```
function ChooseColor(var CC: TChooseColor): Bool;

type
  PFindReplace = ^TFindReplace;
  TFindReplace = record
    lStructSize: Longint;                                        { size of this struct $20 }
    hWndOwner: HWnd;                                             { handle to owner's window }
    hInstance: THandle;                                          { instance handle of.EXE that
                                                                   contains cust. dlg. template }
    Flags: Longint;                                              { one or more of the fr_?? }
    lpstrFindWhat: PChar;                                        { ptr. to search string    }
    lpstrReplaceWith: PChar;                                     { ptr. to replace string   }
    wFindWhatLen: Word;                                          { size of find buffer      }
    wReplaceWithLen: Word;                                       { size of replace buffer   }
    lCustData: Longint;                                          { data passed to hook fn.  }
    lpfnHook: function (Wnd: HWnd; Msg, wParam: Word; lParam: Longint): Word;
                                                                 { ptr. to hook fn. or nil }
    lpTemplateName: PChar;                                       { custom template name     }
  end;

function ReplaceText(var FindReplace: TFindReplace): HWnd;

type
  PChooseFont = ^TChooseFont;
  TChooseFont = record
    lStructSize: Longint;       { }
    hWndOwner: HWnd;                                           { caller's window handle   }
    hDC: HDC;                                                  { printer DC/IC or nil     }
    lpLogFont: PLogFont;                                       { ptr. to a LOGFONT struct }
    iPointSize: Integer;                                       { 10 * size in points of selected font }
    Flags: Longint;                                            { enum. type flags         }
    rgbColors: Longint;                                        { returned text color      }
    lCustData: Longint;                                        { data passed to hook fn.  }
    lpfnHook: function (Wnd: HWnd; Msg, wParam: Word; lParam: Longint): Word;
                                                               { ptr. to hook function    }
    lpTemplateName: PChar;                                     { custom template name     }
    hInstance: THandle;                                        { instance handle of.EXE that contains cust.
                                                                 dlg. template }
    lpszStyle: PChar;                                          { return the style field here must be lf_FaceSize
                                                                 or bigger }
    nFontType: Word;                                           { same value reported to the EnumFonts call back
                                                                 with the extra fonttype_bits added }
    nSizeMin: Integer;                                         { minimum pt size allowed & }
    nSizeMax: Integer;                                         { max pt size allowed if cf_LimitSize is used      }
  end;

{ these are extra nFontType bits that are added to what is returned to the EnumFonts callback routine }

  Italic_FontType = $0200;
  Regular_FontType = $0400;

  wm_ChooseFont_GetLogfont = wm_User + 1;


{ strings used to obtain unique window message for communication between dialog and caller }

  SetRGBString = 'commdlg_SetRGBColor';
  FindMsgString = 'commdlg_FindReplace';
  HelpMsgString = 'commdlg_help';

{ HIWORD values for lParam of commdlg_LBSelChangeNotify message }

  cferr_NoFonts          = $2001;
  cferr_MaxLessThanMin   = $2002;

  fnErr_FilenameCodes    = $3000;
  fnErr_SubclassFailure  = $3001;
  fnErr_InvalidFilename  = $3002;
  fnErr_BufferTooSmall   = $3003;

  frErr_FindReplaceCodes = $4000;
  frErr_BufferLengthZero = $4001;

  ccErr_ChooseColorCodes = $5000;




{
```



```
}
```