

第四章 数组

4.1 概述

- 数组的概念**
- ① 是一种**构造类型**的数据结构，存在固定的定义结构。
 - ② 由一组具有**同一类型的**变量组成，同一个数组中的元素类型必须一致。
 - ③ 在内存占据**连续**的一段存储单元，知道某个元素的起始位置，可以推得下一个元素的位置。
 - ④ 数组中所包含的每个数据称为**数组元素**，它可以通过下标来区分。
 - ⑤ 同一个文件中，数组的名称不能和变量的名称一致。

注意 数组与简单变量的区别

4.2 数组的定义与引用

4.2.1 数组的定义

4.2.1.1 用类型说明语句定义数组

定义方式 **类型说明** **数组名(维说明符 [,维说明符,...])[,数组名,...]**

实例

INTEGER a(-5:5), b(20), temp(20,30)

temp 是一个 20×30 的二维数组。

a 是一个下标从 -5 到 5 的一维数组，默认步长为 1

a(-5:5) 等价于 a(-5:5:1)，其中共有 11 个元素（包含 a(0)）

b 等价于 b(1:20)，下界 1 可以省略

CHARACTER*8 name(50)

该句合法，因为 N 是符号常量

INTEGER,PARAMETER::N=10

该句非法，N 是变量

INTEGER::N=10

执行这一句是调用上文定义的语句，有两种情况。

REAL x(N+2:N*2)

4.2.1.2 用 DIMENSION 语句说明数组

定义方式 **DIMENSION** **数组名(维说明符[,维说明符,...])[,数组名,...]**

这种方法无需定义数组数据类型，其类型可以在随后定义。

实例

DIMENSION a(-5:5),b(3,4)

定义两个数组 a、b 的维信息。

INTEGER a

数据类型定义为整型。

REAL b

数据类型定义为实型。

4.2.1.3 用类型说明语句和 DIMENSION 语句定义数组

定义方式 **类型说明, DIMENSION (维说明符[,维说明符,...]) ::数组名[,数组名,...]**

实例

INTEGER,DIMENSION(4,5)::a,b

这两个数组 a、b 具有相同的维信息和数据类型。

REAL(8),DIMENSION(0:10)::c,d

都是实型数组，每个元素长度为 8，有 11 个元素。

REAL,DIMENSION(0:10)::a,b(20),c(3,5)

b,c 与统一定义有冲突，以后面的定义为主。

4.2.2 数组元素的引用

4.2.2.1 单个数组元素引用：下标法

引用方式 数组名(下标[, 下标, ...])

实例	
INTEGER a(5), b(2,3)	定义一维数组 a 和二维数组 b
a(1), a(2), a(3), a(4), a(5)	读取 a 的全部 5 个元素
b(1,1),b(1,2),b(1,3),b(2,1),b(2,2),b(2,3)	读取 b 的全部 6 个元素
Real i,j i=2.5 j=2.0	
b(i-1,j-1)=10	实际赋值的是 b(1,1)=10

4.2.2.2 多个数组元素引用：片段法

连续片段法 数组名(起始下标: 终止下标) 表示一组连续的元素

实例	
INTEGER a(10)	REAL b(2,3)
a(5:8)	表示数组 a 中 4 个连续的元素 a(5)至 a(8)
a(5:8)=0	表示把 a(5)-a(8) 共 4 个元素都赋值为 0
b(1:1,1:3)	表示 b(1,1),b(1,2),b(1,3)三个元素，即第一行的元素
b(1:2,2:2)	表示 b(1,2),b(2,2)两个元素

下标三元组 数组名([起始下标]:[终止下标][:步长], ...) 把不连续的元素组成数组片段

实例	
INTEGER,DIMENSION(5:45)::a	下标为 5~45 的整型数组（例如经纬度）
INTEGER,DIMENSION(4,5)::b	
a(10:30:1)	表示 a(10)~a(30)中的连续的 21 个元素
a(:15:5)	从最开始，输出 a(5),a(10),a(15)
a(6::10)	表示 a(6),a(16),a(26),a(36)
b(:,1:5:2)=500	表示将 b 数组中第 1,3,5 列元素赋值为 500

