Fortran 90 Reading Notes

Dong Miaowen

2020.06.23-

Contents

1 第一章略 **2**
1.1 章节标题 2
1.1.1 小标题 2

2 First steps in Fortran 90 programming **3**
2.1 From problem to program in three basic steps 3
2.2 Some basic Fortran 90 concepts 3
2.3 Running Fortran programs on a computer 4
2.4 Errors in programs 4
2.5 The design and testing of programs 4
2.6 The old and new Fortran 90 source forms 4
2.7 Program Exercises 5

3 Essential data handling **6**
3.1 The two fundamental types of numbers 6
3.2 REAL and INTERGER varibles 6
3.3 Arithemtic expressions and assingment 6
3.4 List-directed input and output of numeric data 7
3.5 Handling CHARACTER data 9
3.6 Initial values and constants 9
3.7 Creating your own data types 10
3.8 Obsolete forms of declaration, initialization and constant definition 10

Chapter 1

第一章略

1.1 章节标题

1.1.1 小标题

Chapter 2

First steps in Fortran 90 programming

2.1 From problem to program in three basic steps

(1) 明确问题
(2) 分析问题并将其分解为几个基本元素
(3) 根据上步结果编写代码
(4) 测试——重复二、三步直至得到好的结果

2.2 Some basic Fortran 90 concepts

Self-test Exercises 2.1

1. (1) 明确问题
   (2) 分析问题并将其分解为几个基本元素
   (3) 根据上步结果编写代码

2. 测试

3. 以字母开头、不包含特殊符号、长度小于 31 个字符、不区分大小写。

4. program name
   end program name

5. implicit none
   用于禁止一些在更老版本的 Fortran 中出现的一些内容。

   Answer：
   If the last non-blank character of a line is an ampersand (&), then the statement is continued on the next time.

   If the ampersand appears in a character context(that is, in the middle of a character string enclosed in quotation marks or apostrophes), then the first non-blank character of the next line must also be an ampersand, and the character string continues from immediately after that ampersand.

If the ampersand at then end of the first line is not in a character context, then the statment is continued either from the first character after an ampersand, if that is the first non-blank character on the line, or from the start of the next line, if the first non-blank character is not an ampersand.

6. 后跟 comments 用于提示、解释代码作用。
   在整段代码前或语句结束后的任意位置可以添加。

2.3 Running Fortran programs on a computer

2.4 Errors in programs

2.5 The design and testing of programs

编写程序时，我们需要考虑：

1. Elegance. 花更多的时间在设计程序上，以更好地编写更优美的代码。强调易读性和效率。
2. Maintianability. 比起编写新的算法，编写长期可维护的算法更加重要。Remembering "write once and read many times".
3. Portable programs. 程序最终可以适应于大部分计算机。

好的程序需要：

1. 完全掌握程序的目的，确认好所有 input 和 output 。
2. 使 input 尽可能简单易理解；使 output 清晰明确有用。
3. 清楚地写下解决问题的方案，将整个问题划分为更加好实现的子问题。以便后期修改直至得到正确答案。
4. 查看已有的可用代码 in procedure libraries 。
5. Using a modular design to make sure that no single block of code should be longer than about 50 lines, excluding any comments.
6. Use descriptive namees for varibles and program units anf be lavish with comments.
7. Input error checking.
8. 测试程序的每一部分。

2.6 The old and new Fortran 90 source forms

Self-text Exercises 2.2

1. Syntactic error: 语法错误，包括拼写错误和不符合语法规范等。
   Semantic error: 语法没有错误，但不符合逻辑。

   Syntactic error 可被 compiler 检测出，导致 compilation errors 。
   Semantic errors 通常不会被检测出来，属于逻辑硬伤，导致 execution errors 。

2. Elegance(easier to test), maintianability, portability.

3. (1) 完全掌握程序的目的，确认好所有 input 和 output 。
   (2) 使 input 尽可能简单易理解；使 output 清晰明确有用。
   (3) 清楚地写下解决问题的方案，将整个问题划分为更加好实现的子问题。以便后期修改直至得到正确答案。
   (4) 查看已有的可用代码 in procedure libraries 。

4. Answer:

   (1) Ensure that your program carries out as many checks on the validity of the data it reads as is possible (and realistic). A program that attempts to produce a meaningful answer.
   (2) Carry out internal valisity checks at critical points in the calculations.
   (3) Check that a reasonable number of iterations aree being made while trying to converege to a solution.
   (4) Test each part of your program thoroughly before testing the complete grogram.

