```
#include "aporia.h"
#include <string.h>
#include <stdio.h>

extern  int pascal WinMain(unsigned short hInstance,unsigned short hPrevInstanc
e,char far *lpCmdLine,int nCmdShow);
extern  long far pascal AporiaWndProc(unsigned short hWnd, unsigned msg, unsign
ed short wParam, long lParam);
extern  long far pascal ExecWndProc(unsigned short hWnd, unsigned msg, unsigned
 short wParam, long lParam);

char *ProgName = PROGRAM_NAME;

typedef struct tag_notes {
    HANDLE hNotePath;       // name of the note file.
    HANDLE hDisplayProg;    // program to view notes or edit notes.
} Notes;

typedef struct tag_rep {
    HBITMAP hBitmap;    // pointer to bitmap
    POINT Pos;          // position of the icon
} Representation;

typedef struct purpose_rep {
    struct {
        unsigned IsBuiltIn; // a special icon.
    } Flags;
    HANDLE hPath;       // name of program found through DOS PATH.
} Purpose;

typedef struct tag_icon {
    int IconId;         // Id of this icon for searching the list.
    HANDLE hDisplayName;// user's name for this program.
    Purpose Action;     // action associated with this icon.
    Representation Rep; // Representation of this icon.
    Notes Note;         // notes associated with the icon
    Notes Help;         // help associated with the icon
    HANDLE hNext;       // next icon
} Icon;

#define MAXICONS    20
struct Aporia {
    short count;        // number of icons.
    HANDLE hFirstIcon;  // list of icons
    // owner            // owner of this aporia: password etc.
    Notes Note;         // notes associated with aporia
    Notes Help;         // help associated with aporia
    struct {
        unsigned IsChanged;     // Icons have changed, please save on close.
        unsigned ShowBitmaps;   // Display the bitmaps.
    } Flags;
} Aporia;

HWND MakeAporiaWindow(HANDLE hInstance)
{
    // Create the Aporia window
    return CreateWindow(
        ProgName,           // window class
        ProgName,           // window name
        WS_OVERLAPPEDWINDOW,// window style
        200,                // x position
        200,    //CW_USEDEFAULT             // y position
        100,    //CW_USEDEFAULT             // width
        100,                // height
        NULL,               // parent handle
        NULL,               // menu or child ID
        hInstance,          // instance
        NULL);              // passed in lParam field on CREATE
}

#define BITMAPHEIGHT 20
#define BITMAPWIDTH 20
#define BORDERSIZE 10
#define EXECICONCLASS   "IconClass"
int NextStartPos = BORDERSIZE;
HWND MakeIconWnd( HANDLE hInstance, Icon *ip, char *name)
{
    HWND hWnd;
    int StartY, StartX, Height, Width;

    // Default layout:
    //  Icons are on the far right, right adjusted to a border, as wide as
    //  the greater of the text or bitmap, equally spaced downward.

    // width: greater of the length of the string or bitmap.
    Width = max( ((ip->Rep.hBitmap && Aporia.Flags.ShowBitmaps)?BITMAPWIDTH:0),

        (strlen(name) * NewFontWidth) );
    // x position: far right, right adjusted to border.
    StartX = GetSystemMetrics(SM_CXSCREEN) - BORDERSIZE - Width;
    // y position: equally spaced after the previous icon.
    StartY = NextStartPos;
    // height: sum of the bitmap height and a bit more than the leading.
    Height = ((ip->Rep.hBitmap && Aporia.Flags.ShowBitmaps)?BITMAPHEIGHT:0)
        + NewFontLeading + 1;
    // Save NextStartPos: iconHeight + Border
    NextStartPos += Height + BORDERSIZE;

    // Save position in icon.
    ip->Rep.Pos.x = StartX;
    ip->Rep.Pos.y = StartY;

    // Create Window and set the its Icon Num in the extra allocated space.
    if( hWnd = CreateWindow( EXECICONCLASS, name, WS_POPUP,
        StartX, StartY, Width, Height,
        NULL, 0, hInstance,  NULL))
            SetWindowWord( hWnd, 0, ip->IconId );

    return hWnd;

BOOL MakeIcon( HANDLE hInstance, char *Path, char *DisplayName, char *BitmapNam
e, int IsBuiltIn )
{
    HANDLE hNewIcon;
    Icon *ip;
    char *cp;
    HWND hWnd;

    // Allocate the icon and put at head of list
    if( hNewIcon=LocalAlloc(sizeof(Icon),LMEM_MOVEABLE|LMEM_ZEROINIT) ) {
        if( ip=(Icon *)LocalLock( hNewIcon) ) {

            // Insert at head of list
            ip->hNext = Aporia.hFirstIcon;  // take over the head.
            Aporia.hFirstIcon = hNewIcon;   // reset pointer to the head.

            // Create its purpose.
            if( (*Path) && (ip->Action.hPath=LocalAlloc( strlen(Path)+1,
            LMEM_MOVEABLE|LMEM_ZEROINIT)) ) {
                if( cp = LocalLock( ip->Action.hPath) ) {
                    strcpy(cp,Path);
                    LocalUnlock(ip->Action.hPath);
                }
                else {
                    LocalFree(ip->Action.hPath);
                    ip->Action.hPath = NULL;
                }
            }
            ip->Action.Flags.IsBuiltIn = IsBuiltIn;

            // Store the DisplayName.
            if( ip->hDisplayName=LocalAlloc( strlen(DisplayName)+1,
            LMEM_MOVEABLE|LMEM_ZEROINIT) ) {
                if( cp = LocalLock( ip->hDisplayName) ) {
                    strcpy(cp,DisplayName);
                    LocalUnlock(ip->hDisplayName);
                }
                else {
                    LocalFree(ip->hDisplayName);
                    ip->hDisplayName = NULL;
                }
            }

            // The Icon bitmap
            ip->Rep.hBitmap = LoadBitmap(hInst, (LPSTR) BitmapName);

            // The Icon Id
            ip->IconId = Aporia.count;

            // Create new count. NB. must be updated now for searches to work.
            ++Aporia.count;

            // Create a window for the structure.
            if( hWnd = MakeIconWnd( hInstance, ip, DisplayName) ) {
                ShowWindow(hWnd, SW_SHOWNORMAL);    // Show as icon
                UpdateWindow(hWnd);                 // Send WM_PAINT

                LocalUnlock(hNewIcon);
                return TRUE;
            }
            else {
                // clean-up.
                --Aporia.count;
                Aporia.hFirstIcon = ip->hNext;
                LocalUnlock(hNewIcon);
                LocalFree(hNewIcon);
            }
        }
        else
            LocalFree(hNewIcon);
    }
    return FALSE;
}

#define DEFAULT_PROG    "clock.exe"
void LoadIcons(HANDLE hInstance)
{
    int IconId, LastIcon;

    // Load the standard special icons.
    MakeIcon( hInstance, "", "trash", "trash", 1);
    MakeIcon( hInstance, "", "help", "help", 1);
    MakeIcon( hInstance, "", "notes", "notes", 1);
    MakeIcon( hInstance, "", "directory", "dirs", 1);

    // Load the icons from last time as saved in win.ini.
    //  When aporia stores each icon in the win.ini file,
    //  each icon is given a unique id starting from 0:
    //      [Aporia]
    //      Icon0=clock.exe
    //      Count=1
    // Count is the limit.
    for( IconId = 0, LastIcon = GetProfileInt( ProgName, "Count", 0 );
    IconId < LastIcon; IconId++)
    {
```