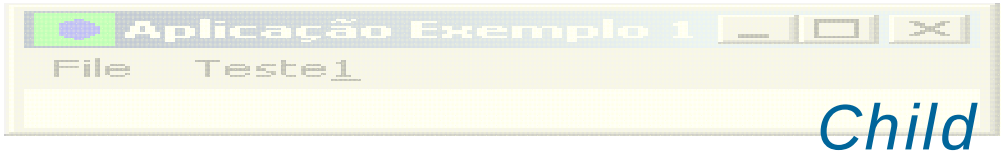





# Modo GUI na API Win32

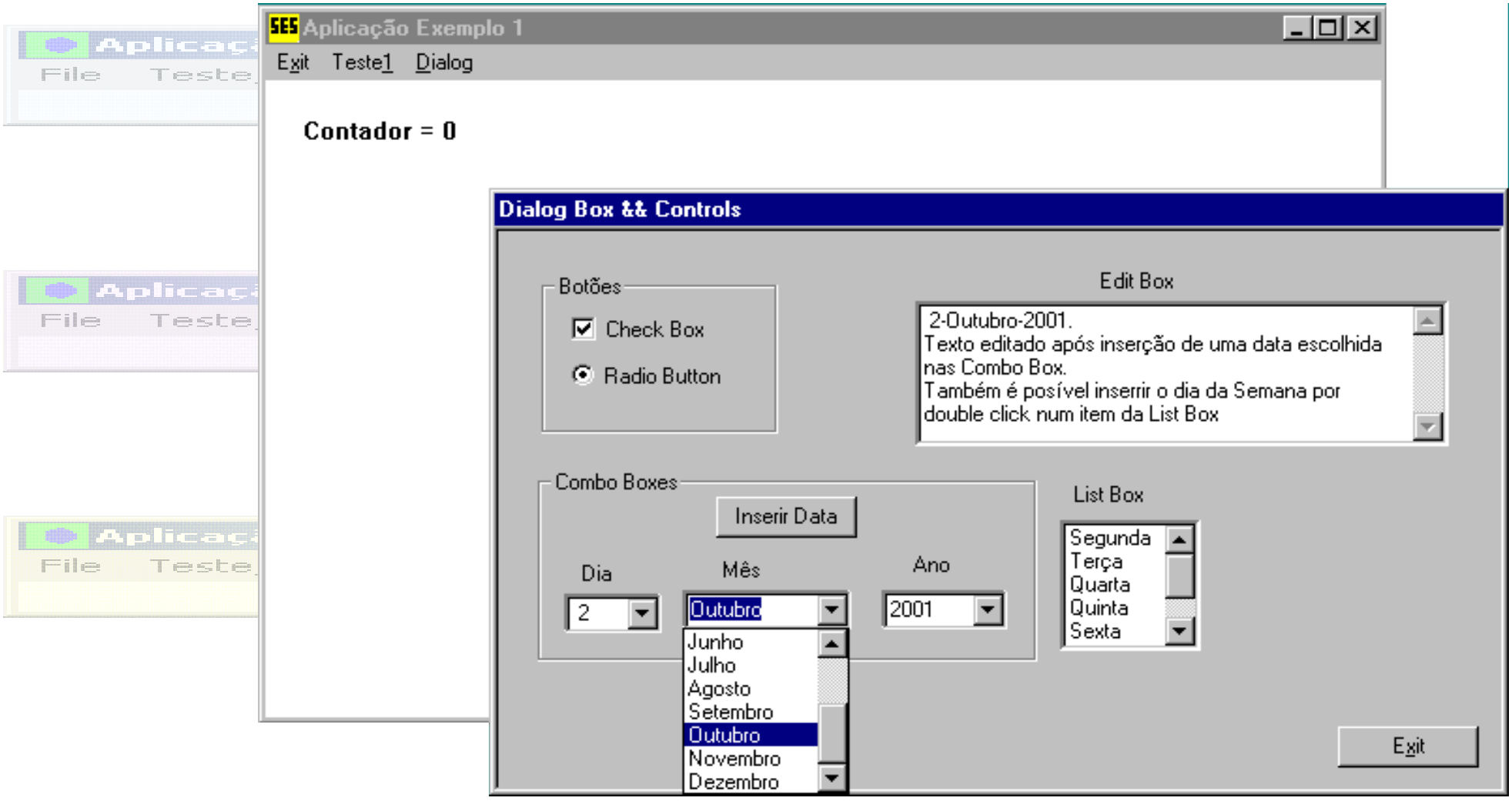
*Anexo 2:*

Janelas de Diálogo

e

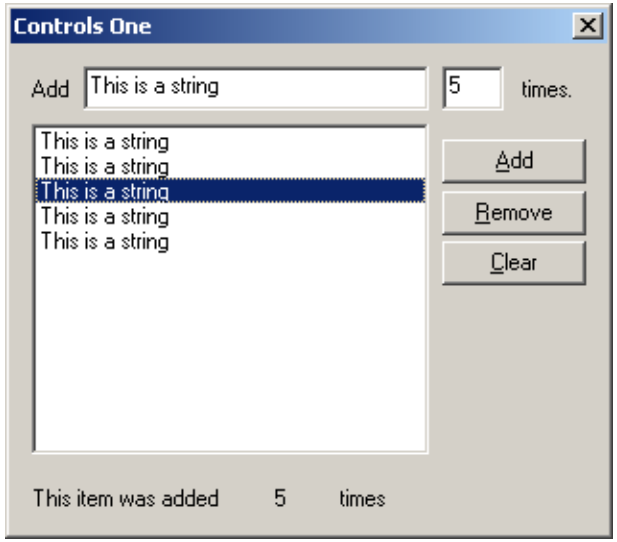
*Child Window Controls*

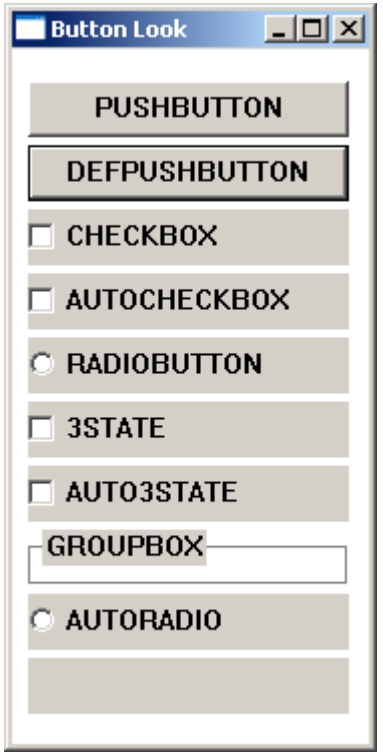
# Janelas de diálogo
# ( *Dialog Boxes* )

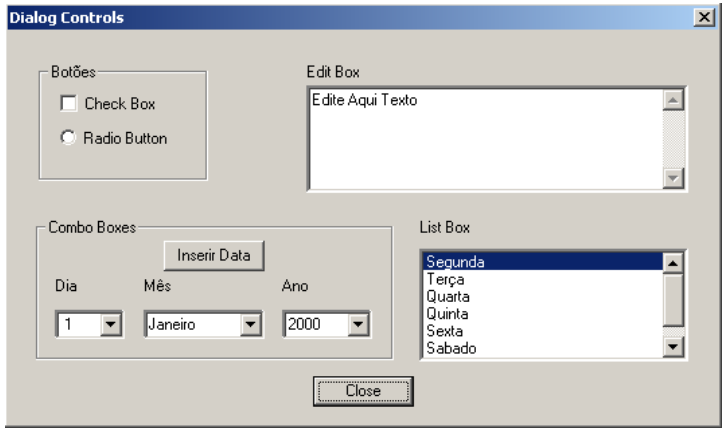


# Child Window controls

- Buttons
- Check boxes
- Edit boxes
- List boxes
- Combo boxes
- Text strings
- Scroll bars



