DOI:10.13471/j.cnki.acta.snus.2017.04.001

Fortran 与 VB.NET 的混合编程*

胡文清，詹杰民
（中山大学工学院应用力学与工程系，广东 广州 510275）

摘 要：Fortran 语言具有很高的执行效率，广泛地应用于数值计算领域，积累了大量高效可靠的源程序，例如，Microsoft 公司的 IMSL Fortran 程序库，为科学计算提供了强大的工具。VB. NET 是完全面向对象的语言，与 VB 相比，VB. NET 具有快速简易地开发功能更强大的 Windows 程序的优点。剖析了 VB. NET 通过动态链接库模式调用 Fortran 的混合编程，针对变量、字符串、数组、结构体、结构体数组等情况给出对应的调用方法，并给出了典型示例。为开发集合 Fortran 和 VB. NET 优点，兼具高执行效率和快速简易开发能力的软件提供技术支持。

关键词：Fortran；VB. NET；混合编程
中图分类号：TP311 **文献标志码**：A **文章编号**：0529－6579（2017）04－0001－08

Mixed-language programming with Fortran and VB.NET

HU Wenqing, *ZHAN Jiemin*
(Department of Applied Mechanics and Engineering, College of Engineering,
Sun Yat-sen University, Guangzhou 510275, China)

Abstract: Fortran language has high execution efficiency. It is widely used in the field of numerical calculation. And it has accumulated a lot of high-efficient and reliable source codes. For example, The Microsoft IMSL-Fortran Library can give powerful mathematical and statistical analysis. VB.NET is a fully object-oriented language. Compared against VB language, VB.NET is quicker and easier to develop a more powerful Windows program. Mixed-language programming of VB.NET calling Fortran through dynamic link library mode is studied. It gives the corresponding calling method for variable, string, array, structure and structure array. Also the typical examples are offered. It provides technical support for the development of software with both advantages of Fortran and VB.NET: high execution efficiency and quick and easy development ability.

Key words: Fortran; VB.NET; mixed-language programming

Fortran 编程语言于 1954 年由 IBM 工程师开发，是世界上最早出现的计算机高级程序设计语言，广泛应用于科学和工程计算领域。Fortran 语法简明严谨，接近数学公式的自然描述，可以直接对矩阵和复数进行运算，在计算机里具有很高的执行效率。长久以来，Fortran 积聚了一大批高效、可靠、经过考验的函数库、软件包等源程序。然而 Fortran 是面向过程的语言，在设计开发人机交互界面方面能力较弱，不利于图形处理，因此一般采用混合编程的形式，使用其他易于开发软件的编程

* **收稿日期**：2016－07－25
基金项目：广东省协同创新与平台环境建设专项（2014B090904066）
作者简介：胡文清（1987 年生），男；**研究方向**：流体力学；E-mail：emp_beren@163.com
通信作者：詹杰民（1963 年生），男；**研究方向**：流体力学；E-mail：stszjm@mail.sysu.edu.cn

语言开发软件界面，利用 Fortran 编写计算核心程序。

在 Windows XP 和更早的时代，程序员经常采用 VB 编程语言来进行软件开发。Visual Basic 是一种由 Microsoft 公司开发的结构化的、模块化的、面向对象的、包含协助开发环境的事件驱动为机制的可视化程序设计语言。VB 是世界上使用人数最多的开发语言之一。利用 VB 开发人机交互界面[1]，计算核心程序用 Fortran 开发是普遍的选择[2]，VB 和 Fortran 的混合编程方法也已被广泛研究[3-4]。

