成都理工大学 | 地球物理学院

《地球与空间探测数据处理方法》

第二章 数据分析工具：IDL编程

陶 丹

Email：adam.tao@hotmail.com

Address: 北翼楼（地物院）5814室

课程要求和考核方式

课程基础

高等数学/线性代数/概率论与数理统计
高级程序语言与程序设计(C)/Matlab 程序设计

学习目的

1. 掌握IDL可视化分析工具的基础内容，能够独立程序编写并处理地球空间数据。
2. 能够熟悉了解空间探测数据类型及数据文件的存储格式，并能够熟练利用IDL可视化工具来实现空间探测数据的读写、数据格式的转换以及相关可视化分析等。
3. 能够熟练掌握空间探测数据的坐标变化、数据的平滑处理、滤波、谱分析、相关性分析、拟合、平均值统计、中值及误差分析等方法。

考核方式

平时成绩（40%）+ 考试成绩（60%）

其中：

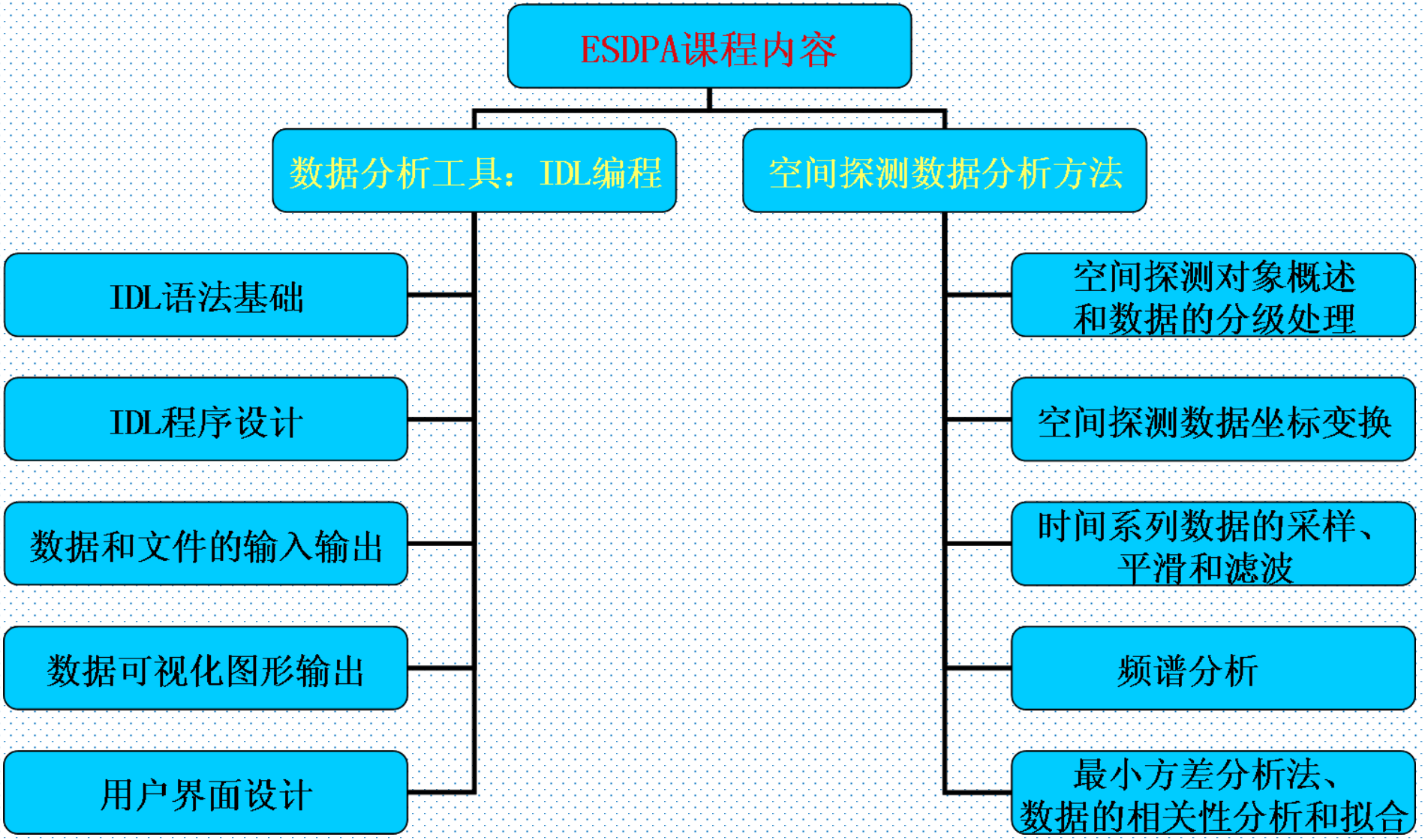
平时成绩（40%）：考勤（10%）+ 大作业/实验报告（30%）

考试成绩（60%）：期末考试（60%，闭卷）

参考书目

- 《IDL可视化工具入门与提高》，闫殿武编著，机械工业出版社，2003.
- 《IDL可视化分析与应用》，韩培友编著，西北工业大学出版社，2006.
- 《IDL程序设计—数据可视化与ENVI二次开发》，董彦卿编著，高等教育出版社，2012.

地球与空间探测数据处理方法课程架构



2.6 数组

一组有序数据的集合，由一组名字相同，而下标不同的多个元素组成。每个数组元素相当于一个内存变量。数组是数据处理分析中最常用的工具。

- **数组格式**
  - 格式：数组名［下标列表］，可以创建1-8维的数组。

    数组名：数组的名字，其命名的规则与变量命名规则相同。

    下标列表：表示数组中数组元素的位置，可以是常量、变量、数组或表达式；若为实数，则自动取整；元素类型必须相同；下标列表中下标要用逗号隔开。