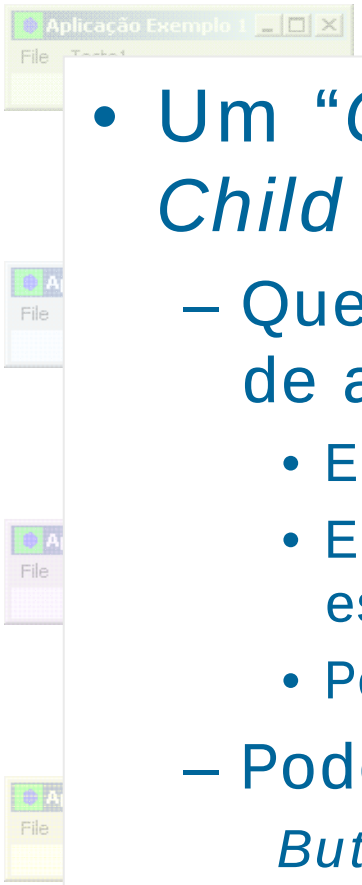



# *Child Window*

## Características de uma *child window*

- É definida com o estilo de WS_CHILD
- É definida com tendo uma *parent window*
- Reside na *client area* da *parent window*
- Por omissão só tem *client area* (a não ser que sejam indicadas explicitamente)
- Geralmente são utilizadas para dividir a *client area* da *parent window* em várias áreas funcionais
- WM_PAINT: O sistema envia WM_PAINT para a parent window e depois envia WM_PAINT para as child window inválidas pela Z order (start from top)
- Input (Mouse) events: o sistema verifica na parent window pela z order (start from top) qual a child window a que destina

# Child Window Controls



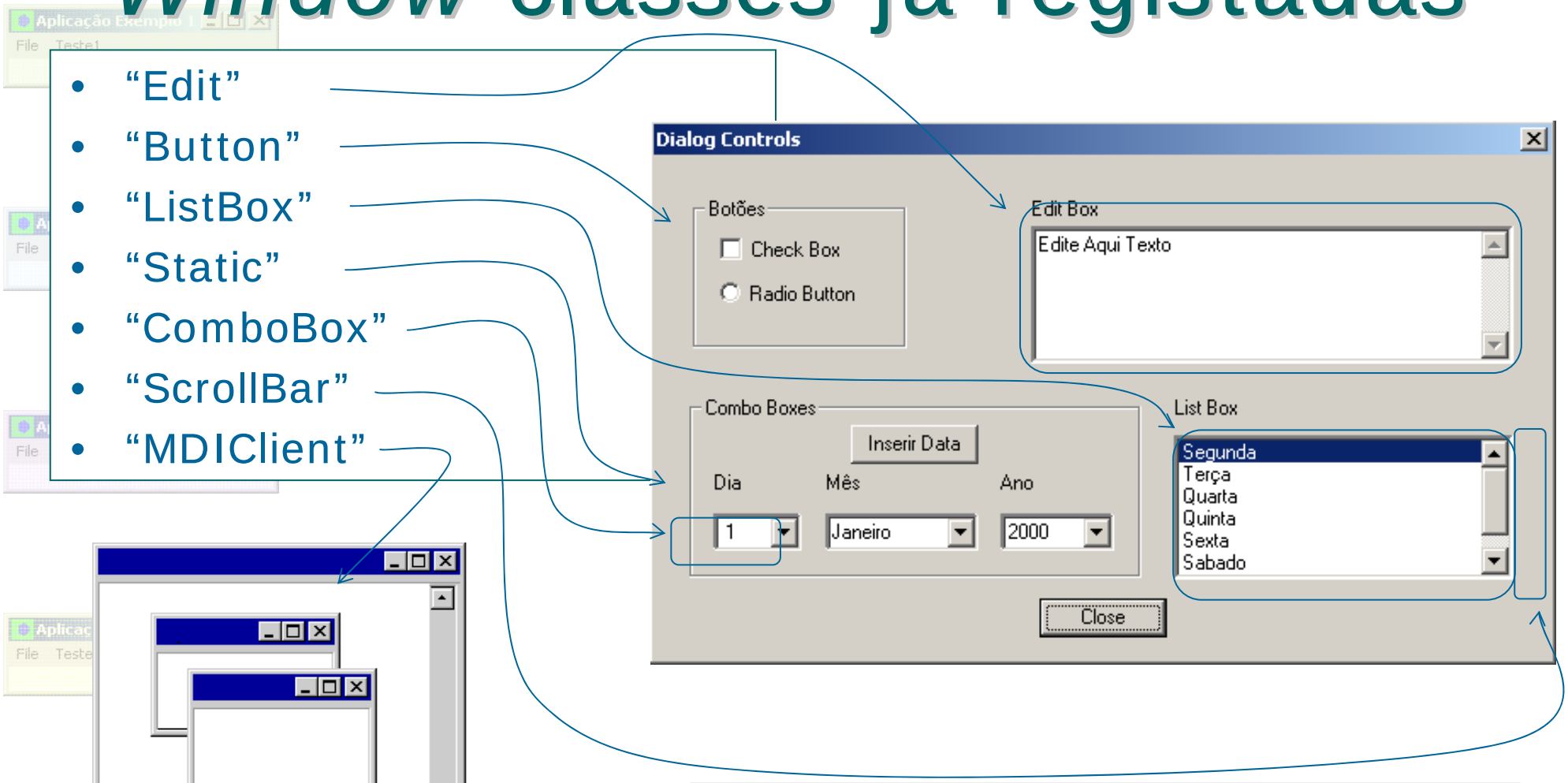
- Um "*Child Window Control*" (CWC) é uma *Child Window* que:
  - Que funciona como um dispositivo de *input/output* de alto nível para a *parent window*
    - Encapsula uma determinada funcionalidade (exº botão)
    - Envia mensagens para a *parent window* quando muda de estado
    - Permite que a *parent window* altere o seu estado
  - Pode-se criar CWC's, ou usar as pré-definidas:

    *Buttons, Check boxes, Edit boxes, List boxes, Combo boxes, Text Strings, Scroll bars*

`Consultar: MSDN /platform SDK/User Interface Servives/Controls`

# *Child Window Controls*
# *Window* classes já registadas

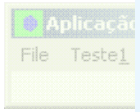
- "Edit"
- "Button"
- "ListBox"
- "Static"
- "ComboBox"
- "ScrollBar"
- "MDIClient"






Nome da window class a colocar no CreateWindowEx.
Estas janelas apenas contêm a funcionalidade base

# Exemplo de interacções com um *CWC – Button*




**Tipos de *Button***

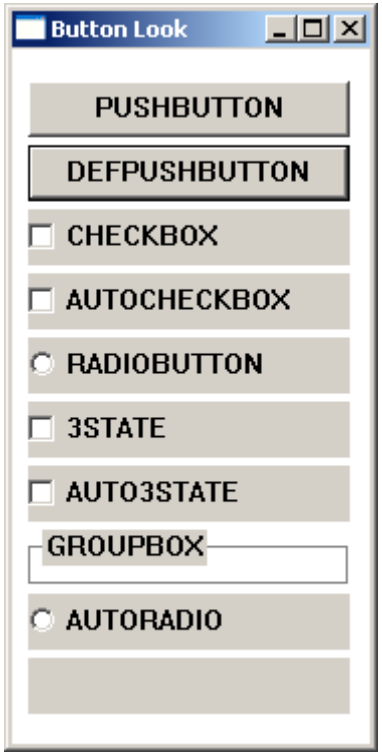
**BS_PUSHBUTTON**
**BS_DEFPUSHBUTTON**
**BS_CHECKBOX**
**BS_AUTOCHECKBOX**
**BS_RADIOBUTTON**
**BS_3STATE**
**BS_AUTO3STATE**
**BS_GROUPBOX**
**BS_AUTORADIOBUTTON**
**BS_OWNERDRAW**
**BS_LEFTTEXT**

**Mensagens enviadas pelo *button* à parent window**

**BN_CLICKED**
**BN_PAINT**
**BN_PUSHED**
**BN_UNPUSHED**
**BN_DISABLE**
**BN_DOUBLECLICKED**
**BN_SETFOCUS**
**BN_KILLFOCUS**

**Mensagens enviadas pela *parent window* ao *button***

**BM_GETCHECK**
**BM_GETIMAGE**
**BM_GETSTATE**
**BM_SETCHECK**
**BM_SETIMAGE**
**BM_SETSTATE**
**BM_SETSTYLE**
**BM_CLICK**

