第八章　数组及应用

1. 数组概述
2. 数组声明
3. 数组元素引用
4. 数组逻辑结构和存储结构
5. 数组输入输出
6. 给数组赋初值
7. 动态数组
8. 数组应用举例

8.1 数组概述

前面章节涉及的问题都是比较简单,编写的程序也不复杂,这些程序的一个突出特点是所处理的数据量非常少,采用的数据类型都是简单的内部数据类型,使用的变量都是普通变量。然而在实际的生产活动中,往往遇到的是一些涉及大量数据的复杂问题,如:

- 统计100年内月平均温度及高于平均温度的月数。
- 统计学校1500名学生的平均身高及大于平均身高的男女生人数。
- 统计某商场近10年月平均销售额及高于平均销售额的月份。
- 求解100元一次方程组。
- 求解高阶微分方程。
- 计算100×100阶矩阵的转置矩阵。
- 对某班30名学生的学习成绩进行排序处理。

以上问题都要涉及几十、几百、几千,甚至上万个数据,如果在程序中采用简单的内部数据类型和变量来求解这类问题,其难度将大大增加,甚至无法设计和编写程序。

（非数组示例1） （非数组示例2） （非数组示例3） （数组示例4）

8.1 数组概述示例1

```
!统计3个数的平均值及大于平均值的数据个数。使用普通变量和顺序结构实现。
PROGRAM  example1
INTEGER a1,a2,a3, num
REAL  sum,av
READ *,a1,a2,a3          !输入3个数
sum=a1+a2+a3 ；av=sum/3   !求3个数之和，计算平均值
num=0                     !赋初值
IF (a1>av) num=num+1      !统计大于平均值的数据个数
IF (a2>av) num=num+1
IF (a3>av) num=num+1
PRINT *,‘平均值为：’,av          !输出平均值
PRINT *,'大于平均值的数据个数为：',num  !输出大于平均值的数据个数
END
```

8.1 数组概述示例2

```
!统计10个数的平均值及大于平均值的数据个数。使用普通变量和顺序结构实现。
PROGRAM  example2
INTEGR a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,num
REAL  sum,av
READ *,a1,a2,a3,a4,a5,a6,a7,a8,a9,a10
sum=a1+a2+a3+a4+a5+a6+a7+a8+a9+a10
av=sum/10;  num=0
IF (a1>av) num=num+1
IF (a2>av) num=num+1

IF (a10>av) num=num+1
PRINT *,'平均值为：',av;  PRINT *,'大于平均值的数据个数：',num
END
```

8.1 数组概述示例3

!统计10个数的平均值及大于平均值的数据个数。使用普通变量和循环结构实现。

```
PROGRAM  example3
INTEGER a,num
REAL  sum,av
sum=0.0
DO i=1,10
  READ *,a
  sum=sum+a
ENDDO
av=sum/10
num=0
DO i=1,10
  READ *,a
  IF (a>av) num=num+1
ENDDO
PRINT *,'平均值为：',av
PRINT *,'大于平均值的数据个数为：
',num
END
```

!该程序虽然变量比较少，程序比较短，但数据被输入两遍。

8.1 数组概述示例4

!统计10个数的平均值及大于平均值的数据个数。使用数组和循环结构实现。

```
PROGRAM  example4
PARAMETER(max_num=10)
INTEGER a(10),num
REAL  sum,av
sum=0.0
DO i=1,10
  READ *,a(i)
  sum=sum+a(i)
ENDDO
av=sum/10

num=0
DO i=1,10

  IF (a(i)>av) num=num+1
ENDDO
PRINT *,'平均值为：',av
PRINT *,'大于平均值的数据个数为：
',num
END
```

!该程序变量少，程序短，数据输入一遍。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●数组是类型相同的一组数据(变量)的有序集合。

任何一组类型相同且有序的数据都可使用数组存储。

●数组分静态数组和动态数组

- 静态数组：在编译时分配存储单元，运行时数组大小不能改变。
- 动态数组：在程序运行时分配存储单元，运行时数组大小可改变。

●数组大小为数组元素个数。

●数组声明有3种方式：类型声明语句、DIMENSION语句、DIMENSION属性

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句进行数组声明。一般格式：

<数组声明语句>→<类型声明符> [::] <数组声明表>

■声明下界为1的数组。例

■声明数组下界不为1的数组。例

■声明赋初值的数组。例

●数组元素个数计算。

REAL a(L1:U1,L2:U2,...,Ln:Un) !数组元素个数m按以下公式计算：

如：REAL arr(2:5,-1:2,4),M=(5-2+1)×(2-(-1)+1)×(4-1+1)=64。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用DIMENSION语句进行数组声明。一般格式：

<DIMENSION语句>→DIMENSION <数组声明表>

例：

```
INTEGER day,count
CHARACTER*10 country
LOGICAL L
DIMENSION num(3),day(12),ihour(5:7),count(-3:3)
DIMENSION sc(-5:-1,5:9),ar(2:5,-1:2,4)
DIMENSION country(15),L(5)
```

DIMENSION语句中不能出现::,不能给数组赋初值。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句和DIMENSION属性进行数组声明。

一般格式：

<带DIMENSION属性的数组声明语句>→<类型声明符>,DIMENSION(<维说明表>) :: <数组声明表>

例：

```
INTEGER ,DIMENSION(2)::num,total,score(10:15),price(4,5)
REAL,DIMENSION(-5:-1,5:9) :: sc,jp
CHARACTER*10,DIMENSION(15) :: country,name(10)
INTEGER , DIMENSION(5) :: S=(/45,78,85,55,95/)
```

必须出现::,可给数组赋初值。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

- 数组声明中，维下界可以为1，维下界为1可缺省。
- 数组声明中，维下界和维上界可以为任意整数，且维下界≤维上界。
- 数组声明中，出现::，才能在声明语句中给数组赋初值。
- 数组声明中，允许维下界>维上界，且数组大小为0。
- 使用类型声明语句声明数组时，可使数组与普通变量混合声明。
- 使用DIMENSION语句声明数组时，可用类型声明语句在前面声明类型。声明中不能有::。
- 使用DIMENSION属性声明数组时，声明中必须有::。

Back

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句进行数组声明。一般格式：

<数组声明语句>→<类型声明符>[::] <数组声明表>

■声明下界为1的数组。例

■声明数组下界不为1的数组。例

■声明赋初值的数组。例

●数组元素个数计算。

```
INTEGER（1）num(3),day(12)  !声明KIND值为1的二个整型数组
INTEGER  N(4),M(5,8)        !声明KIND值为4的二个整型数组
REAL  score(4),prices(15)   !声明二个单精度实型数组
REAL(8)  dr(5)              !声明一个双精度实型数组
COMPLEX  comp(3)            !声明一个单精度复型数组
CHARACTER*10 country(15)    !声明一个长度为10的字符型数组
LOGICAL  L(5)               !声明一个KIND值为4的逻辑型数组
```

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句进行数组声明。一般格式：

<数组声明语句>→<类型声明符> [::] <数组声明表>

- ■声明下界为1的数组。例
- ■声明数组下界不为1的数组。例
- ■声明赋初值的数组。例

●数组元素个数计算。

REAL a(L1:U1,L2:U2,...,Ln:Un) !数组元素个数m按以下公式计算：

```
INTEGER count(-3:3)  !声明一个一维整型数组，有7个数组元素
INTEGER week(0:6)    !声明一个一维整型数组，有7个数组元素
REAL  sc(-5:-1,5:9)  !声明一个二维实型数组，有25个数组元素
REAL  ar(2:5,-1:2,4) !声明一个三维实型数组，有64个数组元素
```

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句进行数组声明。一般格式：

<数组声明语句>→<类型声明符> [::] <数组声明表>

- ■声明下界为1的数组。例
- ■声明数组下界不为1的数组。例
- ■声明赋初值的数组。例

●数组元素个数计算。

REAL a(L1:U1,L2:U2,...,Ln:Un) !数组元素个数m按以下公式计算：

如：REAL arr(2:5,-1:2,4),M=(5-2+1)×(2-(-1)+1)×(4-1+1)=64。

```
INTEGER :: num(3)=(/12,23,34/),day(12)
REAL :: a(-1:3)=(/1.5,2.5,3.5,4.5,5.5/)
INTEGER ::
X(3,4)=(/1,2,3,4,5,6,7,8,9,10,11,12/)
```

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用DIMENSION语句进行数组声明。一般格式：

<DIMENSION语句>→DIMENSION <数组声明表>

例：

```
INTEGER day,count
CHARACTER*10 country
LOGICAL L
DIMENSION num(3),day(12),ihour(5:7),count(-3:3)
DIMENSION sc(-5:-1,5:9),ar(2:5,-1:2,4)
DIMENSION country(15),L(5)
```

DIMENSION语句中不能出现::,不能给数组赋初值。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

●使用类型声明语句和DIMENSION属性进行数组声明。

一般格式：

<带DIMENSION属性的数组声明语句>→<类型声明符>,DIMENSION(<维说明表>) :: <数组声明表>

例：

```
  INTEGER,
DIMENSION(2)::num,total,score(10:15),price(4,
5)
  REAL,DIMENSION(-5:-1,5:9) :: sc,jp
  CHARACTER*10,DIMENSION(15) ::
country,name(10)
  INTEGER，DIMENSION(5) ::
S=(/45,78,85,55,95/)
```

必须出现::,可给数组赋初值。

8.2 数组声明

- 定义
- 声明1
- 声明2
- 声明3
- 说明

- 数组声明中，维下界可以为1，维下界为1可缺省。
- 数组声明中，维下界和维上界可以为任意整数，且维下界≤维上界。
- 数组声明中，出现::，才能在声明语句中给数组赋初值。
- 数组声明中，允许维下界>维上界，且数组大小为0。
- 使用类型声明语句声明数组时，可使数组与普通变量混合声明。
- 使用DIMENSION语句声明数组时，可用类型声明语句在前面声明类型。声明中不能有::。
- 使用DIMENSION属性声明数组时，声明中必须有::。

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

●数组声明只是确定了一个数组的类型、维数和元素个数。

●在程序中需要引用数组的某个元素、某些元素和全部元素,或者给数组元素赋予新的值,

●数组元素可参与某种运算,可作为操作数放在表达式中。

示例程序:

```
INTEGER  :: num(-2:2),sum=0.0
num(-2)=100;num(-
1)=200;num(0)=300;num(1)=400;num(2)=500
av1=(num(-2)+num(-1)+num(0)+num(1)+num(2))/5.0
DO I= -2,2
  sum=sum+num(I)
ENDDO
av2=sum/5.0
PRINT *,'av1=',av1
PRINT *,'av2=',av2
END
```

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

●数组元素引用一般格式：

<数组元素引用>→<数组名>[（<数组下标>{,<数组下标>}）]

如：INTEGER id(10)

id=25 !等价id(1)=25;id(2)=25;...;id(10)=25

id(1)=2*id(1); id(5)=3*id(5); id(10)=2*id(2)+3*id(9)

id(2:4)=15; id(6:9)=35

id(:10)=999 !等价于id(1:10)=999

步骤	id(1)	id(2)	id(3)	id(4)	id(5)	id(6)	id(7)	id(8)	id(9)	id(10)
1	25	25	25	25	25	25	25	25	25	25
2	50	25	25	25	75	25	25	25	25	125
3	50	15	15	15	75	35	35	35	35	125
4	999	999	999	999	999	999	999	999	999	999
999										

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

●数组引用要保证下标值落在下界和上界之间，否则将产生严重后果。

●使用内部函数（UBOUND、LBOUND）分别检测数组的上界和下界。

<UBOUND函数调用> → UBOUND(<数组名>[，[DIM=]<整型表达式>])

<LBOUND函数调用> → LBOUND(<数组名>[，[DIM=]<整型表达式>])

可检测某一维(指定维数，有第二参数)，结果为整数。

也可检测所有维(不指定维数，缺第二参数)，结果为一维整型数组。

例：INTEGER num(-5:-1,11,15,-2:2)

