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：

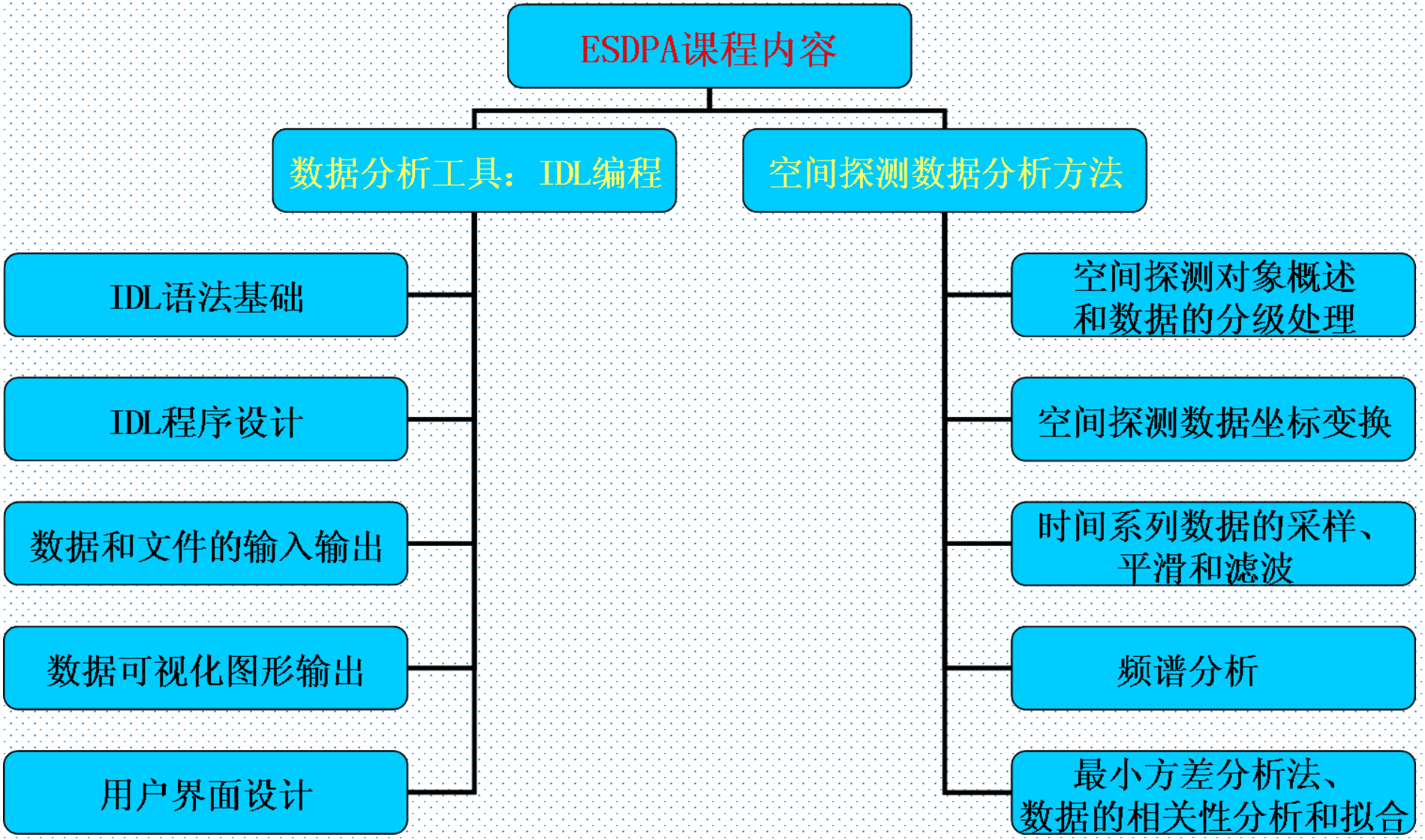
平时成绩（40%）：考勤（10%）+ 大作业/实验报告（30%）

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



二、IDL语法基础：数组的操作

2.9 数组的操作：

函数名称	返回值
N_elements()	数组元素的个数
Size()	数组大小和类型信息
Min()	数组的最小值
Max()	数组的最大值
Mean()	数组平均值
Variance()	数组值的方差
Stddev()	数组的标准偏差
Moment()	平均值（mean），方差（stddev），倾斜度（Skew），峰度（kurtosis），标准差（stddev），平均绝对偏差（mean absolute deviation）
Total()	数组元素的总和

```
IDL> Y=[4.0, 2.2,5.0,9.0]
IDL> PRINT,N_ELEMENTS(Y)
       4
IDL> PRINT,SIZE(Y)
       1(维)  4 (n*m)  4(类型代码)    4(数目)
IDL> PRINT,SIZE(Y，N_DIMENSIONS)
       1
IDL> PRINT,MIN(Y)
       2.20000
IDL> PRINT,MAX(Y)
       9.00000
IDL> PRINT,MEAN(Y)
       5.05000
IDL> PRINT,TOTAL(Y)
       20.2000
```

注意：TOTAL函数总是返回浮点型结果

```
IDL> X=[1,2,3,4,5]
IDL> PRINT,TOTAL(X)
       15.0000
```

二、IDL语法基础：数组的操作

- 求数组最大值：使用**MAX(ARRAY[, MIN=VAR])**函数，且将最小值存储到变量VAR中。

  例如：*IDL>X=INDGEN(8)*

  IDL>PRINT,MAX(X,MIN=VAR0),VAR0

  7 0

- 求数组最小值：使用**MIN(ARRAY[, MAX=VAR])**函数，且将最大值存储到变量VAR中。

  例如：*IDL>X=INDGEN(8)*

  IDL>PRINT,MIX(X,MAX=VAR1),VAR1

  0 7

- 求数组元素个数：使用**N_ELEMENTS(ARRAY)**函数。

  处理未定义的数组或变量时，返回0值。

  例如：*IDL> PRINT,N_ELEMENTS(ARR9)*

  0

二、IDL语法基础：数组的操作

- 求数组大小和类型：使用SIZE函数。

  SIZE(ARRAY[, L64][, /DIMENSIONS | , /N_DIMENSIONS | , /N_ELEMENTS | , /TNAME | , /TYPE])

