Petites aides pour Visual C++

Vous trouverez dans ce modeste livret de nombreuses aides glanées ici et là sur le Web et dans les livres. J'espère qu'elles vous seront utiles.

Des erreurs peuvent s'être glissées à l'intérieur (errare humanum est), dans ce cas, merci de me le faire savoir à l'adresse suivante eric.berthomier@free.fr, je procéderai aux corrections le plus rapidement possible.

Bonne programmation à tous ...

M. BERTHOMIER Eric
Version du 4 Octobre 2000

Tables des petites aides pour Visual C++

Environnement de Visual C++

1. Fichier d'indexation de recherche

La recherche d'une fonction par l'intermédiaire du clic droit de la souris peut, à partir d'un certain moment devenir fausse et de ce fait inefficace. Pour résoudre ce problème, il faut supprimer le fichier \Debug***Nom_du_projet.bsc** {browse source}.

Lors du prochain clic droit, Visual C++ ne trouvera pas son fichier de browse et reconstruira sa table d'indexation.

2. Fichier de class Wizard

De la même façon, le classe Wizard peut devenir plus ou moins scabreux. Il faut alors détruire le fichier ***Nom_du_projet.clw*** {class wizard}.

Ajustement de la taille d'un élément d'une Dialog Box

Mots clés : ***MoveWindow, ScreenToClient***

1. Description générale

Soit Zone_Affichage les nouvelles tailles que l'on désire appliquer à notre élément.

Zone_Affichage.cx = nouvelle largeur.
Zone_Affichage.cy = nouvelle hauteur.

Soit IDC_CADRE_IMAGE, l'élément dont l'on désire changer la taille.

2. Application

```
// Ajustement de la taille de la fenêtre d'affichage
CWnd*  pImg = GetDlgItem (IDC_CADRE_IMAGE);

// On lit les coordonnées de notre élément
CRect    rect2;
pImg->GetWindowRect (&rect2);

// Ces coordonnées sont des coordonnées Ecran, de ce fait, on les transforme en coordonnées Client
ScreenToClient (&rect2);

// On ajuste la taille de la fenêtre par la commande MoveWindow
pImg->MoveWindow (rect2.left, rect2.top, Zone_Affichage.cx+3, Zone_Affichage.cy+3);
```

Changement de la couleur de fond d'une fenêtre

Mots clés : ***WM_ERASEBKGND***

1. Mise en place

Intercepter le message WM_ERASEBKGND grâce au wizard.

2. Application

```
BOOL CMyView::OnEraseBkgnd(CDC* pDC)
{
	// Création d'un pinceau pour le background la fenêtre
	CBrush backBrush (RGB(128, 128, 128));

	// Sauvegarde de l'ancien pinceau
	CBrush* pOldBrush = pDC->SelectObject(&backBrush);

	CRect rect;

	// Effacement de la zone à nettoyer
	pDC->GetClipBox(&rect);
	// Peinture
	pDC->PatBlt(rect.left, rect.top, rect.Width(), rect.Height(),PATCOPY);

	// Remise dans l'état précédent
	pDC->SelectObject(pOldBrush);

	return TRUE;
}
```

Sortie d'une boîte d'édition par Retour-Chariot

Mots clés : ***Retour-chariot.***

1. Déclaration de la boîte d'édition

Il est nécessaire de déclarer la boîte d'édition en mode **multiligne (ES_MULTILINE)** mais sans ES_WANTRETURN. Le format multiligne permet d'intercepter le retour chariot dans les messages WM_KEYDOWN.

```
m_wndNoPage.Create    (ES_MULTILINE | ES_LEFT | WS_BORDER,
                       rect, &m_wndToolBar, ID_NOPAGE))
```

2. Traitement des messages

Afin de ne pas passer à la ligne suivante, le retour chariot va être filtré en envoyant le focus sur la fenêtre principale afin d'obtenir un KillFocus de la même façon qu'avec une tabulation.