随着科技的发展，现在的电脑平台普遍从 XP 时代的 32 位 x86 更换至 WIN7 时代的 64 位 x64，XP 时代的开发平台已不能满足需求。在 windows XP 时代，Visual Basic 的版本还是 VB6.0，Fortran 的标准还是 77、90、95 的 CVF，关于这些版本的混合编程的研究已有很多；在 64 位时代，VB 已升级为 VB.NET。2005 年，微软宣布将不会再对非。NET 版本的 VB 进行支持，因此将开发平台升级至 VB.NET 已是大势所趋。与 VB 相比，VB.NET 是完全面向对象的语言，功能更加强大，开发者可以更快速的可视化开发网络应用程序、网络服务、Windows 应用程序和服务器端组件。但是，由于 VB.NET 是基于 .NET 框架的，所开发的程序都被编译成微软中间语言（Microsoft Intermediate Language）的中间代码，然后通过. NET Framework 的公共语言运行库（Common Language Runtime）来执行。在 Visual Studio 2008 以后的版本，Visual Basic. NET 的特性已经与原来的大有不同。由于改动太大，VB.NET 对 VB 的向后兼容并不好。另一方面，Fortran 也从 CVF 变化为 2003、2008 标准的 Intel Fortran。因此 VB 和 Fortran 的混合编程方法并不能直接使用在 VB.NET 上，如何实现 VB.NET 和 Fortran 的混合编程，是本文的研究重点。另外，本文除了对变量、字符串、数组等一般情况的混合编程方法进行了研究外，还对结构体、结构体数组等情况进行了研究，基本包括了 VB.NET 和 Fortran 混合编程中可能遇到的所有情况。

本文研究的开发平台为 64 位 WIN7，Fortran 为 Intel（R）Visual Fortran Compiler Professional 11.1.038 + Microsoft Visual Studio 2008，VB. NET 为 Microsoft Visual Studio 2010。

1 混合编程的调用约定和实现方法

VB. NET 和 Fortran 的调用方法通常有 2 种[5]：第 1 种是 VB. NET 实用 Shell 命令直接调用 Fortran 编译生成的 exe 文件，这种方法适合于计算量较大而交互操作不多的情况。但是这种方法 VB. NET 程序和 Fortran 程序是异步执行的，并不利于数据的实时交互；第 2 种方法是将 Fortran 程序编译成动态链接库（Dynamic Link Library），在 VB. NET 里通过约定的接口动态调用。这种方法又细分为隐式链接和显式链接，隐式链接需要将 Fortran 编译生成的 LIB 文件加载到 VB. NET 的工程中，即可直接调用 Fortran 的 DLL。显式链接只需要 Fortran 编译生成的 DLL 文件，但是需要在 VB. NET 程序中显式声明需要调用的 Fortran 过程和函数。由于隐式链接方法在更新 Fortran 的 DLL 时需要每次都加载新的 LIB 文件，并不方便，因此一般采用显式链接方法来加载 DLL。综上，本文采用 VB. NET 显式链接加载 Fortran 编译生成的 DLL 的方法来实现 Fortran 与 VB. NET 的混合编程。

不同的语言编写的程序之间的过程调用存在较大差异，因此在混合编程时必须保证调用约定的匹配。Fortran 一共有 3 种调用约定：C、STDCALL、Default，VB. NET 的默认调用约定为 WinAPI，在 32 位 Intel 平台上对应于 STDCALL。由于 STDCALL 调用约定用于调用 Win32 API 函数，也是 VB.NET 的默认调用约定，因此在 Fortran 处不采用默认调用约定，而是使用 STDCALL，以符合 Win32 API 规范，这需要在 Fortran 源程序中声明[6-7]。

表 1 STDCALL 调用约定的实现

Table 1 Calling convention of STDCALL

元素	实现
参数传递约定	按值，除非传递指针或引用类型
堆栈维护职责	调用的函数从堆栈中弹出自己的参数
名称修饰约定	下划线（_）是名称的前缀。 名称后跟后面参数列表中的字节数（采用十进制）的符号（@）
外部例程名称大小写约定	小写

VB.NET 通过. NET Framework 的 CLR 来执行，称作托管代码，而 VB 的代码和 Fortran 编译的 DLL 属于非托管代码。因此 VB.NET 与 Fortran 的混合编程方法相对于 VB 时期有了相当的变化，这在本文后面部分再分别说明。

1.1 Fortran 部分实现方法

Fortran 编译 DLL 时，可以采用 function 或者

subroutine 形式。由于 subroutine 可以利用虚参实现多个返回值，因此一般采用 subroutine 形式。以下是示例代码段：

```
subroutine ftest(na,nb,a,b)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES VALUE::na,nb
! DEC$ ATTRIBUTES REFERENCE::a,b
```