LBOUND(num,1)等于-5

UBOUND(num,1)等于-1

LBOUND(num)等于（/-5，11，-2/）

UBOUND(num)等于（/-1，15，2）

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

```
INTEGER  arr(-5:3,100:110,4)
INTEGER(1)  ub(3),ub1,ub2,ub3,lb(3),lb,lb2,lb3
ub1=UBOUND(arr,1);                              ub2=UBOUND(arr,DIM=2);
ub3=UBOUND(arr,3)
ub=UBOUND(arr)
lb1=LBOUND(arr,1); lb2=LBOUND(arr,DIM=2); lb3=LBOUND(arr,3)
lb=LBOUND(arr)
PRINT *,'函数UBOUND(arr,    1)求数组arr第1维的下标上界是：',ub1
PRINT *,'函数UBOUND(arr,DIM=2)求数组arr第2维的下标上界是：',ub2
PRINT *,'函数UBOUND(arr,    3)求数组arr第3维的下标上界是：',ub3
PRINT *,'函数UBOUND( arr  )求数组arr所有维的下标上界依次是：',ub
PRINT *,'函数LBOUND(arr,    1)求数组arr第1维的下标下界是：’,lb1
PRINT *,'函数LBOUND(arr,DIM=2)求数组arr第2维的下标下界是：',lb2
PRINT *,'函数LBOUND(arr,    3)求数组arr第3维的下标下界是：',lb3
PRINT *,'函数LBOUND( arr  )求数组arr所有维的下标下界依次是：',lb
END （运行结果）
```

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

```
PROGRAM  change_pay
REAL                                    :                    :
teacher(1001:1005)=(/1200.0,1500,3500,2000,2400/)
!下标为教师工号
PRINT *,'输入教师工号和加减工资额:'
READ *,i,a
IF （i>UBOUND(teacher,1) .OR. i<LBOUND(teacher,1)）
THEN
  PRINT *,'教师工号错！！！！拒绝修改工资！！！！'
ELSE
  teacher(i)=teacher(i)+a
  WRITE(*,100) i,teacher(i)
ENDIF
100 FORMAT(1X, '工号为',I4,'的教师修改后工资:',F7.1)
END
```

（运行结果）

Back

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2

```
INTEGER  arr(-5:3,100:110,4)
INTEGER(1)  ub(3),ub1,ub2,ub3,lb(3),lb,lb2,lb3
ub1=UBOUND(arr,1);                    ub2=UBOUND(arr,DIM=2);
ub3=UBOUND(arr,3)
ub=UBOUND(arr)
lb1=LBOUND(arr,1); lb2=LBOUND(arr,DIM=2); lb3=LBOUND(arr,3)
lb=LBOUND(arr)
PRINT *,'函数UBOUND(arr,    1)求数组arr第1维的下标上界是：',ub1
PRINT *,'函数UBOUND(arr,DIM=2)求数组arr第2维的下标上界是：',ub2
PRINT *,'函数UBOUND(arr,    3)求数组arr第3维的下标上界是：',ub3
PRINT *,'函数UBOUND( arr  )求数组arr所有维的下标上界依次是：',ub
PRINT *,'函数LBOUND(arr,    1)求数组arr第1维的下标下界是：’,lb1
PRINT *,'函数LBOUND(arr,DIM=2)求数组arr第2维的下标下界是：',lb2
PRINT *,'函数LBOUND(arr,    3)求数组arr第3维的下标下界是：',lb3
PRINT *,'函数LBOUND( arr  )求数组arr所有维的下标下界依次是：',lb
END （运行结果）
```

运行程序,输出结果为:

```
函数UBOUND(arr,    1)求数组arr第1维的下标上界是:    3
函数UBOUND(arr,DIM=2)求数组arr第2维的下标上界是:  110
函数UBOUND(arr,    3)求数组arr第3维的下标上界是:    4
函数UBOUND( arr )求数组arr所有维的下标上界依次是:  3
110   4
函数LBOUND(arr,    1)求数组arr第1维的下标下界是:   -5
函数LBOUND(arr,DIM=2)求数组arr第2维的下标下界是: 100
函数LBOUND(arr,    3)求数组arr第3维的下标下界是:   1
函数LBOUND( arr )求数组arr所有维的下标下界依次是:  -5
100  1
```

8.3 数组元素引用

- 概述
- 格式
- 检测
- 示例1
- 示例2


```
PROGRAM  change_pay
REAL                                    :                                    :
teacher(1001:1005)=(/1200.0,1500,3500,2000,2400/)
!下标为教师工号
PRINT *,'输入教师工号和加减工资额:'
READ *,i,a
IF （i>UBOUND(teacher,1) .OR. i<LBOUND(teacher,1)）
THEN
  PRINT *,'教师工号错！！！！ 拒绝修改工资！！！！'
ELSE
  teacher(i)=teacher(i)+a
  WRITE(*,100) i,teacher(i)
ENDIF
100 FORMAT(1X, '工号为',I4,'的教师修改后工资:',F7.1)
END
```

（运行结果）

三次运行程序，分别输入以下数据：

- 1004,300↙
- 1000,200↙
- 1010,500↙

程序运行输出结果分别为：

- 工号为1004的教师修改后工资：2300.0
- 教师工号错!!!!拒绝修改工资!!!
- 教师工号错!!!!拒绝修改工资!!!

8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

- 数组是复杂数据结构，有逻辑结构和存储结构两大特征。
- 数组逻辑结构：数组是一个与计算机实现无关的抽象的数据表。
- 数组存储结构：数组是一个与计算机实现有关的具体的数据表(线性表),连续存储在若干存储单元中。