    例如二维数组ARRAY[N,M]，表示N列M行（与其他语言有别），按行排列，0为下标起点，N-1或M-1结束。
  - 数组元素的总数等于各维数组的元素个数的乘积（N×M）。

“先创建后使用”

- **三种创建方式**
  - 直接创建法：数组名=[表达式表]（1-3维）　例如：*IDL>ARRAY=[7.0,5.0]　ARRAY[0]=7.0*
  - 函数创建法：数组名=数组创建函数（1-8维）　例如：*IDL>ARRAY=INDGEN(100) A=FLTARR(2,3)*
  - 特殊命令创建法：REPLICATE 和 MAKE_ARRAY
- **数组赋值**
  - 格式1：数组名[下标列表]=表达式

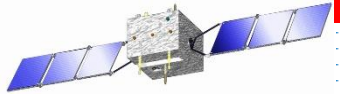
    例如：*IDL>ARRAY=FINDGEN[7,5]*

    IDL>ARRAY[3,4]=89.0
  - 格式2：数组名[(Di1:Di2)]=表达式　（Di1:Di2元素下标范围）

    例如：*IDL>ARRAY=FINDGEN[7,5]*

    IDL>ARRAY[1:4, 2:3]=20170510.

“*号代表任意下标”
“元素必须同类型”



二、IDL语法基础：数组直接创建法

2.6.1 数组的直接创建法应用示范：

- 一维5个元素数组 ARRAY(5)

```
IDL> ARRAY =[0, 1, 2, 3, 4]
IDL> PRINT, ARRAY
      0   1   2   3   4
IDL> ARRAY1=[ARRAY, 5, 6, 7, 8]
IDL> PRINT, ARRAY1
  0     1     2     3     4     5     6     7     8
```