- isalnum permet de filtrer tous les caractères alphanumériques.
- isdigit permet de filtrer tous les caractères numériques.

```
void CMyEditNoPage::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        if (nChar == 0x0D)
                AfxGetMainWnd ()->SetFocus ();
        else
                CEdit::OnKeyDown(nChar, nRepCnt, nFlags);
}

void CMyEditNoPage::OnChar(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        // Filtre pour tous les caractères alphanumériques
        if (isalnum (nChar))
        {
                // Filtre pour les caractères numériques
                if (isdigit (nChar))
                        CEdit::OnKeyDown(nChar, nRepCnt, nFlags);
        }
        else
                // Caractères non alphanumériques
                CEdit::OnKeyDown(nChar, nRepCnt, nFlags);
}
```

- ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView () permet d'obtenir un pointeur sur la fenêtre enfant active.

```
void CMyEditNoPage::OnKillFocus(CWnd* pNewWnd)
{
        CEdit::OnKillFocus(pNewWnd);

        CString  chaine;
        GetWindowText (chaine);

        ((CFrameWnd*) AfxGetMainWnd ())->GetActiveView ()->PostMessage (WM_NEW_PAGE,
         (WPARAM) atoi (chaine));
}
```

CListCtrl : Tri et Création

Mots clés : ***CListCtrl, OnColumnclick, SortItems, SetItemData, InsertItem, SetItem, InsertColumn***

1. Création

Exemple de création d'une CListCtrl avec 3 colonnes :

```
CListCtrl           m_Liste_Fichier;

m_Liste_Fichier.GetWindowRect (&rect);
LV_COLUMN    lvcolumn;

lvcolumn.mask = LVCF_FMT | LVCF_SUBITEM | LVCF_TEXT | LVCF_WIDTH;
lvcolumn.fmt = LVCFMT_CENTER;
lvcolumn.cx = (int) (rect.Width () / 3);

lvcolumn.iSubItem = 0;
lvcolumn.pszText = "Nom du fichier";
m_Liste_Fichier.InsertColumn (0,&lvcolumn);

lvcolumn.iSubItem = 1;
lvcolumn.pszText = "Auteur";
m_Liste_Fichier.InsertColumn (1,&lvcolumn);

lvcolumn.iSubItem = 2;
lvcolumn.pszText = "Taille";
m_Liste_Fichier.InsertColumn (2,&lvcolumn);
```

2. Tri lors de l'appui sur l'entête d'une colonne

Interceptez par le wizard le message LVN_COLUMNCLICK lié à votre ListCtrl.

Exemple pour une Liste de fichier :

```
void CMaClasse::OnColumnclickListeFichier(NMHDR* pNMHDR, LRESULT* pResult)
{
        NM_LISTVIEW* pNMListView = (NM_LISTVIEW*)pNMHDR;

        m_type_tri = (LPARAM) pNMListView->iSubItem;

        m_Liste_Fichier.SortItems ((PFNLVCOMPARE)MySort, m_type_tri);
```

*Paramètre **lParamSor**t passé à la fonction de tri*

```
        *pResult = 0;
}
```

3. Insertion d'éléments dans la liste et prémisses à l'utilisation du tri

Prenons l'exemple d'une structure fichier

```
char    nom [30];
char    auteur [20];
int     taille;
```

Toutes les données relatives aux éléments de notre liste sont en fait du texte ou des images ou les deux. Afin de les associer aux données réelles nous disposons pour chaque élément LV_ITEM d'un pointeur LPARAM lParam.

C'est au programmeur d'associer ce pointeur aux données qu'ils désirent. Ce qui se réalise à l'aide de la fonction : **SetItemData** (<no de l'élément concerné>, (LPARAM) ptr_Liste_Desc);

Pour reprendre notre exemple, prenons un tableau de fichiers

```
fichier tab_fichier [50];
```

Pour chaque élément i,

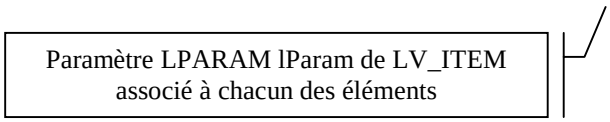
```
lvitem.iItem = i;
lvitem.iSubItem = 0;
lvitem.pszText = "mon_texte";

m_Liste_Fichier.InsertItem (&lvitem);

lvitem.iSubItem = 1;
lvitem.pszText = "mon_texte colonne 2";

m_Liste_Fichier.SetItem (&lvitem);

// Enregistrement des data lié à l'item
m_Liste_Fichier.SetItemData (i, (LPARAM) &tab_fichier [i]);
```



Note:

InsertItem permet de créer un élément
SetItem permet de modifier le contenu d'un élément.

4. Déclaration de la fonction de tri associée

Dans votre objet déclarer la fonction de tri comme ci-dessous:

```
static int CALLBACK MySort(LPARAM lParam1, LPARAM lParam2, LPARAM lParamSort);
```

lParam1, lParam2 sont des pointeurs sur les données à comparer. Ces pointeurs ont été précédemment associées à nos données par la fonction **SetItemData**.