对于一维数组，其逻辑结构是一个线性表。存储结构是一个线性表。

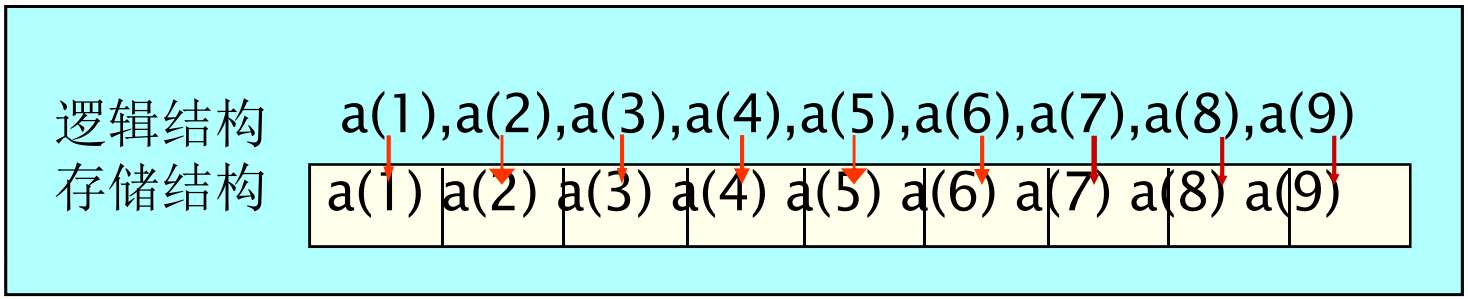
对于二维数组，其逻辑结构是矩阵，存储结构是一个线性表。

对于三维数组，其逻辑结构是三维实体，存储结构是一个线性表。

8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

- 数组说明：INTEGER a（9）。
- 逻辑结构、存储结构、相互关系：

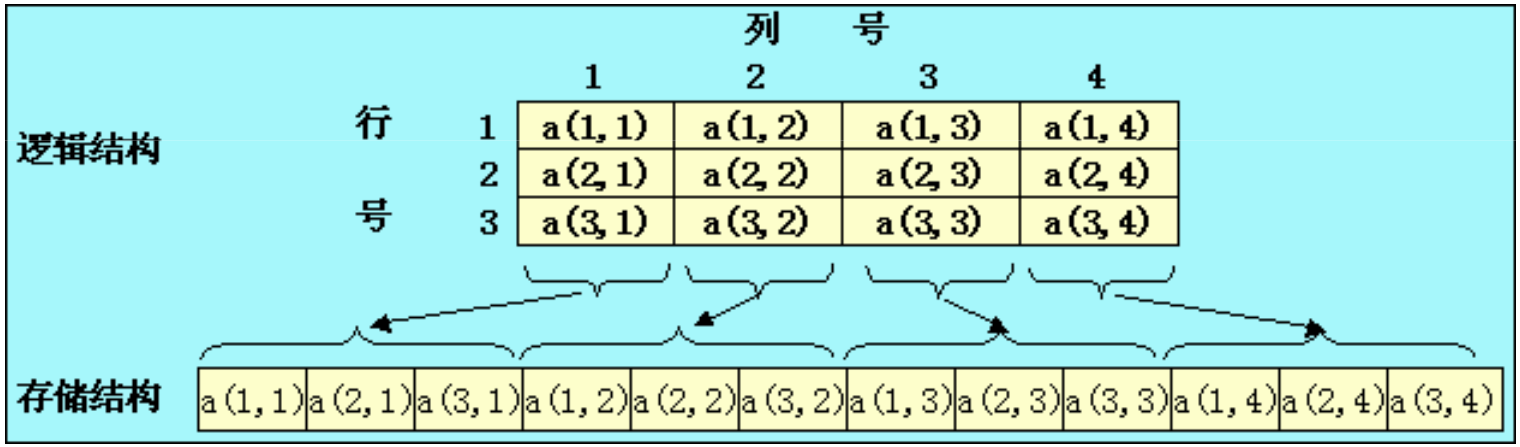


8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

●数组说明：INTEGER a（3，4）。

●逻辑结构、存储结构、相互关系：

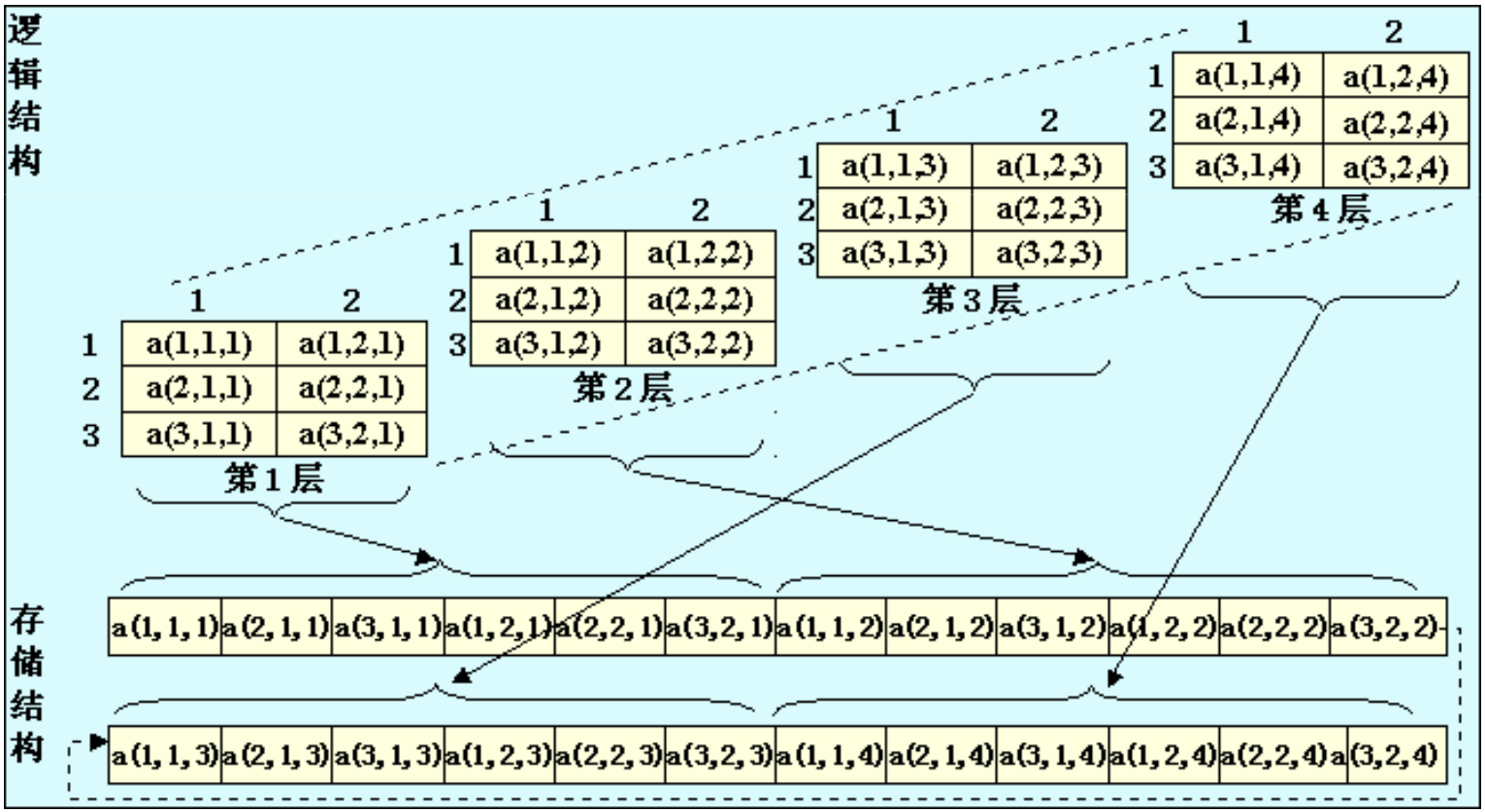


8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

●数组说明：INTEGER a（3，2，4）。

●逻辑结构、存储结构、相互关系：



8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

●数组说明：INTEGER a(L_1:U_1,L_2:U_2,…,L_n:U_n)。

●逻辑结构：

逻辑结构可看成是由U_n-L_n+1个n-1维数组构成,每个n-1维数组又可看成是由$U_{n-1}-L_{n-1}+1$个n-2维数组构成,依次类推,直到每个二维数组又可看成是由U_1-L_1+1个一维数组构成。

数组元素的下标a(i,j,k,…,r,s)决定该数组元素在逻辑结构中的位置。

●存储结构：

存储结构是在计算机内存中分配的一串连续的存储单元,从第一个数组元素（如:a(L_1,L_2,…,L_n)）开始,按照数组逻辑结构以列为主依次将数组元素分配在连续的存储单元中。多维数组的存储结构是一维线性表,多维数组的元素在存储结构中从左向右排列时,其下标变化如同加油站计数器反向变化,先从1维开始变化,最后到n维变化,由此可得到多维数组的存储结构。

●相互关系：一一对应关系。

8.4 数组的逻辑结构和存储结构

- 概述
- 一维
- 二维
- 三维
- n维
- 位置

●数组说明：INTEGER a(L_1:U_1,L_2:U_2,…,L_n:U_n)。

●逻辑结构数组元素a(i,j,k,…,r,s)对应存储结构第几个元素计算：

$(U_1-L_1+1)(U_2-L_2+1)\ldots(U_{n-1}-L_{n-1}+1)(s-L_n)+$
$(U_1-L_1+1)(U_2-L_2+1)\ldots(U_{n-2}-L_{n-2}+1)(r-L_{n-1})+$
$\ldots +$
$(U_1-L_1+1)(U_2-L_2+1)(k-L_3)+$
$(U_1-L_1+1)(j-L_2)+i-L_1+1$

●例：

三维数组声明 INTEGER a(-5:8,-3:4,10,15),有672个数组元素

/

数组元素a(2,1,13)在其存储结构中位于第400个元素。通过计算得到：

[8-(-5)+1]×[4-(-3)+1]×(13-10)+[8-(-5)+1]×[1-(-3)]+2-(-5)+1

=14×8×3+14×4+8=400

Back

8.5 数组的输入输出

- 使用DO循环语句
- 使用隐含DO循环
- 使用数组名片段

●使用DO循环语句输入输出部分或全部数组元素。

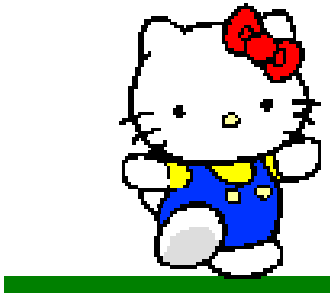
```
INTEGER  arr（8）
DO i=1,8,2 !输入1,3,5,7元素
  READ *,arr（i）
ENDDO
DO i=1,8   !输入全部元素
  READ *,arr（i）
ENDDO
DO i=1,8,2 !输出1,3,5,7元素
  PRINT *,arr（i）
ENDDO
DO i=1,8   !输出全部元素
  PRINT *,arr（i）
ENDDO
```

(示例1)(示例2)

```
INTEGER  arr（3，4）
DO i=1,3 !输入全部元
  素
  DO j=1，4
    READ *,arr（i，j）
  ENDDO
ENDDO
DO i=1,3 !输出全部元
  素
  DO j=1，4
    PRINT *,arr（i，j）
  ENDDO
ENDDO
```

注意：每行输入输出一个

元素值



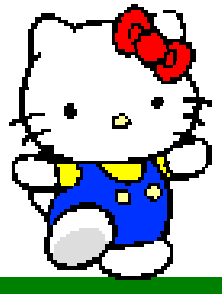
8.5 数组的输入输出

- 使用DO循环语句
- 使用隐含DO循环
- 使用数组名片段

●使用隐含DO循环输入输出部分或全部数组元素。

```
INTEGER  arr1（8），arr2(3,4)
!输入第1、3、5、7数组元素
READ *,(arr1（I）,I=1,8,2)
!输入全部元素
READ *,(arr1（I）,I=1,8)
!输出1、3、5、7数组元素
PRINT *,(arr1（I）,I=1,8,
!输出全部元素
PRINT *,(arr1（I）,I=1,8)
!输入全部元素
READ *,((arr2（I,J）,J=1,4),I=1,3)
!输出全部元素
PRINT *,((arr2（I,J）,J=1,4),I=1,3)
```

注意：每行可输入输出一个或多个数组元素值



(示例1)(示例2)

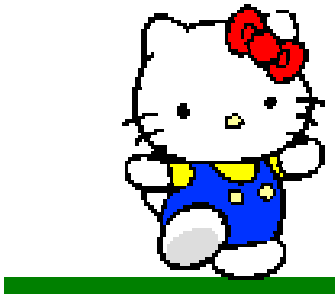
8.5 数组的输入输出

- 使用DO循环语句
- 使用隐含DO循环
- 使用数组名片段

●使用数组名、数组片段输入输出部分或全部数组元素。

```
INTEGER  arr1（8），arr2(3,4)
!输入第1、3、5、7数组元素
READ *,(arr1（I：I）,I=1,8,2)
!按存储结构输入全部元素
READ *,arr1
!输出1、3、5、7数组元素
PRINT *,(arr1（I：I）,I=1
!按存储结构输出全部元素
PRINT *,arr1
!按存储结构输入全部元素
READ *,arr2
!按存储结构输出全部元素
PRINT *,arr2
```

注意：每行可输入输出一个或多个数组元素值



(示例1)(示例2)

Back

使用DO循环语句输入输出数组示例1

[例8.1]：已知5个学生成绩：85,94,78,51,35。按序输入,统计平均成绩,然后修改第1,3,5个学生成绩,再统计平均成绩,输出5个学生最新成绩及修改前后平均成绩。

```
PARAMETER(N=5)
REAL :: sum=0,av1,av2,s
WRITE(*,"(' 输入',I2, '名学生
DO i=1,N
  READ *,score(i);sum=su
ENDDO
av1=sum/N
DO i=1,N,2
  WRITE(*,"(' 输入第',I1, '个
  READ *,score(i)
ENDDO
sum=0
DO i=1,N
  sum=sum+score(i)
ENDDO
av2=sum/N
PRINT *,N,'名学生的最新成绩是：'
DO i=1,N-1
  WRITE（*,"(' ',F5.1,',',\) "）score(i)
ENDDO
WRITE（*,"(' ',F5.1)"）score(N)
WRITE(*,"(' 更新前平均成绩是：',F7.2)")
av1
WRITE(*,"(' 更新后平均成绩是：',F7.2)")
av2
END
```

输入数据：输入 5 名学生的成绩,每行输入一个成绩：

85↙

94↙

78↙

51↙

35↙

输入第1 个学生新成绩：11↙

输入第3 个学生新成绩：33↙

输入第5 个学生新成绩：55↙

输出结果：　　5名学生的最新成绩是：

11.0，94.0，33.0，51.0，55.0

更新前平均成绩是：　68.60

更新后平均成绩是：　48.80

使用DO循环语句输入输出数组示例2

[例8.2]：已知矩阵A,输入矩阵数据并转置输出该矩阵。

矩阵A

12.5	23.8	54.2	78.9
92.4	45.3	0.0	24.4
84.2	32.8	72.8	43.2

```
PROGRAM exam8_2
PARAMETER(M=3,N=4)
REAL A(M,N)
WRITE(*,"(' 以行为主输入',I3,' 个
DO i=1,M
  DO j=1,N
    READ *,A(i,j)
  ENDDO
ENDDO
PRINT *,'输出转置后数组：'
DO i=1,N
  DO j=1,M
    WRITE(*,"(' ',F4.1,' '\)") A(j,i)
  ENDDO
  PRINT*
ENDDO
END
```

输入数据：以行为主输入 12 个矩阵数据,每行输入一个数据：

12.5↙
23.8↙
54.2↙
78.9↙
92.4↙
45.3↙
0↙
24.4↙
84.2↙
32.8↙
72.8↙
43.2↙

输出结果：输出转置后数组：

12.5 92.4 84.2
23.8 45.3 32.8
54.2 0.0 72.8
78.9 24.4 43.2

使用隐含DO循环输入输出数组示例1

[例8.1]：已知5个学生成绩：85,94,78,51,35。按序输入,统计平均成绩,然后修改第1,3,5个学生成绩,再统计平均成绩,输出5个学生最新成绩及修改前后平均成绩。

```
PROGRAM exam8_1
PARAMETER(N=5)
REAL :: sum=0,av1,av2,s
WRITE(*,“(‘ 输入’ ,I2, ‘名学
READ *,(score(i),i=1,N)
DO i=1,N
  sum=sum+score(i)
ENDDO
av1=sum/N
PRINT*, ‘输入第奇数个学生
READ *,(score(i),i=1,N,2)
sum=0
DO i=1,N
  sum=sum+score(i)
ENDDO
av2=sum/N
PRINT *,N,'名学生的最新成绩是：'
WRITE(*,"(F5.1,',',\) “)(score(i),i=1,N-1)
WRITE（*,"(F5.1)"）score(N)
WRITE(*,"(' 更新前平均成绩是： ',F7.2)")
av1
WRITE(*,"(' 更新后平均成绩是： ',F7.2)")
av2
END
```

输入数据：输入 5 名学生的成绩,每行输入多个成绩：

85，94，78，51，35↙

输入第奇数个学生新成绩：

11,33,55↙

输出结果：　　5名学生的最新成绩是：

11.0，94.0，33.0，51.0，55.0

更新前平均成绩是：　68.60

更新后平均成绩是：　48.80

使用隐含DO循环输入输出数组示例2

[例8.2]：已知矩阵A,输入矩阵数据并转置输出该矩阵。

矩阵A

$$\begin{pmatrix} 12.5 & 23.8 & 54.2 & 78.9 \\ 92.4 & 45.3 & 0.0 & 24.4 \\ 84.2 & 32.8 & 72.8 & 43.2 \end{pmatrix}$$

```
PROGRAM exam8_2
PARAMETER(M=3,N=4)
REAL A(M,N)
WRITE(*,“(‘ 按逻辑结构输入',I3,' 个矩阵数据：')")  M*N
READ *,((A(i,j),j=1,N),i=1,M)
 PRINT *,‘输出转置后数组：’
WRITE(*,"(<M>(' ',F4.1,' '))") ((A(i,j),i=1,M),j=1,N)
 END
```