其中，! DEC$ ATTRIBUTES 语句用于定义该 DLL 工程的属性。STDCALL 定义了该子过程使用 STDCALL 调用约定，DLLEXPORT 表明该子过程能被外部其他程序或 DLL 调用。ALIAS 属性定义了该子过程的别名，将编译产生的目标例程名限定为引号内的名称，以保持大、小写混合。使用 ALIAS 属性避免了在 VB.NET 的部分必须使用符合 STDCALL 调用约定的例程名称，而是能按程序员个人的意愿决定该子过程的例程名。VALUE 和 REFERENCE 定义了该子过程的参数是按值传递还是按地址传递。STDCALL 调用约定的默认参数传递约定是按值传递，而 VALUE 和 REFERENCE 属性的定义会覆盖调用约定对参数传递产生的影响，因此推荐将全部参数都显式定义其参数传递方式。

1.2　VB.NET 部分实现方法

在 VB.NET 里，声明外部过程的语句如下：

Declare [Ansi|Unicode|Auto] Sub|Function 〈名称〉 Lib "〈库〉" [Alias "〈别名〉"] [([argumentList])]

对于 1.1 节定义的子过程，假设该 DLL 工程编译的 DLL 为 ftest. dll，该 DLL 文件放在工程运行目录下，则 VB.NET 里使用 Declare 语句定义该过程，Call 语句调用。以下是示例代码段：

```
Declare Ansi Sub FTest Lib "ftest. dll" (ByVal na As Integer, ByVal nb As Integer, ByRef a As Double, ByRef b As Long)
Call FTest(na, nb, a, b)
```

其中，Ansi 用于指定 VB.NET 将所有的字符串封送为 ANSI 值。由于 VB.NET 的字符串使用 Unicode 编码，而 Fortran 的字符串使用 ANSI 编码，因此该处指定封送编码。Sub 语句指定该外部过程为过程。FTest 为过程的名称，与 1.1 节里 ALIAS 属性定义的别名一致。〈库〉指定引用 DLL 的路径，这里使用的是相对路径。Alias 指定在 VB.NET 里的别名，并非必须。argumentList 指定该过程的参数属性，ByVal 为按值传递，ByRef 为按地址传递。VB.NET 的参数属性设置必须与 Fortran 的一致。

2　变量传递

VB.NET 与 Fortran 的变量传递可以使用 ByVal 按值传递，也可以使用 ByRef 按地址传递。相比于 VB，VB.NET 对于 Integer 和 Long 的定义作了修改，因此程序员不能套用旧的 VB 定义。VB.NET 与 Fortran 的常用数据类型对应关系如表 2 所示[8-9]。

表 2　数据类型对应关系

Table 2　Data types correspondence

Fortran 数据类型	VB.NET 数据类型
INTEGER（1）	SByte
INTEGER（2）	Short
INTEGER（4）	Integer
INTEGER（8）	Long
REAL（4）	Single
REAL（8）	Double
REAL（16）	无

以下是示例代码段：

```
Fortran
subroutine ftest(na,nb,a,b)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES VALUE::na,nb
! DEC$ ATTRIBUTES REFERENCE::a,b
REAL(8)::a
INTEGER(8)::b
INTEGER(4):: na,nb
VB.NET
Declare Ansi Sub FTest Lib "ftest. dll" (ByVal na As Integer, ByVal nb As Integer, ByRef a As Double, ByRef b As Long)
Dim a As Double
Dim b As Long
Dim na, nb As Integer
Call FTest(na, nb, a, b)
```

3　字符串传递

Fortran 的字符串传递时默认会在参数的最后附加一隐藏参数，该参数是字符串的长度，按值传递。通过设置调用约定和 VALUE/REFERENCE 属性可以改变传递方式。表 3 为详细情况[8]。

VB.NET 的字符串类型实际是以包含长度和地址信息的结构体形式存储的。因此，VB.NET 和 Fortran 的字符串传递必须以以下形式实现：

表 3 Fortran 传递字符串的 ATTRIBUTE 属性效果

Table 3 Effect of ATTRIBUTE options on character strings passed as arguments in Fortran