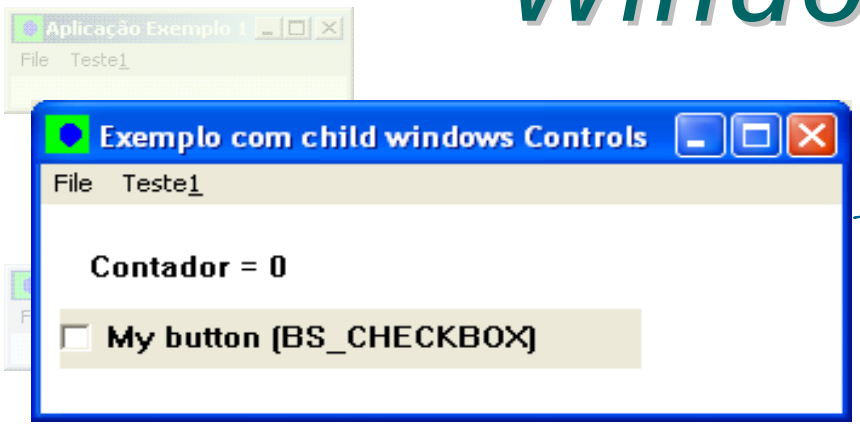




A tag da mensagem segue em HIWORD(wparam)

BS – button style
BN – button notification
BM – button message

# Funcionamento das *Child Window Controls*




**Parent window**

Em **lParam** segue o HWND da CWC, e em **LOWORD(wparam)** código de identificação do CWC **HIWORD(wparam)** segue código da notificação

**CWC envia WM_COMMAND à *parent window***

2

3

***parent window* actualiza o estado da CWC**

SendMessage(hCWC, BM_SETCHECK, BST_UNCHECKED, 0);

My button

**Child Window Control**

1

**Utilizador pressiona com o LB do rato**

# Enviar Mensagens para as janelas

```
LRESULT SendMessage(
    HWND hWnd,      // Handle da Janela a que se destina a Mensagem (WindProc)
    UINT Msg,       // Mensagem
    WPARAM wParam, // Primeiro Parâmetro da mensagem
    LPARAM lParam  // Segundo Parâmetro da mensagem
);
```

- Envia uma mensagem para uma *Window*
- Chama a *window procedure* e não retorna até esta processar a mensagem.
- Se em hWnd é passado HWND_BROADCAST, a mensagem é enviada para todas as Janelas principais do sistema (*top-level windows*)

## Exemplos de Mensagens

**WM_APP – semântica a definir pela aplicação**

**Botões**

BM_GETCHECK
BM_SETCHECK

***Edit Box***

EM_GETSEL
EM_GETLINE

***List Box***

LB_ADDSTRING
LB_DELETESTRING
LB_GETCOUNT
LB_GETCURSEL
LB_GETSEL

***Combo Box***

CB_ADDSTRING
CB_DELETESTRING
CB_GETCOUNT
CB_GETCURSEL
CB_FINDSTRING

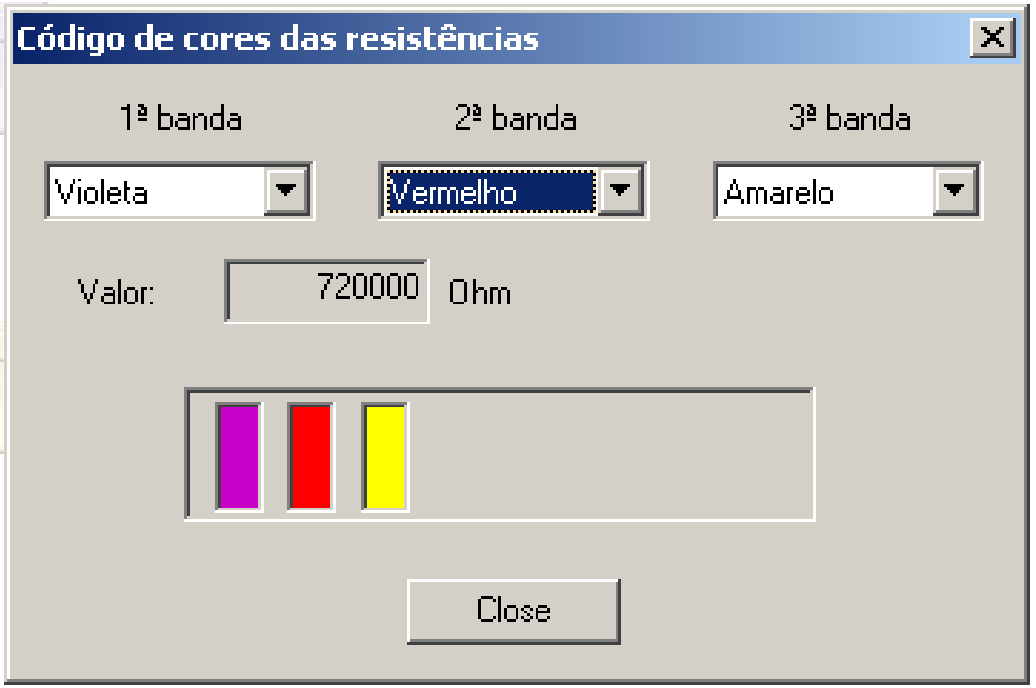
# Exemplo de uma *Child Window Control*

```
LRESULT CALLBACK WndProc(...) {
  ...
  case WM_CREATE:
        cxChar = LOWORD (GetDialogBaseUnits ()) ;
        cyChar = HIWORD (GetDialogBaseUnits ()) ;
        CreateWindow( TEXT("button"),    // registered class name
                      TEXT("My button (BS_CHECKBOX)"),      // window name
                      WS_CHILD | WS_VISIBLE | BS_CHECKBOX,  // window style
                      cxChar,                               // horizontal position of window
                      cyChar * 3,                           // vertical position of window
                      30 * cxChar,                          // window width
                      7 * cyChar / 4,                       // window height
                      hWnd,                                 // handle to parent or owner window
                      (HMENU) 1,                            // child identifier
                      ((LPCREATESTRUCT) lParam)->hInstance, // handle to application instance
                       NULL                                  // window-creation data
                     );
  case WM_COMMAND :
    switch ( LOWORD( wParam ) ) {
      case 1: // Id do child window
        if ( HIWORD(wParam) == BN_CLICKED) {
           buttonChecked = !buttonChecked;
           if ( buttonChecked ) SendMessage((HWND)lParam, BM_SETCHECK, BST_CHECKED, 0);
           else SendMessage((HWND)lParam, BM_SETCHECK, BST_UNCHECKED, 0);
        }
        break;
      ...
```

# Tipos de Janelas de diálogo (Dialog Boxes - DB)

- Modal DB
- Modeless DB
- System Modal



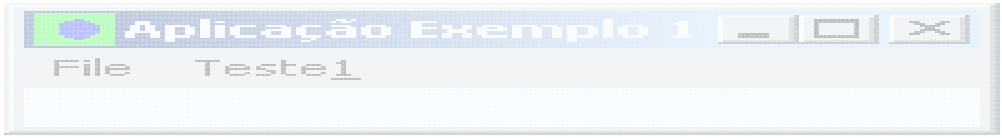
# Dialog Boxes

- Uma *dialog box* é uma janela (*window)*
- Geralmente são *popup windows* com *Child Window Controls*
  - Com processamento de *input* incorporado dos CWCs, com sequência de percurso por Tabs, com controle do focus, ...
  - Definidos numa "dialog box template" no “program's resource script file”
- Têm uma *dialog box procedure*
  - Iniciação das CWC’s
  - Processamento de mensagens originadas nos CWCs
  - Terminação da *dialog box*
  - Não têm processamento de WM_PAINT, mouse e keyboard