输入数据：以行为主输入 12 个矩阵数据,每行输入一个数据：

12.5 23.8 54.2 78.9↙

92.4 45.3 0 24.4↙

84.2 32.8 72.8 43.2↙

输出结果：输出转置后数组：

12.5 92.4 84.2

23.8 45.3 32.8

54.2 0.0 72.8

78.9 24.4 43.2

使用数组名数组片段输入输出数组示例1

[例8.1]：已知5个学生成绩：85,94,78,51,35。按序输入,统计平均成绩,然后修改第1,3,5个学生成绩,再统计平均成绩,输出5个学生最新成绩及修改前后平均成绩。

```
PROGRAM exam8_1
PARAMETER(N=5)
REAL :: sum=0,av1,av2,scor
WRITE(*,“(‘ 输入’ ,I2, ‘名学生
READ *,score  !或READ *,sco
DO i=1,N
  sum=sum+score(i)
ENDDO
av1=sum/N
PRINT*, ‘输入第奇数个学生新
READ *,(score(i：i),i=1,N,2)
sum=0
DO i=1,N
  sum=sum+score(i)
ENDDO
av2=sum/N
PRINT *,N,'名学生的最新成绩是：'
WRITE(*,"(F5.1,',',\) “) score(1:N-1)
WRITE（*,"(F5.1)"）score(N)
WRITE(*,"(' 更新前平均成绩是：',F7.2)") av1
WRITE(*,"(' 更新后平均成绩是：',F7.2)") av2
END
```

输入数据：输入 5 名学生的成绩,每行输入多个成绩：

85，94，78，51，35↙

输入第奇数个学生新成绩：

11,33,55↙

输出结果：　　5名学生的最新成绩是：

11.0，94.0，33.0，51.0，55.0

更新前平均成绩是：　68.60

更新后平均成绩是：　48.80

使用数组名数组片段输入输出数组示例2

[例8.2]：已知矩阵A,输入矩阵数据并转置输出该矩阵。

矩阵A

$$\begin{pmatrix} 12.5 & 23.8 & 54.2 & 78.9 \\ 92.4 & 45.3 & 0.0 & 24.4 \\ 84.2 & 32.8 & 72.8 & 43.2 \end{pmatrix}$$

```
PROGRAM exam8_2
PARAMETER(M=3,N=4)
REAL A(M,N),B(N,M)
WRITE(*,“(‘ 按存储结构输入',I3,' 个矩阵数据：')") M*N
READ *,A
DO I=1,M
  DO J=1,N
    B(J,I)=A(I,J)
  ENDDO
ENDDO
```

```
PRINT *,‘按存储结构输出转置后数组：’
WRITE(*,*) B
END
```

输入数据：按存储结构输入 12 个矩阵数据：

12.5,23.8,54.2,78.9,92.4,45.3,0,24.4,84.2,32.8,72.8,43.2↙

输出结果：输出转置后数组：

12.5 92.4 84.2 23.8 45.3 32.8 54.2 0.0 72.8
78.9 24.4 43.2

8.6 给数组赋初值

- 数组构造器
- DATA语句

- 使用数组构造器按存储结构次序给数组赋初值。
- 数组构造器是由若干数据或隐含DO循环组成的一个线性表。
- 数组构造器一般给一维数组赋初值，对于多维数组需使用标准函数RESHAPE将数组构造器转换为与多维数组逻辑结构一致的形式，给数组赋初值。
- 一般格式：<数组构造器>→（/<数组初值表>/）
  <数组初值表>→<初值项>{,<初值项>}
  <初值项>→<初值>|<隐含DO循环>
- 示例：INTEGER :: num1(5)=(/2,4,6,,8,10/),i=4,j=8
  INTEGER,DIMENSION(5) :: num2,num3,num4,B(2,3),C(8)
  num2=(/2,i,6,j,10/)
  num3=(/2,ABS(-4),6,INT(8.5),10/)
  num4=(/2,4,3*i-j+2,8,10/)
  B=RESHAPE((/1,2,3,4,5,6/),(/2,3/))
  C=(/RESHAPE(B,(/6/)),7,8/)

(示例)

Back

8.6 给数组赋初值

- 数组构造器
- DATA语句

●使用DATA语句可以给普通变量、整个数组、数组片段、单一数组元素、结构体、结构体组件赋初值。称为赋初值语句。

●一般格式：

<DATA语句>→DATA <对象表>/<初值表>/ {,<对象表>/<初值表>/}

<对象表>→<变量>|<数组>|<数组片段>|<数组元素>|<结构体>

|<隐含DO循环>

<初值表>→<初值项>{,<初值项>}

<初值项>→<初值>|<重复系数>*<初值>

●示例：

```
CHARACTER (LEN = 10) name
INTEGER(1) :: m,n,num(-5:5),prices(0:9)
COMPLEX  cstuff
REAL, DIMENSION (100, 100) :: array
DATA m,n/2*1/,num/1,2,7*5,8,9,/,cstuff/(2.5,-3.5)/
DATA name /'Bai Yun'/, prices/3*0,7*25.5/
DATA ((array(k,j),j=1,k),k=1,100) /5050*1.0/
```

使用数组构造器给数组赋初值示例

[例8.3]：已知矩阵A,内部赋初值并按逻辑结构输出该矩阵。

矩阵A

2	8	14	20
4	10	16	22
6	12	18	24

```
PROGRAM exam8_3
PARAMETER(M=3,N=4)
REAL A(M,N)
A=RESHAPE((/2,4,6,8,10,12,14,16,18,20,22,24/),(/3,4/))
!A=RESHAPE((/(I,I=2,24,2)/),(/3,4/))
PRINT*,'输出数组A：'
WRITE(*,"(<N>(' ',I3,' '))") (A(I,J),J=1,N),I=1,M)
END
!输出结果同例题中矩阵。
```

8.7 动态数组

- 概述
- 声明
- 分配
- 示例

- 静态数组优点是存储分配算法简单,运行速度快,缺点是存储开销较大。
- 在元素个数不确定时,为考虑最坏情况,静态数组声明的很大,导致浪费。
- 动态数组优点是存储开销小，缺点是存储分配算法复杂,运行速度慢。
- 示例：统计学生成绩，人数不定。

```
PROGRAM  example871    !使用静态数组程序
INTEGER score(100),i,j,n,x
REAL :: av,sum=0.0
WRITE(*,“(1X,‘请输入班级学生人数：’,\)”)
READ*,n
WRITE(*,“(1X,‘请输入’,I3,‘名学生成绩：’)”) n
READ*,(score(i),i=1,n)
DO i=1,n
sum=sum+score(i)
ENDDO
av=sum/n
WRITE(*,"(1X,'该班学生平均成绩是：',F5.1)") av
END
```

```
PROGRAM  example872          !使用动态数组程序
INTEGER,DIMENSION(:),ALLOCATABLE :: score(:)
INTEGER i,j,n,x
REAL :: av,sum=0.0
WRITE(*,“(1X,‘请输入班级学生人数：’,\)”)
READ*,n
ALLOCATE(score(n))
WRITE(*,“(1X,‘请输入’,I3,‘名学生成绩：’)”) n
READ*,(score(i),i=1,n)
DO i=1,n
sum=sum+score(i)
ENDDO
av=sum/n
WRITE(*,"(1X,'该班学生平均成绩是：',F5.1)") av
END PROGRAM
```

8.7 动态数组

- 概述
- 声明
- 分配
- 示例

●使用动态数组，需进行声明。声明时不指定大小。

●一般格式：

<动态数组声明语句>→<类型声明符>,DIMENSION(:{,:}),ALLOCATABLE::<数组表>

<数组表>→<数组名>{,<数组名>}

或者

<动态数组声明语句>→<类型声明符>,ALLOCATABLE :: <数组表>

<数组表>→<数组名>(:{,:}){,<数组名>(:{,:})}

●示例：

```
INTEGER,DIMENSION(:),ALLOCATABLE :: num,score
REAL,DIMENSION(:,:) A,B
INTEGER,ALLOCATABLE :: num(:),score(:)
REAL,ALLOCATABLE :: A(:,:),B(:,:)
```

8.7 动态数组

- 概述
- 声明
- 分配
- 示例

●动态数组声明后,使用ALLOCATE语句为分配存储空间，指定维下界和上界。

●一般格式：

<ALLOCATE语句>→ALLOCATE(<动态数组声明表>[,STAT=<状态变量>])

<动态数组声明表>→<数组名>(<维说明表>){,<数组名>(<维说明表>)}

●ALLOCATE语句维说明表中的下界和上界可以是整型变量或整型表达式。

●ALLOCATE语句中指定数组维数与声明的数组维数相同,否则产生语法错误。

●ALLOCATE通过状态变量获得执行状态,若成功,则状态为0,否则为错误号。

●示例：

```
INTEGER,DIMENSION(:),ALLOCATABLE :: num
REAL,DIMENSION(:,:),ALLOCATABLE :: arr
CHARACTER*10,ALLOCATABLE :: book(:,:,:),name(:)
n=4;m=5;READ *,k
ALLOCATE(num(5),arr(n,m),book(35,4,k),name(n+m+k))
ALLOCATE(num(n),arr(-5:n,m:m+10),book(m,n,k))
ALLOCATE(num(n),arr(-5:n,m:m+10),book(m,n,k))
ALLOCATE(num(:n),arr(-
 5:n,m:m+10),book(m+1,n+2,k+3),STAT=ierr)
```

8.7 动态数组

- 概述
- 声明
- 分配
- 示例

[例8.4]从键盘输入数目不确定的一批正整数,将这批整数保存在数组中,然后反序输出这批整数。分别采用静态数组和动态数组编写程序实现之。

```
!使用静态数组程序
PROGRAM  static_array
INTEGER,PARAMETER:: max=200
INTEGER  :: i,n=0,array(max)
READ *,i
DO WHILE (i.NE.-1)
  n=n+1
  array(n)=I
  READ *,i
ENDDO
PRINT *,'data_number=',n
PRINT *,(array(i),i=n,1,-1)
END
```