- 2行3列二维数组 ARRAY(3, 2)

```
IDL> ARRAY = [ [1.0, 2.5, 3.8] ,  [4.5, 5.0,6.7] ]
IDL> PRINT,ARRAY
      1.00000      2.50000      3.80000
      4.50000      5.00000      6.70000
IDL> PRINT,ARRAY[0,*]
      1.00000
      4.50000
IDL> PRINT,ARRAY[*,0]
      1.00000      2.50000      3.80000
IDL> PRINT,ARRAY[0,0] & PRINT, ARRAY[0]
      1.00000
      1.00000
```

- 2行（2维）3列（1维）2层（3维）三维数组 ARRAY(3, 2, 2)

```
IDL> ARRAY =[ [ [1, 2, 3] ,  [4, 5 ,6] ] , [ [7,  8, 9] , $ [10,
11 ,12] ]]
IDL> PRINT,ARRAY
      1      2      3
      4      5      6

      7      8      9
     10     11     12
IDL> PRINT,ARRAY[0,*,*]
      1
      4

      7
     10
IDL> PRINT,ARRAY[*,0,*]
      1      2      3

      7      8      9
IDL> PRINT,ARRAY[*,*,0]
      1      2      3
      4      5      6
```

二、IDL语法基础：数组的函数创建法

2.6.2 数组的函数创建法及其应用示范：

数据类型	创建初始值为零的数组	创建初始值为索引号的数组
字节	bytarr	bindgen
16位有符号整型	intarr	indgen
32位有符号整型	lonarr	lindgen
64位有符号整型	lon64arr	l64indgen
16位无符号整型	uintarr	uindgen
32位无符号整型	ulonarr	ulindgen
64位无符号整型	ulon64arr	ul64indgen
浮点型	fltarr	findgen
双精度浮点型	dblarr	dindgen
复数	complexarr	cindgen
双精度复数	dcomplexarr	dcindgen
字符串	strarr	sindgen
指针	ptrarr	-
对象	objarr	-

例如：

```
IDL> ZERO=INTARR(5)
IDL> PRINT,ZERO
       0       0       0       0       0
IDL> INDEX=INDGEN(5)
IDL> PRINT,INDEX
       0       1       2       3       4
IDL> INDEX=INDGEN(5,3)
IDL> PRINT,INDEX
       0       1       2       3       4
       5       6       7       8       9
      10      11      12      13      14
IDL> ZERO_F=FLTARR(2,3)
IDL> PRINT, ZERO_F
      0.00000      0.00000
      0.00000      0.00000
      0.00000      0.00000
IDL> INDEX=FINDGEN(2,3)
IDL> PRINT,INDEX
      0.00000      1.00000
      2.00000      3.00000
      4.00000      5.00000
```

- IDL> PRINT,INTARR(2,/NOZERO) ;/NOZERO初值为随机数
- IDL> PRINT,INDGEN(3 [,*]) ;关键字[,/BYTE |,/COMPLEX |,/FLOAT |,/DOUBLE |,/STRING |,…]

2.6.3 数组的特殊命令创建法：

- **REPLICATE(表达式,D1[,…,D8])** 函数可以创建一个特定维度的数组，并将表达式的值复制到该数组中。

```
IDL> X=REPLICATE(0.0,5,2)
IDL> PRINT,X
      0.000000    0.000000    0.000000    0.000000    0.000000
      0.000000    0.000000    0.000000    0.000000    0.000000
```

- **MAKE_ARRAY(D1[,…,D8][,*][,/VALUE=value][,/INDEX][,/NOZERO])**用来创建一个特定维度的数组，并把VALUE的值赋给该数组。

```
IDL> Y=MAKE_ARRAY(3,2,/FLOAT, VALUE=12.5)
IDL> PRINT,Y
      12.5000     12.5000     12.5000
      12.5000     12.5000     12.5000
IDL> Y=MAKE_ARRAY(3,2,/STRING,/INDEX)
IDL> PRINT,Y
      0       1       2
      3       4       5
IDL> HELP,Y
      Y       STRING    = ARRAY[3, 2]
```

