An Introduction to Fortran 90/95/2003

Xinyu Wen
xwen@pku.edu.cn
Atmospheric and Oceanic Sciences, Peking University

Sep. 2010

编程语言的竞争(TIOBE)

Position Sep 2010	Position Sep 2009	Delta in Position	Programming Language	Ratings Sep 2010	Delta Sep 2009	Status
1	1	=	Java	17.915%	-1.47%	A
2	2	=	C	17.147%	+0.29%	A
3	4	↑	C++	9.812%	-0.18%	A
4	3	↓	PHP	8.370%	-1.79%	A
5	5	=	(Visual) Basic	5.797%	-3.40%	A
6	7	↑	C#	5.016%	+0.83%	A
7	8	↑	Python	4.583%	+0.65%	A
8	18	↑↑↑↑↑↑↑↑↑	Objective-C	3.368%	+2.78%	A
9	6	↓↓↓	Perl	2.447%	-2.08%	A
10	10	=	Ruby	1.907%	-0.47%	A
11	9	↓↓	JavaScript	1.665%	-1.33%	A
12	11	↓	Delphi	1.585%	-0.39%	A
13	13	=	Lisp	1.084%	+0.24%	A--
14	12	↓↓	Pascal	0.790%	-0.17%	A--
15	27	↑↑↑↑↑↑↑↑↑↑↑↑	Transact-SQL	0.771%	+0.40%	A--
16	-	↑↑↑↑↑↑↑↑↑↑↑↑	Go	0.728%	+0.73%	A--
17	21	↑↑↑↑	RPG (OS/400)	0.715%	+0.26%	A--
18	30	↑↑↑↑↑↑↑↑↑↑↑↑	PowerShell	0.686%	+0.42%	B
19	24	↑↑↑↑↑	Ada	0.676%	+0.29%	B
20	14	↓↓↓↓↓↓	PL/SQL	0.637%	-0.18%	A-

Position	Programming Language	Ratings
21	SAS	0.635%
22	NXT-G	0.612%
23	MATLAB	0.605%
24	Lua	0.555%
25	ABAP	0.546%
26	Scheme	0.516%
27	Fortran	0.463%
28	Alice	0.443%
29	Logo	0.436%
30	D	0.414%
31	Tcl	0.413%
32	C shell	0.402%
33	COBOL	0.397%
34	Standard ML	0.392%
35	ActionScript	0.351%
36	Scratch	0.343%
37	Smalltalk	0.334%
38	CL (OS/400)	0.333%
39	R	0.329%
40	Visual Basic .NET	0.328%
41	Modula-2	0.321%
42	Haskell	0.315%
43	Prolog	0.290%
44	Objective Caml	0.280%
45	APL	0.274%
46	F#	0.274%
47	Forth	0.270%
48	IDL	0.253%
49	C++/CLI	0.250%
50	Scala	0.243%

