Arbeiten mit der MFC - Library
(Microsoft Foundation Class Library)

1. Die Microsoft Foundation Classes

Zur Windows - Programmierung in C++ stellt Microsoft mit den MFC eine komplette Sammlung vorhandener Windowsklassen mit Attributen und Methoden bereit. Objektorientierte Grundsätze wie die Polymorphie werden durch eine Vererbungshierarchie bewerkstelligt.
Möchte ein Programmierer eine Anwendung entwickeln, so kann er seine Anwendung von einer Basisklasse ableiten und muss nur noch die speziellen Eigenschaften und Methoden seines Programms Implementieren.

2. Hallo Windows Welt mit MFC

2.1 Die Headerdatei Hello.h

In der Headerdatei Hello.h werden zunächst zwei Klassen eingerichtet:

Da die MFC objektorientiert ausgerichtet ist, beginnt das Programm mit zwei Klassendeklarationen:
CRahmenfenster, die Hallo Welt Klasse wird von der MFC - Klasse CFrameWnd abgeleitet und erbt damit alle Eigenschaften und Methoden eines Rahmenfensters.
Als Methoden werden der Konstruktor und OnPaint( ) zum Zeichnen in das Fenster deklariert.
DECLARE_MESSAGE_MAP( ) ist ein MFC Makro. Hier wird eine Nachrichtentabelle für unser Fenster eingerichtet, das automatisch die Message Loop startet.
CString m_Text ist eine Eigenschaft zur Speicherung des Ausgabetextes. Unter Windows ist eine extra Klasse für Strings definiert (deshalb wird <cstring> eingebunden!).

Die Klasse für die eigentliche Applikation CMeineApp leitet man von CWinApp ab. Damit sind alle Methoden einer Anwendung (z.B. auch WinMain(...)) vererbt und vor dem Programmierer verborgen.
Die virtuelle Methode InitInstance( ) benutzt man zum Erzeugen und Anzeigen des Rahmenfensters.

```
#ifndef HELLO
#define HELLO

//Grundprogramm für Windows mit MFC
#include <afxwin.h>
#include <cstring>

// Deklaration der abgeleiteten Rahmenfensterklasse
class CRahmenfenster : public CFrameWnd
{
public:
        CRahmenfenster( );
        void OnPaint( );

        DECLARE_MESSAGE_MAP( )

private:
        CString m_Text;
};

// Deklaration der abgeleiteten Anwendungsklasse
class CMeineApp : public CWinApp
{
public:
        virtual BOOL InitInstance( );
};

#endif
```

2.2 Die Quelltextdatei Hello.cpp

Die Methoden der Klassen werden in einer Quelltextdatei definiert:

Im ersten Schritt erzeugt man ein Anwendungsobjekt dieApp. Bei der Erzeugung wird automatisch die Methode InitInstance( ) der Anwendung aufgerufen.
Anschließend definiert man den Konstruktor unseres Anwendungsfensters.
Hier wird die Variable classname deklariert, die eine Bezeichnung für eine registrierte Windows - Fensterklasse speichern kann.
Über AfxRegisterWndClass(...) registriert man die Fensterklasse mit Cursor und Hintergrundfarbe.
Create erzeugt die Klasse. Dazu muss der Fensterklassenname, der Fenstertext, der Fensterstil und die Größe als CRect übergeben werden.
m_Text wird mit "Hallo Windows Welt" initialisiert.

```
#include "Hello.h"

//Anwendungs-Objekt erzeugen
CMeineApp dieApp;


// Konstruktor von CRahmenfenster
CRahmenfenster::CRahmenfenster( )
{
  //selbst die Fensterklasse definieren
  LPCTSTR classname;
  classname = AfxRegisterWndClass(CS_HREDRAW|CS_VREDRAW,
              LoadCursor(NULL, IDC_ARROW),
              (HBRUSH) (COLOR_WINDOW+1), 0);

  //Fenster erzeugen
  Create(classname, "MFC Hello Windows World",
        WS_OVERLAPPEDWINDOW,
        CRect(10,10,250,100));

  m_Text = "Hallo Windows Welt!";
}
...
```

Arbeiten mit der MFC - Library
(Microsoft Foundation Class Library)