二、IDL语法基础：数组的函数创建法

- 创建数组的关键字TYPE的代码及含义：

- **INDGEN(D1[,…,D8][,*])**

例如：*IDL> PRINT,INDGEN(3,/FLOAT)*

0.000000 1.00000 2.00000

IDL> X=INDGEN(3,TYPE=7)

IDL> HELP,X

X STRING = ARRAY[3]

* 关键字[,/BYTE |,/COMPLEX |,/DCOMPLEX |,/FLOAT |,/DOUBLE |,/L64 |,/LONG |,/STRING |,/UINT |,/UL64 |,/ULONG][,TYPE=value]

表 2.6 关键字 TYPE 的代码及其含义

创建数组的类型	类型代码	类型名称	创建数组的类型	类型代码	类型名称
未 定 义	0	UNDEFINED	结构型	8	STRUCT
字 节 型	1	BYTE	双精度复数	9	DCOMPLEX
有符号整型	2	INT	指针型	10	POINTER
浮 点 型	4	FLOAT	无符号整型	12	UINT
双精度浮点型	5	DOUBLE	无符号长整型	13	ULONG
复 数	5	COMPLEX	64 位有符号整型	14	LONG64
字 符 串	7	STRING	64 位无符号整型	15	ULONG64

二、IDL语法基础：数组

数组的存储

按列存储
同FORTRAN语言

- 一维数组ARRAY[N]

  ARRAY[0]->ARRAY[1]->…->ARRAY[N-1]

- 二维数组ARRAY[N,M]

  ARRAY[0,0] ->ARRAY[1,0] ->…->ARRAY[N-1,0] ->

  -> ARRAY[0,1] ->ARRAY[1,1] ->…->ARRAY[N-1,1] ->

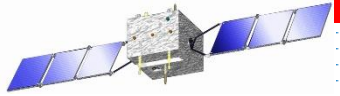
  ……

  -> ARRAY[0,M-1]->ARRAY[1,M-1]->…->ARRAY[N-1,M-1]

- 三维或多维数组类推

  例如：ARRAY[N,M,K]

  ARRAY[N,M,0] ->ARRAY[N,M,1] ->…->ARRAY[N,M,K-1]



二、IDL语法基础：数组

- 数组的引用方式
  - 格式1：数组名（调用整个数组）

    例如：*IDL>ARRAY=INDGEN(6, 8)*

    IDL>PRINT, ARRAY 相当于PRINT, ARRAY[,*] 或ARRAY[0:(N-1),0:(M-1)]*
  - 格式2：数组名[下标列表]（调用下标指定元素）

    例如：*IDL>Z0=ARRAY[0,1] +99*
  - 格式3：数组名[下标范围]（调用局部元素）

    例如：*IDL>Z1=TOTAL(ARRAY[0:1,1:2])*
  - 格式4：(数组名表达式)[下标列表]（调用下标指定元素参加指定运算）

    例如：*IDL>ARRAY1=(ARRAY*3+4)[*,*])*

2.7 表达式和运算法则

表达式是是指用运算符把常量、变量、数组和函数按照一定的语法规则连接起来的有意义的式子。表达式中所有的字符必须写在一行上。

表达式运算后有一个确定的值，根据其类型可分为四种：

数值型表达式、字符型表达式、关系型表达式和逻辑性表达式

- 数值型运算符（算术运算符）

数值型运算符和含义如下：

（）	圆括号	＋	加	＋＋	自加	－	减	－－	自减
MOD	模	*	乘	/	除	^	乘方	＜	求最小
＞	求最大	[ ]	数组或下标	#	矩阵列乘	##	矩阵行乘	•	结构成员操作

