName; // имя меню
    LPCWSTR     lpszClassName; // имя класса окна
    HICON       hIconSm; // дескриптор маленькой иконки
};
```

WinMain (регистрация оконного класса)

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
        hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{
        LPWSTR szClassName = L"MyWindowClass";
        LPWSTR szTitleName = L"My first window";
        WNDCLASSEX wndclass;
        wndclass.cbSize = sizeof(wndclass);
        wndclass.lpszClassName = szClassName;
        wndclass.style = CS_HREDRAW | CS_VREDRAW;
        wndclass.lpfnWndProc = WndProc;
        wndclass.cbClsExtra = 0;
        wndclass.cbWndExtra = 0;
        wndclass.hInstance = hInstance;
        /* ... */
}
```

WinMain (регистрация оконного класса)

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
        hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{       /* … */
   wndclass.hCursor = LoadCursor(NULL, IDC_ARROW);
   wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION);
   wndclass.hbrBackground
        =(HBRUSH)GetStockObject(WHITE_BRUSH);
        // или CreateSolidBrush(RGB(0, 0, 0));
   wndclass.lpszMenuName = NULL;
   wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);

   if (!RegisterClassEx(&wndclass)){
      return false;
   }
}
```

Класс окна

- перед регистрацией класса окна должны быть заполнены все поля структуры WNDCLASSEX
- чтобы не заполнять все поля вручную (если достаточно параметров по умолчанию), можно предварительно обнулить переменную структуры через memset()
- потом можно установить только значения нужных полей

```
void* memset( void* dest, int c, size_t count );
```

```
memset(&wndclass, 0, sizeof(wndclass));
```

Стили класса окна

- начинаются с префикса CS_
- устанавливаются при заполнении полей структуры

CS_DBLCLKS	реакция на двойной щелчок мыши
CS_VREDRAW	перерисовка окна, если меняется его высота
CS_HREDRAW	перерисовка окна, если меняется его ширина
CS_NOCLOSE	запрет закрывать окно

```
WNDCLASSEX wndclass;
wndclass.cbSize = sizeof(wndclass);
wndclass.style = CS_HREDRAW | CS_VREDRAW | CS_NOCLOSE;
```

Иконка (поле hIcon)

- можно использовать стандартную иконку
- используется функция LoadIcon(), где первый параметр – NULL, второй – предопределенный

IDI_ APPLICATION	иконка, назначаемая по умолчанию
IDI_ASTERISK	символ i
IDI_EXCLAMATION	восклицательный знак
ID_HAND	знак «Стоп»
IDI_QUESTION	вопросительный знак

```
WNDCLASSEX wndclass;
wndclass.hIcon = LoadIcon(NULL, IDI_HAND);
```

Курсоры

- начинаются с префикса IDC_
- используется функция LoadCursor()

IDC_ARROW	стандартная стрелка
IDC_CROSS	перекрестье
IDC_UPARROW	вертикальная стрелка
IDC_SIZEWE	горизонтальная двусторонняя стрелка
IDC_SIZENS	вертикальная двусторонняя стрелка
IDC_SIZEALL	четырехсторонняя стрелка
IDC_WAIT	песочные часы или вращающаяся окружность
IDC_BEAM	текстовый курсор

```
WNDCLASSEX wndclass;
wndclass.hCursor = LoadCursor(NULL, IDC_WAIT);
```

Функция для создания окна

```
HWND CreateWindowEx(
        DWORD dwExStyle,     // WS_EX_TOPMOST
        LPSTR lpClassName,
        LPSTR lpWindowName,
        DWORD dwStyle,       // WS_OVERLAPPEDWINDOW
        int X,
        int Y,
        int nWidth,
        int nHeight,
        HWND hWndParent,     // NULL, HWND_DESKTOP
        HMENU hMenu,
        HINSTANCE hInstance,
        LPVOID lpParam);
```

WinMain (создание окна)

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
        hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{   /* ... */
    HWND hWnd = CreateWindowEx(
        WS_EX_TOPMOST,
        szClassName,
        L"Window",
        WS_OVERLAPPEDWINDOW,
        0, 0, 150, 150,
        NULL,
        NULL,
        hInstance,
        NULL);
}
```

Общие стили окна

- начинаются с префикса WS_
- обычно устанавливаются при создании окна
- перекрывающие окна (главные)
  и всплывающие окна (временные)

WS_OVERLAPPEDWINDOW	строка заголовка, рамка, кнопки в правом верхнем углу, меню, линейка прокрутки
WS_OVERLAPPED	строка заголовка, рамка
WS_POPUPWINDOW	рамка; может быть заголовок и меню
WS_POPUP	без рамки

```
hWnd = CreateWindowEx(WS_EX_TOPMOST, szclassName, L"Window",
            WS_OVERLAPPEDWINDOW, 0, 0, 250, 250,
            NULL, NULL, hInstance, NULL);
```

Стили окна

- начинаются с префикса WS_

WS_BORDER	окно с рамкой
WS_CAPTION	окно со строкой заголовка
WS_CHILD	дочернее окно
WS_DISABLED	заблокированное окно
WS_HSCROLL	окно с горизонтальной полосой прокрутки
WS_SYSMENU	окно с меню
WS_VSCROLL	окно с вертикальной полосой прокрутки

```
hWnd = CreateWindowEx(WS_EX_TOPMOST, szclassName,
        L"Window",    WS_OVERLAPPEDWINDOW |
        WS_DISABLED | WS_VSCROLL,  0, 0, 250, 250,
        NULL, NULL, hInstance, NULL);
```