  -输出数组的维数、列、行、类型和元素个数。

  -可选项依次返回各维大小、维数、元素个数、类型名称及类型代码。

  -/L64，默认情况下，SIZE的结果是一个32位整数，如需要，则返回一个64位整数。

表 2.7 数据类型代码表

类型代码	类型名称	数据类型	类型代码	类型名称	数据类型
0	UNDEFINED	Undefined	8	STRUCT	Structure
1	BYTE	Byte	9	DCOMPLEX	Double-precision complex
2	INT	Integer	10	POINTER	Pointer
3	LONG	Longword integer	11	OBJREF	Object reference
4	FLOAT	Floating point	12	UINT	Unsigned integer
5	DOUBLE	Double-precision floating	13	ULONG	Unsigned longword integer
6	COMPLEX	Complex floating	14	LONG64	Longword 64-bit integer
7	STRING	String	15	ULONG64	Unsigned64-bit integer

例如：*IDL> PRINT,SIZE(FINDGEN(5,7,13))*

3 5 7 13 4 455

二、IDL语法基础：数组的操作

- 求数组平均值：使用MEAN(ARRAY[, /DOUBLE])函数。
  -/DOUBLE，返回双精度值。
  例如：*IDL> PRINT,MEAN(FINDGEN(8))*
  3.50000

- 求数组方差：使用VARIANCE(ARRAY[, /DOUBLE])函数。
  -/DOUBLE，返回双精度值。

- 求数组标准偏差：使用STDDEV(ARRAY[, /DOUBLE])函数。
  -/DOUBLE，返回双精度值。

- 求数组平均值、方差、倾斜度及峰值：使用MOMENT函数。
  MOMENT(ARRAY[,/DOUBLE][,MDEV=VAR1][,SDEV=VAR2)
  -/MDEV,平均绝对偏差
  -/SDEV,标准偏差
  例如：*IDL> PRINT,MOMENT(FINDGEN(7,8),MDEV=VAR1,SDEV=VAR2),VAR1,VAR2*
  27.5000 266.000 0.000000 -1.26445
  14.0000 16.3095

二、IDL语法基础：数组的操作

- 数组的改造：使用**REFORM**函数，不改变元素个数前提下生成新的维数的数组。

  REFORM(ARRAY,D1[,…,D8][,/OVERWRITE])

  -/OVERWRITE，覆盖原数组。

  例如：

```
IDL> PRINT,ARR0
       0       1       2       3       4       5       6       7       8
IDL> ARR1=REFORM(ARR0,3,3)
IDL> PRINT,ARR0,ARR1
       0       1       2       3       4       5       6       7       8
       0       1       2
       3       4       5
       6       7       8
IDL> ARR1=REFORM(ARR0,3,3,/OVERWRITE)
IDL> PRINT,ARR0,ARR1
       0       1       2
       3       4       5
       6       7       8
       0       1       2
       3       4       5
       6       7       8
```

二、IDL语法基础：数组的操作

- 数组的反转：使用**REVERSE**函数。

  REVERSE(ARRAY[, SUBSCRIPT_INDEX][,/OVERWRITE])

  -SUBSCRIPT_INDEX，按指定的维数（≤原数组维数），逆序排列后生成新数组。

  默认按第一维数逆序排序。

  -/OVERWRITE，覆盖原数组。

  例如：

  IDL> ARR0=INDGEN(3,3)

  IDL> PRINT,ARR0,REVERSE(ARR0),$

  REVERSE(ARR0,1),REVERSE(ARR0,2)

  ```
  0   1   2
  3   4   5
  6   7   8
  2   1   0
  5   4   3
  8   7   6
  2   1   0
  5   4   3
  8   7   6
  6   7   8
  3   4   5
  0   1   2
  ```

二、IDL语法基础：数组的操作

- 数组的反转：使用**REVERSE**函数。

  REVERSE(ARRAY[, SUBSCRIPT_INDEX][,/OVERWRITE])

  -SUBSCRIPT_INDEX，按指定的维数（≤原数组维数），逆序排列后生成新数组。

  默认按第一维数逆序排序。

  -/OVERWRITE，覆盖原数组。

  例如：

```
IDL> ARR0=INDGEN(3,3)
IDL> PRINT,ARR0,REVERSE(ARR0),$
REVERSE(ARR0,1),REVERSE(ARR0,2)
      0       1       2
      3       4       5
      6       7       8
      2       1       0
      5       4       3
      8       7       6
      2       1       0
      5       4       3
      8       7       6
      6       7       8
      3       4       5
      0       1       2
```

```
IDL> ARR0=INDGEN(3,3,3)
IDL> PRINT,ARR0,REVERSE(ARR0,3)
      0       1       2
      3       4       5
      6       7       8

      9      10      11
     12      13      14
     15      16      17

     18      19      20
     21      22      23
     24      25      26


     18      19      20
     21      22      23
     24      25      26

      9      10      11
     12      13      14
     15      16      17

      0       1       2
      3       4       5
      6       7       8
```

二、IDL语法基础：数组的操作

- 数组的转置：使用转置函数TRANSPOSE将数组的列和行对调。例如：

```
IDL> A=[[1,0,-1,2],[-1,1,3,0],[0,5,7,-6]]
IDL> HELP,A
     A               INT       = ARRAY[4, 3]
IDL> PRINT,A
      1      0     -1      2
     -1      1      3      0
      0      5      7     -6
IDL> B=TRANSPOSE(A)
IDL> HELP,B
     B               INT       = ARRAY[3, 4]
IDL> PRINT,B
      1     -1      0
      0      1      5
     -1      3      7
      2      0     -6
```

用户也可以引入一个可选择的一维数组自变量来实现数组的转置操作。该一维数组的元素大小（整数）代表原来数组的维度顺序。

```
IDL> ARR=INDGEN(2,4,6)
IDL> HELP,TRANSPOSE(ARR)
     <EXPRESSION>    INT       = ARRAY[6, 4, 2]
IDL> HELP,TRANSPOSE(ARR,[1,2,0])
     <EXPRESSION>    INT       = ARRAY[4, 6, 2]
```

二、IDL语法基础：数组的操作

- 数组的旋转(或转置)：使用ROTATE（ARRAY, 旋转参数N）函数将一维或两维数组以90度的增量按顺时针方向旋转。函数的第一个自变量是需要旋转的数组，第二个自变量是旋转的方向标志.