5. Création de la fonction de tri

Note générale :

La fonction de tri doit renvoyer une valeur positive si le premier paramètre doit précéder le second, une valeur négative dans le cas contraire et 0 sir les paramètres sont identiques.

```
int CALLBACK CMaClasse::MySort        (LPARAM lParam1, LPARAM lParam2,
                                       LPARAM lParamSort)
{
        fichier*  lvitem1;
        lvitem1 = (fichier*) lParam1;

        fichier*  lvitem2;
        lvitem2 = (fichier*) lParam2;

        bool      tri_decroissant = (lParamSort >= 10);
```

// lParamSort est un pointeur paramètre défini à votre libre convenance. Dans le cas présent, il permet de savoir si le tri est croissant ou décroissant et sur quel paramètre doit s'effectuer ce tri.

```
	switch (lParamSort)
	{
	case 1:
		{
			// Tri sur 2nde colonne -> Tri sur la taille
			if (lvitem1->taille == lvitem2->taille)
				retour =  0;
			else
			{
				if (lvitem1->taille< lvitem2->taille)
					retour = -1;
				else
					retour = 1;
			}
		break;
	…
	}

	if (tri_decroissant)
		return (-retour);
	else
		return (retour);
}
```

Les contextes graphiques

Mots clés : ***CDC***

1. Création

Lors de sa création, un contexte graphique ne contient rien, ou presque, puisqu'il est initialisé avec un bitmap de 1*1 pixel en Noir et Blanc

```
pDC->CreateCompatibleDC (CDC* Other pDC)
```

Pour créer un contexte graphique, il est préférable de le créer compatible avec un autre, i.e. avec les mêmes caractéristiques de couleurs, de mode…

Pour créer un contexte graphique en mémoire, le plus simple est d'utiliser le DC de l'écran.

```
hdc.CreateIC ("DISPLAY",NULL,NULL,NULL);
DCMem.CreateCompatibleDC (&hdc);
hdc.DeleteDC ();
```

2. Initialisation d'un contexte graphique

Pour initialiser un contexte graphique, il faut sélectionner un objet. Le plus courant des objets à sélectionner est le bitmap. Une fois sélectionné, le contexte graphique prend la forme et la taille de cet objet.

```
DCMem.SelectObject (Trou_noir);
```

ATTENTION seul, SelectObject permet de grandir un contexte graphique

Pour initialiser la palette de couleur, on procède de la façon suivante :

```
// Sauvegarder l'ancienne palette et initialiser avec la nouvelle
CPalette* pOldPal = pDC->SelectPalette (&Palette1,TRUE);

// Appliquer la palette
pDC->RealizePalette ();

// Faire les dessins dans le contexte graphique

// Remettre la précédente palette
pDC->SelectPalette (pOldPal,FALSE);
```

3. Destruction et remplacement des objets sélectionnés

Il n'est pas possible de détruire un objet sélectionné dans un contexte graphique (cause de Bug car erreur non signalée). 2 solutions s'ouvrent donc :

- Sélectionner le nouvel objet que l'on désire mettre dans le contexte graphique et détruire l'ancien.

```
CBitmap* OldBitmap = DCMem.SelectObject (NewBitmap) ;
OldBitmap -> DeleteObject () ;
```

- Se créer un objet NULL, le sélectionner, puis détruire l'ancien.

```
Trou_noir.CreateBitmap (1, 1, 1, 1, NULL);
CBitmap* OldBitmap = DCMem.SelectObject (Trou_noir);
OldBitmap->DeleteObject () ;
```

- Une solution non testée, utiliser detach ?

Gestion de l'impression

Mots clés : ***Printer***

1. Paramètres de l'imprimante par défaut

```
// Lecture des caractéristiques de l'imprimante par défaut
HDC             hdc_default;
DWORD           dwNeeded, dwReturned;

// Information traitement imprimante par défaut
PRINTER_INFO_5 pinfo5[5];

// Recherche de l'imprimante par défaut
if (EnumPrinters (PRINTER_ENUM_DEFAULT, NULL, 5, (LPBYTE) pinfo5,
                sizeof (pinfo5), &dwNeeded, &dwReturned))
        hdc_default = CreateDC (NULL,pinfo5[0].pPrinterName,NULL,NULL);
else
{
        AfxMessageBox ("Aucune imprimante par défaut sélectionnée.");
        return;
}

// Lecture des caractéristiques de l'imprimante

// Taille en cm de la page
GetDeviceCaps (hdc_default, HORZSIZE);
GetDeviceCaps (hdc_default, VERTSIZE);

// Nombre de pixels x,y
GetDeviceCaps (hdc_default, HORZRES);
GetDeviceCaps (hdc_default, VERTRES);

// Résolution en dpi
int     m_ImprResDpi = GetDeviceCaps (hdc_default, LOGPIXELSX);

// Nom de l'imprimante par défaut
pinfo5[0].pPrinterName;
```