直接引用 数组名 直接引用数组名，例如 Print *, r

4.3 数组的逻辑结构与存储结构

4.3.1 一维数组

逻辑结构 由一组类型相同的数据构成的线性表。

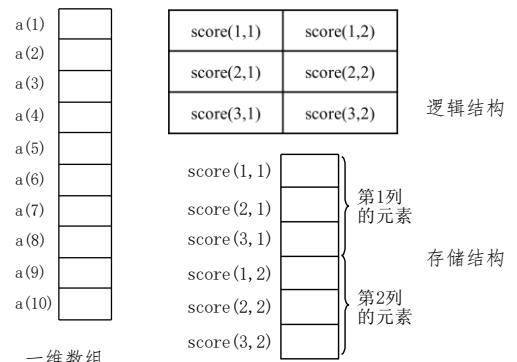
存储结构 与逻辑结构相同，例如 INTEGER a(10) (右图)

4.3.2 二维数组

逻辑结构 一张表格或矩阵。

存储结构 按列顺序存储。例如：REAL score(3,2)

Python 和 C 语言默认都按照行优先存储。



实例

现有某 10*10 区域内对流层低层 850hPa 和高层 200hPa 水平风场 U,V 逐月资料，时段从 1982 年 1 月到 1985 年 12 月，请给出 U、V 风速的数组设置。

U(10,10,48,2)、V(10,10,48,2)

4.4 数组的输入输出

4.4.1 使用 DO 循环输入输出数组元素

4.4.1.1 一维数组的输入和输出

一重循环法 一行输入输出一个数

实例

```
DO i = 1,10  
    READ(*,*) a(i)          必须输入 10 次回车  
END DO  
DO i=1,10,2  
    WRITE(*,200) a(i)  
END DO  
200 FORMAT(1X,2I3)          执行时输入: 输出结果:  
                            1✓  □ □ □ 1  
                            2✓  □ □ □ 3  
                            3✓  □ □ □ 5  
...✓  □ □ □ 7  
10✓  □ □ □ 9
```

如后面变量个数多于 2 个, **2** 生效, 否则不生效。
表示每行输出两个数。

4.4.1.2 二维数组的输入和输出

双重循环法 一行输入输出一个数

实例

```
DO i = 1,3  
    DO j = 1,2  
        READ *, w(i,j)  
    ENDDO  
ENDDO
```

执行时输入:
87✓ 赋给w(1, 1)
80✓ 赋给w(1, 2)
74✓ 赋给w(2, 1)
95✓ 赋给w(2, 2)
93✓ 赋给w(3, 1)
78✓ 赋给w(3, 2)

4.4.2 用数组名或数组片段对数组进行输入和输出

4.4.2.1 一维数组的输入和输出

- 方法一 READ*, a 输入十个整数依次将它们放入数组元素 a(1)~a(10)中, 一行输入 10 个数字。
方法二 READ*,a(1:10:2) 输入五个整数依次将它们放入数组元素 a(1),a(3),a(5),a(7),a(9)中。
方法三 PRINT*, a 输出 a(1)-a(10)的值

4.4.2.2 二维数组的输入和输出

- 方法一 READ *, w 输入数据: 87,74,93,80,95,78✓ 注意:
输入输出顺序总是和数组元素在内存中的存放顺序一致。

- 方法二 PRINT '(1X,2F5.2)', w 输出结果为: □87.00□74.00
□93.00□80.00
□95.00□78.00

- 方法三 PRINT '(1X,3F5.2)', w(1:3,2:2) 精确控制输出结果为: □80.00□95.00□78.00

4.4.3 用隐含的 DO 循环对数组进行输入和输出

一般形式 (输入/输出表, $i = e1, e2 [, e3]$)

- 注意 ① i 是隐含 DO 的循环变量。 ② $e1, e2, e3$ 分别是循环变量的初值、终止、步长。
③ 隐含 DO 循环**必须要用小括号括起来**。

4.4.3.1 一维数组的输入和输出

环境信息 INTEGER a(5) 100 FORMAT(5I3) 200 FORMAT(1X,3I3)
输入方式 READ(*,100)(a(i),i=1,5) 输入所有元素，等效于 READ *, a
输出方式 WRITE(*,200)(a(i),i=1,5,2) 输出 a(1),a(3),(5) 的值，

执行时输入:
□□1□□2□□3□□4□□5 ✓
输出结果为:
□□□1□□□3□□□5

4.4.3.2 二维数组的输入和输出

环境信息 REAL w(3,2) 200 FORMAT(2F6.2)
输入方式 READ(*,200) ((w(i,j),j=1,2), i=1,3)

执行时输入:
□87.00□80.00 ✓
□74.00□95.00 ✓
□93.00□78.00 ✓

4.5 数组赋初值

4.5.1 DATA 语句赋初值

赋值方法 DATA 变量表 1/初值表 1/ [, 变量表 2/初值表 2/] ...

