Лабораторная работа №6

Изучение средств Windows для построения пользовательского интерфейса

Работа выполняется в системе программирования Borland Delphi 2.0 в среде Windows 95. Использование средств визуального программирования и классов VCL не допускается, то есть Delphi используется просто как 32-разрядный компилятор с языка Паскаль для операционной системы Windows. Разрешается использовать функции работы со строками Delphi из модуля SysUtils.

Теоретические положения

При изучении теоретического материала полезно обратиться к описанию ЛР3: шаблон программы Windows.

Оконные органы управления (controls)

В системе Windows реализованы средства для создания современного пользовательского интерфейса, включающего такие элементы как кнопки (buttons), списки (listboxes), полосы прокрутки (scrollbars), окна редактирования (editboxes) и т.д. Все эти элементы управления в программе должны порождаться как дочерние окна окон программы - экземпляры предописанных классов. Такими классами являются (перечисляются идентификаторы, используемые в процедуре CreateWindow):

button - кнопка, переключатель (radio button), флажок (checkbox).
static - просто прямоугольная область.
scrollbar - горизонтальная или вертикальная полоса прокрутки.
edit - поле редактирования, одно- или многострочное.
listbox - список.
combobox - выпадающий список или список, комбинированный со строкой редактора.

При создании органа управления в процедуру CreateWindow, в отличие от создания обычного окна программы, должны быть переданы следующие параметры:

```
function CreateWindow(
  lpClassName: PChar;  // имя класса окна управления, малыми буквами, как представлено выше
  lpWindowName: PChar; // текст окна (надпись на кнопке и т.д.)
  dwStyle: DWORD;  // Стиль окна должен включать флаги WS_CHILD и (обычно)
                   // WS_VISIBLE + стиль органа управления
  X, Y: integer    // Положение и размеры органа управления в координатах
                   // рабочей области родительского окна
  nWidth, nHeight: Integer;  // Ширина и высота
  hWndParent: HWND; // Хэндл окна-родителя
  hMenu: HMENU;     // Идентификатор (номер) органа управления, определяется
                    // программистом
  hInstance: HINST;  // Экземпляр приложения, создающий окно
  lpParam: Pointer   // nil
): HWND;
```

Дочерние окна обрабатывают сообщения от мыши и клавиатуры. Когда пользователь производит с органом управления какие-либо действия, окну-родителю посылается сообщение WM_COMMAND, содержащее информацию о производимом действии и идентификатор (номер) окна управления, присвоенный ему при создании. Оконная процедура родительского окна должна уметь обрабатывать такие сообщения. Параметры сообщения WM_COMMAND:

loword(wParam) - идентификатор дочернего окна управления
hiword(wParam) - код уведомления (операции)
lParam - хэндл дочернего окна управления.

Родительское окно управляет дочерними, тоже посылая им сообщения. Эти сообщения зависят от класса окон управления.

Кроме того, для управления дочерними окнами часто используются функции ShowWindow, MoveWindow, IsWindowVisible, EnableWindow, IsWindowEnabled, SetWindowText, GetWindowText. Подробнее об этих функциях см. Справку по Win32 API.

Для переключения фокуса ввода между окнами управления в диалогах обычно используются клавиши Tab и Shift-Tab. Windows содержит средства для автоматизации этого действия без значительного усложнения пользовательской программы.

Дочерние окна управления автоматически уничтожаются при уничтожении родительского окна.

Цикл обработки сообщений для диалогов

Чаще всего оконными органами управления "оборудуются" окна, предназначенные исключительно для организации диалога с пользователем — ввода значений, установки параметров и т.д. Такие окна называют диалоговыми или просто диалогами. Windows предоставляет программисту возможность без особых затрат труда организовать переключение органов управления при нажатии пользователем стандартных комбинаций клавиш (самые распространенные комбинации — Tab и Shift-Tab). Для этого цикл обработки сообщений программы должен быть модифицирован следующим образом (это может быть как основной цикл программы, так и специально организуемый для конкретного диалога цикл обработки сообщений).