2. OnPreparePrinting

pInfo->SetMinPage (1ère page);
pInfo->SetMaxPage (Dernière page);

3. OnPrint

Permet d'accéder au données liées à CPrintInfo lors de l'impression.

pInfo -> m_bPreview indique si la destination est l'aperçu avant impression.
pInfo-> m_nCurPage donne le numéro de la page en cours d'impression.

Choix d'un Répertoire & Parcours du Réseau

Mots clés : ***BrowseInfo, SHBrowseForFolder***

#include "Shlobj.h"

1.BROWSEINFO

Structure de recherche qui est utilisée par SHBrowseForFolder.

- **HWND hwndOwner :** handle de la fenêtre parent.
  exemple : pour une CView : this -> m_hwnd
- **LPCITEMIDLIST pidlRoot :** Adresse d'une structure ITEMIDLIST spécifiant la localisation du répertoire de départ. Seul le répertoire de départ et ses sous-répertoires seront listés. Peut être NULL, dans ce cas, le répertoire Root est pris par défaut.
- **LPSTR pszDisplayName :** Adresse d'un buffer pour réception du nom du répertoire. Ce buffer doit pouvoir recevoir jusqu'à MAX_PATH octets.
- **LPCSTR lpszTitlte :** Sous-titre de la fenêtre de recherche de répertoire.
- **UINT ulFlags :** 0 ou une combinaison des flags suivants:

- BIF_BROWSEFORCOMPUTER : retourne seulement les machines.
- BIF_BROWSEFORPRINTER : retourne seulement les imprimantes.
- BIF_BROWSEINCLUDEFILES : accepte les fichiers en plus des répertoires.
- BIF_DONTGOBELOWDOMAIN : ne parcours pas le voisinage réseau
- BIF_EDITBOX : boîte de dialogue permettant la saisie
- BIF_RETURNFSANCESTORS : accepte seulement les vieux fichiers système !!!
- BIF_RETURNONLYFSDIRS : retourne seulement les répertoires système.
- BIF_VALIDATE : en cas nom incorrect dans la zone d'édition : appel de la fonction BrowseCallBackProc avec le message BFFM_VALIDATEFAILED. Flag ignoré si pas BIF_EDITBOX.

- **BFFCALLBACK lpfn :** Adresse d'une fonction BrowseCallBackProc qui est appelé quand un événement intervient
- **LPARAM lParam :** Valeur passé en paramètre à la fonction BrowseCallBack.
- **int iImage :** Variable recevant l'image associé avec le répertoire sélectionné. Cette image est spécifiée comme l'index dans la liste des images système.

2.SHBrowseForFolder

```
WINSHELLAPI LPITEMIDLIST WINAPI SHBrowseForFolder(LPBROWSEINFO lpbi);
```

Affiche une boîte de dialogue permettant la sélection d'un répertoire.
Retourne l'adresse d'une liste d'identificateur d'item qui spécifie la localisation du répertoire sélectionné par rapport au root ou NULL si l'utilisateur a cliqué sur Cancel.

lpbi : Adresse sur une structure BROWSEINFO qui contient les informations nécessaires à l'affichage de la boîtes de dialogue. L'application d'appel est responsable de la libération de la liste d'identificateurs utilisée par la tâche.

3. Récupérer le chemin complet

BOOL bSuccess = SHGetPathFromIDList(pidl, Rep);

Cette fonction permet d'obtenir le chemin complet à l'élément choisi. Le problème de base était de pouvoir choisir un répertoire mais, seulement un répertoire. Pour cela il est nécessaire d'implémenter une fonction CALLBACK de contrôle. En effet, les noms de machines sont acceptés par la boîte de dialogue par défaut.