5. 132

6. 40

7. No limit to the number of statements, but of characters.
   By at least one blank (semi-colons).

2.7 Program Exercises

2.3 如果不定义变量，则输出 exactly 为输入。如果定义变量为 REAL ，则输出保留九个有效数字的实数。

2.4 PRINT*,"" 中怎样标""，怎样换行?

2.7 character(len = $*$) 定义变量为长度为 $*$ 的字符串。

Chapter 3

Essential data handling

3.1 The two fundamental types of numbers

integer: -50 000 000　+49 999 999（浮点计数后范围为 $-2\times10^9 \sim +2\times10^9$ ）

real number: -5000.0　+4999.9999 （浮点计数会对范围外的数字取近似用科学计数法表示，范围扩展至 $-10^{38} \sim +10^{38}$ ）

3.2 REAL and INTERGER varibles

About variable declaration:　TYPE :: name1, name2,...
About implicit declaration:　IMPLICIT NONE

It is extremely important that this statement appears at the beginning of every program in order that imolicit declarations of variables are forbidden.

3.3 Arithemtic expressions and assingment

About assignment statement:　READ (name = expression)

运算符号:

Operator	Meaning	Priority
+	addition	Low
−	subtraction	Low
*	multiplication	Medium
/	division	Medium
**	exponentiation	High

Table 3.1: Arthmetic operators

需要注意的是，在混合运算（mixed-mode expression）过程中，结果会因为整数运算涉及的取整而导致结果不准确，甚至错误。这种现象被称为 integer

division 。

解决方案：将全部运算在实数定义下进行，而后根据结果所需要的形式取整或不取整。

例外们：

如果遇到实数和整数混合运算的情况，结果会根据变量的性质取值。例如，运算结果为实数，但变量定义为整数。此时，truncate，即，仅保留整数部分，不四舍五入。

即使在全部都为实数的运算下，也可能因为取有限有效数字而导致计算结果有误。例如，$w = x + y - z, w = x = y = z = 5.678$，因为 $x + z = 11.356$，而计算机四舍五入保留四位有效数字，即 11.36。在减去 5.678 后，结果为 5.682，误差为 0.07%。

支持正负号。

支持指数形式（科学计数法）。例：$10^{-6} = 10E - 6$。

3.4 List-directed input and output of numeric data

The list-directed input statement: READ
The list-directed output statement: PRINT

About termination:

The rule is that each number, or their item, must be followed by a value separator consisting of a comma, a space, a slash (/) or the end of the line; any of these value seaparators may be preceded or followed by any number of consecutive blanks (or space).

If there are tow consescutive commas, then the effect is to read a null value, which results in the value of the corresponding vaarible in the input list being left unchanged.

If the terminating character is a slash then no more data items are read, and processing of the input statement is end. If there are any remaining items in the input list then the result is as though null values had been inout to them; in other words, thire values remian unchanged.

Input data: 1,
3,
5,6.0 failed.

Self-text Exercises 3.1

1. 整数没有小数部分。而实数有小数部分。
2. 整数可储存为精确值，计算无误差。而实数会进行近似储存，计算结果也是近似值。

3. i. 实数可取值范围远大于整数。
   ii. 实数有小数部分，可表示比整数更多的数。

4. 定义变量的类型。

   Answer: A declaration statement is a statement that identifies a name that will be used to represent a variable, and which also sepcifies the type of information (such as real or integer numbers) that will be stored in that variable.

5. (a) INTEGER :: men, women, children
   REAL :: adults_to_children
   (b) REAL :: a_feet, b_feet, h_feet, a_inch, b_inch, h_inch
   (c) REAL :: a, b, h
   (d) REAL :: times
   INTEGER :: photons

6. A implicit declaration is one in which a variable does not appear in a type declaration statement, but takes its type from the first letter of its name.

   An IMPLICIT NONE statement at the beginning of the program, immediately after the PROGRAM statement, prevents implicit declaration and results in an error if a variable is used without first being declaraed. Implicit declaration is very dangerous, and can lead to many types of program errors, for example as result of mistyped name not being deected or a variable accidentally being of wrong type.

