Inhaltsverzeichnis

5. Beispiele / Projekte:

Die Visual Studio Solution MixedLanguageProgramming.sln enthält diverse IVF und VC Projekte (.vcproj, .vfproj), die beispielhaft das zuvor Gesagte illustrieren sollen.

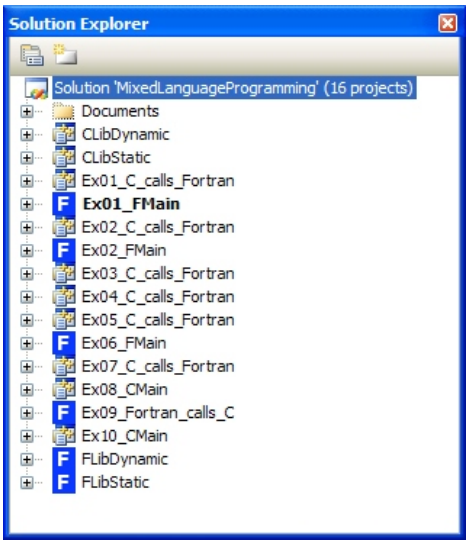

Abb.7: Die Visual Studio Solution MixedLanguageProgramming

A. FLibStatic (f90):
erstellt eine statische Bibliothek (.lib) aus diversen Fortran SUBROUTINEs, FUNCTIONs und MODULEs, die von den anderen Projekten verwendet wird.

B. FLibDynamic (f90):
erstellt eine dynamische Bibliothek (.dll) aus den gleichen Fortran SUBROUTINEs, FUNCTIONs und MODULEs wie FLibStatic. Man beachte die Exportdatei DLLExports.def, die für die "Publizierung" der Symbole sorgt. Die FLibDynamic wird von anderen Projekten verwendet.

1. Ex01_Main (f90):
FMain ruft die SUBROUTINEs FSub01, FSub04 und die FUNCTION FRFun01 aus der FLibStatic.
Das Programm dient der "Einstimmung" und zum Nachweis, daß die Fortran Routinen, die auch in den C++ Programmen verwendet werden, unverändert innerhalb eines Fortran Programms gerufen werden können.

2. Ex02_Main (f90):
FMain ruft die SUBROUTINEs FSub01, FSub04 und die FUNCTION FRFun01 aus der FLibDynamic.
Es dient zum einen dem gleichen Zweck wie Ex01_Main und zeigt überdies die Verwendung einer DLL.

3. Ex01_C_calls_Fortran (cpp):
Das C++ Programm ruft die Fortran
`SUBROUTINE FSub01( i2Arg, r4Arg, i4Arg, r8Arg )`
aus der FLibStatic. Es zeigt Grundsätzliches zum Aufruf einer Fortran SUBROUTINE und zur Typkompatibilität zwischen Fortran und C/C++.

4. Ex02_C_calls_Fortran (cpp):
Das C++ Programm ruft die
`SUBROUTINE FSub01( i2Arg, r4Arg, i4Arg, r8Arg )`
und die
```
FUNCTION FRFun01( rArg, sText1, iArg, sText2 )
                                RESULT(lRet)
```
aus der FLibDynamic. Es zeigt den Aufruf einer Fortran SUBROUTINE und einer FUNCTION mit CHARACTER-Argumenten in einem C++ Programm sowie die Verwendung einer DLL.

5. Ex03_C_calls_Fortran (cpp):
Das C++ Programm kreiert die Struktur

`struct T_PolarCoord`
analog zu denen des Fortran `MODULE FMixLangMod01`
`TYPE T_PolarCoord`,
führt zusätzlich eine Struktur (`struct_doublecomplex`) zur Behandlung von COMPLEXen Zahlen ein und ruft FUNCTIONs des MODULE

```
FUNCTION FCxFun02( tPolarCoord ) RESULT(cxRetVal)
FUNCTION FCxFun03( cxVal ) RESULT(tPolarCoord)
```

aus der FLibStatic.

6. Ex04_C_calls_Fortran (cpp):
   Das C++ Programm ruft die

```
SUBROUTINE FSub04( n, m, iMatrix, rVectorIn,   &
                   rVectorOut )
```

   aus der FLibDynamic. Es zeigt den Umgang mit Feldern (arrays).