4. Contrôle : Est-ce un répertoire ?

Par défaut la fonction Callback renvoie 0, sinon 1.
SHGetPathFromIDList nous permet de savoir si on est en présence d'un répertoire, si ce n'est pas le cas, on envoie un message d'invalidation du bouton OK.

```
int CALLBACK CChoixRep::BrowseNotify(HWND hwnd, UINT uMsg, LPARAM lParam, LPARAM
lpData)
{
	char	Rep [256];

	switch (uMsg)
	{
	case BFFM_SELCHANGED:
		if (!SHGetPathFromIDList ((LPITEMIDLIST) lParam, Rep))
			SendMessage (hwnd,BFFM_ENABLEOK,0,0);
		break;

	default:
		return(0);
	}

	return(1);
}
```

5. Mémoire, oh ma belle mémoire

La documentation liée à SHBrowseForFolder nous indique que la désallocation mémoire est à la charge de l'utilisateur. On utilise donc SHGetMalloc pour récupérer le IMalloc lié à notre fenêtre de choix, puis la commande Free permet de libérer la mémoire.

```
// Desallouer pidl
LPMALLOC alloc;
if (SHGetMalloc(&alloc) == NOERROR)
{
	if (bi.pidlRoot)
	{
		alloc->Free((LPITEMIDLIST)bi.pidlRoot);
	}

	alloc->Free(pidl);
}
```

Attention, en toute apparence, CheckMemory ne permet pas de voir l'oubli de la libération mémoire.

6.Le Code

ChoixRep.h

```
class CChoixRep : public CObject
{
public:
	CChoixRep ();
	~CChoixRep ();

public:
	bool ChoixRep (CString&, CWnd*, CString libelle = "");

private:
	static CALLBACK CChoixRep::BrowseNotify(HWND hwnd, UINT uMsg, LPARAM lParam,
LPARAM lpData);
};
```

ChoixRep.cpp

```
#include "stdafx.h"
#include "ChoixRep.h"
#include "Shlobj.h"

CChoixRep::CChoixRep ()
{
}

CChoixRep::~CChoixRep ()
{
}

bool CChoixRep::ChoixRep (CString& path, CWnd* fen, CString libelle)
{
	char	Rep [MAX_PATH];

	BROWSEINFO bi;
	bi.hwndOwner = fen->m_hWnd;
	bi.pidlRoot = NULL;
	bi.pszDisplayName = NULL;
	bi.lpszTitle = libelle;
	bi.ulFlags = BIF_RETURNONLYFSDIRS;
	bi.lpfn = BrowseNotify;
	bi.lParam = NULL;

	LPITEMIDLIST pidl = SHBrowseForFolder(&bi);
	BOOL bSuccess = SHGetPathFromIDList(pidl, Rep);

	// Desallouer pidl
	LPMALLOC alloc;
	if (SHGetMalloc(&alloc) == NOERROR)
	{
		if (bi.pidlRoot)
		{
			alloc->Free((LPITEMIDLIST)bi.pidlRoot);
		}

		alloc->Free(pidl);
		alloc->Release ();		// Libère le pointeur d'interface
	}

	if (bSuccess)
	{
		path=Rep;
		return (true);
	}
	else
		return (false);

	return (true);
}

int CALLBACK CChoixRep::BrowseNotify(HWND hwnd, UINT uMsg, LPARAM lParam, LPARAM lpData)
{
	char	Rep [256];

	switch (uMsg)
	{
	case BFFM_SELCHANGED:
		if (!SHGetPathFromIDList ((LPITEMIDLIST) lParam, Rep))
```

```
                        SendMessage (hwnd,BFFM_ENABLEOK,0,0);
                break;

        default:
                return(0);
        }

        return(1);
}
```

Note de l'auteur :

Merci à mes amis, qui par leurs soutien ont permis d'éviter l'événement de gauche.