实例

REAL a,b,c,d,e	定义 5 个变量
DATA a,b,c/-1.0,-1.0,-1.0/, d,e/2.5,5.8/	a,b,c 对应 3 个值, d,e 对应 2 个值
INTEGER a(5)	定义有 5 个元素的数组
DATA a/1,2,3,4,5/	允许直接给其赋 5 个值
CHARACTER * 6 chn(10)	定义 10 个长度为 6 的字符串的 chn 数组
DATA chn/10*'aaaaaa' /	10 个数据的初值都一样
INTEGER num(100)	定义有 100 个元素的整数组
DATA (num(i),i=1,10)/10*0/,(num(i),i=91,100)/10*1/	跳跃式赋值

注意

① 在给二维数组赋初值时，一定要注意数据的排列顺序。例如有如下矩阵：

$$M = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

正确的赋值方法是： INTEGER M(4,4) DATA M/1,5,9,13,2,6,10,14,3,7,11,15,4,8,12,16/ 按列

② DATA 语句是非执行语句。它的作用是给编译系统提供信息，在程序编译阶段赋初值，而不是在程序运行阶段。

4.5.2 使用数组赋值符赋初值

赋值方法 类型说明 :: 数组名 (维说明符) = (/ 初值表 /)

实例

INTEGER :: a(5) = (/ 1,2,3,4,5 /)	
INTEGER :: a(5) = (/ 6*2, (i, i = 2,4), 5*2 /)	12 给 a(1), 2,3,4 给 a(2), 10 给 a(5)
INTEGER :: a(5) = (/ i, i = 2,4 /)	该方法错误，没有都给初值

说明

- ① 初值表中可以使用常量、符号常量、常量表达式或隐含 DO 循环，但不能使用变量。在括号和除号之间不能有空格，并且不能省略 “::”。
- ② 这种方法必须对数组中的每个元素都给定初值。如下面的语句是错误的：
- ③ 如果对数组中的元素都赋同样的初值，则上面的语句可以简化为： INTEGER :: a(5)=5，这种方法只有在定义时有效。

4.6 动态数组

静态数组	数组定义时就分配大小，并且大小是固定不变的。
动态数组	说明时不分配存储单元，在程序运行时再由语句分配对应大小的内存，且大小可按需要变化。
使用步骤	① 定义动态数组 ② 为动态数组分配存储空间 ③ 使用完动态数组之后回收其所占内存空间
定义	类型说明, ALLOCATABLE ::数组名 1(维说明符 1) [, 数组名 2(维说明符 2), ...]
分配内存	例如: INTEGER,ALLOCATABLE :: vector(:), matrix(:, :, :) ALLOCATE(动态数组名 1([下界:]上界) [, 动态数组名 2([下界:]上界), ...]) ALLOCATE(vector(-5:5), matrix(5,5))
说明	① 下界和上界可以是整型常量或者变量。 ② 下界为 1 时可以省略。 ③ 如果下界>上界，则数组的大小为零。
释放内存	DEALLOCATE(动态数组名 1[, 动态数组名 2, ...]) DEALLOCATE(vector, matrix) 注意：此语句的括号内只需写数组名而不要写其大小。

4.7 数组常用算法举例

例题：输入南京站 2014 年 7 月份 31 天的气温值，把高于平均温度的日期和气温值输出。

```
integer, parameter :: ndays = 31
character(len=8) :: num(ndays)      ! 存放日期
real :: s(ndays)                  ! 存放气温数据
real :: average, sum_temp
integer :: i, count_above

do i = 1, ndays          ! 初始化日期数组
    write(num(i), '(I8)') 20140700 + i
    num(i) = adjustl(num(i))
end do
```

```
! 输入 31 天的气温数据（示例数据）
s = [28.5, 29.2, 31.0, 32.5, 33.1, 30.8, 29.6, &
     31.2, 32.7, 34.2, 35.0, 33.8, 32.1, 30.5, &
     29.8, 31.5, 33.2, 34.8, 36.2, 35.7, 34.1, &
     32.8, 31.4, 30.2, 32.0, 33.5, 34.9, 35.3, &
     33.7, 31.9, 30.6]
```