Back

```
!使用动态数组程序
PROGRAM  dynamic_array
INTEGER,ALLOCATABLE :: old_array(:),new_array(:)
INTEGER  :: i,n=0
ALLOCATE(new_array(0))
READ *,i
DO WHILE (i.NE.-1)
  n=n+1
  ALLOCATE(old_array(n-1))
  old_array=new_array
  DEALLOCATE(new_array)
  ALLOCATE(new_array(n))
  new_array(1:n-1)=old_array
  new_array(n)=I
  DEALLOCATE(old_array)
  READ *,i
ENDDO
PRINT *,'data_number=',n
PRINT  *,(new_array(i),i=n,1,-1)
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.5]计算N个测试数据的平均值和标准偏差。测试数据个数不定。

解：已知：N个测试数据data为：$X_1,X_2,X_3,\ldots,X_n$。从键盘输入。实型。

求：平均值Xa和标准偏差Xs。实型。

平均值计算公式为：$X_a=(X_1+X_2+X_3+\ldots+X_n)/N$

标准偏差计算公式为：$X_s=\sqrt{\dfrac{\sum_{i=1}^{N}(X_i-X_a)^2}{N-1}}$

测试数据用一个一维数组X表示,平均值用变量Xa表示,标准偏差用变量Xs表示。测试数据个数不确定,假设最大个数max为50,实际个数为N。从键盘输入数据,数据以非数值字符为结束标志。

程序：

输入数据：25.32，18.35，44.78，57.39，85.2，A

输出结果：测试数据： 25.32 18.35 44.78 57.39 85.20

平均值 ： 46.21

标准偏差： 26.74

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.6]根据试卷难易程度,需要对学生成绩做适当调整(加减分数)。给定某班学生成绩和加减分数值,根据加减分数值调整学生成绩,调整成绩大于100分按100分记,小于0分按0分记,按优、良、中、及格和不及格确定学生成绩等级。统计调整前和调整后平均成绩。输出调整前学生成绩和平均成绩,以及调整后学生成绩、成绩等级和平均成绩。

0~59　60~69　70~79　80~89　90~100

不及格　及格　中　良　优

解：已知：学生姓名name、成绩score、加减分数delta。从键盘输入。

求：调整前平均成绩old_av、调整后平均成绩new_av、调整后成绩等级grades。

用四个数组：names、old_scores、new_scoress、grades,分别存储调整前姓名、成绩、调整后成绩、等级。其中：names和grades为字符串数组,old_scores和new_scores为整型数组。

学生人数不确定,假设最大数max为200,实际学生人数为n。

程序:

输入数据和输出结果:

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.7]编写程序实现在给定任意长度的句子字符串中查找指定单词并替换为新的单词,输出替换后的字符串。

解：已知：旧句子串string、旧单词old_word、新单词new_word。

从键盘输入。字符型。

求：新句子string、字符型。

求解该问题需要涉及到三个长度不确定的字符串：句子字符串string、旧单词字符串old_word、新单词字符串new_word。

由于三个字符串长度不定,所以使用三个动态字符数组。

声明三个整型变量：n1、n2、n3,分别存放字符串：string、old_word、new_word的长度。

程序：

输入数据：This is a pen.↙

pen↙

table↙

输出结果：This is a table.

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.8]已知某班学生姓名和成绩。查找在某一分数段内的学生姓名和成绩以及查找到的学生人数。编写程序实现之 。

解：已知:姓名names、成绩scores、分数段下界l_score、上界h_score。

names和scores为数组,用DATA语句赋初值, l_score和h_score

从键盘输入。学生人数为N,假定N=10。

求：分数段内学生姓名names(p)、成绩score(p)，人数num。

采用顺序查找法,用p指示数组scores的查找位置,p初值为1,用scores(p)和变量l_score、h_score比较,如果l_score≤scores(p)≤h_score,则查找成功,否则p增1,重复该过程,直到查找到数组末尾。

程序:

输入数据：60,80↙

输出结果：查找到的学生姓名和成绩：

李强　　65

张大伟　　72

朱小军　　75

查找到的学生人数为：　　3

8.8 数组应用举例

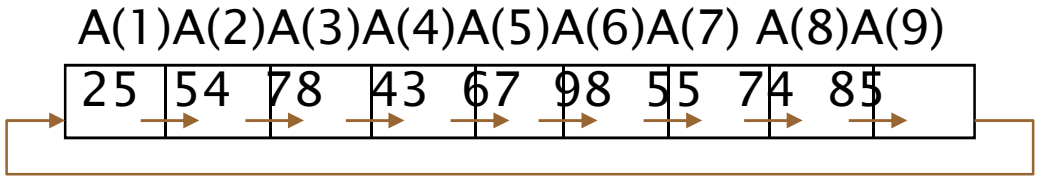
- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.9]已知9个测试数据,对这批数据进行循环移动,即每个数据（除最后一个）向后移动一个位置,最后一个数据移到第一个位置。

解：已知:测试数据A,数组,用DATA语句赋初值。数据个数为N,假定N=9。

求：对数组A进行循环移位。

数组元素移动情况示意如图8-4所示。



变量i指示当前数组元素位置,向后移动语句A(I+1)=A(I)完成:

移动顺序从最后元素开始,将其放在辅助变量temp中,然后从元素A(8)起执行上述赋值语句,直到元素A(1),最后将temp值赋予A(1)。这一移动工作由循环语句实现。

程序:

移动前数据： 25 54 78 43 67 98 55 74 85

输出结果：85 25 54 78 43 67 98 55 74

8.8 数组应用举例

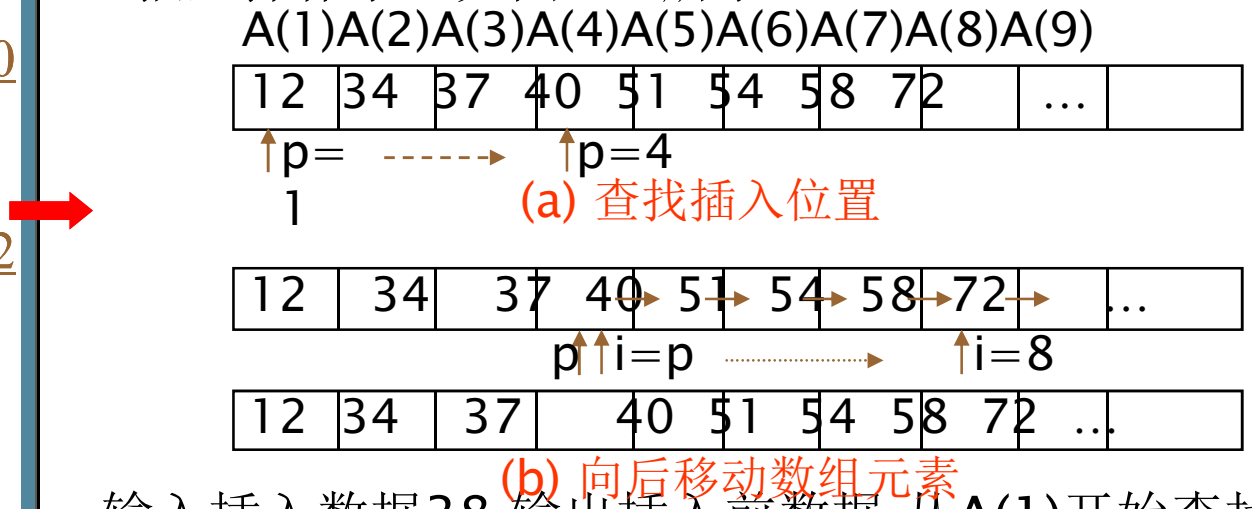
- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.10]已知一组有序整数,将一整数插入到整数序列中使之仍保持有序。

解：已知:有序整数A,数组,用DATA语句赋初值。数据个数为N,假定N=8。

求：将整数x插入到数组A中适当位置。

插入操作示意如图8-5所示。



输入插入数据38,输出插入前数据,从A(1)开始查找插入位置p,从A(8)开始到插入位置p,依次向后移动一位。

程序:

插入前数据： 12 34 37 40 51 54 58 72

输出后数据： 12 34 37 38 40 51 54 58 72

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.11]已知8个无序整数,将这组整数由小到大（或由大到小）排序。

解：已知:无序整数A,数组,用DATA语句赋初值。数据个数为N,假定N=8。

求：将一组无序整数A从小到大排序。

采用选择法排序。基本思想：从第1个数据起在N个待排序数据中选择一个最小数,将最小数与第1个数据交换,称该步为第1趟排序;在从第2个数据起在剩余的N-1个数据中选择一个次最小数,将次最小数与第2个数据交换,称该步为第2趟排序;依次类推,直到在从第N-1个数据起在剩余的2个数据中选择一个较小数,将较小数与第N-1个数据交换,称该步为第N-1趟排序,直至排序任务完成,得到一组有序数据。

排序过程示意如图8-6所示。

程序:

排序前数据： 51 34 37 54 72 12 58 40

排序后数据： 12 34 37 40 51 54 58 72

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.12]已知M×N矩阵A(如：M=3,N=4),求转置矩阵A′以及它们的乘积。

$$A=\begin{bmatrix} 2.0 & 8.0 & 3.0 & 4.0 \\ 5.0 & 4.0 & 1.0 & 8.0 \\ 9.0 & 7.0 & 6.0 & 1.0 \end{bmatrix}$$

解：已知: M×N矩阵A,二维数组,用RESHAPE函数赋初值。假定M=3,N=4。

求：转置矩阵A′及矩阵乘积A•A′。

转置矩阵A′与矩阵A的关系是：

A′(i,j)=A(j,i),i=1,2,…,M, j=1,2,…,N。

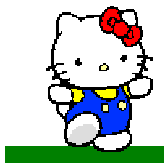
M×N矩阵A与N×M矩阵A′相乘,结果为M×M矩阵C,C与A与A′关系是：

$$C(i,j)=\sum_{k=1}^{N} A(i,k)\times A'(k,j)\quad i=j=1,2,\ldots,M$$

用二维整型数组A存放矩阵A,用二维整型数组B存放A的转置矩阵A′,用一个静态整型数组C存放A与A′的乘积矩阵C。

程序1， 程序2（用矩阵函数实现）， 矩阵函数：

输出结果：



8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.5]计算N个测试数据的平均值和标准偏差。测试数据个数不定。

解：已知：N个测试数据data为：$X_1,X_2,X_3,\ldots,X_n$。从键盘输入。实型。

求：平均值Xa和标准偏差Xs。实型。

平均值计算公式为：$X_a=(X_1+X_2+X_3+\ldots+X_n)/N$

标准偏差计算公式为：$X_s=\sqrt{\frac{\sum_{i=1}^{N}(X_i-X_a)^2}{N-1}}$

测试数据用一个一维数组X表示,平均值用变量Xa表示,标准偏差用变量Xs表示。测试数据个数不确定,假设最大个数max为50,实际个数为N。从键盘输入数据,数据以非数值字符为结束标志。

程序：

输入数据：25.32，18.35，44.78，57.39，85.2，A

输出结果：测试数据： 25.32 18.35 44.78 57.39 85.20

平均值 ： 46.21

标准偏差： 26.74

```
PROGRAM  array881
!说明变量和数组
PARAMETER（max=50）
INTEGER :: N=0
REAL
X(max),Xa,Xs,sum=0.0,data
!从键盘输入数据并求和
DO WHILE (.TRUE.)
  READ(*,*,IOSTAT=io) data
  IF (io<>0) EXIT
  N=N+1
  X(N)=data
  sum=sum+data
ENDDO
Xa=sum/N
sum=0.0
DO i=1,N
  sum=sum+(X(i)-Xa)**2
ENDDO
Xs=SQRT(sum/(N-1))
WRITE(*,"(1X,'  测  试  数  据  :
',<N>F10.2)") (X(i),i=1,N)
WRITE(*,"(1X,'平均值：',F10.2)") Xa
WRITE(*,"(1X,'标准偏差：',F10.2)") Xs
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.6]根据试卷难易程度,需要对学生成绩做适当调整(加减分数)。给定某班学生成绩和加减分数值,根据加减分数值调整学生成绩,调整成绩大于100分按100分记,小于0分按0分记,按优、良、中、及格和不及格确定学生成绩等级。统计调整前和调整后平均成绩。输出调整前学生成绩和平均成绩,以及调整后学生成绩、成绩等级和平均成绩。

0~59　60~69　70~79　80~89　90~100

不及格　及格　中　良　优

解：已知：学生姓名name、成绩score、加减分数delta。从键盘输入。

求：调整前平均成绩old_av、调整后平均成绩new_av、调整后成绩等级grades。

用四个数组：names、old_scores、new_scoress、grades,分别存储调整前姓名、成绩、调整后成绩、等级。其中：names和grades为字符串数组,old_scores和new_scores为整型数组。