Utilisation de la base de registre sur les répertoires

Mots clés : ***IShellFolder, Base de registres, Répertoires***

```
bool	GetDirectoryList (CString &chemin, char** table_rep, short* nbre, short max_rep)
{
	// Chaque répertoire est écrit sur un maximum de 9 caractères (8 + 0)
	// nbre représente le nombre de répertoire trouvé
	// max_rep représente le nombre maximum de répertoire que l'on peut ajouter

	// Afin de ne pas casser le chemin
	CString  path;

	// Ajout de l'\ si non présent
	if (chemin [chemin.GetLength () - 1] != '\\')
		path = chemin + '\\';
	else
		path = chemin;

	// Nombre de répertoires
	*nbre = 0;

	// Index sur la table des répertoires
	// Gestionnaire des allocations mémoires Shell
	LPMALLOC pMalloc;

	// IshellFolder sur le desktop (bureau)
	LPSHELLFOLDER psfDeskTop = NULL;

	// IShellFolder sur le répertoire concerné
	LPSHELLFOLDER psfDocFiles = NULL;

	// Liste des documents (fichiers)
	LPITEMIDLIST pidlDocFiles = NULL;

	// Elément lu dans la liste énumérée
	LPITEMIDLIST pidlItems = NULL;

	// Liste énumérée des sous-répertoires
	LPENUMIDLIST ppenum = NULL;

	// Nom du fichier (struct variant)
	STRRET strDispName;

	TCHAR pszSourceFiles[256];

	// Nombre de caractères lus durant l'exécution de ParseDisplayName
	ULONG chEaten;

	// Nombre d'éléments restant dans la liste d'une énumération d'objets
	ULONG celtFetched;

	// Code erreur
	HRESULT hr;

	pszSourceFiles[0] = '\0';

	// Obtenir une occurrence du pointeur d'allocation mémoire
	hr = SHGetMalloc(&pMalloc);
```

```
// Obtenir un pointeur sur le bureau
hr = SHGetDesktopFolder(&psfDeskTop);

WCHAR           ole_path [_MAX_PATH * 2];

MultiByteToWideChar (CP_ACP, 0, path, -1, ole_path, _MAX_PATH * 2);

// Obtenir l'IDL correspondant au répertoire
hr = psfDeskTop->ParseDisplayName(NULL, NULL, ole_path, &chEaten, &pidlDocFiles, NULL);

if (FAILED (hr))
{
        // Libération du IshellFolder du bureau
        hr = psfDeskTop->Release();

        // Terminer
        return (false);
}

// Création d'un IshellFolder lié au IIDL du répertoire
hr = psfDeskTop->BindToObject(pidlDocFiles, NULL, IID_IShellFolder, (LPVOID *) &psfDocFiles);

// Libération du IshellFolder du bureau
hr = psfDeskTop->Release();

// Liste de tout ce qu'il y a dans le répertoire
hr = psfDocFiles->EnumObjects (NULL, SHCONTF_FOLDERS, &ppenum);

if (FAILED (hr))
{
        // Libération du pointeur d'énumération lié à la recherche du contenu du répertoire
        ppenum->Release();

        // Libération de la mémoire utilisée par le IshellFolder du répertoire
        pMalloc->Free(pidlDocFiles);

        // Libération du IShellFolder du répertoire
        psfDocFiles->Release();

        return (false);
}

// 256 répertoires dans un sous-répertoire maximum
char    table_desc [256][9];

// Tant qu'il y a des éléments
while ((hr = ppenum->Next(1, &pidlItems, &celtFetched) != S_FALSE) && (celtFetched) == 1)
{
        ULONGrgfInOut = SFGAO_DISPLAYATTRMASK;

        // Attributs du répertoire : provoque sortie de DLL en -1
        // hr = psfDocFiles->GetAttributesOf  (1, (const struct _ITEMIDLIST **) &pidlItems,
                                                &rgfInOut);

        // Demander le nom du fichier / répertoire
        hr = psfDocFiles->GetDisplayNameOf  ((LPCITEMIDLIST) pidlItems,
                                                SHGDN_FORPARSING, &strDispName);

        // Chaine de caractères pour affichage
```

```
                CString  str;

                // Transfomer en chaîne de caractères
                PrintStrRet (str, pMalloc, pidlItems, &strDispName);

                // Chemin
                char       m_short_path [_MAX_PATH];

                // Détermination du nom court du répertoire
                GetShortPathName (str, m_short_path, _MAX_PATH);

                if (*nbre < max_rep)
                {
                        // Extraction du nom court du répertoire, attention certains noms de répertoire n'ont
                        //pas de nom court
                        if (GetFileTitle (m_short_path, (char*) &table_desc [*nbre], 9) == 0)
                                (*nbre) ++;
                }
        }

        // Libération du pointeur d'énumération lié à la recherche du contenu du répertoire
        ppenum->Release();

        // Libération de la mémoire utilisée par le IshellFolder du répertoire
        pMalloc->Free(pidlDocFiles);

        // Libération de l'IDL du répertoire
        pMalloc->Free(pidlItems);

        // Libération du IShellFolder du répertoire
        psfDocFiles->Release();

        // Taille du résultat
        int        tai_res = (*nbre) * 9;

        // Allocation de la mémoire pour le résultat
        *table_rep = new char [tai_res];

        // Copie du résultat
        CopyMemory (*table_rep, table_desc, tai_res);

        // Ok tout c'est bien passé
        return (true);
}

void PrintStrRet (CString& str, LPMALLOC g_pMalloc, LPITEMIDLIST pidl, LPSTRRET lpStr)
{
        LPSTR lpsz;
        int cch;

        switch (lpStr->uType)
        {
        case STRRET_WSTR:
                cch = WideCharToMultiByte( CP_ACP, 0, lpStr->pOleStr, -1, NULL, 0, NULL, NULL);
                lpsz = (LPSTR) g_pMalloc->Alloc (cch);

                 if (lpsz != NULL)
                 {
                        WideCharToMultiByte( CP_ACP, 0, lpStr->pOleStr, -1, lpsz, cch, NULL, NULL);
                        str.Format ("%s", lpsz);
```

```
                        g_pMalloc->Free (lpsz);
                }
        break;

        case STRRET_OFFSET:
                str.Format ("%s", ((char *) pidl) + lpStr->uOffset);
                break;

        case STRRET_CSTR:
                str.Format ("%s", lpStr->cStr);
                break;
        }
}
```