-N=0,1,2,…,7。其中N=0-3分别表示旋转0,90,180,270度;N=4-7分别表示转置的同时旋转。

```
IDL> ARR=INDGEN(4,3)
IDL> PRINT,ARR
       0       1       2       3
       4       5       6       7
       8       9      10      11
IDL> PRINT,ROTATE(ARR,1)        ；1是按顺时针方向旋转90度
       8       4       0
       9       5       1
      10       6       2
      11       7       3
IDL> PRINT,ROTATE(ARR,2)        ；2是按顺时针方向旋转180度
      11      10       9       8
       7       6       5       4
       3       2       1       0
IDL> PRINT,ROTATE(ARR,3)        ；3是按顺时针方向旋转270度
       3       7      11
       2       6      10
       1       5       9
       0       4       8
```

二、IDL语法基础：数组的操作

- 数组的旋转(或转置)：使用ROTATE（ARRAY, 旋转参数N）函数将一维或两维数组以90度的增量按顺时针方向旋转。函数的第一个自变量是需要旋转的数组，第二个自变量是旋转的方向标志.

  -N=0,1,2,…,7。其中N=0-3分别表示旋转0,90,180,270度;N=4-7分别表示转置的同时旋转。

```
IDL> ARR=INDGEN(4,3)
IDL> PRINT,ARR
       0       1       2       3
       4       5       6       7
       8       9      10      11
IDL> PRINT,ROTATE(ARR,4)        ；4是转置后按顺时针方向旋转0度
       0       4       8
       1       5       9
       2       6      10
       3       7      11
IDL> PRINT,ROTATE(ARR,5)        ；5是转置后按顺时针方向旋转90度
       3       2       1       0
       7       6       5       4
      11      10       9       8
IDL> PRINT,ROTATE(ARR,6)        ；6是转置后按顺时针方向旋转180度
      11       7       3
      10       6       2
       9       5       1
       8       4       0
IDL> PRINT,ROTATE(ARR,7) )      ；7是转置后按顺时针方向旋转270度
       8       9      10      11
       4       5       6       7
       0       1       2       3
```

二、IDL语法基础：数组的操作

- 数组的平移：使用SHIFT（ARRAY, 平移维度参数Si）函数将数组元素沿着某个维度平移。-Si为正则向前平移，否则向后平移。

```
IDL> ARR=INDGEN(5)
IDL> PRINT,ARR
       0       1       2       3       4
IDL> PRINT,SHIFT(ARR,1)
       4       0       1       2       3
IDL> PRINT,SHIFT(ARR,-2)
       2       3       4       0       1
IDL> ARR=INDGEN(3,3)
IDL> PRINT,ARR
       0       1       2
       3       4       5
       6       7       8
IDL> PRINT,SHIFT(ARR,-1,1)  ；列左（负）移1列，行下（正）移1行
       7       8       6
       1       2       0
       4       5       3
IDL> ARR=INDGEN(3,3)
IDL> PRINT,SHIFT(ARR,1,-1)  ；列右（正）移1列，行上（负）移1行
       5       3       4
       8       6       7
       2       0       1
```

二、IDL语法基础：数组的操作

- 删除数组行或列：
  在某些情况下，用户根据需求删除数组中特定的列或行。

```
IDL> ARR0=FINDGEN(12,999)
IDL> DELROW=[13,45,67,456,789]   ；要删除的5行

IDL> DIMS=SIZE(ARR0,/DIMENSIONS)
IDL> PRINT,DIMS
        12         999

IDL> NROWS=DIMS[1]
IDL> INDEX=REPLICATE(1L,NROWS)
IDL> INDEX[DELROW]=0L

IDL> KEEPROWS=WHERE(INDEX EQ 1L)
IDL> ARR0=ARR0[*,KEEPROWS]
IDL> HELP,ARR0
        ARR0            FLOAT     = ARRAY[12, 994]
```

二、IDL语法基础：数组的操作

- 数组的排序：使用SORT(ARRAY[, /L64])函数将数组所有元素按值的大小升序排列，返回的是排列后的下标数组。

```
IDL> ARR=[5,3,2,-1]
IDL> PRINT,SORT(ARR)
       3       2       1       0
IDL> PRINT,ARR[SORT(ARR)]
      -1       2       3       5
```

如果输入数组中所有元素或者某些元素的值相同，那么SORT函数返回的索引数组将不按升序排列：

```
IDL> A=[1,4,1,2,2]
IDL> PRINT,SORT(A)
       2       0       3       4       1
IDL> PRINT,A[SORT(A)]
       1       1       2       2       4
```

二、IDL语法基础：数组的操作

- 抽取数组唯一值：使用UNIQ(ARRAY [, Index])函数返回已经排过序的数组中唯一的值的下标。因此，要使用UNIQ函数，必须先对数组排序。

```
IDL> ARR0=[1, 2, 1, 2, 3, 4, 5, 6, 6, 5]
IDL> PRINT,SORT(ARR0)
       2       0       3       1       4       5       9       6       8       7
IDL> ARR1=ARR0[SORT(ARR0)]
IDL> ARR2=ARR1[UNIQ(ARR1)]
IDL> PRINT,ARR2
       1       2       3       4       5       6
```

=

若数组非单调，则可使用可选项[,Index(=SORT(ARRAY))]

```
IDL> PRINT,UNIQ(ARR0,SORT(ARR0))
       0       1       4       5       6       7
IDL> PRINT,ARR0[UNIQ(ARR0,SORT(ARR0))]
       1       2       3       4       5       6
```

如果需要抽取数组唯一值，同时保持元素顺序不变，则需利用SORT和UNIQ函数获取索引数组：

```
IDL> ARR3=[4,5,3,5,6,2,12,4,11]
IDL> SORT_INDEX=SORT(ARR3)
IDL> PRINT,SORT_INDEX
       5       2       7       0       3       1       4       8       6
IDL> UNIQ_INDEX=UNIQ(ARR3[SORT_INDEX])
IDL> PRINT,UNIQ_INDEX
       0       1       3       5       6       7       8
IDL> INDEX0=SORT_INDEX[UNIQ_INDEX]
IDL> PRINT,INDEX0
       5       2       0       1       4       8       6
IDL> INDEX0=INDEX0[SORT(INDEX0)]
IDL> PRINT,ARR3[INDEX0]
       4       5       3       6       2      12      11
```