参数	Default	C	STDCALL
字符串	按地址传递，有字符串长度参数	第一个字符转换成 INTEGER(4)，按值传递	第一个字符转换成 INTEGER(4)，按值传递
VALUE 属性的字符串	错误	第一个字符转换成 INTEGER(4)，按值传递	第一个字符转换成 INTEGER(4)，按值传递
REFERENCE 属性的字符串	按地址传递，可能有字符串长度参数	按地址传递，没有字符串长度参数	按地址传递，没有字符串长度参数

VB.NET 使用按值传递，Fortran 使用按地址传递。VB.NET 使用按值传递将间接引用存储字符串的结构体，实际上传递的只有字符串的地址，与 Fortran 所期望的一致。

在 VB.NET 里，字符串变量使用 String 定义，不能直接定义定长字符串，字符串长度在赋值后确定。同时，VB.NET 字符串在最后会增加一个 NULL 字符作为结束标志，因此 VB.NET 字符串的总长度为字符数加一。混合编程时应先对 VB.NET 的字符串赋初值，使其长度与 Fortran 定长字符串一致。

Fortran 里汉字的存储是两字节的 ANSI 码，因此 VB.NET 里需要声明 Ansi 属性。Fortran 里统计字符串长度时一个汉字占用 2 个长度，但是在 VB.NET 里，统计字符串长度时一个汉字只占 1 个长度。这点程序员编写程序时应注意。

以下是示例代码段：

```
Fortran
subroutine ftest(chtt,chtt2)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES REFERENCE::chtt,chtt2
character*5::chtt,chtt2
do i=1,len_trim(chtt)
chtt2(i:i)=chtt(len_trim(chtt)-i+1:len_trim(chtt)-i+1)
end do
VB.NET
Declare Ansi Sub FTest Lib "ftest.dll" (ByVal chtt As String, ByVal chtt2 As String)
Dim chtt, chtt2 As String
chtt = "a 中 cd"
chtt2 = "aaaaa"
Call FTest(chtt, chtt2)
```

该代码段的结果为 chtt =” a 中 cd”，chtt2 =” dc 兄 a”，观察汉字的 GBK 码（中 D6D0，兄 D0D6）可证上述结论。

4 数组传递

VB.NET 和 Fortran 的数组传递只能使用按地址传递的方式。实际传递时 VB.NET 需要传递数组的第一个元素，按地址传递时这将传递给 Fortran 数组的开始地址，正如 Fortran 所需。

以下是示例代码段：

```
Fortran
subroutine ftest(c, ni, nj, nk, nl)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES VALUE::ni, nj, nk, nl
! DEC$ ATTRIBUTES REFERENCE::c
integer(4)::ni, nj, nk, nl
real(4),dimension(1:ni,1:nj,1:nk,1:nl)::c
VB.NET
Declare Ansi Sub FTest Lib "ftest.dll" (ByRef c As Single, ByVal ni As Integer, ByVal nj As Integer, ByVal nk As Integer, ByVal nl As Integer)
Dim ni, nj, nk, nl As Integer: Dim c(,,,) As Single
ni = 5: nj = 4: nk = 3: nl = 2
ReDim c(0 To nl-1, 0 To nk-1, 0 To nj-1, 0 To ni-1)
Call FTest(c(0, 0, 0, 0), ni, nj, nk, nl)
```

可以见到 VB.NET 构造数组与 Fortran 有相当大的不同。以下两节将做出详细描述。

4.1 构造 VB.NET 数组

VB.NET 的数组下标必须是 0，不同于 VB，VB 可以使用 option base 语句来修改下标为 0 或 1。尽管如此，VB.NET 仍然可以构建下标不为 0 的数组。VB.NET 采用 CLR 来执行，而 CLR 数组可以使用 Array.CreateInstance 方法来指定数组创建的维数和每维的起始索引值。示例如下：

```
Dim a(,,,), b(,,,) As Single
```

```
ReDim a(0 To nl - 1, 0 To nk - 1, 0 To nj - 1, 0 To ni - 1)
b = Array. CreateInstance( GetType( Single), {nl, nk, nj, ni}, {1, 1, 1, 1})
```

a 和 b 数组均为类型相同的四维数组，每维的大小相等，区别只是起始索引值不同，a 从 0 开始，b 从 1 开始。Array. CreateInstance 方法的第 1 个属性指定数组类型，第 2 个属性指定每维的大小，第 3 个属性指定每维的起始索引值。但是，这种方法不适用于一维数组。

```
Dim c(), d() As Single: ReDim c(0 To ni - 1)
d = Array. CreateInstance(GetType(Single), {ni}, {1})
```