# Dialog Boxes

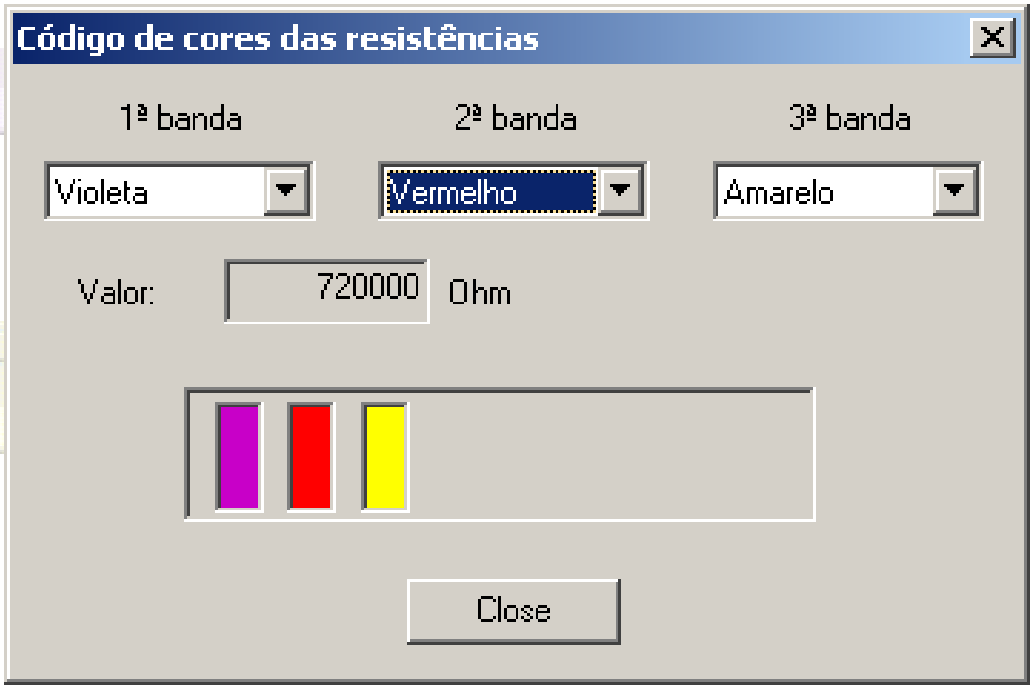

- Tipos de *Dialog boxes*:
  - ***Modal*** – o utilizador tem de encerrar a *dialog box*, antes de poder seleccionar outra janela da aplicação
  - ***Modeless*** – *Dialog Box* sem Parent Window, permite activar outras janelas da aplicação
  - ***System Modal*** – A *Dialog Box* permanece sempre como a janela principal (não permite mudar sequer para outra janela de outra aplicação)

# Tipos de Janelas de diálogo (Dialog Boxes - DB)

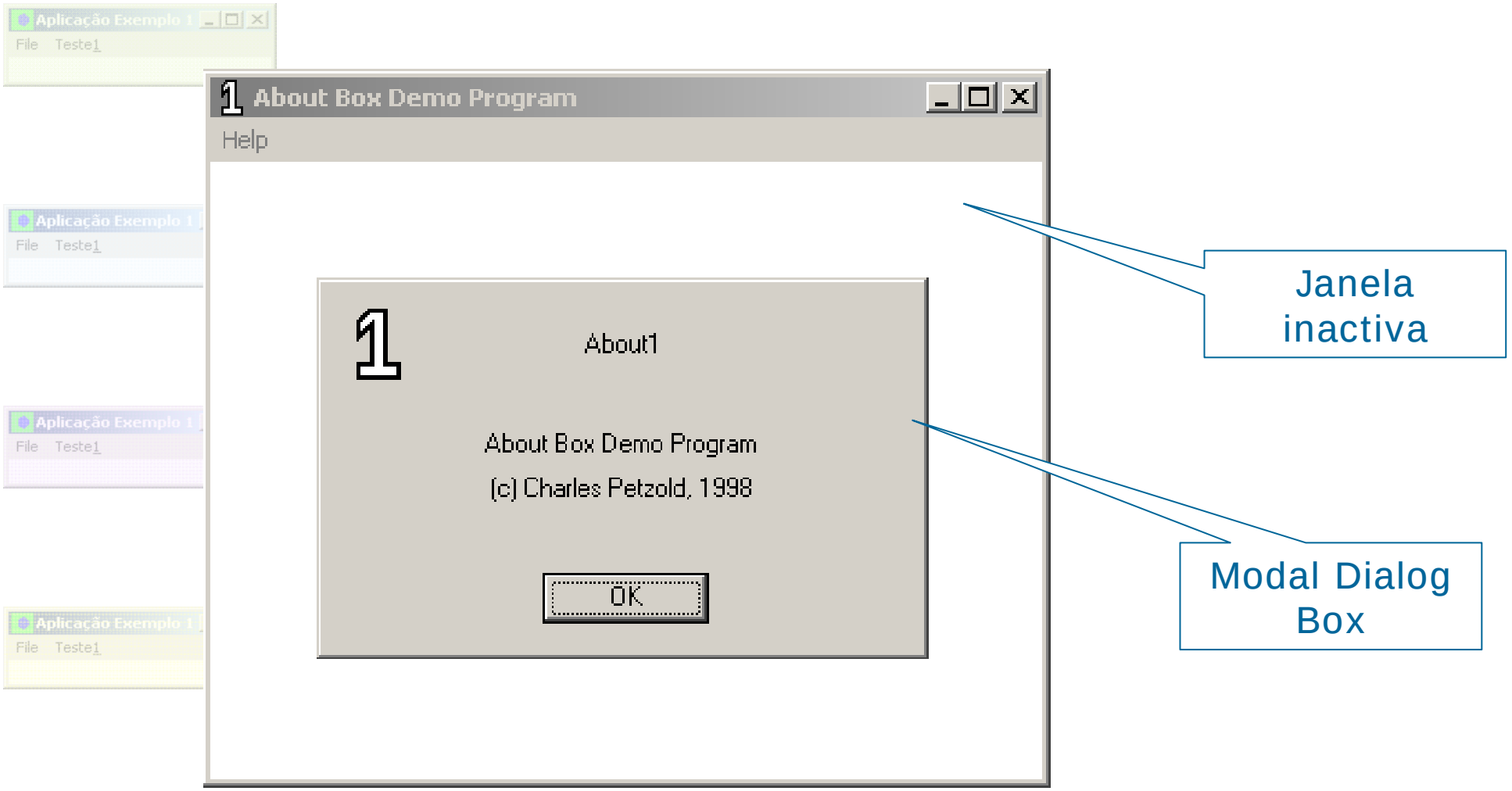



- Modal DB





# Modal Dialog Box



# Menu e DB *template* (VS - edição de *resources*)


```
ABOUT1 MENU
BEGIN
    POPUP "&Help"
    BEGIN
        MENUITEM "&About About1...", IDM_APP_ABOUT
    END
END

ABOUT1  ICON "About1.ico"
```

**Menu e Icon**

Nome da dialog box

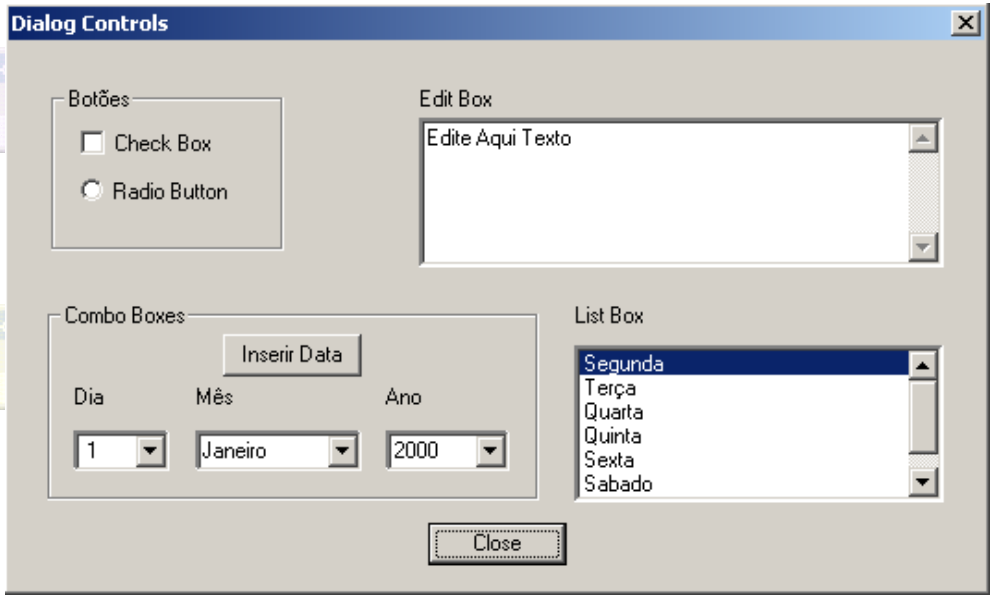
```
ABOUTBOX DIALOGEX 32, 32, 180, 102
STYLE DS_SETFONT | DS_MODALFRAME | WS_POPUP
FONT 8, "MS Sans Serif", 0, 0, 0x0
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,66,80,50,14
    ICON            "ABOUT1",IDC_STATIC,6,7,21,20
    CTEXT           "About1",IDC_STATIC,40,12,100,8
    CTEXT           "About Box Demo Program",IDC_STATIC,6,40,167,8
    CTEXT           "(c) Charles Petzold, 1998",IDC_STATIC,6,52,167,8
END
```