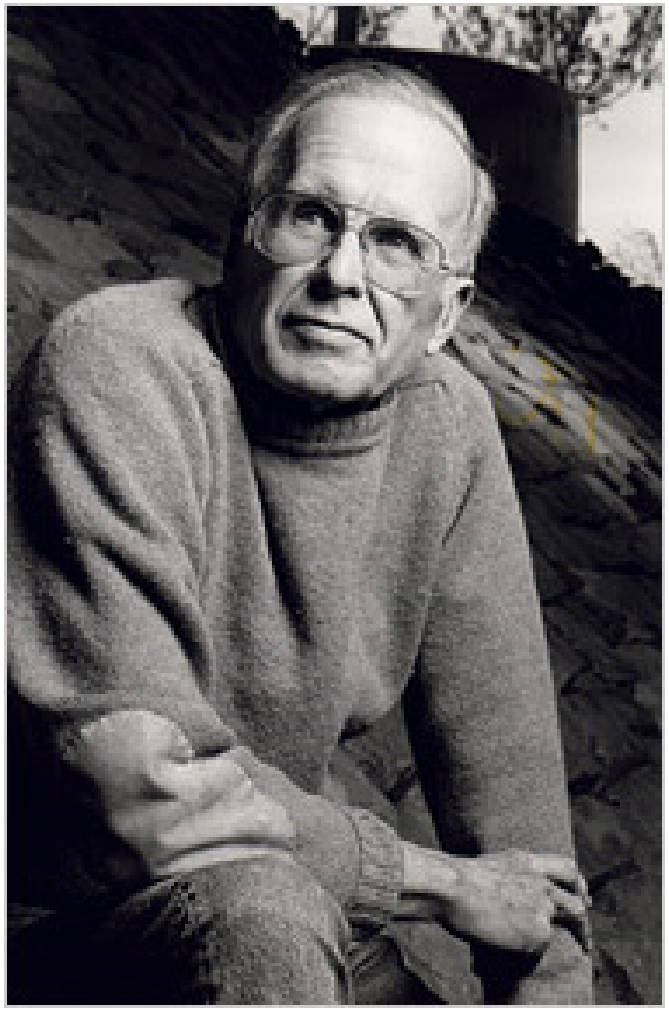
History of Fortran

- 1957: 美国IBM公司工程师John Backus根据汇编语言改进发明FORmula TRANslator，在IBM704机器上实现Fortran I，包含32种语句
- 1958: 引进subroutine，推出商业版Fortran II
- 1960: 设计Fortran III，存在严重缺陷，依赖机器硬件，不易移植，未实际发布和运行
- 1962: 推出Fortran IV，向前不兼容性，应用受限；引入逻辑运算
- 1966: ANSI推出Fortran 66，解决兼容性问题
- 1978: ANSI推出Fortran 77，引入结构化程序思想（60年代末的思潮，如Pascal、C等）

History of Fortran

- 1991: ANSI推出**Fortran 90**，引入现代的结构化、模块化设计思想，但同时保留了所有之前的冗余特性，如goto/common/block/等
- 1995: ANSI推出**Fortran 95**，主要引入forall和where等语句，为SM型并行计算打基础
- 2003: ISO推出**Fortran 2003**，主要引入OOP，现在只有Sun的编译器支持此标准
- 2008: ISO修正F2003几个小问题，推出Fortran 2008

John Backus （1924-2007）



- 30岁在IBM发明Fortran语言，这是世界进入计算机时代后第一个高级语言，在此之前，人们只能使用复杂的汇编语言进行编程。他改变了人机“交互方式”。
- 美国计算机学会评奖委员会联合主席吉姆·霍宁评价说，Fortran语言的诞生是计算机领域的一个巨大突破，它改变了“游戏方式”，“而这种性质的改变在计算机业的历史上仅有过两三次。”
- 1977年Backus因为Fortran的发明获得图灵奖（计算机界最高奖）
- 1991年从IBM退休，他一生都为IBM服务。2007年在家中病逝。

Features of Fortran 90/95/2003

- 完整的结构化和模块化
- 矩阵运算
- 简单的子程序接口，方便传递矩阵
- 功能强大而简单的Namelist输入输出
- 对并行计算提供特别支持
- 编译代码执行效率高
- 冗余特性太多，混合代码难于理解
- 内置函数较少
- 大小写不敏感
- 数组在内存中左侧有限
- 图形和字符功能有限

程序是思想的延伸

1. 逻辑的训练场
2. 开放的心态
3. 每一行都是你的杰作

Linux环境下使用Fortran

- 编译器：gfortran
  GNU GCC系列，支持Fortran95，部分支持Fortran2003，免费
- 文本编辑器：vi, emacs, nano
- 编译：
  gfortran –c module.f95
  gfortran –c main.f95
  gfortran –c io.f95 –I/usr/local/netcdf/include
  gfortran –o run module.o main.o
  gfortran –o run *.o –L/usr/local/netcdf/lib –lnetcdf
- 运行：
  ./run
  ./run < namelist.in

Fortran 95: 格式

- 自由格式
- 大小写不分
- &为换行连写
- !为注释语句
- 程序中任何地方用stop可推出程序
- 建议每个域（program、module、function、subrouting）第一行都写implicit none

Fortran 95: 基本数据类型和运算符

- integer: 234 2356323_long 23_short
- real: 0.234 .34 00.30_double .123_quad
- Character: ‘a’ “Xinyu Wen”
- complex: (3.14_double, -7) (1,-1)
- logical: .true. .false.

所有REAL型变量或常量，请**务必写小数点**！显式地标识它是real而不是integer

- 数学运算：+ - * / **
- 逻辑运算：< <= == /= >= > .not. .and. .or.
- 字符运算：//

Fortran 95: 变量声明

- Integer :: a,b,c
- Integer, dimension(100) :: xarray1d
- Real, dimension(100,20) :: xarray2d
- Real, parameter :: pi = 3.1415926
- Character(len=20) :: name = “George W. Bush”
- Character(len=20), dimension(10) :: name10boys
- Logical :: onoff = .TRUE.
- Integer, dimension(3) :: temp = (/ 24, 28, 30 /)
- Character(len=10), dimension(2) :: names = (/ “Tom”, “Mike” /)

数组下标默认是1，不是0

Fortran 95: 控制结构

- If (a==b) then
  …
  else
  …
  end if
- important: if (a/=b) then
  …
  else
  …
  end if important