这里 c 和 d 并不是类型相同的数组，c 的类型为 single [ ]，而 d 的类型为 single [*]。这是因为 CLR 中有一种数组是被特殊照顾的，那就是以 0 开始的一维数组，这种数组也被称为“Vector”。Vector 数组的类型为 xxx [ ]，Array. CreateInstance 方法构造的一维数组类型为 xxx [*]。因此，当使用一维数组混合编程时，VB.NET 里只能以 0 开始，程序员需要注意和 Fortran 的不同；使用多维数组时，则可以使用 Array. CreateInstance 方法构造和 Fortran 数组起始索引值一致的数组。

4.2　VB.NET 数组和 Fortran 数组的差异

习惯上，我们将数组前面的维度称为列，将后面的维度称为行。Fortran 数组存储时先改变前面的维度，因此，习惯称之为“列优先”存储。而 VB.NET 数组与之相反，为“行优先”存储。举例来说，VB.NET 的数组 arr（4，3，2，1）与 Fortran 的 arr（2，3，4，5）维度、大小均一致。

以下是示例代码段：

```
Fortran
subroutine ftest(c,d, ni, nj, nk, nl)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES VALUE::ni, nj, nk, nl
! DEC$ ATTRIBUTES REFERENCE::c, d
integer(4)::ni, nj, nk, nl
real(4),dimension(1:ni,1:nj,1:nk,1:nl)::c,d
do l=1,nl
do k=1,nk
do j=1,nj
do i=1,ni
d(i,j,k,l) =c(i,j,k,l) + i + nj * (j - 1) + nj * nk * (k - 1) + nj * nk * nl * (l - 1)
end do
end do
end do
end do
VB.NET
Declare Ansi Sub FTest Lib "ftest. dll" (ByRef c As Single, ByRef d As Single, ByVal ni As Integer, ByVal nj As Integer, ByVal nk As Integer, ByVal nl As Integer)
Dim ni, nj, nk, nl As Integer: Dim c(,,,), d(,,,) As Single
ni = 5: nj = 4: nk = 3: nl = 2
ReDim c(0 To nl - 1, 0 To nk - 1, 0 To nj - 1, 0 To ni - 1)
ReDim d(0 To nl - 1, 0 To nk - 1, 0 To nj - 1, 0 To ni - 1)
For l = 1 To nl
For k = 1 To nk
For j = 1 To nj
For i = 1 To ni
c(l - 1,k - 1, j - 1,i - 1) = i + nj * (j - 1) + nj * nk * (k - 1) + nj * nk * nl * (l - 1)
Next
Next
Next
Next
Call FTest(c(0, 0, 0, 0), d(0, 0, 0, 0), ni, nj, nk, nl)
```

该代码段的 VB.NET 部分对 c 按照“行优先”赋值，Fortran 部分对 d 按照“列优先”赋值，并且加上 c 的对应元素的值。结果 d 的每个元素的值均为 c 的对应元素的值的两倍，证明了 VB.NET 数组与 Fortran 数组的差异。

5　结构体传递

结构体能够实现复杂的数据结构，在实际应用中经常使用到。因此 VB.NET 与 Fortran 混合编程的结构体传递问题是研究的重点。在 VB.NET 里使用 Structure 语句定义结构体，Fortran 里使用 type 语句定义。VB.NET 里结构体的定义方法与 VB 相比有了很大的变化。VB.NET 与 Fortran 使用结构体传递需要注意以下 2 个问题：① VB.NET 属于托管代码，定义的结构体中的数组成员不能声明初始大小，需要在结构体里构造初始化函数重定义数组大小；② 无论是 VB.NET 还是 Fortran，结构体中的成员在存储结构上是自然对齐的，未必按照声明次序来存储。

