Кнопки.
Текстовые поля

Кнопки

- относятся к классу BUTTON
- создаются с помощью функции CreateWindow или CreateWindowEx
- получают сообщения от родительского окна
- посылают родительскому окну сообщения WM_COMMAND

Создание кнопки (пример)

```
#define ID_BUTTON  3000

HWND hWndParent, hWndBut;
// …
hWndBut = CreateWindowEx(WS_EX_TOPMOST,
                         L"BUTTON",
                         L"Click",      // Текст на кнопке
                         WS_CHILD | WS_VISIBLE, // Стиль
                         10, 10, 200, 200,
                         hWndParent,  // Родитель
                         (HMENU) ID_BUTTON,  // ID кнопки
                         hInstance,
                         NULL);
if (!hWndBut){ return false; }
```

Стили кнопок

- часто начинаются с префикса BS_

BS_PUSHBUTTON	Обычная кнопка
BS_DEFPUSHBUTTON	Обычная кнопка с жирной рамкой
BS_CHECKBOX	Флажок с текстом (включен или выключен)
BS_AUTOCHECKBOX	Флажок без текста
BS_RADIOBUTTON	Переключатель с текстом
BS_AUTORADIOBUTTON	Переключатель без текста
BS_TEXT	Элемент управления – это текст
BS_GROUPBOX	Область для группировки кнопок
BS_ICON	Элемент управления – это иконка
BS_NOTIFY	Посылает родительскому окну сообщения **BN_DBLCLK**, **BN_KILLFOCUS** и **BN_SETFOCUS**

Сообщения кнопок

- посылают родительскому окну сообщения WM_COMMAND
- младшее слово параметра wParam (LOWORD(wParam)) – идентификатор кнопки
- старшее слово параметра wParam (HIWORD(wParam)) – код извещения
- параметр lParam – дескриптор окна кнопки

Получение идентификатора

```
int GetDlgCtrlID( HWND hwndCtrl);
```

```
int btnID;
/* … */
case WM_LBUTTONDOWN:
    btnID = GetDlgCtrlID(hBtn);
    /* … */
    break;
```

Получение дескриптора

```
HWND GetDlgItem( HWND hwnd,   // дескриптор окна
                 int nIDDlgItem  // ID элемента управления
                 );
```

```
HWND hBtn;
/* … */
case WM_LBUTTONDOWN:
   hBtn = GetDlgItem(hWnd, 101);
   /* … */
   break;
```

Передача сообщений

```
BOOL PostMessage( HWND hwnd, UINT uMsg,
                  WPARAM wParam, LPARAM lParam);
```

- передача сообщения в очередь (асинхронные сообщения)

```
LRESULT SendMessage(HWND hwnd, UINT uMsg,
                    WPARAM wParam, LPARAM lParam);
```

- передача сообщения сразу окну (синхронные сообщения)

Проверка состояния кнопки

- чтобы проверить состояние кнопки, посылается сообщение BM_GETSTATE

```
WORD nState = SendMessage(hButton, BM_GETSTATE, 0, 0);
```

- чтобы проверить состояние переключателя или флажка, посылается сообщение BM_GETCHECK

```
WORD nState = SendMessage(hButton, BM_GETCHECK, 0, 0);
        // 0 – не нажата; 1 – нажата; 2 – неактивный флажок
    (BST_UNCHECKED, BST_CHECKED, BST_INDETERMINATE)
```

Изменение состояния кнопки

- чтобы изменить состояние кнопки, посылается сообщение BM_SETSTATE

```
SendMessage(hButton, BM_SETSTATE, TRUE, 0);  // нажата
SendMessage(hButton, BM_SETSTATE, FALSE, 0); // не нажата
```

- чтобы изменить состояние переключателя или флажка, посылается сообщение BM_SETCHECK; не действует для обычных кнопок

```
SendMessage(hButton, BM_SETCHECK, BST_CHECKED, 0);
SendMessage(hButton, BM_SETCHECK,  BST_UNCHECKED, 0);
```

Обработка сообщений (пример)

```
case WM_COMMAND:
   switch(LOWORD(wParam)) {
      case ID_BUTTON1:
         MessageBox(hwnd, _TEXT("But 1"), _TEXT("1"), MB_OK);
         SendMessage((HWND)lParam, BM_SETSTATE, TRUE, 0);
         if (!SendMessage(hWndButton2, BM_GETCHECK, 0, 0))
            SendMessage(hWndButton2, BM_SETSTATE, FALSE, 0);
         break;
      case ID_BUTTON2:
         MessageBox(hwnd, _TEXT("But 2"), _TEXT("2"), MB_OK);
         if (!SendMessage(hWndButton1, BM_GETCHECK, 0, 0))
            SendMessage(hWndButton1, BM_SETSTATE, FALSE, 0);
      break;
     }
break;
```

Текстовые поля (Edit box)