学生人数不确定,假设最大数max为200,实际学生人数为n。

程序：

输入数据和输出结果：

```
PROGRAM  array882
PARAMETER（max=200）
INTEGER :: n=0,io,delta,score,old_scores(max),new_scores(max)
CHARACTER*10 :: name,names(max),grades(max)*6
REAL :: sum=0,old_av,new_av
OPEN(1,FILE='array881.dat')
DO WHILE （.NOT. EOF(1)）
  READ(1,*) name,score
  n=n+1;sum=sum+score;names(n)=name;old_scores(n)=score
ENDDO
old_av=sum/n
WRITE(*,"(1X,'输入一个加减分数值：',\)")
READ*,delta
DO i=1,n   !调整学生成绩
  new_scores(i)=old_scores(i)+delta; sum=sum+new_scores(i)
 ENDDO
```

```
WHERE(new_scores>100) new_scores=100; WHERE(new_scores<0)
new_scores=0
sum=0.0
DO i=1,n
  sum=sum+new_scores(i)
  IF (new_scores(i)<60) THEN
     grades(i)='不及格'
  ELSEIF (new_scores(i)<70) THEN
     grades(i)='及格'
  ELSEIF (new_scores(i)<80) THEN
     grades(i)='中'
  ELSEIF (new_scores(i)<90) THEN
     grades(i)='良'
  ELSE
     grades(i)='优'
    ENDIF
  ENDDO
```

```
new_av=sum/n
!输出数据
WRITE(*,"(1X,'学生姓名',2X,'调整前成绩',2X,'调整后成绩',2X,'成绩等级')")
WRITE(*,"(1X,44A1)") ('-',i=1,44）
DO i=1,n
    WRITE(*,100) names(i),old_scores(i),new_scores(i),grades(i)
ENDDO
WRITE(*,"(1X,44A1)") ('-',i=1,44)
WRITE(*,"(1X,'调整前平均成绩：',F7.2)") old_av
WRITE(*,"(1X,'调整后平均成绩：',F7.2)") new_av
100 FORMAT(1X,A10,2X,I10,2X,I10,2X,A6)
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.6]根据试卷难易程度,需要对学生成绩做适当调整(加减分数)。给定某班学生成绩和加减分数值,根据加减分数值调整学生成绩,调整成绩大于100分按100分记,小于0分按0分记,按优、良、中、及格和不及格确定学生成绩等级。统计调整前和调整后平均成绩。输出调整前学生成绩和平均成绩,以及调整后学生成绩、成绩等级和平均成绩。

0~59　60~69　70~79　80~89　90~100

不及格　及格　中　良　优

解：已知：学生姓名name、成绩score、加减分数delta。从键盘输入。

求：调整前平均成绩old_av、调整后平均成绩new_av、调整后成绩等级grades。

用四个数组：names、old_scores、new_scoress、grades,分别存储调整前姓名、成绩、调整后成绩、等级。其中:names和grades为字符串数组,old_scores和new_scores为整型数组。

学生人数不确定,假设最大数max为200,实际学生人数为n。

程序:

输入数据和输出结果:

使用数据文件,按规定格式输入有关输入数据,如：

王刚　,45
赵平全　,78
司马俊男,87
杨莉红　,98
张军　,58

运行程序,从键盘输入数据：输入一个加减分数值：10↙

程序运行后,输出结果为：学生姓名　调整前成绩　调整后成绩　成绩等级

王刚	45	55	不及格
赵平全	78	88	良
司马俊男	87	97	优
杨莉红	98	100	优
张军	58	68	及格

调整前平均成绩：　73.20
调整后平均成绩：　81.60

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.7]编写程序实现在给定任意长度的句子字符串中查找指定单词并替换为新的单词,输出替换后的字符串。

解：已知：旧句子串string、旧单词old_word、新单词new_word。

从键盘输入。字符型。

求：新句子string、字符型。

求解该问题需要涉及到三个长度不确定的字符串：句子字符串string、旧单词字符串old_word、新单词字符串new_word。

由于三个字符串长度不定,所以使用三个动态字符数组。

声明三个整型变量：n1、n2、n3,分别存放字符串：string、old_word、new_word的长度。

程序：

输入数据：This is a pen.↙

pen↙

table↙

输出结果：This is a table.

```
PROGRAM  find_replace883
  CHARACTER,DIMENSION(:),ALLOCATABLE                    ::
     string,old_word,new_word
  CHARACTER line(80)
  READ 100, n1, (line(i), i= 1, n1)
  ALLOCATE(string(n1));  string=line
  READ 100, n2, (line(i), i= 1, n2)
  ALLOCATE(old_word(n2)); old_word=line
  READ 100, n3, (line(i), i= 1, n3)
  ALLOCATE(new_word(n3)); new_word=line
100 FORMAT (Q,80A1)
  WRITE(*,"(1X,\)")
  k=1
  DO WHILE (k .LE. n1)
    DO WHILE (string(k).NE.old_word(1) .AND. k .LE. n1)
      WRITE(*,"(A1,\)") string(k); k=k+1
    ENDDO
```

```
    IF (k .LE. n1) THEN
      i=k;j=1
      DO WHILE (string(i)= =old_word(j) .AND. i .LE. n1 .AND. j .LE.
n2)
                  i=i+1;j=j+1
      ENDDO
      IF (j>n2) THEN
            k=k+n2;  WRITE(*,"(<n3>A1,\)") (new_word(m),m=1,n3)
      ELSEIF (i>n1) THEN
        WRITE(*,"(A1,\)") (string(m),m=k,n1); k=n1+1
      ELSE
        WRITE(*,"(A1,\)") string(k);  k=k+1
      ENDIF
    ENDIF
 ENDDO
 PRINT*
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.8]已知某班学生姓名和成绩。查找在某一分数段内的学生姓名和成绩以及查找到的学生人数。编写程序实现之 。

解：已知:姓名names、成绩scores、分数段下界l_score、上界h_score。

names和scores为数组,用DATA语句赋初值, l_score和h_score

从键盘输入。学生人数为N,假定N=10。

求：分数段内学生姓名names(p)、成绩score(p)，人数num。

采用顺序查找法,用p指示数组scores的查找位置,p初值为1,用scores(p)和变量l_score、h_score比较,如果l_score≤scores(p)≤h_score,则查找成功,否则p增1,重复该过程,直到查找到数组末尾。

程序:

输入数据：60,80↙

输出结果：查找到的学生姓名和成绩:

李强	65
张大伟	72
朱小军	75

查找到的学生人数为： 3

```
PROGRAM  array884
 !说明变量和数组
  PARAMETER（N=5）
  INTEGER :: score,l_score,h_score,num=0
  INTEGER,DIMENSION(N) :: scores
  CHARACTER*10,DIMENSION(N) :: names
  !用DATA语句给数组赋予学生姓名和成绩数据
  DATA  names/'王枫','李强','张大伟','朱小军','杨惠娟'/
  DATA  scores/58,65,72,75,95/
  !从键盘输入查找分数段下界和上界
  PRINT *,'输入查找分数段下界和上界：'
  READ *,l_score,h_score
  PRINT*,'查找到的学生姓名和成绩：'
```

```
  PRINT*,'查找到的学生姓名和成绩：'
  DO p=1,N
    IF    (l_score<=scores(p).AND.scores(p)<=h_score)
THEN
      WRITE(*,"(1X,A10,I5)") names(p),scores(p)
      num=num+1
    ENDIF
  ENDDO
 !输出查找到的人数
 WRITE(*,"(1X, '查找到的学生人数为：', I4)") num
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

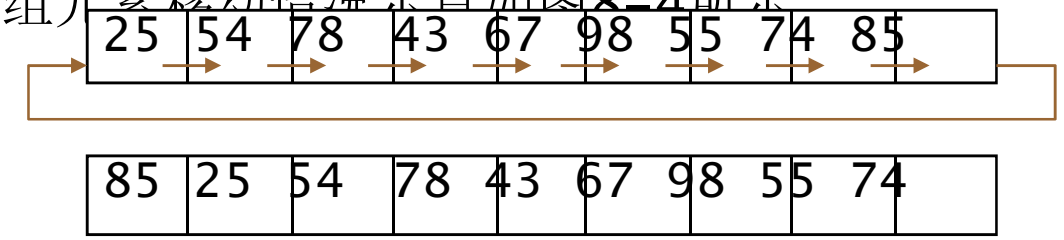
[例8.9]已知9个测试数据,对这批数据进行循环移动,即每个数据（除最后一个）向后移动一个位置,最后一个数据移到第一个位置。

解：已知:测试数据A,数组,用DATA语句赋初值。数据个数为N,假定N=9。

求：对数组A进行循环移位。

数组元素移动情况示意如图8.4所示。

A(1)A(2)A(3)A(4)A(5)A(6)A(7) A(8)A(9)



变量i指示当前数组元素位置,向后移动语句A(I+1)=A(I)完成:

移动顺序从最后元素开始,将其放在临时变量temp中,然后从元素A(8)起执行上述赋值语句,直到元素A(1),最后将temp值赋予A(1)。

这一移动工作由循环语句实现。

程序:

移动前数据： 25 54 78 43 67 98 55 74 85

输出结果：85 25 54 78 43 67 98 55 74

```
PARAMETER（N=9）
INTEGER :: temp,I,A(N)
DATA  A/25,54,78,43,67,98,55,74,85/
PRINT*,'移动前数据：'
WRITE(*,”(1X,9I4)”) (A(I),I=1,N)
temp=A(N)
DO I=N-1, 1, -1
  A(I+1)=A(I)
ENDDO
A(1)=temp
PRINT*,'移动后数据：'
WRITE(*,”(1X,9I4)”) (A(I),I=1,N)
END
```

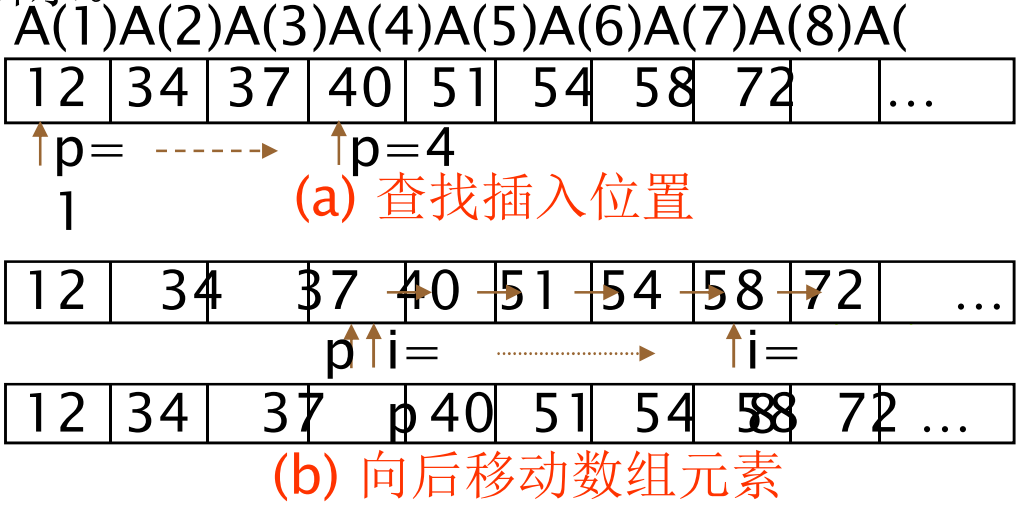
8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.10]已知一组有序整数,将一整数插入到整数序列中使之仍保持有序。

解：已知:有序整数A,数组,用DATA语句赋初值。数据个数为N,假定N=8。

求：将整数x插入到数组A中适当位置。插入操作示意如图8-5所示。



输入插入数据38,输出插入前数据,从A(1)开始查找插入位置p,从A(8)开始到插入位置p,依次向后移动一位。

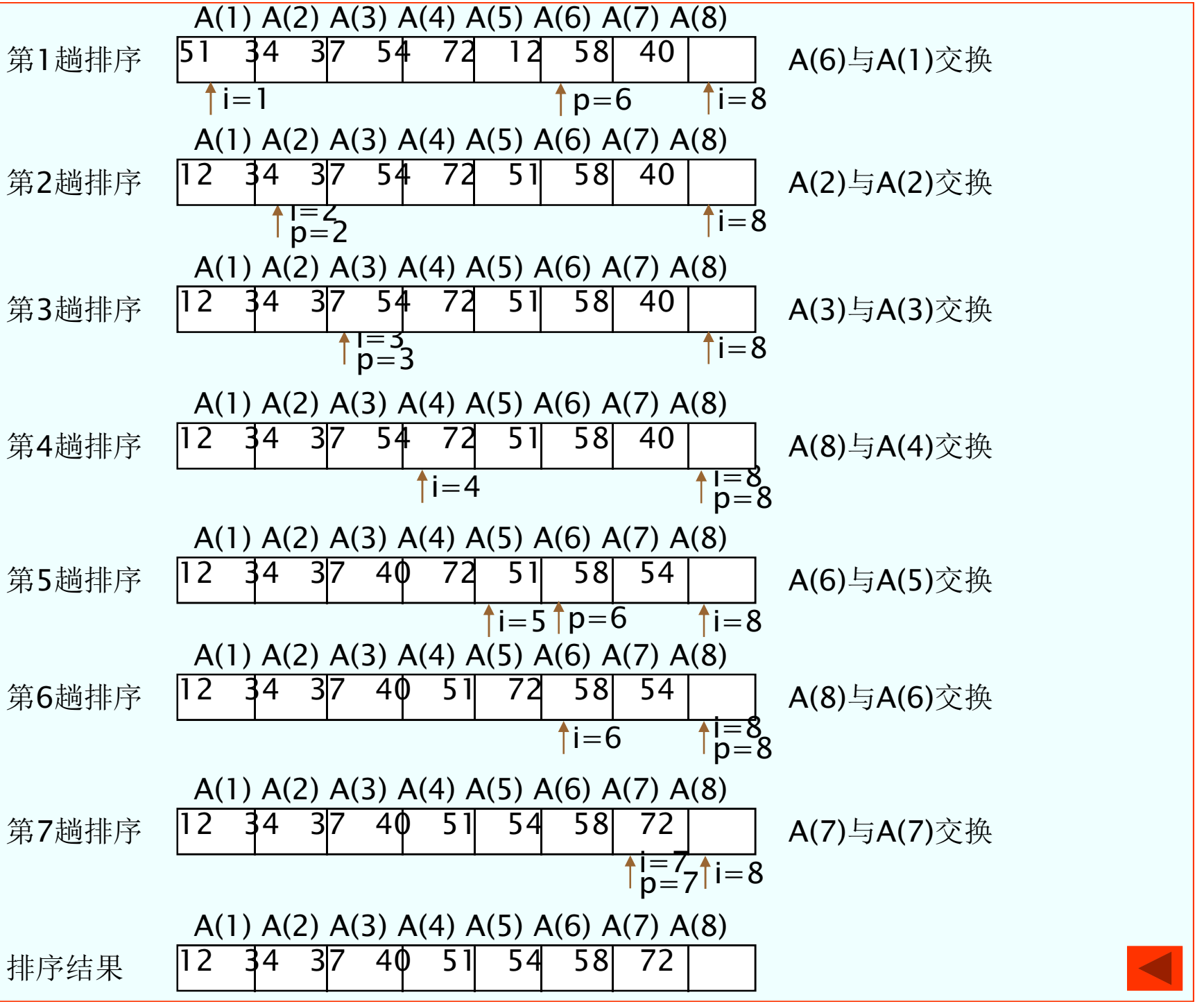
程序:

移动前数据： 12 34 37 40 51 54 58 72

插入后数据： 12 34 37 38 40 51 54 58 72