为解决这 2 个问题，需要在 VB.NET 里引入 System. Runtime. InteropServices 命名空间。利用此命名空间的 MarshalAs 属性，可以指定如何在托管内存与非托管内存之间封送数据，从而解决问题。

以下是示例代码段：

```
Fortran
subroutine ftest(tt)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
```

```
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES REFERENCE::tt
type test
sequence
real(8)::cc
integer(8)::iii
real(4),dimension(0:3)::ccc
character*12::str
integer(4)::abc
integer(4)::cba
end type
type(test)::tt
VB.NET
Imports System.Runtime.InteropServices
Declare Ansi Sub FTest Lib "ftest.dll" (ByRef tt As test)
<StructLayout(LayoutKind.Sequential, CharSet:=CharSet.Ansi, pack:=4)> Public Structure test
Dim cc As Double: Dim iii As Long
<MarshalAs(UnmanagedType.ByValArray, sizeconst:=(4))> Dim ccc() As Single
<MarshalAs(UnmanagedType.ByValTStr, SizeConst:=12)> Dim str As String
Dim abc As Integer: Dim cba As Integer
Public Sub int()
ReDim ccc(0 To 3)
End Sub
End Structure
Dim tt As test: tt.int(): Call FTest(tt)
```

如以上代码段所示，为避免问题2，必须在VB.NET和Fortran都声明为顺序存储。Fortran的实现方法是在结构体定义里加入SEQUENCE属性，VB.NET则是使用StructLayout属性定义。LayoutKind.Sequential规定了结构体成员按声明顺序存储。CharSet属性设置结构体采用ANSI编码，原因见第3部分。由于Fortran的默认对齐长度为4字节，因此VB.NET需要使用pack属性设置结构体存储时按4字节对齐。

由于VB.NET字符串不能声明初始长度，结构体里数组也不能声明初始大小，因此需要采用MarshalAs属性指明封送数据的大小。其中UnmanagedType.ByValArray作用于数组，UnmanagedType.ByValTStr作用于字符串，SizeConst属性指定大小。为实际分配空间，需要在结构体里构造初始化函数int()，先初始化后再Call调用。

利用MarshalAs属性可以为结构体的数组成员指明大小，但是这种方法只适用于一维数组。为了构造VB.NET结构体的多维数组成员，可行的方法是将多维数组拆分为多个一维数组，这样就可以为每个维度数组设置MarshalAs属性。

以下是示例代码段：

```
Fortran
subroutine ftest(tt)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES REFERENCE::tt
type test
sequence
real(4),dimension(1:6,1:2)::ddd
end type
type(test)::tt
VB.NET
Imports System.Runtime.InteropServices
Declare Ansi Sub FTest Lib "ftest.dll" (ByRef tt As test)
<StructLayout(LayoutKind.Sequential, CharSet:=CharSet.Ansi, pack:=4)> Public Structure test
<MarshalAs(UnmanagedType.ByValArray, sizeconst:=(2))> Dim ddd() As structest
Public Sub int()
ReDim ddd(0 To 1)
For i = 0 To 1
ReDim ddd(i).ddda(0 To 5)
Next
End Sub
End Structure
<StructLayout(LayoutKind.Sequential)> Public Structure structest
<MarshalAs(UnmanagedType.ByValArray, sizeconst:=(6))> Dim ddda() As Single
End Structure
Dim tt As test: tt.int():Call FTest(tt)
```

如以上代码段所示，将二维数组拆分为2个一维数组，则可分别为每个维度设置MarshalAs属性指明大小。需要注意的是，多维数组拆分时仍需注意第4部分所述的Fortran与VB.NET的存储顺序差异，保证Fortran遵照“列优先”存储，VB.NET遵照“行优先”存储。

6 结构体数组传递

VB.NET与Fortran的结构体数组传递不能使用第4部分数组传递的方法，使用普通数组传递的方法时结构体数组只能传递第一个数组元素。这是因为VB.NET属于托管代码，是CLR自动分配内存地址和垃圾回收的。而Fortran属于非托管代码，VB.NET与Fortran进行结构体数组传递时，托管对象可能已经被垃圾回收器回收了，就会出问题。这