Notes diverses

1. MRU : Most Recent Used Files

Pour accéder à la liste des fichiers située dans le menu Fichier. Utilisez la variable membre m_pRecentFileList de CWinApp en ajoutant #include « Afxpriv.h » afin d'obtenir la déclaration de la classe CRecentFileList.

Réf : Au cœur des MFC p 409-412

2. GetBitmapInfoHeader

GetBitmapInfoHeader n'initialise pas biYPelsPerMeter, seul, biXPelsPerMeter est initialisé.

Réf : Visual C++ 5.0

3. Titre d'une DialogBox

Dans le OnInitDialog :

```
this->SetWindowText ("<Nouveau Titre>");
```

4. ON_MESSAGE

Il est impossible d'utiliser ON_MESSAGE dans un **OnUpdate** ou un **OnInitialUpdate**.

5. Variant

Pour récupérer la valeur d'un variant :

```
variant.bstrVal
```

6. Utilisation de noms courts

Style de boîte de dialogue : OFN_NOLONGNAME
Recherche :

```
WIN32_FIND_DATA     Arg_recherche;   // Argument de recherche
HANDLE          Rec;                // Handle de recherche

Rec = FindFirstFile (path, &Arg_recherche);
FindNext (Rec, &Arg_recherche);
```

Réf: Symbcom

7. Utilisation d'un même accélérateur sur plusieurs fenêtres

1. Créer un ID virtuel dans la table des accélérateurs (Création automatique si nom inexistant)
2. Lier cet ID à un ensemble de touches d'accélérateur
3. Appliquer ClassWizard sur l'ID de cet accélérateur
4. Envoyer le message à l'ID concerné

```
SendMessage (WM_COMMAND,ID_CONCERNE);
```

8. Aide .hlp

```
{bm[x]  image.bmp}
[x]     l : left
        c : center
        r : right
```

9. Boîte de dialogue : UpdateData

UpdateData (true) permet de récupérer les données (Mise à jour)
UpdateDate (false) permet d'initialiser les données sans les mettre à jour à l'affichage.

10. Constructeur

Attention, les fonctions dans un constructeur ne sont pas effectuées de manière linéaire (?).

11. PragmaPack

Dans une structure, un byte peut faire 2 octets si pas PragmaPack (1) (selon l'alignement des octets).

12. SetFocus & GetFocus

SetFocus permet de donner le focus d'entrée à une fenêtre.
GetFocus permet d'obtenir une pointeur sur la fenêtre qui possède le focus.

13. CArchive

Utilisation explicite d'un CArchive combiné avec un fichier binaire.

Sauvegarde

```
CFile     m_fichier;
m_fichier.Open ("c:\\Test.bb", CFile::modeCreate | CFile::modeReadWrite);

char      toto [7] = "Coucou";
toto [6] = 0;

m_fichier.Write (toto,7);

CArchive          archive (&m_fichier, CArchive::store);

CString  m_str;
for (int i=0; i<10; i++)
{
          m_str.Format ("Phrase essai n°%d.",i+1);
          archive << m_str;
}

archive.Close ();

m_fichier.Close ();
}
```

Lecture

```
CFile     m_fichier;
m_fichier.Open ("c:\\Test.bb", CFile::modeRead);

char      toto [7];
m_fichier.Read (toto,7);
AfxMessageBox (toto);

CArchive          archive (&m_fichier, CArchive::load);

CString  m_str;
for (int i=0; i<10; i++)
{
          archive >> m_str;
          AfxMessageBox (m_str);
}

archive.Close ();

m_fichier.Close ();
}
```