***Dialog box template***

# DB Resource Controls

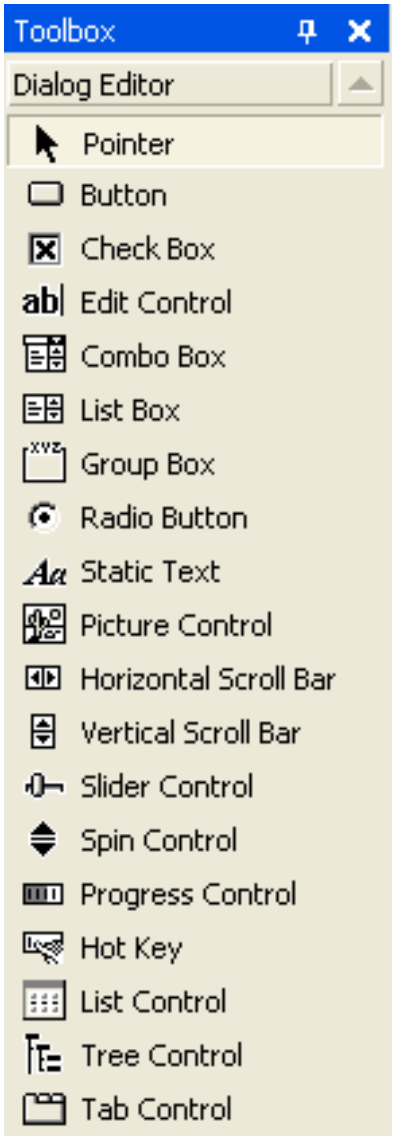
**Controls**: os componentes de uma Janela de diálogo

- Exemplos: button, edit, combo box, list box, group box, static text
- São objectos gráficos geralmente compostos por janelas pré-definidas e com uma funcionalidade de alto nível

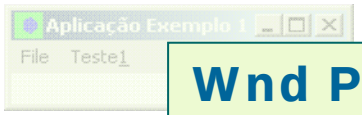



Controls disponíveis no VS

Controls inseridos numa janela de diálogo




# Criar e *Dialog Box Procedure*



**Wnd Procedure "Main window"**

```
case WM_COMMAND :
   switch (LOWORD (wParam)) {
      case IDM_APP_ABOUT :
         DialogBox (hInstance, TEXT ("AboutBox"), hwnd, AboutDlgProc) ;
         break ;
```

Nome da *dialog box*

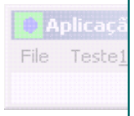
**Dialog box procedure**

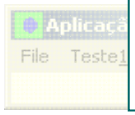
```
BOOL CALLBACK AboutDlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM
                                                                    lParam) {

     switch (message) {
       case WM_INITDIALOG: return TRUE ;
       case WM_COMMAND:
          switch (LOWORD (wParam)) {
            case IDOK:
            case IDCANCEL: EndDialog(hDlg, 0); return TRUE ;
          }
          break ;
     }
     return FALSE ;
}
```

# Criar uma *Dialog Box*

**DialogBox**

```
int DialogBox(          // macro que cria uma Dialog Box a partir de um resource template
  HINSTANCE hInstance, // Handle do programa que contém o resource template
  LPCTSTR lpTemplate,  // String ou ID do template. MAKEINTRESOURCE do ID do resource
  HWND hWndParent,     // Handle da Janela Parent.
                       // NULL se a Dialog for a Janela Principal, Dialog Box MODELESS
  DLGPROC lpDialogFunc // dialog box procedure
 );
```





- DialogBox não retorna o controlo até à Dialog Procedure (lpDialogFunc) invocar EndDialog
- Comportamento:
  - Chama CreateWindowEx(..) para criar uma *Window* do tipo *Dialog Box*.
  - Envia a mensagem WM_INITDIALOG para a função *Dialog Procedure*;
  - Desactiva a janela *parent* (MODAL *Dialog Box*) e inicia o *Message Loop*;
  - Quando a *Dialog Procedure* chama EndDialog, a Dialog Box é destruída,
  - termina o *Message Loop* e é activada a janela *parent* (se a *Dialog Box* é MODAL).
  - Retorna o valor nResult

```
Retorna:
   nResult, retornado pela função EndDialog() quando termina a Dialog Box
   0,  erro o parâmetro hWndParent é inválido
   -1, outro erro (chamar GetLastError)
```

# Dialog Box Procedure

## Dialog Box Procedure

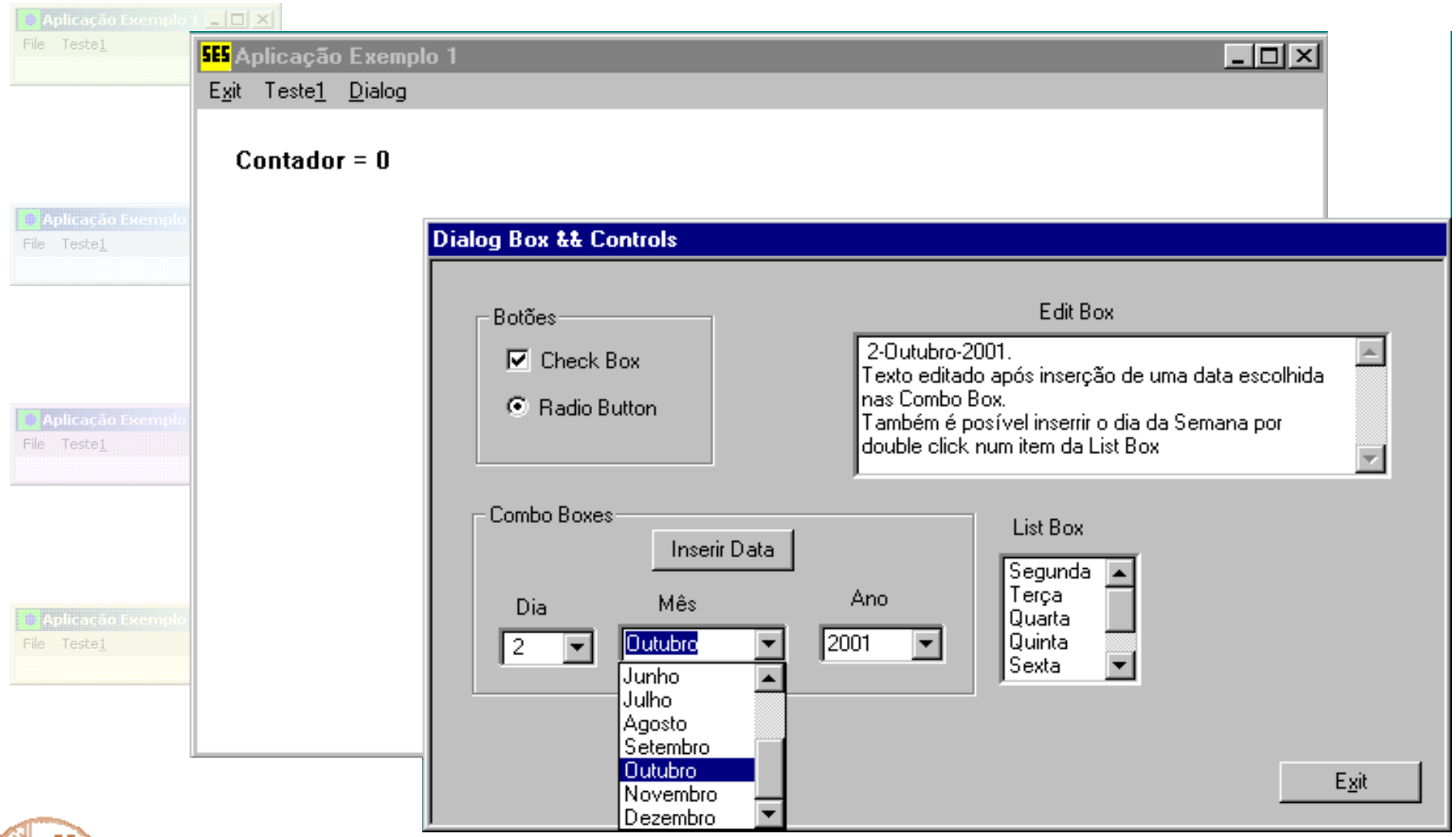
```
INT_PTR CALLBACK DialogProc (
  HWND hwndDlg,  // handle to dialog box
  UINT uMsg,     // mensagem
  WPARAM wParam, // Primeiro Parâmetro da mensagem
  LPARAM lParam  // Segundo Parâmetro da mensagem
);
```