```
PROGRAM  array8810
PARAMETER（max=20）
INTEGER :: x, p, i, A(max), N=8
!用DATA语句给数组赋初值
DATA  A/12, 34, 37, 40, 51, 54, 58, 72/
READ *,x    !输入插入数据x
!输出插入前数组数据
PRINT*,‘插入前数据：’
WRITE(*,100) (A(i),i=1,N)
IF (N>=max) THEN  !检查数组大小
   PRINT *, ‘数组太小,空间不够,不能插入数据。’
ELSE
   p=1
```

```
    DO WHILE （x>=A(p) .AND. p<=N）
      p=p+1
    ENDDO
    DO i=N, p, -1
      A(i+1)=A(i)
    ENDDO
    A(p)=x
    PRINT*,‘插入后数据：’
    WRITE(*,200) (A(i),i=1,N+1)
  ENDIF
100  FORMAT(1X,(1X,<N>I4))
200  FORMAT(1X,(1X,<N+1>I4))
  END
```


A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第1趟排序
51 34 37 54 72 12 58 40
i=1
p=6
i=8
A(6)与A(1)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第2趟排序
12 34 37 54 72 51 58 40
i=2
p=2
i=8
A(2)与A(2)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第3趟排序
12 34 37 54 72 51 58 40
i=3
p=3
i=8
A(3)与A(3)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第4趟排序
12 34 37 54 72 51 58 40
i=4
i=8
p=8
A(8)与A(4)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第5趟排序
12 34 37 40 72 51 58 54
i=5
p=6
i=8
A(6)与A(5)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第6趟排序
12 34 37 40 51 72 58 54
i=6
i=8
p=8
A(8)与A(6)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
第7趟排序
12 34 37 40 51 54 58 72
i=7
p=7
i=8
A(7)与A(7)交换
A(1) A(2) A(3) A(4) A(5) A(6) A(7) A(8)
排序结果
12 34 37 40 51 54 58 72

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.11]已知8个无序整数,将这组整数由小到大（或由大到小）排序。

解：已知:无序整数A,数组,用DATA语句赋初值。数据个数为N,假定N=8。

求：将一组无序整数A从小到大排序。

采用选择法排序。基本思想：从第1个数据起在N个待排序数据中选择一个最小数,将最小数与第1个数据交换,称该步为第1趟排序;在从第2个数据起在剩余的N-1个数据中选择一个次最小数,将次最小数与第2个数据交换,称该步为第2趟排序;依次类推,直到在从第N-1个数据起在剩余的2个数据中选择一个较小数,将较小数与第N-1个数据交换,称该步为第N-1趟排序,直至排序任务完成,得到一组有序数据。

排序过程示意如图8-6所示。

程序:

排序前数据： 51 34 37 54 72 12 58 40

排序后数据： 12 34 37 40 51 54 58 72

```
 PARAMETER（N=8）
 INTEGER :: temp, p, i, j, A(N)
 DATA  A/51, 34, 37, 54, 72, 12, 58, 40/
 PRINT*,'排序前数据：'
 WRITE(*,100) (A(i),i=1,N)
 DO i=1, N-1
   p=I
   DO j=i, N  !查找最小数位
     IF (A(j)<A(p))  p=j
   ENDDO
   temp=A(p);A(p)=A(i);A(i)=temp
 ENDDO
PRINT*,‘排序后数据：’
WRITE(*,”(1X,<N>I4)”) (A(i),i=1,N)
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.12]已知M×N矩阵A(如：M=3,N=4),求转置矩阵A′以及它们的乘积。

$$A=\begin{bmatrix} 2.0 & 8.0 & 3.0 & 4.0 \\ 5.0 & 4.0 & 1.0 & 8.0 \\ 9.0 & 7.0 & 6.0 & 1.0 \end{bmatrix}$$

解：已知: M×N矩阵A,二维数组,用RESHAPE函数赋初值。假定M=3,N=4。

求：转置矩阵A′及矩阵乘积A•A′。

转置矩阵A′与矩阵A的关系是：

A′(i,j)=A(j,i),i=1,2,…,M, j=1,2,…,N。

M×N矩阵A与N×M矩阵A′相乘,结果为M×M矩阵C,C与A与A′关系是：

$$C(i,j)=\sum_{k=1}^{N} A(i,k)\times A'(k,j)\quad i=j=1,2,\ldots,M$$

用二维整型数组A存放矩阵A,用二维整型数组B存放A的转置矩阵A′,用一个静态整型数组C存放A与A′的乘积矩阵C。

程序1， 程序2（用矩阵函数实现）， 矩阵函数：

输出结果：

```
PROGRAM  array8812
!说明变量和数组
  PARAMETER（M=3, N=4）
  INTEGER :: i,j,k
  REAL :: A(M,N),B(N,M),C(M,M)
  !用数组构造器和内部标准函数RESHAPE给数组赋初值

     A=RESHAPE((/2.0,5.0,9.0,8.0,4.0,7.0,3.0,1.0,6.0,4.0,8.0,1.0/)
     ,(/3,4/)
  PRINT*,‘原矩阵：’
  WRITE(*,”(<N>(F5.1,2X))”) ((A(i,j),j=1,N),i=1,M)
  DO i=1, M          !求转置矩阵A',即数组B
     DO j=1, N
       B(j,i)=A(i,j)
     ENDDO
  ENDDO
```

```
  !输出转置矩阵A',即数组B
  PRINT*,‘转置矩阵：’
  WRITE(*,”(<M>(F5.1,2X))”) ((B(i,j),j=1,M),i=1,N)
  DO i=1, M    !求矩阵A与转置矩阵A'的乘积
    DO j=1, M
      C(i,j)=0
      DO k=1, N
        C(i,j)=C(i,j)+A(i,k)*B(k,j)
      ENDDO
    ENDDO
  ENDDO
  !输出乘积矩阵C
  PRINT*,‘乘积矩阵：’
  WRITE(*,”(<M>(F7.2,2X))”) ((C(i,j),j=1,M),i=1,M)
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.12]已知M×N矩阵A(如：M=3,N=4),求转置矩阵A′以及它们的乘积。

$$A=\begin{bmatrix} 2.0 & 8.0 & 3.0 & 4.0 \\ 5.0 & 4.0 & 1.0 & 8.0 \\ 9.0 & 7.0 & 6.0 & 1.0 \end{bmatrix}$$

解：已知: M×N矩阵A,二维数组,用RESHAPE函数赋初值。假定M=3,N=4。

求：转置矩阵A′及矩阵乘积A•A′。

转置矩阵A′与矩阵A的关系是：

A′(i,j)=A(j,i),i=1,2,…,M, j=1,2,…,N。

M×N矩阵A与N×M矩阵A′相乘,结果为M×M矩阵C,C与A与A′关系是：

$$C(i,j)=\sum_{k=1}^{N} A(i,k)\times A'(k,j)\quad i=j=1,2,\ldots,M$$

用二维整型数组A存放矩阵A,用二维整型数组B存放A的转置矩阵A′,用一个静态整型数组C存放A与A′的乘积矩阵C。

程序1， 程序2（用矩阵函数实现）， 矩阵函数：

输出结果：

```
PROGRAM  array8812
PARAMETER（M=3, N=4）
  INTEGER :: i,j,k
  REAL :: A(M,N),B(N,M),C(M,M)
  !用数组构造器和内部标准函数RESHAPE给数组赋初值