Пусть хэндл диалогового окна хранится в переменной hDlg. тогда цикл обработки сообщений должен выглядеть как:

```
while GetMessage(msg,0,0,0)=true do begin {получить очередное сообщение}
  if not IsDialogMessage(hDlg,msg)
         {Если Windows не распознает и не обрабатывает клавиатурные сообщения
          как команды переключения между оконными органами управления,
          тогда чообщение идет на стандартную обработку}
  then begin
    TranslateMessage(msg);   {Windows транслирует сообщения от клавиатуры}
    DispatchMessage(msg);    {Windows вызовет оконную процедуру}
  end;
end; {выход по wm_quit, на которое GetMessage вернет FALSE}
```

Функция IsDialogMessage в этом случае анализирует текущее сообщение MSG и, если сообщение содержит информацию о нажатии стандартной комбинации клавиш, служащей для перемещения между органами управления диалога, обрабатывает его. Если сообщение обрабатывается, функция возвращает TRUE и программа обрабатывать такое сообщение уже не должна, в противном случае возвращается FALSE и сообщение обрабатывается на общих основаниях.

В ходе выполнения функции IsDialogMessage Windows проверяет, есть ли в окне с указанным хэндлом (hDlg) оконные органы управления вообще, и если они есть, то выполняется последовательная посылка ряда сообщений им для смены текущего органа управления.

Кнопки (класс button)

Поддерживаются следующие стили кнопок:

Стиль	Характеристика

BS_PUSHBUTTON	Обычная кнопка ("нажимаемая кнопка").
BS_DEFPUSHBUTTON	Кнопка с более жирной рамкой. В диалогах, описываемых в ресурсах, считается нажатой при нажатии пользователем клавиши Enter.
BS_CHECKBOX	Флажок, т.е. надпись, слева или справа от которой расположено квадратное поле, которое может быть отмечено крестиком.
BS_AUTOCHECKBOX	То же, что предыдущее, но отметка или снятие отметки происходит автоматически, не требуя участия родительского окна.
BS_3STATE	То же, что флажок, но возможно также "третье состояние", когда поле флажка отображается серым цветом.
BS_AUTO3STATE	То же, что предыдущее, но переключение между тремя возможными состояниями происходит автоматически при щелчке по флажку.
BS_RADIOBUTTON	Индикатор исключающего выбора: кружок, слева от которого размещена строка текста. Кружок может быть отмечен точкой в центре.
BS_AUTORADIOBUTTON	То же, что предыдущее, но при щелчке по индикатору он автоматически становится выбранным, а все другие индикаторы в той же группе (того же родительского окна) — невыбранными.
BS_GROUPBOX	Рамка с текстом в верхнем левом углу, обычно служит для визуального объединения радио-кнопок в группу. Само по себе не реагирует на события от клавиатуры и мыши.
BS_OWNERDRAW	Кнопка пользователя, отрисовкой внешнего вида кнопки должна заниматься оконная процедура родительского окна, обрабатывая сообщение WM_DRAWITEM.

При нажатии кнопок в сообщении WM_COMMAND содержится код уведомления BN_CLICKED.

Родительское окно может посылать окнам кнопок управляющие сообщения:

Сообщение	**Описание**
BM_GETCHECK	Возвращает для флажков и индикаторов выбора состояние выбора: 0 (не помечено), 1 (помечено), 2 (третье состояние).
BM_SETCHECK	Устанавливает состояние выбора для флажков и индикаторов выбора. В wParam должен содержаться код состояния (см. выше).
BM_GETSTATE	Возвращает состояние как комбинацию битовых флагов: биты 0 и 1 — содержат код, возвращаемый BM_GETCHECK; бит 2 — нажата (1) или отпущена (0) кнопка; бит 3 — имеет ли кнопка фокус.
BM_SETSTATE	Для нажимаемых кнопок устанавливает состояние. wParam = 0 — кнопка отпущена, wParam = 1 — кнопка нажата.
BM_SETSTYLE	Позволяет изменить стиль кнопки после ее создания. wParam - содержать новый стиль как комбинация битовых флагов,

lParam - 1 - перерисовать кнопку, 0 - не перерисовывать.

Например, изменить состояние окна-флажка с хэндлом hCheckBox1 на противоположное можно с помощью следующей конструкции:

```
SendMessage(hCheckBox1, BM_SETCHECK,
            SendMessage(hCheckBox1, BM_GETCHECK,0,0) xor 1,
            0);
```

Для того, чтобы изменить способ отображения кнопок или их реакцию на внешние события, необходимо создать собственный оконный класс на основе класса button (рассматривается ниже).