数值型运算符的优先级，按由高到低依次如下：

（） [] → • → ^ ＋＋ －－ → * # ## / MOD → ＋ －

数值型运算符的结合方向：同等级运算符由左向右结合。

- ++和--：自加或自减

  例如：*++/--X：X的值先加减1，在使用；X++/--：先使用X的值，然后再加减1*

- < 和 >：求最小值和最大值

  例如：*IDL>PRINT,13>15>67>4*

  67

- # 和 ##：矩阵列乘和行乘
- 表达式 **expN**：

  例如：*9*sqrt(x^2-3*y)-8*

二、IDL语法基础：表达式和运算法则

- 字符型运算符
  - +：将两个字符串直接连接成新字符串
  - 表达式**expC**

    例如：*IDL> “WELCOME TO”+ “CDUT!”*

- 关系型运算符

关系型运算符及其含义如下：

EQ	等于	LT	小于	GT	大于
NE	不等于	LE	小于等于	GE	大于等于

关系型运算符的优先级：均为同一等级。

关系型运算符的结合方向：由左向右结合。

- 表达式 **expL**

  运行结果为逻辑值，即真（1）和假（0）。

逻辑型运算符

逻辑型运算符及其含义如下：

～	逻辑非运算	&&	逻辑与运算	||	逻辑或运算
NOT	按位非运算	AND	按位与运算	OR	按位或运算XOR 按位异或运算

逻辑型运算符的优先级由高到低依次为 NOT→AND→OR/XOR→～→&&→||。

结合方向：NOT 和～ 的结合方向为由右向左；其他的结合方向为由左向右。

表达式 expL

逻辑运算的真值表见表2.8和2.9（A、B均为关系型或者逻辑性表达式）

表 2.8　逻辑运算的真值表

A	B	A&&B	A‖B	～A
真	真	真	真	假
真	假	假	真	假
假	真	假	真	真
假	假	假	假	真

表 2.9　按位逻辑运算的真值表

		AND	OR	XOR	NOT
1	1	1	1	0	0
1	0	0	1	1	0
0	1	0	1	1	1
0	0	0	0	0	1

二、IDL语法基础：表达式和运算法则

- 赋值表达式
  - 赋值运算符(op)

赋值运算符及其含义如下：

=	# # =	# =	* =	+ =	− =	/=	<=	>=	AND=
EQ=	GE=	GT=	LE=	LT=	MOD=	NE=	OR=	XOR=	^=

赋值运算符的结合方向：由右向左。

  - 赋值表达式
    - 格式：变量1 op（变量2 op（…(变量n op 表达式））)

      将表达式的值依次赋给变量n、n-1、…、1。

  - 符合运算符（cop） （除 = 之外）
    - 格式：变量cop表达式

      将变量和表达式的值先进行复合运算，然后再把运算结果赋值给变量。

例如：*IDL>Y=77*

*IDL>X=(Y *= 3)* *;等价于Y=Y*3，然后X=Y*

IDL>PRINT,X,Y

231 231

二、IDL语法基础：表达式和运算法则

- 表达式一般格式
  - 运算符（op）优先级和结合方向

表 2.10　IDL 6.3 运算符的优先级和结合

优先级	运　　算　　符	结合方向
1	( )圆括号　[ ]数组	由内向外
2	.成员访问　( )函数调用圆括号　[ ]数组下标	从右向左
3	* 指针　^(从左向右)　++　--	从右向左
4	* 乘　/　#　##　MOD	从左向右
5	+　-　<　>　NOT(从右向左)	从左向右
6	EQ　NE　LE　LT　GT　GE	从左向右
7	AND　OR　XOR	从左向右
8	&&　||　~(从右向左)	从左向右
9	expR1? expr2:expr3	由内向外
10	=　##=　#=　*=　+=　-=　/=　<=　>=　AND=　EQ=　GE=　GT=　LE=　LT=　MOD=　NE=　OR=　XOR=　^=	从右向左