     A=RESHAPE((/2.0,5.0,9.0,8.0,4.0,7.0,3.0,1.0,6.0,4.0,8.0,1.0/)
     ,(/3,4/)
  PRINT*,'原矩阵：'
  WRITE(*,”(<N>(F5.1,2X))”) ((A(i,j),j=1,N),i=1,M)
  B=TRANSPOSE(A)   !求转置矩阵A',即数组B
  PRINT*,'转置矩阵：'
  WRITE(*,”(<M>(F5.1,2X))”) ((B(i,j),j=1,M),i=1,N)
  C=MATMUL(A,B)    !求矩阵A与转置矩阵A‘的乘积
  PRINT*,'乘积矩阵：'
  WRITE(*,”(<M>(F7.2,2X))”) ((C(i,j),j=1,M),i=1,M)
END
```

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.12]已知M×N矩阵A(如：M=3,N=4),求转置矩阵A′以及它们的乘积。

$$A=\begin{bmatrix} 2.0 & 8.0 & 3.0 & 4.0 \\ 5.0 & 4.0 & 1.0 & 8.0 \\ 9.0 & 7.0 & 6.0 & 1.0 \end{bmatrix}$$

解：已知: M×N矩阵A,二维数组,用RESHAPE函数赋初值。假定M=3,N=4。

求：转置矩阵A′及矩阵乘积A•A′。

转置矩阵A′与矩阵A的关系是：

A′(i,j)=A(j,i),i=1,2,…,M, j=1,2,…,N。

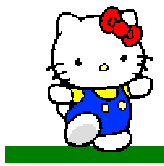
M×N矩阵A与N×M矩阵A′相乘,结果为M×M矩阵C,C与A与A′关系是：

$$C(i,j)=\sum_{k=1}^{N} A(i,k)\times A'(k,j)\quad i=j=1,2,\ldots,M$$

用二维整型数组A存放矩阵A,用二维整型数组B存放A的转置矩阵A′,用一个静态整型数组C存放A与A′的乘积矩阵C。

程序1， 程序2（用矩阵函数实现）， 矩阵函数：

输出结果：



函 数 名	功　能
ALL(mask[,dim])	判断逻辑数组mask在选定维上的所有元素是否都为真
ANY(mask[,dim])	判断逻辑数组mask在选定维上的任一元素是否为真
COUNT(mask[,dim])	求逻辑数组mask在选定维上的为真元素个数
CSHIFT(array,shift[,dim])	对数组选定维上的数组元素进行一或多次循环移位
DOT_PRODUCT(vector_a,vector_b)	执行向量的点积运算
LBOUND(array[,dim])	求数组在选定维上的下界
UBOUND(array[,dim])	求数组在选定维上的上界
MATMUL(array_a,array_b)	执行二维矩阵的乘积运算
MAXLOC(array[,mask])	求满足指定掩码条件的最大值的位置
MINLOC(array[,mask])	求满足指定掩码条件的最小值的位置
MAXVAL(array[,dim][,mask])	求在所选定维上的满足指定掩码条件的最大值
MINVAL(array[,dim][,mask])	求在所选定维上的满足指定掩码条件的最小值
SUM(array[,dim][,mask])	求在所选定维上的满足指定掩码条件的数组元素之和
MERGER(tsource,fsource,mask)	根据所给定的掩码条件合并两个数组
RESHAPE(source,shape[,pad][,order])	重新定义数组的形状
TRANSPOSE(array)	执行二维矩阵的转置运算
SIZE(array[,dim])	求数组在所选定维上的大小

8.8 数组应用举例

- 例8.5
- 例8.6
- 例8.7
- 例8.8
- 例8.9
- 例8.10
- 例8.11
- 例8.12

[例8.12] 已知M×N矩阵A(如：M=3,N=4),求转置矩阵A′以及它们的乘积。

$$A=\begin{bmatrix} 2.0 & 8.0 & 3.0 & 4.0 \\ 5.0 & 4.0 & 1.0 & 8.0 \\ 9.0 & 7.0 & 6.0 & 1.0 \end{bmatrix}$$

解：已知: M×N矩阵A,二维数组,用RESHAPE函数赋初值。假定M=3,N=4。

求：转置矩阵A′及矩阵乘积A•A′。

转置矩阵A′与矩阵A的关系是：

$A'(i,j)=A(j,i), i=1,2,\ldots,M,\ j=1,2,\ldots,N$。

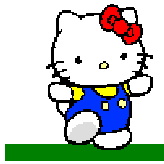
M×N矩阵A与N×M矩阵A′相乘,结果为M×M矩阵C,C与A与A′关系是：

$$C(i,j)=\sum_{k=1}^{N} A(i,k)\times A'(k,j)\quad i=j=1,2,\ldots,M$$

用二维整型数组A存放矩阵A,用二维整型数组B存放A的转置矩阵A′,用一个静态整型数组C存放A与A′的乘积矩阵C。

程序1， 程序2（用矩阵函数实现）， 矩阵函

输出结果：



输出结果：原矩阵：

2.0 8.0 3.0 4.0
5.0 4.0 1.0 8.0
9.0 7.0 6.0 1.0

转置矩阵：

2.0 5.0 9.0
8.0 4.0 7.0
3.0 1.0 6.0
4.0 8.0 1.0

乘积矩阵：

93.00 77.00 96.00
77.00 106.00 87.00
96.00 87.00 167.00

习题八

1. 何谓数组？有何特征？试举例说明。
2. 何谓静态数组和动态数组？两者有何异同点？
3. 有哪些数组声明方式？举例说明这些方式。
4. 按照下面的数组声明,指出每个数组的类型、大小及数组每维的下界和上界。

```
INTEGER  count(-15:11),day(7),high(5:15,8)
REAL prices(-5:10,5:20),scores(35),temp(7,12)
COMPLEX  c1(5,5:10,-4:5),c2(-10:-2,2:8,9,2:10)
CHARACTER*10 str1(15),str2(-1:20,1:20)
LOGICAL   log1(4:10),log2(-3:12,-8:-2,9)
```

5. 执行下面含有数组元素引用的赋值语句,给出每个赋值语句执行后的数组变化情况。

```
INTEGER :: vector(10)=(/1,2,3,4,5,6,7,8,9,10/)
i=2;j=4
vector(3:7)=15
```

习题八

vector=4*vector(i)+1
vector(:5)=vector(10)
vector(i+j:10)=2*vector(i+j+4)
vector(:)=3
vector(i-1:j+1)=vector(6:i*j+2)+j

6. 何谓数组的逻辑结构何存储结构？哪一种结构与计算机存储器有关？FORTRAN 90规定数组在计算机存储器中以什么方式存储？

7. 已知数组声明“INTEGER num(3,4)”,数组内容如下面所示：

$$num=\begin{bmatrix}1 & 4 & 7 & 10\\ 2 & 5 & 8 & 11\\ 3 & 6 & 9 & 12\end{bmatrix}$$

按照逻辑结构和存储结构次序分别写出数组输入和输出语句(如：READ、WRITE、DATA、赋值语句等)。

8. 动态数组声明与静态数组声明有何不同？动态数组使用过程中还需要有哪些配套语句？试举例说明。

习题八

9．输入10个整数,求其逆序并输出,编写程序实现之。

10．已知10个整数,输入一个待删除整数,在10个整数中查找,若有该整数,则将其从10个整数中删除,输入输出删除前后这组整数。编写程序实现之。

11．已知一组整数(个数不定),将数组中所有相同的数删除到只剩一个。编写程序实现之。

12．在一个M×N的矩阵中求绝对值最大和最小元素所在的行号和列号。编写程序实现之。

13．输入一个班学生（人数不多于40人）的学号和5门课成绩,统计每位学生的平均成绩,打印该班学生成绩表。编写程序实现之。

14．输入一个班学生(人数≤40)的学号和5门课成绩,统计每个分数段(0-59,60-69，70-79,80-89,90-99,100)的学生人数及所占比例,输出统计结果。编写程序实现之。

15．输入4个整数放在数组中,数组元素向左循环移位1、2、3、4个位置,结果也放在数组中,输出移位前后的结果数据。如输入：2、5、8、3,输出结果为右面方阵所示。编写程序实现之。

2	5	8	3
5	8	3	2
8	3	2	5
3	2	5	8
2	5	8	3

习题八

16. 从n个数中查找最小数,并将其插入到第一个数前。编写程序实现之。

17. 已知N×N方阵,求对
线元素之和,并输出。
编写程序实现之。

18. 对两个同阶矩阵相加
编写程序实现之。

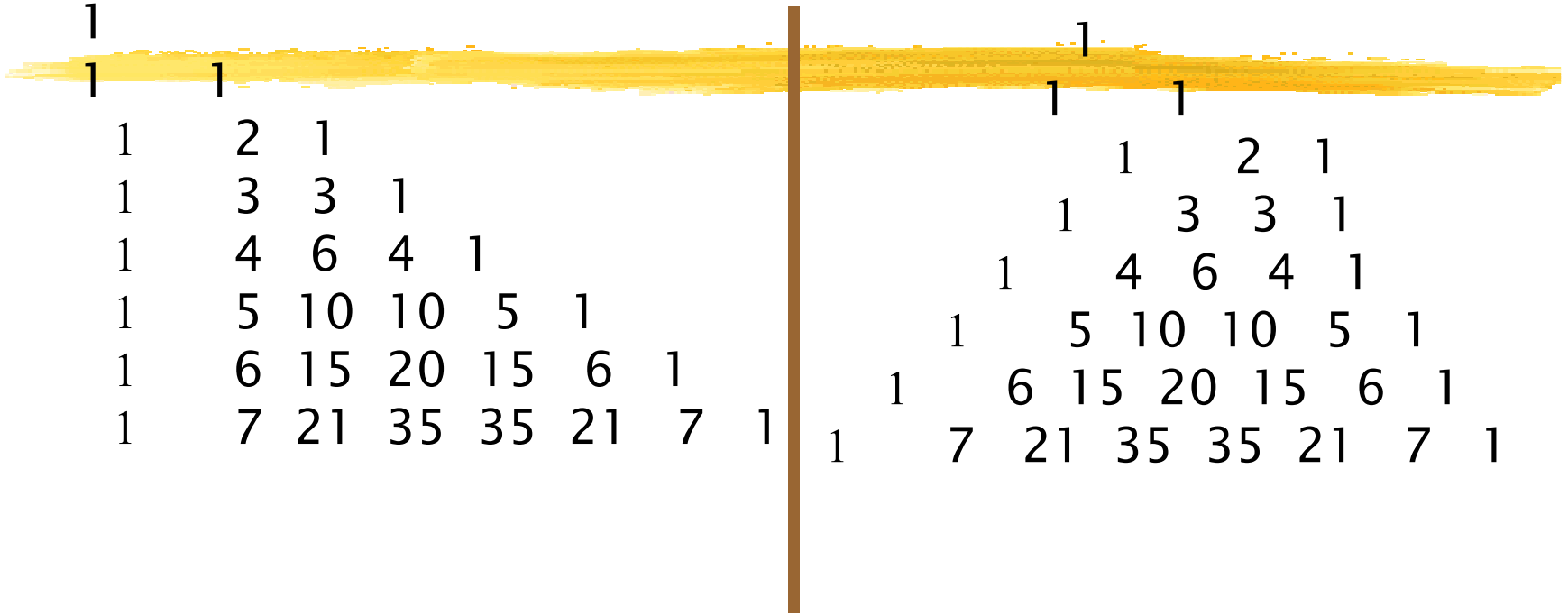
19. 已知M×N矩阵,求每
行元素之和,将和最
大的行与第一行对调,
输出对调前后矩阵。
编写程序实现之。

20. 用数组生成九九乘法
表,按右侧格式打印该
表。编写程序实现之。

```
***** 九九乘法表 *****
     1   2   3   4   5   6   7   8   9
-------------------------------------
---------
  1     1
  2     2   4
  3     3   6   9
  4     4   8  12  16
  5     5  10  15  20  25
  6     6  12  18  24  30  36
  7     7  14  21  28  35  42  49
  8     8  16  24  32  40  48  56  64
  9     9  18  27  36  45  54  63  72  81
-------------------------------------
---------
```

习题八

21．编写程序打印下面两种格式的杨辉三角形。



22．打印“魔幻方阵。魔幻方阵是指方阵的每一行、每一列、对角线元素之和均相等,其值为$n(n^2+1)/2$,n代表方阵的行数和列数。一个n阶奇数方阵由1到n^2个自然数构成,其组成规律是:

(1)将1放在第1行的中间一列;

(2)由1开始依次从左下到右上方向放连续数,如m放在A(i,j),则m+1放在A(i-1,j+1);

习题八

(3)如果i-1<1,则i-1为n;
(4)如果j+1>n,则j+1为1;
(5)如果A(i-1,j+1)已放数,则将数放在A(i+1,j)中。
以下是五阶魔幻方阵:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

23. 试采用其它方法编写程序,对一组整数进行排序。

24. 用高斯消元法求解右面齐次线形方程组。编写程序实现之。

$$\begin{cases} a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+\ldots+a_{2n}x_n=b_2 \\ \vdots \\ a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n=b_n \end{cases}$$

The end

文件名格式：班级 学号 姓名 简略实验名称

邮件标题同文件名

Any questions please 发送至

xingzhengwu@163.com