二、IDL语法基础：数组大小的调整

D1[, ..., D8]

- **REBIN（ARRAY,M,N,可选关键字）**函数将数组调整为大小为M列N行数组，一维数组N省略。
  - 可选关键字/SAMPLE或SAMPLE=1表示按最近邻居抽样法进行放大或缩小。

```
IDL> ARR=[20,40,60]
IDL> PRINT,REBIN(ARR,9,/SAMPLE)
      20      20      20      40      40      40      60      60      60

IDL> PRINT,REBIN(ARR,3,2,/SAMPLE)
      20      40      60
      20      40      60
```

- 原始维度的整数倍或因子
- 每个维度的放大或缩小独立于其他维度
- ?REBIN

  - 如果省略关键字/SAMPLE(或SAMPLE=0)，则采用线性内插法放大数组。由于REBIN函数遵循整数放大因子（原来数组下标“0, 1, 2”放大3倍为“0，3，6”），末尾差数直接复制最后的那个元素代大小作为新数组最后几位元素。

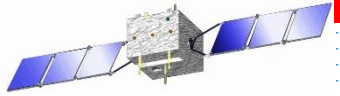
```
IDL> PRINT,REBIN(ARR,9)
      20      26      33      40      46      53      60      60      60
```

一维数组线性内插法插值：

```
SRX = FLOAT(DIMS[0]-M1)/(X-M1)*FINDGEN(X)
ARR_R = INTERPOLATE(ARR, SRX)
```

  - 如果M或M×N小于原来数组的元素大小，则REBIN函数采用临近平均法缩小数组：

```
IDL> ARR=FINDGEN(6)
IDL> PRINT,ARR
      0.000000      1.00000      2.00000      3.00000      4.00000      5.00000
IDL> PRINT,REBIN(ARR,2)
      1.00000      4.00000
IDL> PRINT,REBIN(ARR,3)
      0.500000      2.50000      4.50000
```



二、IDL语法基础：数组大小的调整

- **REBIN和REFORM函数联合创建多维数组**

```
IDL> ARR0=[3,4,5]
IDL> PRINT,REBIN(ARR0,3,3,/SAMPLE)
       3       4       5
       3       4       5
       3       4       5
IDL> PRINT,REBIN(REFORM(ARR0,1,3),3,3,/SAMPLE)
       3       3       3
       4       4       4
       5       5       5
```

二、IDL语法基础：数组大小的调整

- **REBIN和REFORM联合创建多维数组**

```
IDL> ARR0=[3,4,5]
IDL> PRINT,REBIN(ARR0,3,3,/SAMPLE)
       3       4       5
       3       4       5
       3       4       5
IDL> PRINT,REBIN(REFORM(ARR0,1,3),3,3,/SAMPLE)
       3       3       3
       4       4       4
       5       5       5
```

```
IDL> PRINT,REBIN(REFORM(ARR0,1,1,3),3,3,3,/SAMPLE)
       3       3       3
       3       3       3
       3       3       3

       4       4       4
       4       4       4
       4       4       4

       5       5       5
       5       5       5
       5       5       5
```

二、IDL语法基础：数组大小的调整

- CONGRID（ARRAY,X,Y,Z,[,/MINUS_ONE][,/INTERP][,/CUBIC]）函数可以放大或缩小一维、两维和三维数组至同维任意大小。
  - 默认的方法是使用临近抽样法，且右边元素优先，不够的依次从右到左抽样。

```
IDL> X=[1.0,2.0,3.0]
IDL> PRINT,CONGRID(X,4)
   1.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,5)
   1.00000   2.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,6)
   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,7)
   1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,8)
   1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,9)
   1.00000   1.00000   2.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
```

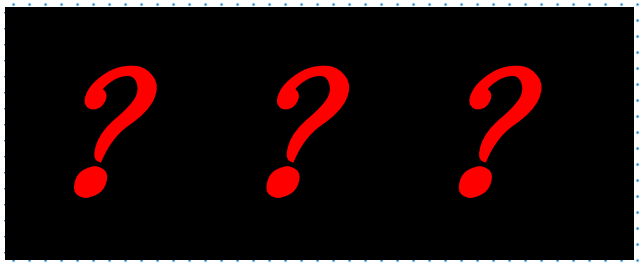
- REBIN在收缩数组时平均多个点
- CONGRID只是对数组进行重采样
- ?CONGRID

二、IDL语法基础：数组大小的调整

- CONGRID（ARRAY,X,Y,Z,[,/MINUS_ONE][,/INTERP][,/CUBIC]）函数可以放大或缩小一维、两维和三维数组至同维任意大小。
  - 默认的方法是使用临近抽样法，且右边元素优先，不够的依次从右到左抽样。

```
IDL> X=[1.0,2.0,3.0]
IDL> PRINT,CONGRID(X,4)
1.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,5)
1.00000   2.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,6)
1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,7)
1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,8)
1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,9)
1.00000   1.00000   2.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
```

- REBIN在收缩数组时平均多个点
- CONGRID只是对数组进行重采样
- ?CONGRID




```
IDL> I=3
IDL> N=9
IDL> INDEX=(FLOAT(I)/FLOAT(N))*FINDGEN(N)
IDL> PRINT,INDEX
      0.000000   0.333333   0.666667   1.00000   1.33333   1.66667   2.00000   2.33333   2.66667
IDL> INDEX=ROUND(INDEX)
IDL> PRINT,INDEX
       0       0       1       1       1       2       2       2       3
IDL> PRINT,X[INDEX]
    1.00000   1.00000   2.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
```

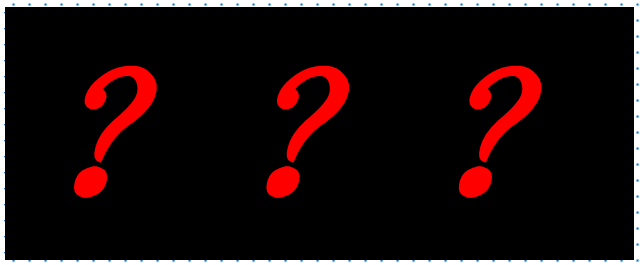
超过最大下标值，自动转换为最大下标，即采用临近抽样法时总是复制最后的值。

二、IDL语法基础：数组大小的调整

- CONGRID（ARRAY,X,Y,Z,[,/MINUS_ONE][,/INTERP][,/CUBIC]）函数可以放大或缩小一维、两维和三维数组至同维任意大小。
  - 默认的方法是使用临近抽样法，且右边元素优先，不够的依次从右到左抽样。

```
IDL> X=[1.0,2.0,3.0]
IDL> PRINT,CONGRID(X,4)
1.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,5)
1.00000   2.00000   2.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,6)
1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,7)
1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,8)
1.00000   1.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
IDL> PRINT,CONGRID(X,9)
1.00000   1.00000   2.00000   2.00000   2.00000   3.00000   3.00000   3.00000   3.00000
```