Статические окна (класс static)

Эти окна отображаются как закрашенные прямоугольники или рамки с текстом. Обычно эти окна не обрабатывают событий от клавиатуры и мыши. Они могут использоваться как контейнеры для других дочерних окон управления. Подробнее см. Справку по Win32 API, функция CreateWindow.

Полосы прокрутки

Окна Windows могут иметь оконные горизонтальные и вертикальные полосы прокрутки, что определяется включением в стиль окна флагов WS_VSCROLL и WS_HSCROLL. Помимо этого существуют также дочерние окна управления класса scrollbar, которые выглядят точно так же, но, в отличие от полос прокрутки окна, могут располагаться в любом месте и иметь произвольный размер. Далее будет рассмотрено использование обоих типов полос прокрутки.

Окна с полосами прокрутки (полосы — не дочерние окна)

Чтобы окно имело вертикальную полосу прокрутки, в его стиле при вызове функции CreateWindow должен присутствовать флаг WS_VSCROLL, горизонтальную — WS_HSCROLL. Такие полосы прокрутки (назовем их оконными) размещаются в окне вдоль правой и нижней границ и имеют протяженность вдоль всей границы окна в пределах рабочей области. Пространство, занятое оконными полосами прокрутки, не включается в рабочую область. Ширина полос прокрутки является общесистемным параметром и ее можно узнать с помощью функции GetSystemMetrics (см. также SystemParametesInfo).

Оконные полосы прокрутки управляются при помощи мыши, они не реагируют на нажатия клавиш, поэтому программист должен "вручную" реализовывать прокрутку при нажатии на клавиши управления курсором в оконной процедуре.

Полосы прокрутки характеризуются диапазоном, определяемым двумя целыми числами — минимальным и максимальным положением бегунка. Положение бегунка всегда дискретно и соответствует целому числу из диапазона возможных положений от минимального до максимального. Так, если диапазон задан как "от 0 до 100" (по умолчанию), то бегунок может находиться в одном из 101 возможных положений, и его положение будет возвращаться как целое число от 0 до 100. Диапазон по умолчанию не всегда удобен, поэтому есть возможность переопределить его при помощи вызова функции

```
function SetScrollRange(hWnd: HWND; nBar, nMinPos, nMaxPos: Integer;
                        bRedraw: BOOL): BOOL;
```

Узнать текущий диапазон оконной полосы прокрутки можно с помощью функции

```
function GetScrollRange(hWnd: HWND; nBar: Integer; var nMinPos, nMaxPos:
                        Integer): BOOL;
```

Здесь hWnd - хэндл окна, nBar - SB_VERT или SB_HORZ, nMinPos и nMaxPos - минимальное и максимальное положение полос прокрутки, bRedraw - определяет, требуется ли перерисовка полосы прокрутки сразу после установления новых параметров. При неуспешном завершении операции возвращается FALSE.

Если не выполняется условие nMinPos<nMaxPos, то полоса прокрутки становится невидимой.

Для работы с положением бегунка используются функции

```
function SetScrollPos(hWnd: HWND; nBar, nPos: Integer; bRedraw: BOOL): Integer;
function GetScrollPos(hWnd: HWND; nBar: Integer): Integer;
```

Здесь hWnd — хэндл окна, nBar - SB_VERT или SB_HORZ, bRedraw - определяет, требуется ли перерисовка полосы прокрутки сразу после установления новых параметров. Возвращаемое значение — старое положение бегунка (оно же во втором случае — текущее).

Сообщения полос прокрутки

При действиях с полосами прокрутки оконной процедуре посылаются сообщения WM_VSCROLL (вертикальная прокрутка) и WM_HSCROLL (горизонтальная прокрутка). В младшем слове wParam содержится код операции с полосой прокрутки, lParam для сообщений оконных полос прокрутки можно игнорировать. Кодами операции в младшем слове wParam могут быть:

SB_BOTTOM	Прокрутить в самый конец
SB_ENDSCROLL	Кнопка мыши отпущена, операция завершена (это сообщение как правило можно игнорировать)
SB_LINEDOWN	Прокрутка вниз на одну строку, возникает при нажатии на стрелку внизу (справа) полосы прокрутки.
SB_LINEUP	Прокрутка вверх на одну строку, возникает при нажатии на стрелку вверху (слева) полосы прокрутки.
SB_PAGEDOWN	Прокрутка на страницу вниз, возникает при щелчке мыши на полосе прокрутки ниже (справа от) бегунка.
SB_PAGEUP	Прокрутка на страницу вверх, возникает при щелчке мыши на полосе прокрутки выше (слева от) бегунка.
SB_THUMBPOSITION	Установить позицию бегунка. Позиция содержится в старшем слове wParam. Возникает после перетаскивания бегунка мышью.
SB_THUMBTRACK	Позиция в время перетаскивания. Позиция содержится в старшем слове wParam. Возникает во время перетаскивания ползунка мышью.
SB_TOP	Прокрутка в самое начало.

Как видно из приведенной таблицы, положение бегунка в сообщениях SB_THUMBPOSITION и SB_THUMBTRACK передается в виде 16-разрядного целого, поэтому "полнофункциональная" полоса прокрутки должна иметь диапазон, определяемый числами в диапазоне от 0 до 65535. Функции SetScrollRange, SetScrollPos, GetScrollRange и GetScrollPos оперируютс 32-разрядными целыми, однако положение бегунка, выходящее за 16-разрядный диапазон, не может быть корректно передано в теле сообщения. При обработке сообщения SB_THUMBPOSITION есть возможность узнать 32-разрядное

положение бегунка при помощи функции GetScrollPos, однако в случае SB_THUMBTRACK с ее помощью узнать положение бегунка невозможно, так как функция GetScrollPos не отражает положения бегунка в процессе его перетаскивания.

При перетаскивании бегунка в очередь сообщений окна помещается множество сообщений с признаком SB_THUMBTRACK, после чего по окончании операции посылается сообщение с SB_THUMBPOSITION.

Простейший способ реакции на сообщения WM_VSCROLL и WM_HSCROLL — вызов функции ScrollWindow для прокрутки клиентской области окна. Прокрутка состоит в том, что изображение в окне соответствующим образом сдвигается, а "открывшиеся" после сдвига участки становятся недействительными, и в очередь сообщений помещается сообщение WM_PAINT. Дочерние окна и начало координат окна также сдвигаются.

Полосы прокрутки — окна класса scrollbar

Полоса прокрутки может быть помещена в произвольном месте окна. Для этого необходимо создать ее как дочернее окно управления класса scrollbar, указав стиль SBS_VERT или SBS_HORZ (см. Справку по WIN32 API: CreateWindow). Полоса прокрутки при этом занимает весь указанный в параметрах CreateWindow (или MoveWindow) прямоугольник, т.е. ее ширина и высота могут быть произвольными. Системные настройки ширины стандартных оконных полос прокрутки можно узнать, вызвав функции GetSystemMetrix(SM_CYHSCROLL) или GetSystemMetrix(SM_CXVSCROLL).

Окна класса scrollbar, как и стандартные полосы прокрутки окна, посылают родительскому окну сообщения WM_VSCROLL или WM_HSCROLL вместо сообщений WM_COMMAND, посылаемых остальными оконными органами управления. Отличия этих сообщений от сообщений, посылаемых оконными полосами прокрутки, состоят в том, что для оконных полос прокрутки lParam содержит 0, а для дочерних окон - полос прокрутки — хэндл такого дочернего окна. В отличие от стандартных оконных полос прокрутки, окна класса scrollbar могут получать фокус и обрабатывать нажатия клавиш клавиатуры. Нажатия клавиш интерпретируются следующим образом:

Клавиша	wParam сообщения VM_xSCROLL
Home	SB_TOP
End	SB_BOTTOM
Page Up	SB_PAGEUP
PageDown	SB_PAGEDOWN
Стрека влево, стрелка вверх	SB_LINEUP
Стрелка вправо, стрелка вниз	SB_LINEDOWN

Управление полосами прокрутки класса scrollbar осуществляется теми же функциями GetScrollPos, GetScrollRange, SetScrollPos и SetScrollRange, что и для стандартных оконных полос прокрутки, однако в параметре hWnd следует указывать хэндл самой полосы прокрутки, а в nBar — константу SB_CTL.