7. An assignment statement cause the result of an expression to be assigned to a variable; that is, to be stored in the memory location identified by the variable name.

8. 按优先级排序：$**$ (exponentiation), $*$ (multipication) and / (division), $+$ (addition) and $-$ (subtraction).

9. REAL :: a, b, ave
   ave = (a+b)/2

10.
```
 6.50000000    10.0000000    0.649999976
          6             10              0
```

   Answer:
   The exact spacing of the numbers, and the number of decimal places for the real values may vary from computer to computer, but will follow essentially the same layout as shown above. Note that the second number on the second line will be printed as 9 if the result of multiplying 2.5 by 4.0 resulted in a value of, for exmaple, 9.999 999 9 as a result of round-off errors during the calculation.

11. Separated by enter: 1.2
    3.456
    7.89
    42.0
    Separated by space: 1.2 3.456 7.89 42.0

Separated by comma: 1.2,3.456,7.89,42.0
Separated by salash: 1.2/3.456/7.89/42.0

3.5 Handling CHARACTER data

Definition 1. The **numeric storage unit** is that art of the memory of the computer in which a single REAL or INTEGER number can be stored; The **character storge units**, typically occupying 8 or 16 bits, ecah of which can hould exactaly one characterr in a code form.

Definition 2. A **character variable** consists of a sequence of one or more consencutive character storge units.

Four ways of writing CHARACTER statement:
CHARACTER(LEN=*length*) :: *name1,mane2, ...*
CHARACTER(*length*) :: *name1,mane2, ...*
CHARACTER**length* :: *name1,mane2, ...*
CHARACTER(LEN=*length*) :: *name1, name2*len_2, name3*len_3*

Definition 3. The process of combining two strings to form a third composite string is called **concatenation**. It is carried out by means of the **concatenation operator**, consisting two consecutive slashes.

Self-text Exercises 3.2

1. *Fortran Character Set*: 包含 26 个英文字母，0-9 十个数字，下划线和 21 个其他特殊符号。
   default character set: set of characters normally available on the computer system being used without any special action on the part of the user.

2. The declaration of an integer or real variable identifies a name that will store a number.
   The declaration of a character variable is always with a length sepcification, and used to identifies a name that will store a string.

3. CHARACTER(LEN=20) :: a,b,c,d
   CHARACTER(LEN=1) :: e
   CHARACTER(LEN=9) :: month

4. CHARACTER(LEN=20) :: a,b,c,d,e*1,month*9

5. A small step for a man
   A giant leap for mankind

3.6 Initial values and constants

变量可以在 Variable declaration 的时候直接赋值。

3.7 Creating your own data types

自定义变量类型:
TYPE new_type
    compontent_defination
    .
    .
    .
END TYPE new_type

Note 1. 可以直接写在程序中。不需要另写文件定义。

Example 1. structure constructor
TYPE person
    CHARATER(LEN=12) :: first_name, middle_initial*1,last_name
    INGTER :: age
    CHARACTER :: sex   ! M or F
    CHARACTER(LEN=11) :: social_security
END TYPE person

TYPE(person) :: jack, jill

jack = person("Jack","R","Hagenbach",47,"M","123-45-6789")
jill = person("Jill","M","Smith",39,"F","987-65-4321")

3.8 Obsolete forms of declaration, initialization and constant definition

Self-text Exercises 3.3

1. An entity which given an initial value in its declaration statement can have that value changed later in the program. An entity with the PARAMETER attrobute is a constant, and its value cannot subsequently be changed.

2. A derived type is a user-defined data type. It consists of one or more components each of which is either of an intrinsic tyoe or of another derived type. A derived type is, therefore, ultimately derived from entities of intrinsic types.

3. 可以定义符合问题的变量，方便使用。

4. type address
       character(len=*) :: province, city, district, street
       integer :: number
       character :: building
   end type address

5. type(address) :: my_address
   my_address = address("Beijing"," ","Haidian","Yiheyuan","5","PKU")

6. type person
   character(len=20) :: first_name, last_name
   type(address) :: address
   end type person

   type(person) :: indivual

   print *,"Please input your name and address in the order"
   print *,"first name, last name, province, city,"
   print *,"district, street, number, building"
   read *, indivual

Program Exercises