- 一般格式：
  表达式 op 表达式 … op 表达式

二、IDL语法基础：表达式和运算法则

- 其它运算法则

- 表达式结果类型由“=”号右边优先级最高的变量类型决定。双精度最优先，…，“byte”优先级最低。

例如：

IDL> PRINT,12.3+1

13.3000

高类型 -> 低类型

数值范围

常用算符含义和优先顺序

数学运算符	含义	优先级（1最高）	关系或逻辑（布尔）运算符	含义	优先级（1最高）
()	小括号	1	Not	布尔非	4
*	指针符号	2	Eq	等于	5
^	指数符号	2	Ne	不等于	5
*	标量乘	3	Le	小于等于	5
#	数组乘	3	Lt	小于	5
##	矩阵乘	3	Ge	大于等于	5
/	除	3	Gt	大于	5
Mod	求模	3	And	布尔与	6
+	加	4	Or	布尔或	6
-	减或负	4	Xor	布尔异或	6
<	最小	4	?:	三重运算	7
>	最大	4			
=	赋值号	8			

表达式中多重括号由内往外运算。

二、IDL语法基础：表达式和运算符

- 运算符使用注意事项：
  - 避免两个整数直接相除。由于表达式结果类型由右边的计算变量决定，所以两个整数直接相除会发生截断，从而得到错误结果。正确的除法需要将其中的一个变为浮点数 。例如：

```
IDL> X=4/5                          IDL> X=4/5.0
IDL> PRINT,X    ?                   IDL> PRINT,X    ?
```

  - 避免使用（EQ, NE) 来比较两个不同类型的变量。

```
IDL> A=7.01
IDL> B=7.01D
IDL> IF (A EQ B) THEN PRINT,'TRUE' ELSE PRINT,'FALSE'
      FALSE
IDL> A=7.01
IDL> B=7.01
IDL> IF (A EQ B) THEN PRINT,'TRUE'
      TRUE
```

  - 关系运算符和布尔运算符联合使用时同括号分隔（常用于IF语句）。如：

```
IDL> A=[1.0,4.5,6.7,8.9,10.1]
IDL> FOR I=0,4 DO BEGIN & IF (A[I] GT 1.0) AND (A[I] LT 10.0) THEN PRINT,A[I] & ENDFOR
      4.50000    6.70000    8.90000
IDL> FOR I=0,4 DO BEGIN & IF (A[I] GT 10.0) OR (A[I] LT 5.0) THEN PRINT,A[I] & ENDFOR
      1.00000    4.50000    10.1000
```

- 取大“>”或取小“<”运算可以将数组中一些错误数据设置为0或其他判断参数：

```
IDL> PRINT, 4<5, 5<5, 6<5
       4       5       5
IDL> PRINT, (-1)>3, 3>1
       3       3
IDL> X=[-999.999,20.0,100.0,-999.999,300.0,-999.999]
IDL> Y=X>0
IDL> PRINT,Y
      0.000000     20.0000     100.000     0.000000     300.000     0.000000
```

二、IDL语法基础：运算符在数组中的运用

- 查找数组元素中符合条件的数组元素的下标（数组元素的定位）

  - 格式：**INDEX = WHERE (*Array_Expression* [, *Count*] [, *COMPLEMENT*=*variable*1] [, *NCOMPLEMENT*=*variable*2])**
  - 功能：返回数组中满足关系运算式的数组元素的下标，同时也可以**COMPLEMENT**=***variable1*** 中的**variable1**记录下数组中不满足条件的元素的下标。
  - 使用**WHERE**函数时，数组使用一维下标。

  - **Array_expression:** 是由数组名或者数组元素组成的关系运算符或逻辑表达式。
  - **Index:** 存储满足条件的数组元素的下标（一维数组）。
  - **Count:** 存储满足条件的元素的个数。
  - ***Variable*1：** 存储不满足条件的数组元素的下标。
  - ***Variable*2：** 存储不满足条件的数组元素的个数。