Окна редактирования (класс edit)

Окна редактирования представляют собой одно- или многострочные текстовые редакторы. Они могут иметь клавиатурный фокус и позволяют вводить текст, редактировать его при помощи клавиш управления, выделять при помощи клавиш и мыши и производить

операции копирования и вставки с буфером обмена при помощи клавиш Ctrl-Ins, Shift-Ins и Shift-Del. Окна редактирования могут иметь собственные стандартные полосы прокрутки, что определяется включением флагов WS_VSCROLL и WS_HSCROLL в стиль окна редактирования при его создании.

Окно редактирования создается как экземпляр класса edit. В стиле окна могут присутствовать следующие специфичные для окон-редакторов флаги:

ES_MULTILINE Многострочный редактор.

ES_AUTOHSCROLL Автоматически используется горизонтальная полоса прокрутки, если текст не умещается по ширине окна. При отсутствии этого флага производится автоматический перенос на следующую строку.

ES_AUTOVSCROLL Автоматически появляется вертикальная полоса прокрутки, если текст не умещается по высоте окна.

ES_NOHIDESEL Выделенный текст подсвечивается, даже когда окно теряет фокус, иначе при потере фокуса подсветка текста гасится.

С помощью функции SetWindowText можно задать текст в окне редактирования, с помощью GetWindowtextLength и GetWindowText — получить текст. Многострочные окна редактирования воспринимают последовательность #13#10 как конец строки; следующие символы выводятся со следующей строки, а символы конца строки не отображаются. Окна редактирования не обрабатывают символ табуляции (#9) и отображают его в виде вертикальной черты ("|"). Чтобы в обычном окне (не в диалоге) обрабатывать нажатия клавиш Tab, Shift-Tab, Enter как управляющие, необходимо вводить новую оконную процедуру для окон редактирования (см. ниже). Размер редактируемого текста ограничен примерно 32 килобайтами, при этом используется область памяти, выделенная приложению, содержащему окно редактора.

Окна редактора посылают родительскому окну сообщения WM_COMMAND со следующими кодами уведомления в старшем слове wParam:

EN_SETFOCUS Редактор получил фокус ввода

EN_KILLFOCUS Редактор потерял фокус ввода

EN_CHANGE Содержимое изменилось, посылается после вывода на экран

EN_UPDATE Содержимое изменилось, посылается перед обновлением экрана

EN_ERRSPACE Буфер редактирования переполнился

EN_MAXTEXT Буфер редактирования переполнился при вставке

Программа может управлять окном редактора, посылая ему сообщения. Наиболее важные из них:

EM_GETLINECOUNT — возвращает число строк в многострочном редакторе.

EM_LINEINDEX, wParam = номер строки — возвращает смещение начала строки относительно начала буфера редактирования. Строки нумеруются с нуля. wParam=-1 позволяет узнать смещение строки, содержащей курсор.

EM_LINELENGTH, wParam = номер строки — возвращает длину указанной строки.

EM_GETLINE, wParam = номер строки, lParam = указатель на буфер — копирует указанную строку в заданный буфер, заканчивая ее нулем.

См. также сообщения WM_CUT, WM_COPY, WM_PASTE, EM_GETSEL, EM_SETSEL, EM_REPLACESEL, с помощью которых реализуются операции с буфером обмена.

Списки (класс listbox)

Окна класса listbox представляют собой списки Windows. Список — это набор текстовых строк, которые выводятся в прямоугольном прокручиваемом окне в один или несколько столбцов. Окно списка может получать фокус и способно обрабатывать сообщения клавиатуры. Когда окно списка имеет фокус, одна из строк отображается с пунктирной рамкой и считается имеющей фокус. Кроме того, строки могут быть выделены цветом, в этом случае они считаются выбранными. Наличие у строки фокуса в общем случае не означает, что она выбрана.

По умолчанию списки не посылают родительскому окну сообщения WM_COMMAND; чтобы посылка извещений происходила, необходимо включать в стиль окна списка флаг LBS_NOTIFY. Другими важными флагами стиля являются:

LBS_SORT	Строки списка автоматически сортируются
LBS_MULTIPLESEL	Позволяет выделять несколько строк списка (список с множественным выбором)

Для того, чтобы список имел рамку и полосу прокрутки, необходимо включить в стиль окна списка флаги WS_BORDER и WS_VSCROLL. Интересные результаты получаются при использовании с окном списка флагов WS_CAPTION, WS_SIZEBOX, WS_HSCROLL.