- REBIN在收缩数组时平均多个点
- CONGRID只是对数组进行重采样
- ?CONGRID




```
IDL> I=3
IDL> N=9
IDL> INDEX=(FLOAT(I-1)/FLOAT(N-1))*FINDGEN(N)
IDL> PRINT,INDEX
      0.000000    0.250000    0.500000    0.750000     1.00000     1.25000     1.50000     1.75000     2.00000
IDL> INDEX=ROUND(INDEX)
IDL> PRINT,INDEX
       0       0       1       1       1       1       2       2       2
IDL> PRINT,X[INDEX]
    1.00000     1.00000     2.00000     2.00000     2.00000     2.00000     3.00000     3.00000     3.00000
```

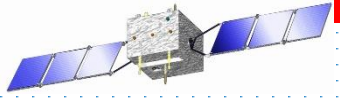
= CONGRID(X,9,/MINUS_ONE)

二、IDL语法基础：数组大小的调整

- **CONGRID（ARRAY,X,Y,Z,[,/INTERP][,/CUBIC]）函数可以放大或缩小一维、两维和三维数组至同维任意大小。**
  - 在放大或缩小一维或两维数组时，采用关键字/INTERP可使用线性内插法（此时右边优先不再有效）。对于三维数组，CONGRID函数总是使用线性内插法调整数组。

```
IDL> X=[1.0,2.0,3.0]
IDL> PRINT,CONGRID(X,5,/INTERP)
      1.00000      1.60000      2.20000      2.80000      3.00000
IDL> PRINT,CONGRID(X,7,/INTERP)
      1.00000      1.42857      1.85714      2.28571      2.71429      3.00000      3.00000
IDL> PRINT,CONGRID(X,9,/INTERP)
      1.00000      1.33333      1.66667      2.00000      2.33333      2.66667      3.00000      3.00000      3.00000
```

- ?CONGRID
- CONGRID.PRO



二、IDL语法基础：数组大小的调整


```
Function CONGRID, arr, x, y, z, $ CENTER=center, $ CUBIC
= cubicIn, $ INTERP=interp, $ MINUS_ONE=minus_one

  COMPILE_OPT idl2, hidden
  ON_ERROR, 2      ;Return to caller if error

  ndim = SIZE(arr, /N_DIMENSIONS)
  dims = SIZE(arr, /DIMENSIONS)

  if ((ndim lt 1) or (ndim gt 3)) then $
      Message, 'Array must have 1, 2, or 3 dimensions.'

  ;;Supply defaults = no interpolate, and no minus_one.
  int = KEYWORD_SET(interp)
  m1 = KEYWORD_SET(minus_one)
  cubic = (N_ELEMENTS(cubicIn) gt 0) ? cubicIn : 0
  if (cubic ne 0) then int = 1;Cubic implies interpolate
     offset = KEYWORD_SET(center) ? 0.5 : 0.0
  ; Construct new interpolate coordinates.
  ; Skip this for 2D nearest-neighbor since we use
POLY_2D instead.
  if ((ndim ne 2) or ((ndim eq 2) and int)) then begin
  ; Note that we need to use "offset" twice: Once to
shift the new
  ; coordinates to the midpoint, and again to shift the
location of
  ; the original coordinates to their midpoint.

  switch ndim of  ; Fall through for ndim>1.
         3: srz = float(dims[2]-m1)/(z-m1)*(findgen(z) +
offset) - offset
         2: sry = float(dims[1]-m1)/(y-m1)*(findgen(y) +
offset) - offset
         1: srx = float(dims[0]-m1)/(x-m1)*(findgen(x) +
offset) - offset
  endswitch
  endif
```

C:\Program files\exelis\idl83\lib\congrid.pro

```
case ndim of
     1: begin      ; *** ONE DIMENSIONAL ARRAY
        arr_r = (int) ? INTERPOLATE(arr, srx,
CUBIC = cubic) : $  arr[ROUND(srx)]
        end
     2: begin      ; *** TWO DIMENSIONAL ARRAY
        if (int) then begin ;bilinear or cubic
        arr_r = INTERPOLATE(arr, srx, sry,
/GRID, CUBIC=cubic)
        endif else begin  ; nearest neighbor
        ; Note: For expansion, divide by (x-1)
so that CONGRID
        ; will agree with REBIN.
        expand = (x gt dims[0])
        xm1 = (m1 or expand) ? x-1 : x
        arr_r = POLY_2D(arr,[[0,0],[(dims[0]-
m1)/float(xm1),0]], $ ;Use poly_2d
[[0,(dims[1]-m1)/float(y-m1)],[0,0]],int,x,y)
        endelse
        end
     3: begin     ; *** THREE DIMENSIONAL ARRAY
        ; Only supports linear interpolation.
        arr_r = INTERPOLATE(arr, srx, sry,
srz, /GRID)
        end
endcase

return, arr_r
end
```

二、IDL语法基础：数组大小的调整

- **CONGRID（ARRAY,X,Y,Z,[,/INTERP][,/CUBIC]）函数可以放大或缩小一维、两维和三维数组至同维任意大小。**
  - 在放大或缩小一维或两维数组时，采用关键字/INTERP可使用线性内插法（此时右边优先不再有效）。对于三维数组，CONGRID函数总是使用线性内插法调整数组。

```
IDL> X=[1.0,2.0,3.0]
IDL> PRINT,CONGRID(X,5,/INTERP)
      1.00000      1.60000      2.20000      2.80000      3.00000
IDL> PRINT,CONGRID(X,7,/INTERP)
      1.00000      1.42857      1.85714      2.28571      2.71429      3.00000      3.00000
IDL> PRINT,CONGRID(X,9,/INTERP)
      1.00000      1.33333      1.66667      2.00000      2.33333      2.66667      3.00000      3.00000      3.00000
```