注意：

1. **条件不要是两个real变量相等的判断，会有微小错误，改为两变量之差小于一个小量**
2. 条件表达式用括号括起来
3. 中途可用exit推出if结构
4. 太长的if结构，前面可标名称

Fortran 95: 控制结构

- Where (SST<0) Ice=1
- Where (SST<0)
  ice = 1
  elsewhere
  ice = 0
  end where

注意：

1. **Where判断只适用于完全相同的数组（等位数组）**
2. 此语句只用于简化对等位数组的判断和赋值操作

Fortran 95: 控制结构

- Do
  …
  if (…) then exit
  …
  end do
- Do I = 100, 2, -2
  …
  end do
- Do while (…)
  …
  end do

注意：

1. 无限循环一定要有exit出口
2. 循环可以用cycle强行进入下一轮循环；用exit强行推出当前循环
3. 大型循环体前面可标名称
4. **Do循环不要用real型变量作循环变量**

Fortran 95: 函数和子程序

- Function add(a,b)
  implicit none
  real, intent(in) :: a, b
  real :: add
  add = a + b
  end function

  主程序调用：
  b = 2.0
  x = add(23.1,b)

Note：

1. **所有输入的变量都用intent(in)或intent(inout)显式限定，说明改动不改动**
2. 返回变量只能是单一变量，也要说明类型，但不用制定intent
3. 保持function的独立性是非常重要的
4. 主程序中用一般函数方法调用

Fortran 95: 函数和子程序

- subroutine add(a,b,x)
  implicit none
  real, intent(in) :: a, b
  real, intent(out) :: add
  add = a + b
  end subroutine

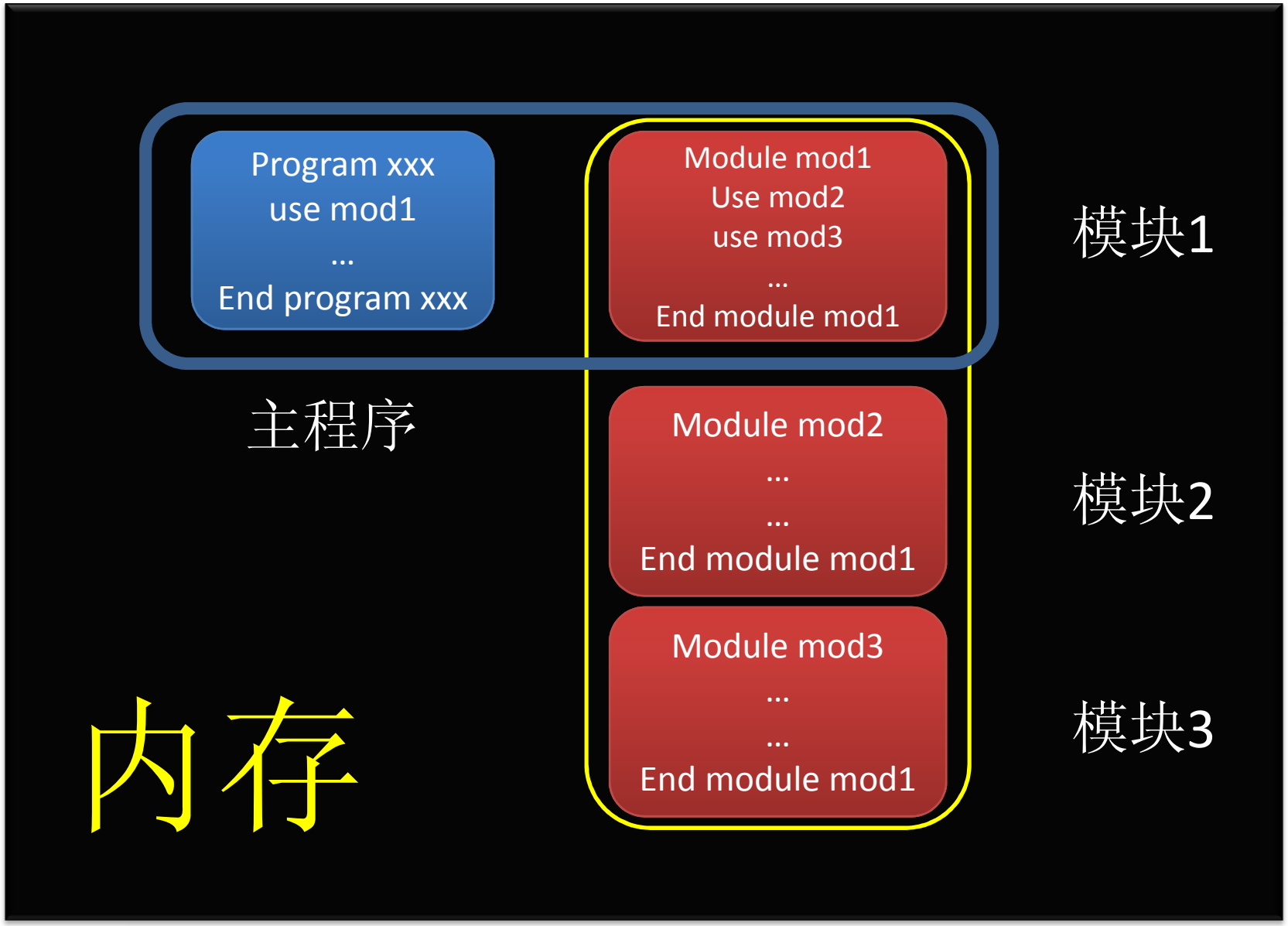
  主程序调用：
  b = 2.0
  x = 10
  call add(23.1,b,x)