**Deverá retornar :**
**TRUE** – mensagem processada
**FALSE** – mensagem não processada, indica para ser executada a acção por omissão associada à mensagem

## Terminar a *Dialog Box*

```
BOOL EndDialog(
  HWND hDlg,      // Handle da Dialog Box a Terminar
  INT_PTR nResult // valor a retornar à função que criou a Dialog Box
);
```

# Exemplo de uma *Modal Dialog Box*



# Criar a *Dialog Box* Windows Procedure


```
LRESULT CALLBACK WndProc ( . . .) {
  switch (uMsg) {
    case WM_COMMAND :
      switch ( LOWORD( wParam ) ) {
        …
        case ID_DIALOG :  // opção do menu que cria a dialog box
          if (!DialogBox(hInst, MAKEINTRESOURCE(IDD_DIALOG), hWnd, (DLGPROC)Dialog) )
            MessageBox(hWnd,"Erro ao criar a Dialog Box","Debug",MB_OK);
          break;
        …
      } // switch ( LOWORD( wParam ) )
      break;
      …
  } // switch uMsg

return msg.wParam;
}
```

*GetDlgItem*

```
HWND GetDlgItem( HWND hDlg, int nIDDlgItem ); // devolve o handle do item
```

# Dialog Procedure
# Interacção com os CWCs

```
LRESULT CALLBACK DialogControls(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
  // Data actual
  TCHAR diaActual[16], mesActual[16], anoActual[16], Buffer[64];int indice;

  switch (message) {
    case WM_INITDIALOG :
      MessageBox(hDlg, TEXT("WM_INITDIALOG"), TEXT("DEBUG"), MB_OK);
      InitDialog(hDlg, wParam, lParam);  // iniciações dos Child Window Controls
      return TRUE;

    case WM_COMMAND :
      switch ( LOWORD(wParam) ) {
        case IDOK: EndDialog(hDlg, TRUE); return TRUE;  // Close button message

        case IDC_INSERTDATA: // Insert data button message
          SendMessage(GetDlgItem(hDlg, IDC_CBDIA), WM_GETTEXT, 12, (LPARAM)diaActual);
          SendMessage(GetDlgItem(hDlg, IDC_CBMES), WM_GETTEXT, 12, (LPARAM)mesActual);
          SendMessage(GetDlgItem(hDlg, IDC_CBANO), WM_GETTEXT, 12, (LPARAM)anoActual);
          _stprintf( Buffer, TEXT("%s-%s-%s"), diaActual, mesActual, anoActual);
          SendMessage(GetDlgItem(hDlg, IDC_EDIT1), WM_SETTEXT, 0, (LPARAM)Buffer);
        return TRUE;
. . .
```

# Dialog Procedure (cont.)

```
. . .
        case IDC_LBDIASEM:  // list box message
          if ( HIWORD(wParam) == LBN_DBLCLK) {
            indice = SendMessage(GetDlgItem(hDlg,IDC_LBDIASEM), LB_GETCURSEL, 0, 0);
            SendMessage(GetDlgItem(hDlg,IDC_LBDIASEM), LB_GETTEXT, indice,
                                                          (LPARAM)Buffer);
            SetDlgItemText(hDlg,IDC_EDIT1, (LPCTSTR)Buffer);
          };
          return TRUE;
        case IDC_RADIO1:  // radio button message
          if ( HIWORD(wParam) == BN_CLICKED) {
            bRadio = bRadio == BST_CHECKED ? BST_UNCHECKED : BST_CHECKED;
            SendMessage(GetDlgItem(hDlg, IDC_RADIO1), BM_SETCHECK, (LPARAM) bRadio, 0);
            };
          return TRUE;
        case IDC_CHECK1: // Check button message
          if ( HIWORD(wParam) == BN_CLICKED) {
            bCheck = bCheck == BST_CHECKED ? BST_UNCHECKED : BST_CHECKED;
           SendMessage(GetDlgItem(hDlg, IDC_CHECK1), BM_SETCHECK, (LPARAM) bCheck, 0);
            };
          return TRUE;
      } // switch ( LOWORD(wParam) )
      break;
  }; // switch (message)
  return FALSE; }
```

# Init Dialog

```
static BOOL bCheck, bRadio;

void InitDialog(HWND hDlg,WPARAM wParam,LPARAM lParam) {

    TCHAR mesesAno[12][16]={  // Meses do ano
         TEXT("Janeiro"), TEXT("Fevereiro"),TEXT("Março"),
         TEXT("Abril"),   TEXT("Maio"),     TEXT("Junho"),
         TEXT("Julho"),   TEXT("Agosto"),   TEXT("Setembro"),
         TEXT("Outubro"), TEXT("Novembro"), TEXT("Dezembro")  };

    TCHAR diasSemana[7][16] = {  // Dias da semana
         TEXT("Segunda"), TEXT("Terça"),   TEXT("Quarta"),  TEXT("Quinta"),
         TEXT("Sexta"),   TEXT("Sabado"),  TEXT("Domingo")  };

  TCHAR dia[8], ano[8];

  SendMessage(GetDlgItem(hDlg,IDC_EDIT1), WM_SETTEXT, 0 ,
                                              (LPARAM)TEXT("Edite Aqui Texto"));

  bCheck = BST_CHECKED;
  SendMessage(GetDlgItem(hDlg, IDC_RADIO1), BM_SETCHECK, (LPARAM)bRadio, 0);

  bRadio = BST_CHECKED;
  SendMessage(GetDlgItem(hDlg, IDC_CHECK1), BM_SETCHECK, (LPARAM)bCheck, 0);
 . . .
```

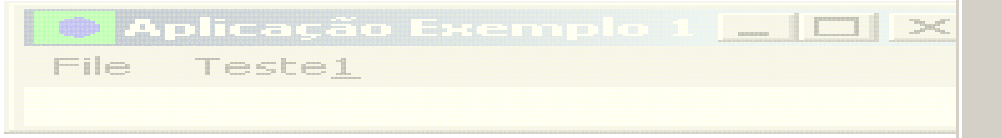
# *Init Dialog* (cont.)