- **WHERE函数的应用实例：**

```
ARRAY = INDGEN(10)
PRINT,'ARRAY= ', ARRAY
    ARRAY = 0 1 2 3 4 5 6 7 8 9

INDEX = WHERE(ARRAY GT 5, COUNT, COMPLEMENT=B_C,
 NCOMPLEMENT=COUNT_C)
; PRINT HOW MANY AND WHICH ELEMENTS MET THE SEARCH CRITERIA:

 PRINT, 'NUMBER OF ELEMENTS > 5: ', COUNT
      NUMBER OF ELEMENTS > 5: 4
 PRINT, 'SUBSCRIPTS OF ELEMENTS > 5: ', INDEX
      SUBSCRIPTS OF ELEMENTS > 5: 6 7 8 9
 PRINT, 'NUMBER OF ELEMENTS <= 5: ', COUNT_C
      NUMBER OF ELEMENTS <= 5: 6
 PRINT, 'SUBSCRIPTS OF ELEMENTS <= 5: ', B_C
      SUBSCRIPTS OF ELEMENTS <= 5: 0 1 2 3 4 5
```

2.8 数组的运算

- 元素个数和维数都相同的两个数组进行运算，就是相同下标元素的运算：

```
IDL> A=[1.0,3.0,5.0,7.0,9.0]
IDL> B=[2.0,4.0,6.0,8.0,10.0]

IDL> PRINT,A+B                    ;矩阵的加法
       3.00000      7.00000      11.0000      15.0000      19.0000
IDL> PRINT,A/B
       0.500000     0.750000     0.833333     0.875000     0.900000
```

- 数组和一个标量运算，数组每个元素都和标量运算一次（矩阵的数量乘法）：

```
IDL> X=[20.0,11.0,25.0,45.0]
IDL> Y=X/5.0     ；全部元素值同时除以一个标量。
IDL> PRINT,Y
       4.00000      2.20000      5.00000       9.00000

IDL> A=[10,10,10,10]
IDL> PRINT,A-5
       5       5       5       5
```

二、IDL语法基础：数组的运算

- 如果表达式中两个数组大小不同维数相同，那么作为结果的数组包含的元素等于表达式中最小的数组所包含的元素：

```
IDL> A=[1.0,2.0]
IDL> B=[3.0,4.0,5.0]
IDL> PRINT,A*B
       3.00000      8.00000
```

- 如果大小和维数都不同，则取表达式中最小的数组（元素最少的）的大小和维度：

```
IDL> A=[10,10,10,10,10,10,10,10]
IDL> B=[[1,2],[3,4]]
IDL> PRINT,A*B
       10      20
       30      40
```

- 元素个数相同但维数不同的，则取表达式中左边的数组所包含的维数：

```
IDL> A=[10,10,10,10]
IDL> B=[[1,2],[3,4]]
IDL> PRINT,A*B
       10      20      30      40
IDL> PRINT,B*A
       10      20
       30      40
```

二、IDL语法基础：数组的运算

- #数组列乘：将第一个数组A的列乘以第二个数组B的行，要求B的行数=A的列数。

```
IDL> A=[[1.0,2.0,3.0],[4.0,5.0,6.0]]
IDL> B=[[-1.0,0],[1.0,2.0],[3.0,4.0]]
IDL> PRINT,A#B                          ;C=A#B，C具有A的列和B的行数
      -1.00000    -2.00000    -3.00000
       9.00000     12.0000     15.0000
       19.0000     26.0000     33.0000
```

- ##数组行乘（矩阵乘）：将第一个数组的行乘以第二个数组的列，要求B的列数=A的行数。

```
IDL> PRINT,A##B                         ;C=A##B，C具有B的列和A的行数
       10.0000     16.0000
       19.0000     34.0000
```

附录1：矩阵和的运算

- 设$A=[a_{ij}]$和$B=[b_{ij}]$是$m\times n$矩阵，则它们的和是：

$$A+B=[a_{ij}+b_{ij}]=\begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & \cdots & a_{1n}+b_{1n} \\ a_{21}+b_{21} & a_{22}+b_{22} & \cdots & a_{2n}+b_{2n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ a_{m1}+b_{m1} & a_{m2}+b_{m2} & \cdots & a_{mn}+b_{mn} \end{bmatrix}$$