Для управления списком используются следующие сообщения, посылаемые окну списка при помощи функции SendMessage:

LB_ADDSTRING, lParam = указатель на Z-строку — добавляет строку в конец несортированного списка, в сортированном списке строка оказывается на нужном месте в соответствии с алгоритмом сортировки.

LB_INSERTSTRING, wParam = позиция, lParam = указатель на Z-строку — вставляет строку в указанной позиции, все остальные строки сдвигаются. Если список сортированный, пересортировка не производится. При wParam=-1 строка добавляется в конец списка.

LB_DELETSTRING, wParam = позиция — удаляет строку в заданной позиции.

LB_RESETCONTENT — полная очистка списка.

WM_SETREDRAW, wParam = 0 - разрешить, 1 - запретить — устанавливает флаг, разрешающий или запрещающий перерисовку списка при изменении его содержимого. При последовательном добавлении или удалении нескольких строк полезно сбросить этот флаг перед началом операции и взвести по окончании.

LB_GETCOUNT — возвращает количество элементов в списке.

LB_SETCURSEL, wParam = номер строки — в списке с одиночным выбором делает указанную строку выбранной, при wParam=-1 ни одна строка не выбрана.

LB_SELECTSTRING, wParam = номер первой строки, lParam = указатель на Z-строку для поиска — делает выбранной первую подходящую строку, начиная с wParam. Если wParam=-1, то поиск начинается с начала списка. Выбирается строка, начинающаяся со строки, задаваемой lParam. При невозможности найти подходящую строку возвращается LB_ERR.

LB_GETCURSEL — возвращает номер текущей выбранной строки или -1.

LB_GETTEXTLEN, wParam = номер строки — возвращает длину указанной строки.

LB_GETTEXT, wParam = номер строки, lParam = адрес буфера — записывает в буфер заданную строку списка, завершая ее нулем.

LB_SETSEL, wParam = 0 - снять выделение, 1 - включить выделение, lParam - номер строки — в списках с множественным выбором выделяет или снимает выделение указанного элемента списка, если lParam=-1, то выделяется или снимается выделение у всех элементов списка.

LB_GETSEL, wParam = номер строки — возвращает 0, если строка не выбрана, и другое значение, если выбрана.

Окна списков в свою очередь извещают родительские окна о действиях с ними путем посылки им сообщений WM_COMMAND со следующими кодами уведомления в старшем слове wParam:

LBN_ERRSPACE	При добавлении строки не хватило памяти.
LBN_SELCHANGE	Изменен текущий выбор.
LBN_DBLCLICK	Был двойной щелчок мышью по списку.
LBN_SETFOCUS	Список получил фокус.
LBN_KILLFOCUS	Список потерял фокус.

Создание собственных окон управления. Subclassing и superclassing

Windows позволяет программисту создавать собственные оконные классы на основе уже существующих и модифицировать поведение уже созданных окон. В Справке по Win 32 API (из Software Development Kit) при описании этих вопросов используются термины subclassing (создание подклассов) и superclassing (создание суперклассов). Эти термины представляются не очень удачными, поскольку в объектно-ориентированном программировании термин "суперкласс" имеют несколько иное значение, а здесь наблюдается прямая аналогия с принципами ООП. "Subclassing" в документации по Windows означает подмену оконной процедуры для конкретного окна или для всего существующего класса другой оконной процедурой, "superclassing" — создание нового оконного класса на основе уже существующего с заменой оконной процедуры. (В ООП суперклассом принято называть класс-предок, в документации же по Windows под суперклассом подразумевается надстройка над существующим классом, т.е. то, что в ООП называлось бы классом-потомком.)

Замена оконной процедуры существующего окна

Этот прием используется, когда необходимо модифицировать поведение существующего окна. Используя его можно, например, сделать так, чтобы некоторые клавиатурные сообщения от дочернего окна управления, имеющего фокус, попадали в родительское окно (например, о нажатиях клавиш Tab и Enter). Для этого необходимо:

1. Описать новую оконную процедуру, предусмотрев в ней новую обработку для нужных сообщений; остальные сообщения должны передаваться старой оконной процедуре.

2. Прочитать адрес заменяемой оконной процедуры из локальной памяти окна.

3. Записать адрес новой оконной процедуры в локальную память окна.