Immer wenn ein Fenster aktualisiert wird, weil es gerade erzeugt bzw. verschoben usw. wurde, sendet Windows die WM_PAINT Botschaft an das Fenster. Der Job des Programmierers ist es, dafür zu sorgen, dass das Fenster seinen Inhalt neu zeichnet. Dies geschieht in der Methode OnPaint( ).
Hier wird der Gerätekontext zum Zeichnen in den Clientbereich des Rahmenfensters erzeugt. Anschließend wird mit TextOut(...) der Inhalt von m_Text in das Fenster geschrieben.
Zum Schluss muss noch die Originalmethode OnPaint( ) von CFrameWnd aufgerufen werden.

```
//Antwortfunktion zu WM_PAINT
void CRahmenfenster::OnPaint( )
{
  // DC im Rahmenfenster
  CClientDC dc(this);
  dc.TextOut(45, 10, m_Text);

  // Originalmethode aufrufen
  CFrameWnd::OnPaint( );
}
```

Für die einfache Botschaftsverarbeitung nutzt man wie bereits erwähnt eine Nachrichtentabelle. BEGIN_MESSAGE_MAP(...) kennzeichnet den Anfang der Nachrichtentabelle und übernimmt dazu den Typ unserer Rahmenfensterklasse sowie die Basisklasse). Da unser Rahmenfenster nur auf die Botschaft WM_PAINT reagieren soll, ist der einzige Eintrag ON_WM_PAINT( ). Der Botschaft wird in der Tabelle ein ON_ vorangestellt.
Komplexere Anwendungen haben natürlich wesentlich mehr Einträge in der Message Map.
END_MESSAGE_MAP( ) kennzeichnet das Ende der Nachrichtentabelle für das Rahmenfenster.

```
// Antworttabelle
BEGIN_MESSAGE_MAP(CRahmenfenster, CFrameWnd)
        ON_WM_PAINT( )
END_MESSAGE_MAP( )
```

Im letzten Schritt initialisiert man die Anwendung in der Methode InitInstance( ), die beim Erzeugen der Applikation automatisch aufgerufen wird.
In m_pMainWnd wird ein Zeiger auf das Rahmenfenster im Heap gespeichert. m_pMainWnd ist in der Basisklasse CWinApp deklariert und wurde von unsrer Basisklasse geerbt.
ShowWindow(...) bzw. UpdateWindow(...) sorgen für eine Korrekte Anzeige des Rahmenfensters.

```
// Anwendung initialisieren
BOOL CMeineApp::InitInstance( )
{
        // Rahmenfenster-Objekt erzeugen
        // und Fenster anzeigen
        m_pMainWnd = new CRahmenfenster;
        m_pMainWnd->ShowWindow(m_nCmdShow);
        m_pMainWnd->UpdateWindow( );

        return TRUE;
}
```

- ☞ **Zeichnen Sie ein Klassenstrukturdiagramm für Ihre Anwendung!**
- ☞ **Nutzen Sie die MSDN - Library, um andere Fensterstile zu finden. Erzeugen Sie Ihr Rahmenfenster mit einem anderen Fensterstil (WS_ ...).**
- ☞ **Nutzen Sie SetWindowText(...) um den Titel Ihrer Anwendung zu ändern!**
- ☞ **Ändern Sie den Ausgabetext und die Position des Ausgabetextes!**

2.3 MFC im Vergleich zu API

- ☞ **Beschreiben Sie die Unterschiede eines MFC - basierten Windows - Programms im Vergleich zu einem API - Programm!**

Im Vergleich zur Windowsprogrammierung mit der API fällt auf, dass die Nutzung der MFC einige komplexe Details vor dem Programmierer verbirgt. WinMain(...) und die Fensterfunktion WndProc(...) sind bereits in den MFC - Klassen gekapselt. Durch das Ableiten seiner eigenen Klassen von MFC - Basisklassen kann der Programmierer relativ einfach komplexe Anwendungen entwickeln. Grundlage dafür ist natürlich das Verständnis der objektorientierten Programmierung in C++.
Die Message Loop ist in der sogenannten Message Map versteckt. Hier stellt man einfach eine Tabelle mit allen Botschaften und Methoden zusammen, auf die das Programm reagieren soll.

Insgesamt gesehen enthält die MFC alles, was für die Windowsprogrammierung notwendig ist. Die Schwierigkeit besteht nicht mehr darin, etwas Neues zu programmieren, sondern die entsprechende Funktionalität in der MFC zu finden und anzuwenden.