- относятся к классу EDIT
- нужны для ввода текста с клавиатуры и его редактирования
- по умолчанию – однострочные
- создаются с помощью функции CreateWindow или CreateWindowEx
- получают сообщения от родительского окна
- посылают сообщения WM_COMMAND родительскому окну

Создание текстового поля

```
#define ID_EDIT 100

HWND hWndParent, hEdit;
// ...
hEdit = CreateWindowEx(WS_EX_TOPMOST,
                       L"EDIT",     // класс
                       L"???",    // текст в поле редактора
                       WS_CHILD | WS_VISIBLE,  // стили
                       10, 10, 100, 50,
                       hWndParent,  // родительское окно
                       (HMENU) ID_EDIT, // идентификатор
                       hInstance,
                       NULL);
```

Стили текстовых полей

- начинаются с префикса ES_

ES_LEFT, ES_RIGHT, ES_CENTER	Выравнивание текста
ES_UPPERCASE	Преобразовать текст к верхнему регистру
ES_LOWERCASE	Преобразовать текст к нижнему регистру
ES_MULTILINE	Многострочное текстовое поле
ES_WANTRETURN	При нажатии кнопки <Enter> перейти на следующую строку в текстовом поле
ES_READONLY	Запрет на редактирование текста в поле
ES_NUMBER	Разрешить вводить только цифры
ES_PASSWORD	Отображать при вводе не буквы, а *
ES_AUTOHSCROLL, ES_AUTOVSCROLL	Автоматическая прокрутка текста по вертикали или по горизонтали

Изменение стиля

- при выполнении стиль уже созданного элемента часто нельзя изменить
- в некоторых случаях можно изменить стиль с помощью функции SetWindowLong

```
LONG WINAPI SetWindowLong(HWND hWnd, int ind, LONG newS);
```

```
LONG style;
/* … */
case WM_LBUTTONDOWN:
    style = GetWindowLong(hEdit, GWL_STYLE);
    SetWindowLong(hEdit, GWL_STYLE, style | ES_NUMBER);
    break;
```

Сообщения текстовому полю

- чтобы послать сообщение текстовому полю, используется функция SendMessage

```
LRESULT SendMessage(HWND hwnd, UINT uMsg,
                    WPARAM wParam, LPARAM lParam);
```

```
case ID_BUT_DELETE:
    SendMessage(hEdit1, WM_CUT, 0, 0);
    break;
case ID_BUT_PASTE:
    SendMessage(hEdit2, WM_PASTE, 0, 0);
    SetFocus(hEdit1);
    break;
```

Коды сообщений

Код сообщения	Значение wParam	Значение lParam	Действие
WM_CLEAR	0	0	Удалить выделенный текст
WM_CUT	0	0	Переместить выделенный текст в буфер обмена
WM_COPY	0	0	Скопировать выделенный текст в буфер обмена
WM_PASTE	0	0	Вставить текст из буфера обмена
EM_GETLINE	Line	Buf	Копировать строку Line в буфер Buf
WM_GETTEXT	max	Buf	Копировать не более чем max символов в буфер Buf
WM_SETTEXT	0	&Line	Копировать текст из строки Line в текстовое поле
EM_SETSEL	Start	End	Выделить текст от позиции Start до End

Сообщения от текстовых полей

- родителю посылается сообщение WM_COMMAND
  - lParam – дескриптор текстового редактора
  - LOWORD(wParam) – идентификатор редактора
  - HIWORD(wParam) – код сообщения

```
case WM_COMMAND:
    switch (LOWORD(wParam)) {
        case ID_EDIT:
            if (HIWORD(wParam) == EN_ERRSPACE ) {  /* … */ }
            break;
      // …
    }
    break;
```

Коды сообщений

Код сообщения	Назначение
EN_CHANGE	Текст в редакторе будет меняться
EN_MAXTEXT	Превышена заданная для редактора длина текста
EN_UPDATE	Последняя операция над текстом выполнена, но поле редактора еще не обновлено
EN_SETFOCUS	Текстовый редактор получил фокус ввода
EN_KILLFOCUS	Текстовый редактор потерял фокус ввода
EN_ERRSPACE	Не хватает памяти, чтобы выполнить действие
EN_HSCROLL	По горизонтальной линейке прокрутки щелкнули мышью
EN_VSCROLL	По вертикальной линейке прокрутки щелкнули мышью

Обработка сообщений (пример)

```
wchar_t buf[100];
// …
case WM_COMMAND:
   switch(LOWORD(wParam)) {
      case ID_BUTTON:
         SendMessage(hEdit, EM_GETLINE, 0, (LPARAM)buf);
         MessageBox(hwnd, buf, L"Ваш текст", MB_OK);
         SetWindowText(hEdit, NULL);
         SetFocus(hEdit);
      break;
     // …
}
break;
```

Вопросы?