- ?CONGRID
- CONGRID.PRO

以一维数组为例，按线性内插法插值：

```
SRX = FLOAT(DIMS[0]-M1)/(X-M1)*FINDGEN(X)
ARR_R = (INT) ? INTERPOLATE(ARR, SRX) : ARR[ROUND(SRX)]
```

```
IDL> CONGRID(X,7,/INTERP)
```

=

```
IDL> SRX= FLOAT(3)/(7)*FINDGEN(7)
IDL> PRINT,SRX
      0.000000     0.428571     0.857143      1.28571      1.71429      2.14286      2.57143
IDL> INTERPOLATE(X, SRX)
```

- 在采用关键字/CUBIC时，使用卷积内插法，且变量的取值范围在-1到0之间。

二、IDL语法基础：数组大小的调整

- INTERPOLATE(P, X [, Y [, Z]][,/GRID][,CUBIC=VAR）函数可以对已经存在的数组元素进行线性、双线性或三线性内插操作，以缩放一维、两维或三维数组。将数组调整为同维任意大小的数组。
  与CONGRID函数的区别：INTERPOLATE函数可以帮助用户在特定的位置插入值。
  -X,Y,Z表示生成数组维数的下标组成的一维数组，用于定位插值元素的位置。
  -默认线性内插；CUBIC时卷积内插，变量取值范围-1~0之间。
  -/GRID，数组的插值点按网格生成，否则按线性生成。
  -生成新数组的维数不能改变，但大小可任意改变，且能可以任意定位插值点。

• 线性插值

```
IDL> X=[1.0,2.0,3.0]
IDL> LOCATE=[0.0,0.5,1.0,1.5,2.0]
IDL> PRINT,INTERPOLATE(X,LOCATE)
      1.00000      1.50000      2.00000      2.50000      3.00000

IDL> LOCATE=[0.0,0.2,1.0,1.8,2.0]   ；位置参数的大小决定插入值的大小
IDL> PRINT,INTERPOLATE(X,LOCATE)
      1.00000      1.20000      2.00000      2.80000      3.00000
```

• 双线性插值

```
IDL>P = FINDGEN(4,4)
IDL>PRINT, INTERPOLATE(P, [.5, 1.5, 2.5], [.5, 1.5, 2.5], /GRID)

      2.50000   3.50000   4.50000
      6.50000   7.50000   8.50000
      10.5000   11.5000   12.5000
```

■ /GRID关键字控制位置数组如何指定需要插入的位置，在线性插值的情况下没有效果。

CORRESPONDING TO THE LOCATIONS:
(.5,.5), (1.5, .5), (2.5, .5),
(.5,1.5), (1.5, 1.5), (2.5, 1.5),
(.5,2.5), (1.5, 2.5), (2.5, 2.5)

二、IDL语法基础：数组大小的调整

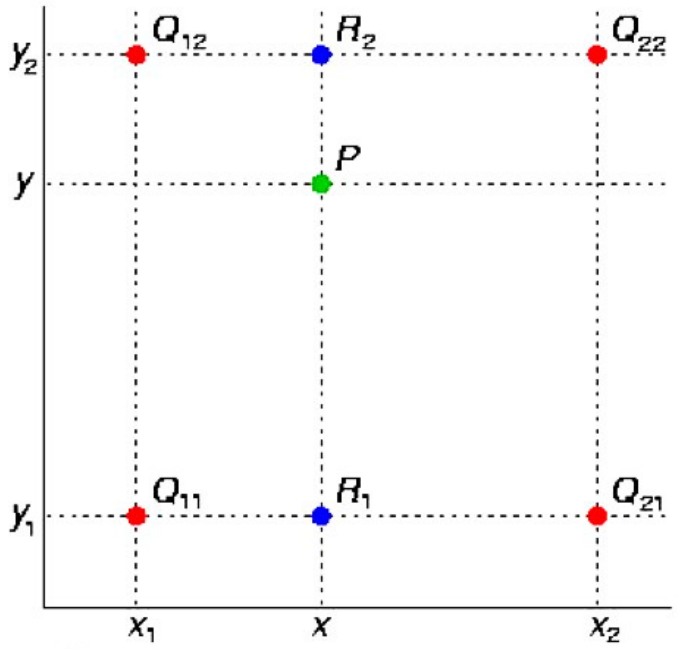
- **INTERPOLATE(P, X [, Y [, Z]][,/GRID][,CUBIC=VAR）函数可以对已经存在的数组元素进行线性、双线性或三线性内插操作，以缩放一维、两维或三维数组。将数组调整为同维任意大小的数组。-/GRID，数组的插值点按网格生成，否则按线性生成。**

0.00	1.0	2.0	3.0
	(.5,.5)	(1.5,.5)	(2.5,.5)
4.0	5.0	6.0	7.0
	(.5,1.5)	(1.5,1.5)	(2.5,1.5)
8.0	9.0	10.0	11.0
	(.5,2.5)	(1.5,2.5)	(2.5,2.5)
12.0	13.0	14.0	15.0

- 双线性插值

```
IDL>P = FINDGEN(4,4)
IDL>PRINT, INTERPOLATE(P, [.5, 1.5, 2.5], [.5, 1.5, 2.5], /GRID)
```

```
   2.50000   3.50000   4.50000
   6.50000   7.50000   8.50000
   10.5000   11.5000   12.5000
```




```
IDL> PRINT, INTERPOLATE(P, [.3], [.6], /GRID)
      2.70000
```

距离靠近，权重越高

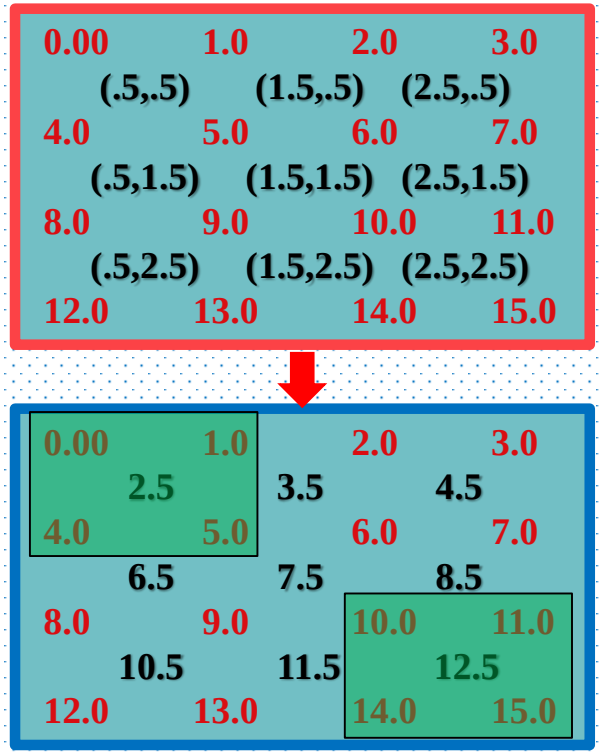
$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{where} \quad R_1 = (x, y_1)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{where} \quad R_2 = (x, y_2)$$