! 实际应用中应该从文件或用户输入获取
READ(*,*) (num(i),s(i),i=1,N)

```
! 计算平均温度
sum_temp = sum(s)
average = sum_temp / ndays
```

```
! 输出平均温度
write(*,'(A,F6.2,A)') '南京站 2014 年 7 月份平均气温:
```

```
', average, '°C'
```

```
! 查找并输出高于平均温度的日期和气温值
count_above = 0
write(*,'(A)') '高于平均温度的日期和气温:'
write(*,'(A10,A10)') '日期', '气温(°C)'

do i = 1, ndays
    if (s(i) > average) then
        count_above = count_above + 1
        write(*,'(A10,F10.1)') num(i), s(i)
    end if
end do

write(*,'(A,I0,A)') '共有', count_above, '天的气温高于
平均值'
```

格点数据的处理

题目：2014 年 4 月 23 日 08 时在(135-145E,50-60N)范围内 500hPa 等压面上存在位势高度最低值，请将该区域内网格点上位势高度最低值输出，并将格点对应经纬度给出。（假设水平分辨率为 $5^\circ \times 5^\circ$ ）

问题分析：
① 数据结构：这是一个确定的二维数组，数组的行数 m 和列数 n （即 X 和 Y 方向格点数）根据网格区域和水平分辨率计算。定义变量 h 表示位势高度， lon 表示经度， lat 表示纬度。注意：X 正向代表经度方向，Y 正向代表

纬度方向。

② 算法思路：先根据网格区域和水平分辨率计算 m 和 n ；用 data 语句给 $h(m, n)$ 赋初值；根据水平分辨率和经纬度范围，可以用循环语句给 $lon(m, n), lat(m, n)$ 赋值；设定一个最低位势高度值 $hmin$ ，通过循环比较 $h(m, n)$ 与 $hmin$ 大小找到真正的最低值，同时将格点信息给出。

！ 定义区域范围和分辨率

```
real, parameter :: lon_min = 135.0, lon_max = 145.0  
real, parameter :: lat_min = 50.0, lat_max = 60.0  
real, parameter :: resolution = 5.0
```

！ 计算网格点数

```
integer :: m, n  
integer :: i, j  
real :: hmin  
integer :: min_i, min_j
```

！ 根据区域和分辨率计算网格点数

```
m = int((lon_max - lon_min) / resolution) + 1  
n = int((lat_max - lat_min) / resolution) + 1
```

！ 声明数组

```
real, dimension(3, 3) :: h ! 位势高度  
real, dimension(3, 3) :: lon ! 经度  
real, dimension(3, 3) :: lat ! 纬度
```

！ 给经纬度数组赋值

```
do i = 1, m  
    do j = 1, n  
        lon(i, j) = lon_min + (i - 1) * resolution  
        lat(i, j) = lat_min + (j - 1) * resolution  
    end do  
end do
```

！ 给位势高度赋初值（使用 data 语句）

```
h(1, 1) = 5420.0  
h(2, 1) = 5415.0  
h(3, 1) = 5430.0  
h(1, 2) = 5410.0  
h(2, 2) = 5400.0 ! 最低值点  
h(3, 2) = 5425.0
```

$h(1, 3) = 5425.0$

$h(2, 3) = 5415.0$

$h(3, 3) = 5435.0$

！ 输出位势高度场数据

```
write(*,'(A)') '位势高度场数据(单位: gpm):'  
write(*,'(10X,3A12)') ('Longitude', i=1, m)  
do j = n, 1, -1  
    write(*,'(F6.1,1X,3F12.1)') lat(1, j), (h(i, j), i=1, m)  
end do  
write(*,'(7X,3F12.1)') (lon(i, 1), i=1, m)  
write(*,*)
```

！ 查找最低位势高度值及其位置

```
hmin = h(1, 1)  
min_i = 1  
min_j = 1
```

```
do i = 1, m  
    do j = 1, n  
        if (h(i, j) < hmin) then  
            hmin = h(i, j)  
            min_i = i  
            min_j = j  
        end if  
    end do  
end do
```

！ 输出最低位势高度值及其位置信息

```
write(*,'(A,F8.1,A)') '区域内最低位势高度值为:',  
hmin, ' gpm'  
write(*,'(A,F7.1,A)') '对应经度:', lon(min_i, min_j), '°E'  
write(*,'(A,F7.1,A)') '对应纬度:', lat(min_i, min_j), '°N'  
write(*,'(A,I0,A,I0)') '对应网格点位置: (', min_i, ',',  
min_j, ')'
```

```
end program min_geopotential_height
```