- 矩阵和数组的区别：
  矩阵的下角标以1开始，而数组是以0开始的，且不同元素之间要用“,”隔开。
  例如数组ARRAY

$$array=\begin{bmatrix} a_{0,0},a_{1,0},a_{2,0} \\ a_{0,1},a_{1,1},a_{2,1} \end{bmatrix}$$

附录2：矩阵的数量乘法

- 设$A=[a_{ij}]$是$m\times n$矩阵，k是常数，则k和A的数量乘积：

$$kA=[ka_{ij}]=\begin{bmatrix} ka_{11} & ka_{12} & \cdots & ka_{1n} \\ ka_{21} & ka_{22} & \cdots & ka_{2n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ ka_{m1} & ka_{m2} & \cdots & ka_{mn} \end{bmatrix}$$

附录3：矩阵乘法

- 设$A=[a_{ij}]$是$m\times n$矩阵，$B=[b_{ij}]$是$n\times s$矩阵：

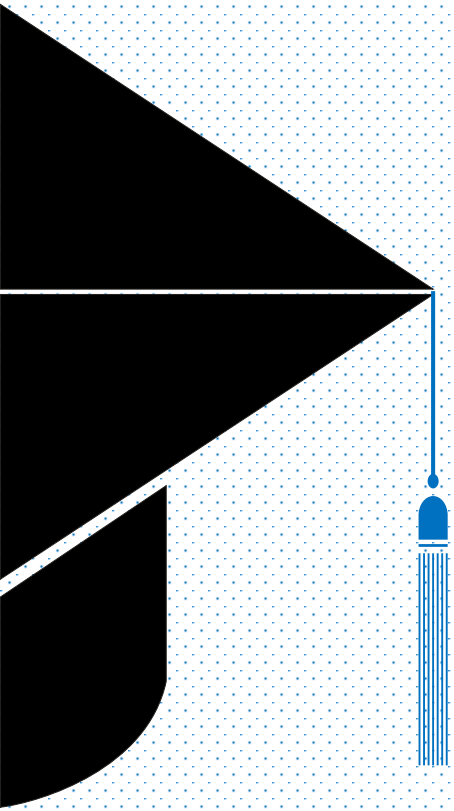
$$A=[a_{ij}]=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad B=[b_{ij}]=b\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1s} \\ b_{21} & b_{22} & \cdots & b_{2b} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{ns} \end{bmatrix}$$

则A与B之乘积AB（记作$C=[c_{ij}]$是一个$m\times s$矩阵：$c_{ij}=a_{i1}b_{1j}+a_{i2}b_{2j}+\ldots+a_{in}b_{nj}=\sum_{k=1}^{n}a_{ik}b_{kj}$

$$AB=\begin{bmatrix} 1 & 0 & -1 & 2 \\ -1 & 1 & 3 & 0 \\ 0 & 5 & 7 & -6 \end{bmatrix}\begin{bmatrix} 0 & -4 \\ 1 & 2 \\ -3 & -2 \\ -1 & 1 \end{bmatrix}$$

$$=\begin{bmatrix} 1\times0+0\times1+(-1)\times(-3)+2\times(-1) & 1\times(-4)+0\times2+(-1)\times(-2)+2\times1 \\ 1\times0+0\times1+(-1)\times(-3)+2\times(-1) & 1\times(-4)+0\times2+(-1)\times(-2)+2\times1 \\ 0\times0+5\times1+7\times(-3)+(-6)\times(-1) & 0\times(-4)+5\times2+7\times(-2)+(-6)\times1 \end{bmatrix}=\begin{bmatrix} 1 & 0 \\ -8 & 0 \\ -10 & -10 \end{bmatrix}$$

成都理工大學
CHENGDU UNIVERSITY OF TECHNOLOGY



演示完毕

Thank You