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2).$$

二、IDL语法基础：数组大小的调整

- **INTERPOLATE(P, X [, Y [, Z]][,/GRID][,CUBIC=VAR）**函数可以对已经存在的数组元素进行线性、双线性或三线性内插操作，以缩放一维、两维或三维数组。将数组调整为同维任意大小的数组。
  -/**GRID**，数组的插值点按网格生成，否则按线性生成。



- 双线性插值

IDL>P = FINDGEN(4,4)

IDL>PRINT, INTERPOLATE(P, [.5, 1.5, 2.5], [.5, 1.5, 2.5], /GRID)

2.50000 3.50000 4.50000
6.50000 7.50000 8.50000
10.5000 11.5000 12.5000

- /GRID关键字控制位置数组如何指定需要插入的位置，在线性插值的情况下没有效果。

IDL>PRINT, INTERPOLATE(P, [.5, 1.5, 2.5], [.5, 1.5, 2.5])

CORRESPONDING TO THE LOCATIONS:
(.5,.5), (1.5, 1.5), (2.5, 2.5),

2.50000 7.50000 12.5000

二、IDL语法基础：数组大小的调整

在放大或缩小两维数组时（如IMAGE），需要选择一种计算法则来计算内插值在数组中的位置。例如要将一个32×32的数组扩展为500×300的数组。

- DIST函数创建一个数组，其中每个数组元素的值与其频率成正比。
- TVSCL图像显示程序将图像的强度值缩放到直接图形图像显示设备的范围内，并将数据输出到指定位置的图像显示。

```
IDL> A=DIST(32,32)
IDL> TVSCL,A
IDL> NX=500
IDL> NY=300
IDL> DIMS=SIZE(A,/DIMENSIONS) ； 关键字DIMENSIONS是返回每维大小
IDL> PRINT,DIMS
      32       32
IDL> XLOC=(FINDGEN(NX)+0.5)*(DIMS[0]/FLOAT(NX))-0.5    ;计算列的大小
IDL> YLOC=(FINDGEN(NY)+0.5)*(DIMS[1]/FLOAT(NY))-0.5    ;计算行的大小
;0.5表示图片长或宽占输出窗口大小的实际比率（长和宽范围0-1）。

IDL> B=INTERPOLATE(A,XLOC,YLOC,/GRID)
IDL> HELP,B
      B               FLOAT     = ARRAY[500, 300]
IDL> TVSCL,B
```

关键字/GRID允许位置数组在输出数组的每个维度上可以大小不同，否则必须相同。

2.10 字符串的操作

- 系统日期：SYSTIME函数
  以24个字符的串形式：WW MM DD HH:MM:SS YEAR返回当前系统日期。
  ```
  IDL> PRINT,SYSTIME()
       Thu Sep 19 14:55:13 2019
  ```

- 求字符串长度：STRLEN(expC)函数
  计算字符串长度值。若expC为空字符串，则返回0值。
  ```
  IDL> PRINT,STRLEN('WELCOME TO CDUT!')
       16
  ```

- 比较字符串：STRCMP函数
  STRCMP(expC1, expC2 [, N][, /FOLD_CASE])
  比较两个字符串是否相同，若相同则返回1，否则返回0。
  -N，只比较前N个字符。
  -/FOLD_CASE，不区分大小写。
  ```
  IDL> PRINT,STRCMP('Welcome to CDUT!','WELCOME TO CHENGDU!',10)
       0
  IDL> PRINT,STRCMP('Welcome to CDUT!','WELCOME TO CHENGDU!',10,/FOLD_CASE)
       1
  ```

二、IDL语法基础：字符串的操作

- 压缩字符串：**STRCOMPRESS函数**

  STRCOMPRESS(expC[, /REMOVE_ALL])

  去除字符串中单词间多于一个的空格（空格和制表符TAB）。

  -/REMOVE_ALL，去除所有空格（空格和制表符TAB）。

  IDL> PRINT,STRCOMPRESS('Welcome to CDUT!')

  Welcome to CDUT!

  IDL> PRINT,STRCOMPRESS('Welcome to CDUT! ',/REMOVE_ALL)

  WelcometoCDUT!

- 字符串大小写转换：**STRUPCASE和STRLOWCASE函数**

  - **STRUPCASE(expC)**

  将字符串中所有小写字母转换成大写字母，其它非字母字符不变。

  - **STRLOWCASE(expC)**

  将字符串中所有大写字母转换成小写字母，其它非字母字符不变。

  IDL> PRINT,STRUPCASE('Welcome to CDUT!')

  WELCOME TO CDUT!

  IDL> PRINT,STRLOWCASE('Welcome to CDUT!')

  welcome to cdut!

二、IDL语法基础：字符串的操作

- 字符串匹配：STRMATCH函数

  STRMATCH(expC1, expC2 [, /FOLD_CASE])

  检查expC1和expC2是否匹配。若匹配则返回1，否则返回0。

  -/FOLD_CASE，不区分大小写 。

```
IDL> PRINT,STRMATCH(['Welcome', 'to', 'CDUT!'],'cdut!')
       0   0   0
IDL> PRINT,STRMATCH(['Welcome', 'to', 'CDUT!'],'cdut!',/FOLD_CASE)
       0   0   1
```

- 字符串截取：STRMID函数

  STRMID(expC, N [, expN][, /REVERSE_OFFSET)

  从字符串的第N个（从0开始）字符开始截取字符，默认取到字符串末尾。

  -expN，从第N个字符开始，取expN个字符。

  -/REVERSE_OFFSET，N的位置从字符串末尾开始计数。

```
IDL> PRINT,STRMID('Welcome to CDUT!',3)
       come to CDUT!
IDL> PRINT,STRMID('Welcome to CDUT!',3,/REVERSE_OFFSET)
       DUT!
IDL> PRINT,STRMID('Welcome to CDUT!',3,2,/REVERSE_OFFSET)
       DU
```

二、IDL语法基础：字符串的操作

- 字符串连接：STRJOIN函数

  STRJOIN(expC1 [, ‘expC2’])

  使用定界符expC2将expC1中的字符串连接起来。

  -expC2])，若省略则直接连接 。

  IDL> PRINT,STRJOIN(['Welcome', 'to', 'CDUT!'])