Новая оконная процедура может поступать с сообщениями следующим образом:

- передавать их на обработку старой оконной процедуре;

- модифицировать сообщения и передавать их старой процедуре;

- обрабатывать сообщения самостоятельно, не передавая старой процедуре.

Вместо вызова DefWindowProc в новой оконной процедуре сообщения как правило передаются старой оконной процедуре при помощи функции

```
function CallWindowProc(lpPrevWndFunc: TFNWndProc;
        hWnd: HWND; Msg: UINT; wParam: WPARAM; lParam: LPARAM): LRESULT;
```

Первый параметр этой функции — указатель на старую оконную процедуру, остальные параметры — параметры для вызова оконной процедуры.

Для получения адреса оконной процедуры окна hwnd используется вызов

```
pOldProc := pointer( GetWindowLong(hwnd, GWL_WNDPROC) );
```

получаемое 32-битное целое должно быть затем преобразовано к типу pointer. Можно совместить получение адреса старой оконной процедуры с заменой его на новый, так как функция SetWindowLong возвращает старое значение изменяемой ячейки памяти окна:

```
pOldProc := pointer( SetWindowLong(hwnd, GWL_WNDPROC, integer(@NewWndProc) ) );
```

Создание нового оконного класса на основе существующего

Часто бывает необходимо изменить поведение не одного окна, а всех окон какого-либо класса, например, чтобы все кнопки передавали родительскому окну на обработку сообщения от клавиатуры. Для этого можно изменить оконную процедуру у каждой из создаваемых кнопок, но логичнее было бы создать новый класс кнопок, которые ведут себя так же, как и кнопки button, но еще и отдают сообщения от клавиатуры главному окну.

Для этого необходимо:

1. Получить описание базового класса.

2. Модифицировать описание базового класса, создав таким образом описание нового класса. В частности, обычно вводится новая оконная процедура, обрабатывающая часть сообщений и передающая остальные старой процедуре при помощи вызова CallWindowProc, как это описано выше.

3. Зарегистрировать новый класс в системе.

4. Использовать зарегистрированный класс для порождения окон.

Описание класса в виде структуры TWndClassEx можно получить с помощью функции GetClassInfoEx, затем в полученной структуре заменяются поля указателя на оконную процедуру и имени класса. Указатель на оконную процедуру базового класса должен быть сохранен для ее вызова из оконной процедуры нового класса. Модифицированное описание класса регистрируется с помощью RegisterClassEx.

Задание

Выполняется при самостоятельной подготовке:

1. Изучить теоретическую часть описания ЛР и материал соответствующих лекций.

2. По материалам Справки (Help) изучить описание следующих сообщений Windows и связанных с ними функций и структур данных:

Сообщения:

WM_COMMAND, WM_VSCROLL, WM_HSCROLL;

BM_GETCHECK, BM_SETCHECK, BM_GETSTATE, BM_SETSTATE, BM_SETSTYLE;

EM_GETLINECOUNT, EM_LINEINDEX, EM_LINELENGTH, EM_GETLINE, WM_CUT, WM_COPY, WM_PASTE, EM_GETSEL, EM_SETSEL, EM_REPLACESEL;

LB_RESETCONTENT , LB_ADDSTRING, LB_INSERTSTRING, LB_DELETSTRING, WM_SETREDRAW, LB_GETCOUNT, LB_SETCURSEL, LB_SELECTSTRING, LB_GETCURSEL, LB_GETTEXTLEN, LB_GETTEXT, LB_SETSEL, LB_GETSEL.

Функции:

SetWindowText, GetWindowtextLength, GetWindowText, SendMessage, CreateWindow, ShowWindow, MoveWindow, ScrollWindow, IsWindowVisible, EnableWindow, IsWindowEnabled, GetWindowLong, GetDlgCtrlId, SetScrollPos, GetScrollPos, SetScrollRange, GetScrollRange, GetWindowLong, SetWindowLong, SendDlgItemMessage, CheckDlgButton, CheckRadioButton, IsDlgButtonChecked.

3. Продумать алгоритм решения индивидуального задания.

Выполняется в лаборатории:

1. Включить компьютер. Запустить систему Delphi 2.0.

2. Загрузить исходный текст примера LAB6.PAS, а также модули WINDOWS.PAS, MESSAGES.PAS, изучить логику работы программы.