Note：

1. **所有输入的变量都用intent(in)或intent(inout)或intent(out)显式限定，说明传来的参数用不用？改动不改动？**
2. 可传递和修改大规模数组
3. 保持subrouting的独立性非常重要
4. Subrouting一般完成一些比function复杂得多的独立任务
5. 主程序中用call调用

Fortran 95: 主程序与模块

- Fortran程序主要由一个主程序program，和若干个module组成（也可以没有module）
- program和各个module，都在内存中占据独立空间。
- 当program调用某个module，或一个module调用另一个module时，实际是把对方的内存空间包进了自己的内存空间。
- Module是Fortran90最大的贡献，极大地解放了大规模数据和子程序/函数的共享，提高协同编程的效率和有效性。


Program xxx
use mod1
…
End program xxx
主程序
Module mod1
Use mod2
use mod3
…
End module mod1
Module mod2
…
…
End module mod1
Module mod3
…
…
End module mod1
模块1
模块2
模块3
内存

Fortran 95: 主程序与模块

- Program hello
  use add
  implicit none
  real :: a=2.0, b=2.3, x

  call add(a,b,x)
  end program hello

- Module add
  implicit none
  subroutine add(a,b,c)
  implicit none
  real, intent(in) :: a, b
  real, intent(out) :: c
  c = a+b
  end subroutine
  end module add

Fortran 95: I/O

简单的I/O

- Read *, a, b
- Print *, a, b, “hello”

复杂的格式化I/O

- Read “(f5.1, f8.2)”, x1, x2
- Print “(f5.1, f8.2)”, x1, x2

Note:

1. 若配合文件读写，则需要制定“设备”，则要用
   read(unit=1,fmt=*) a, b, c
   write(unit=2,fmt=“(f3.1,i5,a5)”) x,I,name
2. Fortran的格式字符串很灵活，请参加其它资料
3. **请不要用Fortran77中的format语句！严重降低程序可读性**

Fortran 95: I/O

- Open (unit=1, file="data.txt")
  read(unit=1,fmt=*) a, b, c
  close(unit=1)

Note:
1. 文本文件的读写很容易
2. 二进制Binary的文件读写相对复杂，请参考其它资料

- Open(unit=99, file="data.txt", action="write", status="new")
  write(unit=99, fmt=*) a, b, c
  close(unit=99)

Fortran 95: namelist读写机制

- Fortran90提供了特殊的读写机制：namelist
- 通过namelist文本文件，方便参数读取
- Fortran95改进了此机制，使得namelist进一步完善，目前很多大气模式都是用这种方法向主程序传递众多参数
- 灵活地使用namelist机制，很方便我们调整程序，减少读写文件造成的错误

Fortran 95: namelist读写机制

- Real :: a, b
  namelist /para/ a, b
  read(*, nml=para)
  print *, a, b

Namelist内容

&para
A = 234.2
B = 12.2
/

- Real :: a=3.4, b=12.0
  character(len=15) :: name="John Backus"
  namelist /info/ a, b, name
  write(*, nml=info)

Note:

1. Namelist语句位置和变量定义的位置一样，不能写在其它位置上
2. Namelist文件中不能有多出来的变量，但可以少于需求的变量

编程实践

- 在三维空间里，积分如下常微分方程：

  dx/dt = sigma (y-x)
  dy/dt = rho x - y - xz
  dz/dt = xy - beta z

  注意使用module，进行模块化编程，结果输出到文件中，分三列分别代表x, y, z，用gnuplot查看结果
- Gnuplot>splot “result.txt” with linespoints