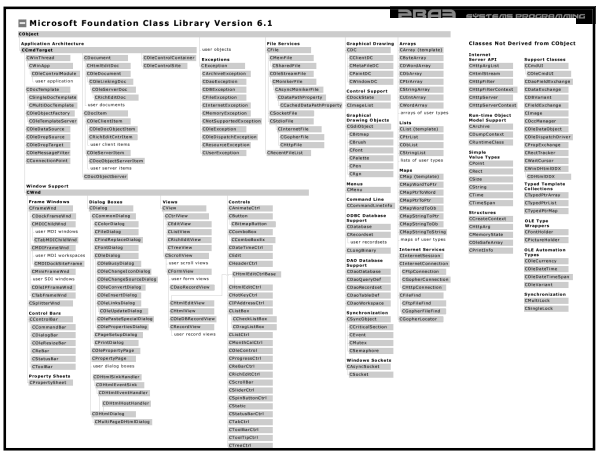
MFC fundamentals

- The Microsoft Foundation Classes are a C++ - based class hierarchy.
- At the top is a class called CObject, which provides some run-time support features, such as serialization.
  - This feature allows an object to store itself on disk and be restored later, e.g. on subsequent invocation of the program
- Its functionality is contained in all of its derived classes.


```
class CMainWin : public CFrameWnd
{
public:
     CmainWin();
     afx_msg void OnPaint();
     afx_msg void OnChar();
     DECLARE_MESSAGE_MAP();
};
```

Classes derived from CObject

- CCmdTarget encapsulates the functionality which handles messages
- Cfile provides support for file operations
- CDC encapsulates device-context support
  - a device context is needed to define a display environment, such as a screen, and allows drawing of simple objects in a window

Classes derived from CObject - cont.

- CGdiObject is the base class for GDI (Graphics Device Interface) objects, such as brushes, bitmaps, and pens
- CException provides exception handling
- CMenu encapsulates menu support
- Other classes directly derived from CObject support such things as thread synchronisation, databases, arrays and lists.

CWnd

- From CCmdTarget is derived one very important class: CWnd
- This is the base class from which all windows are derived, including:
  - Frame windows (normal)
  - Dialog boxes
  - various control windows

CFrameWnd

- CFrameWnd is the most commonly used class derived from CWnd.
- This class provides support for SDI (Single Document Interface) windows.
- It provides the main window for most MFC-based Windows applications.
- It is within classes derived from CFrameWnd that you will typically declare your application's main message map

CWinThread

- CWinThread defines a thread of execution for your application,
- It provides support for thread-based multi-tasking
- From CWinThread is derived the most important class that you will deal with directly in an MFC program: CWinApp

CWinApp

- CWinApp encapsulates an MFC-based application, and controls the startup, initialisation, execution, and shutdown of the program.
- Each MFC program will have only one object of type CWinApp.
- When this object is instantiated, the application begins running.

AFXWIN.H

- All MFC Windows programs must include AFXWIN.H
- This defines the MFC classes
- It also includes windows.h
- MFC also provides several global functions, prefixed with Afx, e.g. AfxMessageBox()
- In general, Afx functions are independent of the class hierarchy

An MFC application Skeleton

- In its simplest form, an MFC program consists of two classes that you must derive from the MFC hierarchy.
  - an application class, derived from CWinApp
  - a window class, derived from CFrameWnd
- Your program is free to derive any other classes that it wishes, but it must include these two.

MFC programming - the steps

- Derive a window class from CFrameWnd
- Derive an application class from CWinApp
- Define a message map.
- Override CWinApp's InitInstance method.
- Create an instance of your own application class

```
#include <afxwin.h>

// derive essential classes
//this is the main window class
class CMainWin : public CFrameWnd
{
public:
  CMainWin();
  DECLARE_MESSAGE_MAP();
};

//construct a window
CMainWin::CMainWin()
{
  Create(NULL, "2BA3 First MFC Application");
}
```

```
// This is the application class
class CApp : public CWinApp
{
public:
  BOOL InitInstance();
};


//Initialise the application
BOOL CApp::InitInstance()
{
  m_pMainWnd = new CMainWin;
  m_pMainWnd->ShowWindow(m_nCmdShow);
  m_pMainWnd->UpdateWindow();

  return TRUE;
}
```

```
//The application's message map
BEGIN_MESSAGE_MAP(CMainWin, CFrameWnd)
END_MESSAGE_MAP()

//Instantiate the application
CApp App;
```

The MFC program -1

- The MFC program derives a class called cMainWin from the CFrameWnd class.
- You must do this to create a frame window.
- Inside this class two members are declared:
  - the constructor
  - the DECLARE_MESSAGE_MAP() macro
- Any window that will process messages must include this, and it should always be the last member declared within the class.
- The window itself is created inside the constructor by the Create() method.

The MFC program -2

- The application class, CApp, is derived from CWinApp.
- InitInstance() is a member of CWinApp, and is a virtual function that must be overridden by your application class:

```
virtual BOOL CWinApp::InitInstance();
```

- Aside: Virtual Functions:
  - A virtual function is declared in a base class, but re-defined by a derived class, and is defined as:

```
virtual int InitShape();
```

  - A pure virtual function is a virtual function that has no definition within the base class, and is defined as:

```
virtual int DrawShape() = 0;
```

The MFC program -3

- InitInstance() is called every time a new instance of your program is started.
- It creates an object of type CMainWin (using the new operator... more later).
- The new operator returns a pointer to the new object, and this pointer is stored in m_pMainWnd.
- m_pMainWnd is a pointer of type CWnd*
- This variable is used by nearly all MFC-based programs, because it is a pointer to your program's main window object.
- It is used to call various member functions related to your main window, as in the next two lines.

The MFC program -4

- ShowWindow() determines how the window should be shown (e.g. SW_MAXIMIZE, SW_MINIMIZE etc...)
- UpdateWindow() tells Windows to send a message to your application that the main window needs to be updated.

Processing Messages

- MFC provides a set of predefined message handler functions
- If your program implements one of these handlers, then that function will be called whenever its associated message is received.
- When a message has additional information associated with it, this information will be passed to the handler as an argument.

Responding to Messages

- To respond to a message, your program must perform these three steps:
  - The message macro corresponding to the message must be added to your program's message map
  - The prototype for the message handler must be added to the window class that will process the message
  - You must implement the message handler associated with the message.

Adding Message Macros to the Message Map

- MFC message macros have the same name as standard Windows messages, but they all begin withe ON_ and end with ( )
  - The only exception is `WM_COMMAND`, which has the associated message macro `ON_COMMAND()`
- Examples:
  - `ON_WM_LBUTTONDOWN(), ON_WM_PAINT(), ON_WM_CLOSE(), ON_WM_MOVE(), ON_WM_CHAR()`
- To add a message macro to your message map, simply include it as follows:

```
BEGIN_MESSAGE_MAP(CMainWin, CFrameWnd)
     ON_WM_CHAR()
END_MESSAGE_MAP()
```

Adding Message Handlers to your Window Class

- Each message that your program responds to must be associated with a message handler
- All message handlers are members of the CWnd class, and may be overidden by your program
- Usually the name of the message handler is the name of the message, without the WM_ and precede by On.
  - E,g, the handler for WM_CHAR is `OnChar()`
- You must add the handler to the window class that your program defines.