3. Откомпилировать и запустить пример. Изучить поведение созданных окон.

4. Написать и отладить программу по индивидуальному заданию (см. ниже). Продемонстрировать результаты работы преподавателю.

5. Завершить работу в Delphi. Оставить компьютер включенным.

Варианты заданий:

Написанная программа должна обеспечивать перемещение между органами управления с помощью стандартных комбинаций клавиш клавиатуры. Приветствуется расширение функциональных возможностей окна по инициативе студента, но с при выполнении основного условия задания. Разрешается использовать функции работы со строками Delphi из библиотеки SysUtils.

1. "Регулятор цвета". Окно, содержащее три полосы прокрутки и три информационных (не модифицируемых) поля ввода для красной, зеленой и синей составляющей цвета. При перемещении полосы прокрутки значение соответствующей цветовой составляющей должно отображаться в соответствующем информационном окошке. Заданный цвет должен произвольным образом отображаться в рабочей области главного окна. Каждая составляющая — число от 0 до 255.

2. "Регулятор цвета". Окно, содержащее три информационных (не модифицируемых) полосы прокрутки и три поля ввода для красной, зеленой и синей составляющей цвета. После ввода составляющей положение соответствующей полосы прокрутки должно изменяться. При неверном вводе значения должно выдаваться сообщение об ошибке. Заданный цвет должен произвольным образом отображаться в рабочей области главного окна. Каждая составляющая — число от 0 до 255.

3. "Калькулятор" с числовыми кнопками, полем индикации, клавишами четырех арифметических действий, стирания и равенства. Все клавиши должны быть реализованы в виде органов управления - кнопок.

4. "Список с поиском". Окно содержит список (изначально чем-то заполненный) и поле ввода. При выборе элемента списка в поле ввода должна появляться выбранная строка. В процессе набора в поле ввода подсветка в списке должна перемещаться к строке, начало которой совпадает с вводимой строкой. если совпадения нет, подсветка должна исчезать.

5. "Ханойская башня" или что-то отдаленно ее напоминающее. Три списка, строку из конца каждого списка можно переносить в конец любого другого списка. Проверок строк на допустимость переноса выполнять не требуется. Предусмотреть органы управления, позволяющие задать, в какой список (и из какого, если это требуется) осуществляется перенос.

6. "Транслитератор". В многострочном редакторе пользователь имеет возможность набрать произвольный русский текст ограниченной длины (2000 символов). После нажатия соответствующей кнопки текст переносится в другой редактор, при этом все русские буквы заменяются на эквивалентные латинские буквы или буквосочетания, например В→V, Ж→ZH и т.д. Редактор с транслитерированным текстом также допускает редактирование.

7. "Транслитератор". В многострочном редакторе пользователь имеет возможность набрать произвольный русский текст ограниченной длины (2000 символов). В процессе ввода (при вводе и удалении символов) текст подвергается транслитерации, при этом все русские буквы заменяются на эквивалентные латинские буквы или буквосочетания, например В→V, Ж→ZH и т.д. Транслитерированный текст отображается в другом поле ввода, редактирование в котором запрещено.

8. В программе два окна — "администратор" и "диалог". В окне администратора пользователь имеет список паролей, в который может добавлять пароль через строку редактора или удалять его (предусмотреть как). В окне диалога пароль вводится в "секретном" режиме (символы заменяются звездочками), после нажатия кнопки "Проверка" пользователю выдается сообщение (например, "Доступ открыт" и "Доступ запрещен"), свидетельствующее о наличии или отсутствии введенного пароля в списке паролей.

9. "Калькулятор" для перевода целых чисел из одной системы счисления в другую. В поле ввода набирается число в заданной системе счисления. В поле вывода появляется преобразованное число. Поддерживать системы счисления с основанием 2, 8, 10, 16. Предусмотреть орган управления для задания системы счисления.

10. "Создатель окон". Пользователь вводит параметры окна, которое он хочет создать, нажимает соответствующую кнопку, и окно с заданными параметрами появляется на экране. Числовые атрибуты вводить в редакторах, остальные выбирать из списков возможных значений. Предусмотреть как минимум ввод характеристик:

> ширина-высота, положение на экране, наличие заголовка, сист. меню, кнопок минимизации и максимизации, типа рамки, ввод строки заголовка, цвет клиентской области (хотя бы черный-белый-серый).