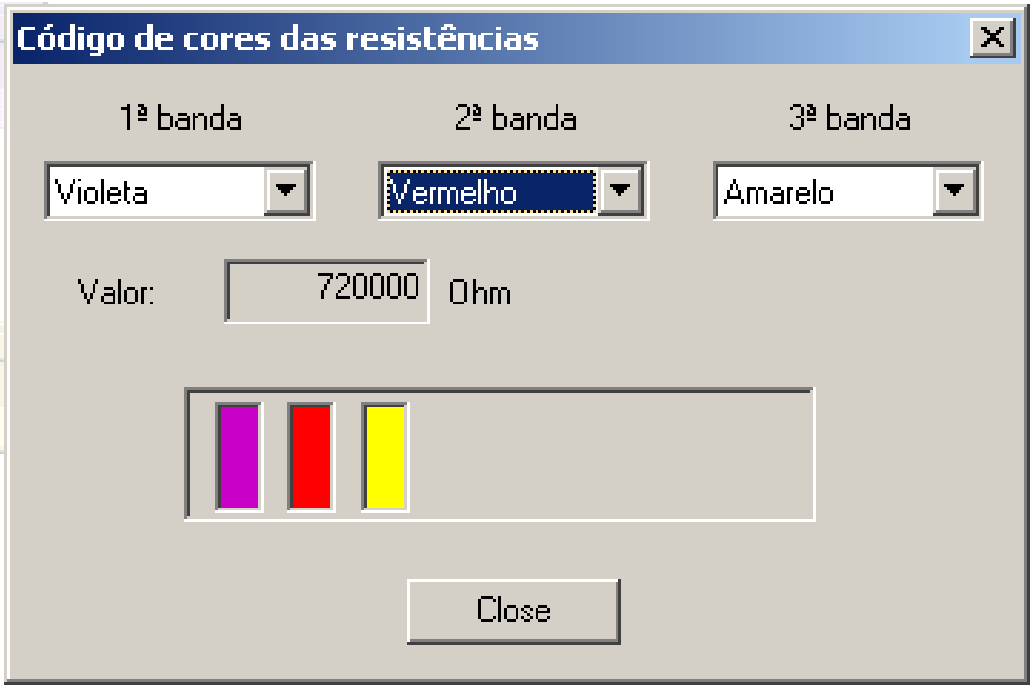
```
  . . .
  for (int indiceDia=1; indiceDia<= 31; indiceDia++) {
    _stprintf( dia, TEXT("%2d"), indiceDia);
    SendMessage(GetDlgItem(hDlg, IDC_CBDIA), CB_ADDSTRING, 0, (LONG) (LPSTR)dia);
  }
  SendMessage(GetDlgItem(hDlg, IDC_CBDIA), CB_SETCURSEL, 0, 0);

  for (int indiceMes=0; indiceMes<12; indiceMes++)
    SendMessage( GetDlgItem(hDlg, IDC_CBMES), CB_ADDSTRING, 0,
                                              (LONG)(LPSTR)mesesAno[indiceMes]);
  SendMessage(GetDlgItem(hDlg, IDC_CBMES), CB_SETCURSEL, 0, 0);

  for (int indiceAno=0; indiceAno<6; indiceAno++) {
    _stprintf( ano, TEXT("%4d"), 2000+indiceAno);
    SendMessage(GetDlgItem(hDlg,IDC_CBANO), CB_ADDSTRING, 0, (LONG) (LPSTR)ano);
  }
  SendMessage(GetDlgItem(hDlg, IDC_CBANO), CB_SETCURSEL, 0, 0);

  for (int indiceSem=0; indiceSem<7; indiceSem++)
    SendMessage(GetDlgItem(hDlg, IDC_LBDIASEM),LB_ADDSTRING, 0,
                                              (LONG)(LPSTR)diasSemana[indiceSem]);
  SendMessage(GetDlgItem(hDlg, IDC_LBDIASEM),  LB_SETCURSEL , 0, 0);
}
```

# Tipos de Janelas de diálogo (Dialog Boxes - DB)

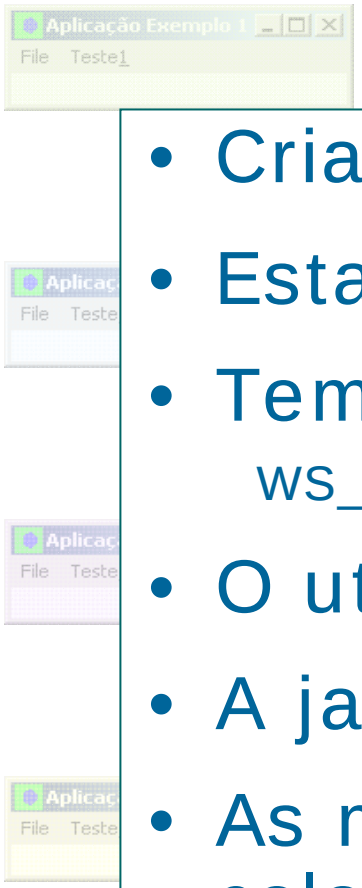


- Modeless DB



# *Modeless Dialog Boxes*




- Criadas pela função: `CreateDialog`
- Esta função retorna imediatamente
- Template de Style para criar uma MLessDB

  WS_POPUP | WS_CAPTION | WS_SYSMENU | WS_VISIBLE
- O utilizador pode mover a janela
- A janela pode ser criada sem estar visível
- As mensagens de uma MLessDB são colocadas na *message queue* da *thread*
- Usar DestroyWindow para eliminar a MLessDB

# Processamento de mensagens de uma Modeless DB


**Wnd Procedure**

```
While( GetMessage(&msg, NULL, 0, 0) > 0 )
{
  if ( hDlgModeless == 0 || !IsDialogMessage ( hDlgModeless, &msg))
  {
     TranslateMessage(&msg);
     DispatchMessage(&msg);
  }
}
```

HWND hDlgModeless – variável static que contém o handle da dialog box

A função **IsDialogMessage** verifica se a mensagem é para a window com o handle recebido, e em caso afirmativo entrega-o à wndProc dessa window.





# *Message Crackers*
# *e*
# Controls

# *Message Crackers* e *Controls*

- As *Message Crackers* não se limitam a fornecer macros para simplificação do tratamento de mensagens
- Existem *Message Crackers* específicas para trabalhar com os *controls*
- Estas macros permitem simplificar a manipulação dos *controls*

Por exemplo:

- Para modificar o *icon* de um *control* do tipo `Static` é necessário enviar a mensagem STM_SETICON ao *control*:

```
SNDMSG((hwndCtl), STM_SETICON, (WPARAM)(HICON)(hIcon), 0L))
```

- Utilizando *Message Crackers* será assim:

```
Static_SetIcon(hwndCtl, hIcon)
```

# Message Crackers (Control Message APIs)

São uma série de macros já definidas (em **windowsx.h**) para acesso a: Static, Button, Edit, ListBox, ComboBox, ScrollBar **controls**

**Alguns exemplos:**

```
#define Static_GetText(hwndCtl, lpch, cchMax)   GetWindowText((hwndCtl), (lpch), (cchMax))
#define Static_SetText(hwndCtl, lpsz)           SetWindowText((hwndCtl), (lpsz))
```

```
#define Button_GetCheck(hwndCtl)       ((int)(DWORD)SNDMSG((hwndCtl), BM_GETCHECK, 0L, 0L))
#define Button_SetCheck(hwndCtl, check)   ((void)SNDMSG((hwndCtl), BM_SETCHECK,  . . .
```

```
#define Edit_GetText(hwndCtl, lpch, cchMax)     GetWindowText((hwndCtl), (lpch), (cchMax))
#define Edit_SetText(hwndCtl, lpsz)             SetWindowText((hwndCtl), (lpsz))
```

```
#define ListBox_AddString(hwndCtl,lpsz) ((int)(DWORD)SNDMSG((hwndCtl),LB_ADDSTRING,0L,. . .
#define ListBox_GetText(hwndCtl, index, lpszBuffer)  ((int)(DWORD)SNDMSG((hwndCtl), . . .
```

```
#define ComboBox_AddString(hwndCtl, lpsz) ((int)(DWORD)SNDMSG((hwndCtl), CB_ADDSTRING, . . .
#define ComboBox_GetCurSel(hwndCtl)    ((int)(DWORD)SNDMSG((hwndCtl), CB_GETCURSEL, 0L, 0L))
#define ComboBox_GetItemData(hwndCtl, index) ((LRESULT)(ULONG_PTR)SNDMSG((hwndCtl), . . .
```

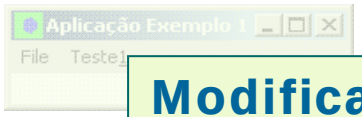
# Funções de manipulação de Controls dentro de uma janela de Diálogo

Alguns controls têm funções que permitem a sua manipulação, quando inseridos dentro de uma Dialog Box (ver MSDN).