7. Ex05_C_calls_Fortran (cpp):
   Das C++ Programm verwendet globale Variablen, die in dem Fortran `MODULE FMixLangMod02` deklariert und initialisiert sind. Es kreiert zudem Strukturen analog denen des MODULEs und zeigt den Zugriff auf deren Komponenten. Das MODULE wird über die statische Library FLibStatic eingebunden.

8. Ex06_FMain (f90):
   Das Programm greift auf initialiserte Variablen des COMMON Block

```
COMMON / Ex06CmnBl / cType, rVector, lOnOff,   &
                     cName, iMatrix
```

   zu, der sich in der statischen Library FLibStatic befindet.

9. Ex07_C_calls_Fortran (cpp):
   Das C++ Programm greift wie das Projekt zuvor auf initialiserte Variablen des COMMON Block

```
COMMON / Ex06CmnBl / cType, rVector, lOnOff,   &
                     cName, iMatrix
```

   zu, der in der statischen Library FLibStatic abgelegt ist. Der COMMON Block enthält auch CHARACTER Komponenten. Für den COMMON Block wird in C++ eine äquivalente Struktur erzeugt (`struct_CmnBl`), und damit wird der COMMON Block "importiert". Seine Werte werden mithilfe der C++ Funktion `CDispCommonVars` direkt angesprochen und angezeigt. Eine ähnliche Funktion demonstriert wie man den COMMON Block als Argument einer Funktion verwenden kann (`CDispCmnBlStruct`).
   Das Programm weist auf mögliche Probleme bei Verwendung in einer DLL hin.

10. Ex08_CMain (cpp):
    Das C++ Programm ruft die C++ Funktionen

```
extern "C" void CSub01( int iArg, float fArg,
                        double &dArg )
extern "C" size_t CFun02( const char* szTextIn,
                          char* szTextOut,
                          const size_t DimTextOut,
                          bool &lInToOut )
extern "C" int* CreateIntArray( int iDim )
```

    der statischen Bibliothek CLibStatic, die auch vom folgenden Fortran Programm verwendet werden.

11. Ex09_Fortran_calls_C (f90):
    Das Fortran Programm ruft die C++ Funktionen

```
extern "C" void CSub01( int iArg, float fArg,
                        double &dArg )
```

```
extern "C" size_t CFun02( const char* szTextIn,
                          char* szTextOut,
                          const size_t DimTextOut,
                          bool &lInToOut )
extern "C" int* CreateIntArray( int iDim )
```

aus der dynamischen Bibliothek CLibDynamic sowie die Standard C-Funktion

```
size_t strlen( const char *str )
```

der C/C++ Laufzeitbibliotheken. Außerdem wird die Windows Betriebssystem Funktion (WinAPI)

```
void CopyMemory( PVOID Destination,
                 const VOID* Source,
                 SIZE_T Length )
```

gerufen.

Im `MODULE Ex09CInterfaces` sind die Schnittstellen (INTERFACEs) zu den C++ Funktionen definiert, so daß ein Fortran Programm die C++ Funktionen rufen kann. Gezeigt werden

+ der Aufbau der INTERFACE Blöcke zum Import der C++Funktionen unter Verwendung der Direktiven des VF Compilers (`!DEC$ ATTRIBUTES`).
+ was bei der Verwendung von CHARACTER Argumenten zu beachten ist
+ wie in VF mit einem C/C++ Pointer zu verfahren ist, der auf eine Speicheradresse zeigt.

Von dem MODULE existiert eine Fortran 2003 basierende Alternative `Ex09CInterfacesF2003.f90`, die jedoch derzeit nur mit Intel Visual Fortran 10.1 übersetzt werden kann. Um sie zu nutzen ist im Programm `Ex09FMain` die Zeile 7 auszukommentieren und Zeile 5 zu kommentieren.

12. Ex10_CMain (cpp):
    Das C++ Programm ruft die Funktion
    ```
    extern "C" int* CreateIntArray( int iDim );
    ```
    auf, um deren Funktion in C++ zu demonstrieren.

Index

K

L

M

N

P

R

S

T

U

V

W

Z