里就需要使用 System. Runtime. InteropServices 命名空间的 GCHandle 类，该类提供用于从非托管内存访问托管对象的方法，利用其 Alloc 方法将托管对象设置为 Pinned 类型，固定对象的内存地址，从而防止垃圾回收器移动对象回收掉。最后结束时使用 Free 方法释放已分配的句柄。

以下是示例代码段：

```
Fortran
subroutine ftest(ta,na,nb)
! DEC$ ATTRIBUTES STDCALL,DLLEXPORT::ftest
! DEC$ ATTRIBUTES ALIAS:"FTest"::ftest
! DEC$ ATTRIBUTES REFERENCE::tt
type test
sequence
real(8)::cc
integer(8)::iii
real(4),dimension(0:3)::ccc
character*12::str
real(4),dimension(1:6,1:2)::ddd
integer(4)::abc
integer(4)::cba
end type
type(test),dimension(1:na,1:nb)::ta
integer(4 ):: na,nb
VB.NET
Imports System. Runtime. InteropServices
Declare Ansi Sub FTest Lib "ftest. dll" (ByVal pt As IntPtr, ByVal na As Integer, ByVal nb As Integer)
<StructLayout(LayoutKind. Sequential, CharSet:=CharSet. Ansi, pack:=4)> Public Structure test
Dim cc As Double: Dim iii As Long
<MarshalAs(UnmanagedType. ByValArray, sizeconst:=(4))> Dim ccc() As Single
<MarshalAs(UnmanagedType. ByValTStr, SizeConst:=12)> Dim str As String
<MarshalAs(UnmanagedType. ByValArray, sizeconst:=(2))> Dim ddd() As structest
Dim abc As Integer: Dim cba As Integer
Public Sub int()
ReDim ccc(0 To 3): ReDim ddd(0 To 1)
For i = 0 To 1
ReDim ddd(i). ddda(0 To 5)
Next
End Sub
End Structure
<StructLayout(LayoutKind. Sequential)> Public Structure structest
<MarshalAs(UnmanagedType. ByValArray, sizeconst:=(6))> Dim ddda() As Single
End Structure
Dim na, nb As Integer: Dim ta(,) As test
Dim ipt As IntPtr: Dim pt As IntPtr: ReDim ta(0 To nb-1, 0 To na-1)
For l = 0 To nb -1
For k = 0 To na-1
ta(l, k). int()
Next
Next
Dim tryon() As Byte: Dim gc As GCHandle
ReDim tryon(Marshal. SizeOf(GetType(test))*na*nb-1)
gc = GCHandle. Alloc(tryon, GCHandleType. Pinned): pt = gc. AddrOfPinnedObject
For j = 0 To nb-1
For i = 0 To na-1
ipt = pt. ToInt32 + Marshal. SizeOf(GetType(test))*i + Marshal. SizeOf(GetType(test))*j*na
Marshal. StructureToPtr(ta(j, i), ipt, True)
Next
Next
Call FTest(pt, na, nb)
For j = 0 To nb-1
For i = 0 To na-1
ipt = pt. ToInt32 + Marshal. SizeOf(GetType(test))*i + Marshal. SizeOf(GetType(test))*j*na
ta(j, i) = Marshal. PtrToStructure(ipt, GetType(test))
Next
Next
gc. Free()
```

如以上代码段所示，VB.NET 和 Fortran 进行结构体数组传递时需要注意以下 3 点：

1）使用 GCHandle 类来传递结构体数组时，需要使用 Byte 数组来作数据存储。这是因为来用于固定数组的 GCHandle 的 Alloc 方法不能以结构体作为参数，因此需要利用 Byte 数组作媒介。使用时需要将该 Byte 数组的大小重定义与结构体数组的大小严格一致。单个结构体的大小使用 Marshal 类的 SizeOf 语句获得。