Расширенные стили окна

- начинаются с префикса WS_EX_
- обычно устанавливаются при создании окна

WS_EX_TOPMOST	окно всегда отображается поверх других окон
WS_EX_TOOLWINDOW	окно для плавающей панели инструментов
WS_EX_ACCEPTFILES	окно будет принимать перетаскиваемые файлы
и др.	

```
hWnd = CreateWindowEx(WS_EX_TOPMOST, szclassName,
L"Window",     WS_OVERLAPPEDWINDOW, 0, 0, 250, 250,
            NULL, NULL, hInstance, NULL);
```

WinMain (показ окна, цикл сообщений)

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
        hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{   /* … */
    if (!hWnd) { return false; }
    ShowWindow(hWnd, 1);
    UpdateWindow(hWnd);

    MSG msg;
    while ( GetMessage(&msg, NULL, 0, 0) ){
        TranslateMessage(&msg);
        DispatchMessage(&msg);
     }
     return msg.wParam;
}
```

WinProc

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT iMsg,
        WPARAM wParam, LPARAM lParam)
{
   HDC hdc;                  // Если нужно рисовать  в окне
   PAINTSTRUCT ps;
   RECT rect;

   switch(iMsg) {
        /* … */
   }

   return 0;
}
```

WinProc (обработка сообщений)

```
switch(iMsg) {
   case WM_PAINT :
      hdc = BeginPaint(hwnd, &ps);
      GetClientRect(hwnd, &rect);
      DrawText(hdc, L"Hello, Step", -1, &rect, DT_CENTER |
                                          DT_SINGLELINE | DT_VCENTER);
      EndPaint(hwnd, &ps);
      break;
   case WM_DESTROY:
      PostQuitMessage(0);
      break;
   default:
      return DefWindowProc(hwnd, iMsg, wParam, lParam);
}
```

MessageBox (окно сообщений)

```
int  WINAPI MessageBoxW(
        HWND hWnd,              // дескриптор родителя
        LPCWSTR lpText,         // текст внутри окна
        LPCWSTR lpCaption,      // заголовок окна
        UINT uType);            // тип окна, набор кнопок
```

- если hWnd = NULL, то родительского окна нет
- если lpCaption = NULL, то заголовок «Ошибка»

```
MessageBoxW(NULL, L"Hello, Step!", L"MessageBox", MB_OK);
```

Флаги

Флаг	Имена кнопок
MB_ABORTRETRYIGNORE	«Стоп», «Повтор», «Пропустить»
MB_OK	«ОК»
MB_OKCANCEL	«ОК», «Отмена»
MB_RETRYCANCEL	«Повтор», «Отмена»
MB_YESNO	«Да», «Нет»
MB_YESNOCANCEL	«Да», «Нет» , «Отмена»

Флаг	Иконка
MB_ICONWARNING	Восклицательный знак
MB_ICONINFORMATION	Символ «i»
MB_ICONQUESTION	Вопросительный знак
MB_ICONSTOP, MB_ICONERROR	Знак остановки

Возвращаемое значение

Символьная константа	Нажатая кнопка
IDABORT	«Стоп»
IDCANCEL	«Отмена»
IDIGNORE	«Пропустить»
IDNO	«Нет»
IDOK	«ОК»
IDRETRY	«Повтор»
IDYES	«Да»

Задание

Напишите простое приложение, проверяющее работу MessageBox’ов. На экран последовательно выводятся четыре окна сообщения. Все окна должны быть разных видов: окно предупреждения, окно информирования, окно вопроса и окно ошибки. Кроме того, на окнах должны быть разные сочетания кнопок (например, на одном – «Да» и «Нет», на другом – «ОК» и «Отмена» и т. д.). Каждое окно должно иметь уникальный заголовок и уникальный текст.

Таймер

- посылает сообщение WM_TIMER
- создается функцией SetTimer()
- удаляется функцией KillTimer()

```
UINT SetTimer(HWND hWnd, UINT_PTR idEvent, UINT uInterval,
              TIMERPROC fnTimerFunc);
```

```
BOOL KillTimer(HWND hWnd, UINT_PTR idEvent);
```

Обработка сообщений таймера

```
switch(iMsg) {
    case WM_CREATE:
         SetTimer(hWnd, 1, 1000, NULL);
         break;
    case WM_TIMER:
        SetClassLong(hWnd, GCL_HBRBACKGROUND,
                                (LONG)CreateSolidBrush(RGB(10, 10, 10));
        InvalidateRect(hWnd, NULL, true);
        break;
     case WM_DESTROY:
         KillTimer(hWnd, 1);
         PostQuitMessage(0);
         return 0;
}
```

Изменение стиля

- используется функция SetClassLong

```
DWORD SetClassLong(HWND hWnd, int indexOldVal, long newVal);
```

GCL_HCURSOR	установить дескриптор курсора
GCL_HICON, GCL_HICONSM	установить дескриптор иконки
GCL_HBRBACKGROUND	установить дескриптор кисти для фона
GCL_MENUNAME	установить адрес строки с именем меню
GCL_STYLE	установить биты стиля оконного класса

```
DWORD cursorOldIndex = SetClassLong(hWnd, GCL_HCURSOR,
                                    LoadCursor(NULL, IDC_WAIT));
```

Вопросы?