**Alguns exemplos:**

| | |
|---|---|
| **CheckDlgButton**<br>**CheckRadioButton**<br>**IsDlgButtonChecked** | Button controls |
| DlgDirListComboBox<br>DlgDirSelectComboBoxEx<br>GetComboBoxInfo | Combo Box Controls |
| EditWordBreakProc | Edit Controls |
| DlgDirList<br>DlgDirSelectEx<br>GetListBoxInfo | List Box Controls |

# Exemplos de Funções de Acesso aos Controls




## Modificar o Estado de um Botão (Radio ou Check)

```
BOOL CheckDlgButton(
  HWND hDlg,      // Handle da Dialog box que contém o botão
  int nIDButton, // ID Resource do Botão para modificar
  UINT uCheck ); // estado do botão: BST_CHECKED, BST_UNCHECKED
```


## Obter o Estado de um Botão (Radio ou Check)


```
UINT IsDlgButtonChecked(
  HWND hDlg,      // Handle da dialog box que contém o botão
  int nIDButton   // ID Resource do Botão a testar se está Checked
);                // Devolve o estado do botão: BST_CHECKED, BST_UNCHECKED
```

Resultam no envio de mensagens de BM_SETCHECK e BM_GETCHECK


## Gets or sets the title or text of a control in a dialog box

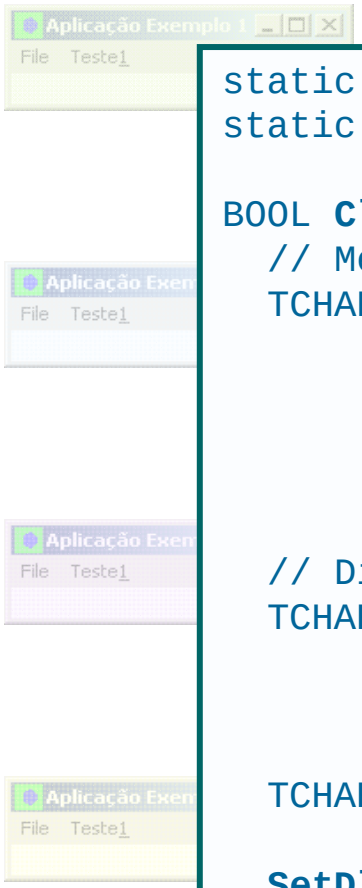
```
BOOL SetDlgItemText( HWND hDlg, int nIDDlgItem, LPCTSTR lpString );
UINT GetDlgItemText( HWND hDlg, int nIDDlgItem, LPTSTR lpString, int nMaxCount);
```

Resultam no envio de mensagens de WM_GETTEXT e WM_SETTEXT

# Exemplo anterior com *Message Crackers*


```
LRESULT CALLBACK DialogControls(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{

  switch (message) {
     HANDLE_MSG(hDlg, WM_INITDIALOG, Cls_OnInitDlgDemo);
     HANDLE_MSG(hDlg, WM_COMMAND,    Cls_OnCommandDemo);
     HANDLE_MSG(hDlg, WM_CLOSE,      Cls_OnCloseDemo);
  }

  return FALSE;
}
```


# Exemplo com *Message Crackers* (cont.)

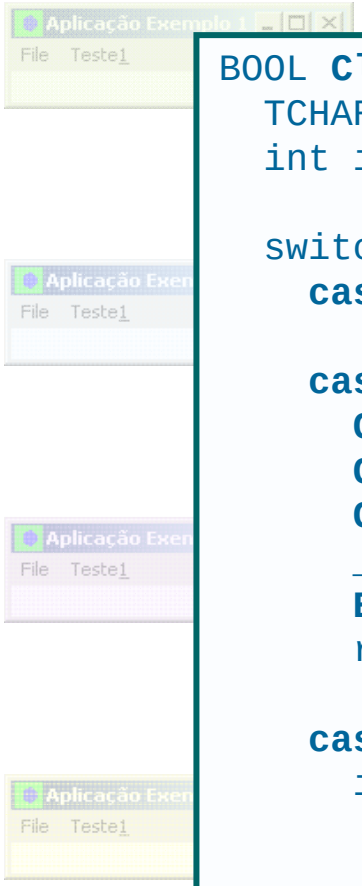



```
static BOOL bCheck=FALSE;
static BOOL bRadio=TRUE;

BOOL Cls_OnInitDlgDemo(HWND hDlg, HWND hwndFocus, LPARAM lParam) {
  // Meses do ano
  TCHAR mesesAno[12][16]={
        TEXT("Janeiro"), TEXT("Fevereiro"),       TEXT("Março"),
        TEXT("Abril"),   TEXT("Maio"),    TEXT("Junho"),
        TEXT("Julho"),   TEXT("Agosto"),  TEXT("Setembro"),
        TEXT("Outubro"), TEXT("Novembro"),TEXT("Dezembro")  };

  // Dias da semana
  TCHAR diasSemana[7][16] = {
        TEXT("Segunda"), TEXT("Terça"),   TEXT("Quarta"),  TEXT("Quinta"),
        TEXT("Sexta"),   TEXT("Sabado"),  TEXT("Domingo")  };

  TCHAR dia[8], ano[8];

  SetDlgItemText(hDlg, IDC_EDIT1, TEXT("Edite Aqui Texto") );

  Button_SetCheck( GetDlgItem(hDlg, IDC_CHECK1), BST_UNCHECKED);
  Button_SetCheck( GetDlgItem(hDlg, IDC_RADIO1), BST_UNCHECKED);
  . . .
```

# Exemplo com *Message Crackers* (cont.)

```
  . . .
  for (int indiceDia=1; indiceDia<= 31; indiceDia++) {
    _stprintf( dia, TEXT("%2d"), indiceDia);
    ComboBox_AddString( GetDlgItem(hDlg, IDC_CBDIA), (LPSTR)dia );
  }
  ComboBox_SetCurSel( GetDlgItem(hDlg, IDC_CBDIA), 0);

  for (int indiceMes=0; indiceMes<12; indiceMes++)
    ComboBox_AddString(GetDlgItem(hDlg,IDC_CBMES),(LPSTR)(LPSTR)mesesAno[indiceMes] );
  ComboBox_SetCurSel( GetDlgItem(hDlg, IDC_CBMES), 0);

  for (int indiceAno=0; indiceAno<6; indiceAno++) {
    _stprintf( ano, TEXT("%4d"), 2000+indiceAno);
    ComboBox_AddString( GetDlgItem(hDlg, IDC_CBANO), (LPSTR)ano );
  }
  ComboBox_SetCurSel( GetDlgItem(hDlg, IDC_CBANO), 0);

  for (int indiceSemana=0; indiceSemana<7; indiceSemana++)
    ListBox_AddString(GetDlgItem(hDlg,IDC_LBDIASEM),(LPSTR)diasSemana[indiceSemana] );
  ListBox_SetCurSel( GetDlgItem(hDlg, IDC_LBDIASEM), 0);

  return TRUE;
}
```

# Exemplo com *Message Crackers* (cont.)



```
BOOL Cls_OnCommandDemo(HWND hDlg, int id, HWND hwndCtl, UINT codeNotify) {
  TCHAR diaActual[16], mesActual[16], anoActual[16], Buffer[64];
  int indice=0;

  switch ( id ) {
    case IDOK: EndDialog(hDlg, TRUE); return TRUE;

    case IDC_INSERTDATA:
      ComboBox_GetText( GetDlgItem(hDlg, IDC_CBDIA), diaActual, 12);
      ComboBox_GetText( GetDlgItem(hDlg, IDC_CBMES), mesActual, 12);
      ComboBox_GetText( GetDlgItem(hDlg, IDC_CBANO), anoActual, 12);
      _stprintf( Buffer, TEXT("%s-%s-%s"), diaActual, mesActual, anoActual);
      Edit_SetText( GetDlgItem(hDlg, IDC_EDIT1), (LPCTSTR)Buffer);
      return TRUE;

    case IDC_LBDIASEM:
      if ( codeNotify==LBN_DBLCLK ) {
        indice = ListBox_GetCurSel( hwndCtl );
        ListBox_GetText( hwndCtl, indice, (LPCTSTR)Buffer);
        Edit_SetText( GetDlgItem(hDlg,IDC_EDIT1), (LPCTSTR)Buffer);
      }
      return TRUE;
   . . .
```

# Exemplo com *Message Crackers* (cont.)


```
  . . .
    case IDC_RADIO1:
      if ( codeNotify==BN_DBLCLK ) {
        CheckDlgButton(hDlg, IDC_RADIO1, bRadio ? BST_UNCHECKED : BST_CHECKED);
        bRadio = IsDlgButtonChecked(hDlg, IDC_RADIO1) == BST_CHECKED;
      }
      return TRUE;

    case IDC_CHECK1:
      CheckDlgButton(hDlg,IDC_CHECK1, bCheck ? BST_UNCHECKED : BST_CHECKED);
      bCheck=IsDlgButtonChecked(hDlg,IDC_CHECK1) == BST_CHECKED;
      return TRUE;
  } // switch ( LOWORD(wParam) )

  return FALSE;
}
```

```
void Cls_OnCloseDemo(HWND hDlg) {
  EndDialog(hDlg, TRUE);
}
```