  WelcometoCDUT!

  IDL> PRINT,STRJOIN(['Welcome', 'to', 'CDUT!'],'+')

  Welcome+to+CDUT!

- 字符串替换：STRPUT函数

  STRPUT(expC1, expC2 [, N])

  从第N个（从0开始）字符开始，用expC2代替expC1中的字符，且保持expC1的长度不变，多出部分裁掉。

  IDL> STR0='Welcome to CDUT, CHENGDU 2019!'

  IDL> STRPUT,STR0,'National Day!',11

  IDL> PRINT,STR0

  Welcome to National Day! 2019!

- 字符串去空格：STRTRIM(expC [, N])函数

  去除expC头部或尾部的空格，并返回处理后的字符串。

  -N=0或缺省，去掉尾部空格；N=1，去掉头部空格；N=2，去掉头尾部空格。

  IDL> PRINT,STRTRIM(' Welcome to CDUT! ',2)

  Welcome to CDUT!

二、IDL语法基础：字符串的操作

- 字符串检测：STRPOS函数

  STRPOS(expC1, expC2 [, POS] [, /REVERSE_OFFSET][, /REVERSE-SEARCH])

  检测expC2是否在expC1中首次出现位置。若出现则返回位置，否则返回值-1（注意大小写）。

  -POS，开始搜索的字符位置。如果POS小于零，则起始位置为零。

  -/REVERSE_OFFSET，POS从字符串末尾开始计数。若使用关键字而未指定POS, 则返回-1。

  -/REVERSE_SEARCH，反向检测expC2首次出现位置。STRPOS通常从POS开始，移动到字符串的末尾寻找匹配项。如果设置了/REVERSE_SEARCH，搜索将移动到字符串的开头。

```
IDL> PRINT,STRPOS('Welcome to CDUT!','CDUT')
        11
IDL> PRINT,STRPOS('Welcome to CDUT!','CDUT',/REVERSE_OFFSET)
        -1
IDL> PRINT,STRPOS('Welcome to CDUT!','to',7,/REVERSE_OFFSET)
        8
IDL> PRINT,STRPOS('Welcome to CDUT!','to',6,/REVERSE_OFFSET)
        -1
IDL> PRINT,STRPOS('Welcome to cdut!','c',/REVERSE_SEARCH)
        11
IDL> PRINT,STRPOS('Welcome to CDUT!','CDUT',4,/REVERSE_SEARCH)
        -1
IDL> PRINT,STRPOS('Welcome to CDUT!','CDUT',15,/REVERSE_SEARCH)
        11
```

和POS配合使用！！！

从POS位置开始反向。省略POS，则从末尾开始。

二、IDL语法基础：字符串的操作

- 字符串分割：STRSPLIT函数

  STRSPLIT(expC1 [, 'expC2'] [, /EXTRACT | , LENGTH=ARRAY] [, ESCAPE=expC3]
  [, COUNT=VAR] [, /PRESERVE_NULL])

  使用定界符expC2将字符串expC1分割子串。默认定界符为空格或制表符TAB。

  -/EXTRACT，返回子串数组。默认返回包含子串首位置的值的数组。

  -LENGTH=ARRAY，将每个子串的长度值存储到数组中，与/EXTRACT不可同用。

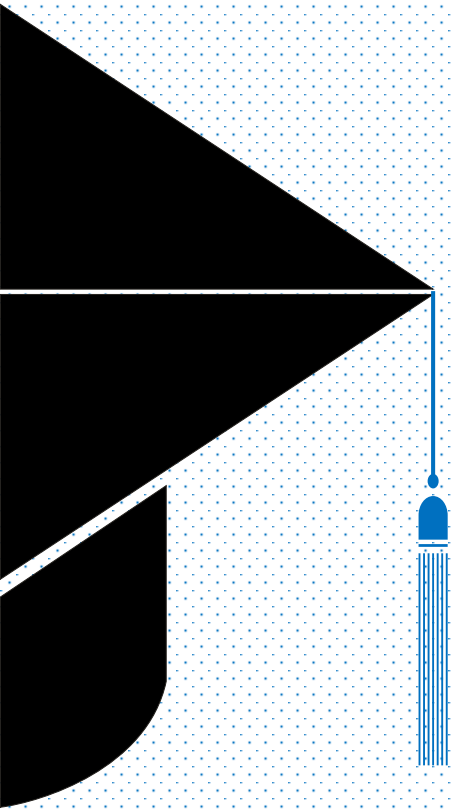
  -ESCAPE=expC3，expC1中包含expC3的字符串被忽略。

  -COUNT=VAR，将提取的子串的个数存储到变量中。

  -/PRESERVE_NULL，提取所有子串。若省略则忽略空子串。

```
IDL> PRINT,STRSPLIT('HTTP:\\WWW.C DU T.ED U.CN')
       0       13       16       21
IDL> PRINT,STRSPLIT('HTTP:\\WWW.CDUT.EDU.CN','.',/EXTRACT)
       HTTP:\\WWW CDUT EDU CN
IDL> PRINT,STRSPLIT('HTTP:\\WWW.CDUT.EDU.CN','.')
       0       11       16       20
IDL> PRINT,STRSPLIT('HTTP:\\WWW.CDUT.EDU.CN','.',LENGTH=ARR0),ARR0
       0       11       16       20
      10        4        3        2
IDL> PRINT,STRSPLIT('HTTP:\\WWW.CDUT.EDU.CN','.',/EXTRACT,ESCAPE=':',COUNT=VAR0),VAR0
       HTTP\\WWW CDUT EDU CN
       4
```

成都理工大學
CHENGDU UNIVERSITY OF TECHNOLOGY



演示完毕

Thank You