2）使用 IntPtr 类型作为该 Byte 数组的指针来作为函数参数传递，这点与单一结构体传递时直接将结构体作为函数参数不同。IntPtr 用于表示指针或句柄的平台特定类型，使用 GCHandle 的 AddrOfPinnedObject 属性将 IntPtr 变量指向该 Byte 数组的内存首地址。利用 Marshal 的 StructureToPtr 方法通过 IntPtr 变量给 Byte 数组赋值，同样调用后需要用 Marshal 的 PtrToStructure 方法通过 IntPtr 变量将 Byte 数组的值传递回目标结构体数组中。

3）无论是给 Byte 数组赋值时还是将 Byte 数

组的值传递回目标结构体数组，均需要使用循环结构对目标结构体数组的每一个成员进行操作。这里仍然需要注意本文第4部分所述的Fortran与VB.NET的存储顺序差异，保证Fortran遵照“列优先”存储，VB.NET遵照“行优先”存储。

7 总 结

本文详细介绍了Fortran与VB.NET在64位Windows平台上混合编程的方法，并给出了相应的示例。通过对变量、字符串、数组、结构体、结构体数组等情况的混合编程方法的研究，基本解决了在Fortran和VB.NET混合编程中可能出现的各种问题，从而为在64位平台上，以DLL为纽带充分发挥Fortran强大的计算能力和VB.NET的可视化程序开发提供了坚实的技术基础。

参考文献：

[1] 陈晓翔,柯栋,李芳,等. 非标准格式地理空间数据的GIS方式直接读取研究[J]. 中山大学学报(自然科学版), 2002, 41(2): 117－118.
CHEN X X, KE D, LI F, et al. A study of GIS－based direct reading of nonstandard geographic spatial data [J]. Acta Scientiarum Naturalium Universitatis Sunyatseni, 2002, 41(2): 117－118.

[2] 刘祚秋,温少荣,周翠英,等. 桩、土、刚性承台相互作用下桩基内力计算新方法[J]. 中山大学学报(自然科学版), 2004, 43(4): 33－37.
LIU Z Q, WEN S R, ZHOU C Y, et al. The new computing method of internal force of piles under the Interaction of piles, soils and rigid cap [J]. Acta Scientiarum Naturalium Universitatis Sunyatseni, 2004, 43(4): 33－37.

[3] 欧阳永忠,王瑞,陆秀平,等. VC、VB与FORTRAN的混合编程技术及其实现[J]. 海洋测绘, 2004, 24(1): 54－59.
OUYANG Y Z, WANG R, LU X P, et al. On mixed－language programming and realization with VC, VB and Fortran [J]. Hydrographic Surveying and Charting, 2004, 24(1): 54－59.

[4] 李学哲, 白云, 陈国新. Fortran 90与VB混合编程技术的研究与实现[J]. 苏州科技学院学报(工程技术版), 2008, 21(4): 76－80.
LI X Z, BAI Y CHEN G X. The research and implementation of VB and Fortran90 mixed－language programming technology [J]. J of University of Science and Technology of Suzhou (Engineering and Technology), 2008, 21(4): 76－80.

[5] 何萌, 柴军瑞. VB与FORTRAN混合编程的两种方法及其比较[J]. 水电能源科学, 2005(1): 60－62.
HE M, CHAI J R. Two kinds of methods of mixing VB and Fortran in one program and their comparison [J]. Water Resources and Power, 2005(1): 60－62.

[6] 毕苏萍, 周振红. Visual Fortran创建Win32 API式的DLL[J]. 计算机工程与设计, 2008, 29(18): 4868－4871.
BI S P, ZHOU Z H. Creating DLL in accordance with win32 API in visual Fortran [J]. Computer Engineering and Design, 2008, 29(18): 4868－4871.

[7] 颜国红, 周振红. 工程计算平台CVF的应用接口资源扩充[J]. 长江科学院院报, 2008, 25(4): 106－110.
YAN G H, ZHOU Z H. How to extend API in engineering computation platform of CVF [J]. Journal of Yangtze River Scientific Research Institute, 2008, 25(4): 106－110.

[8] INTEL CORP. Intel Fortran compiler 11.1 user and reference guides [M]. USA: Intel Corporation, 2009.

[9] MICROSOFT CORP. Microsoft visual studio 2010 documentation [M]. USA: Microsoft Corporation, 2010.