14. Message utilisateur

1. Définir la valeur du message par
   #define WM_MON_MESSAGE WM_USER + …

2. Dans le .cpp de la classe concerné, rajouter à la liste des messages :
   ON_MESSAGE (WM_MON_MESSAGE, OnMonMessage)

3. Dans le .h définir la fonction OnMonMessage de la façon suivante :
   Positionner vous après la déclaration des messages du wizard :
   //}}AFX_MSG

   afx_msg LRESULT OnMonMessage (WPARAM wParam, LPARAM lParam);
   ou
   afx_msg void OnMonMessage ();
   ou …

   et avant la déclaration de fin de table des messages
   DECLARE_MESSAGE_MAP()

15. Choix d'un lecteur

Utilisation de la fonction SetupPromptForDisk …

Les codes des touches virtuelles

Décimal	*Hexa*	*Identificateur Windows*	*Clavier IBM*
1	01	VK_LBUTTON	
2	02	VK_RBUTTON	
3	03	VK_CANCEL	Ctrl Break
4	04	VK_MBUTTON	
8	08	VK_BACK	Backspace
9	09	VK_TAB	Tabulation
12	0C	VK_CLEAR	Touche 5 Pavé avec verrouillage numérique éteint
13	0D	VK_RETURN	Entrée
16	10	VK_SHIFT	Majuscule
17	11	VK_CONTROL	Ctrl
18	12	VK_MENU	Alt
19	13	VK_PAUSE	Pause
20	14	VK_CAPITAL	Verrouillage Majuscule
27	1B	VK_ESCAPE	Echappement
32	20	VK_SPACE	Barre d'espace
33	21	VK_PRIOR	Page Préc.
34	22	VK_NEXT	Page Suiv.
35	23	VK_END	Fin
36	24	VK_HOME	Début
37	25	VK_LEFT	Flèche vers la gauche
38	26	VK_UP	Flèche vers le haut
39	27	VK_RIGHT	Flèche vers la droite
40	28	VK_DOWN	Flèche vers le bas
41	29	VK_SELECT	
42	2A	VK_PRINT	
43	2B	VK_EXECUTE	
44	2C	VK_SNAPSHOT	Impression Ecran
45	2D	VK_INSERT	Insertion
46	2E	VK_DELETE	Suppression
47	2F	VK_HELP	
48-57	30-39		Touches 0 à 9 du clavier principal

65-90	41-5A		A à Z
96	60	VK_NUMPAD0	Touche 0 du Pavé Numérique (verrouillage actif)
97	61	VK_NUMPAD1	Touche 1 du Pavé Numérique (verrouillage actif)
98	62	VK_NUMPAD2	Touche 2 du Pavé Numérique (verrouillage actif)
99	63	VK_NUMPAD3	Touche 3 du Pavé Numérique (verrouillage actif)
100	64	VK_NUMPAD4	Touche 4 du Pavé Numérique (verrouillage actif)
101	65	VK_NUMPAD5	Touche 5 du Pavé Numérique (verrouillage actif)
102	66	VK_NUMPAD6	Touche 6 du Pavé Numérique (verrouillage actif)
103	67	VK_NUMPAD7	Touche 7 du Pavé Numérique (verrouillage actif)
104	68	VK_NUMPAD8	Touche 8 du Pavé Numérique (verrouillage actif)
105	69	VK_NUMPAD9	Touche 9 du Pavé Numérique (verrouillage actif)
106	6A	VK_MULTIPLY	* du pavé numérique (clavier étendu)
107	6B	VK_ADD	+ du pavé numérique (clavier étendu)
108	6C	VK_SEPARATOR	
109	6D	VK_SUBTRACT	- du pavé numérique (clavier étendu)
110	6E	VK_DECIMAL	. du pavé numérique
111	6F	VK_DIVIDE	/ du pavé numérique (clavier étendu)
112	70	VK_F1	Touche de fonction F1
113	71	VK_F2	Touche de fonction F2
114	72	VK_F3	Touche de fonction F3
115	73	VK_F4	Touche de fonction F4
116	74	VK_F5	Touche de fonction F5
117	75	VK_F6	Touche de fonction F6
118	76	VK_F7	Touche de fonction F7
119	77	VK_F8	Touche de fonction F8
120	78	VK_F9	Touche de fonction F9
121	79	VK_F10	Touche de fonction F10
122	7A	VK_F11	Touche de fonction F11 (clavier étendu)
123	7B	VK_F12	Touche de fonction F12 (clavier étendu)
124	7C	VK_F13	
125	7D	VK_F14	
126	7E	VK_F15	
127	7F	VK_F16	
144	90	VK_NUMLOCK	Verrouillage numérique
145	91	VK_SCROLL	Verrouillage défilement