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：

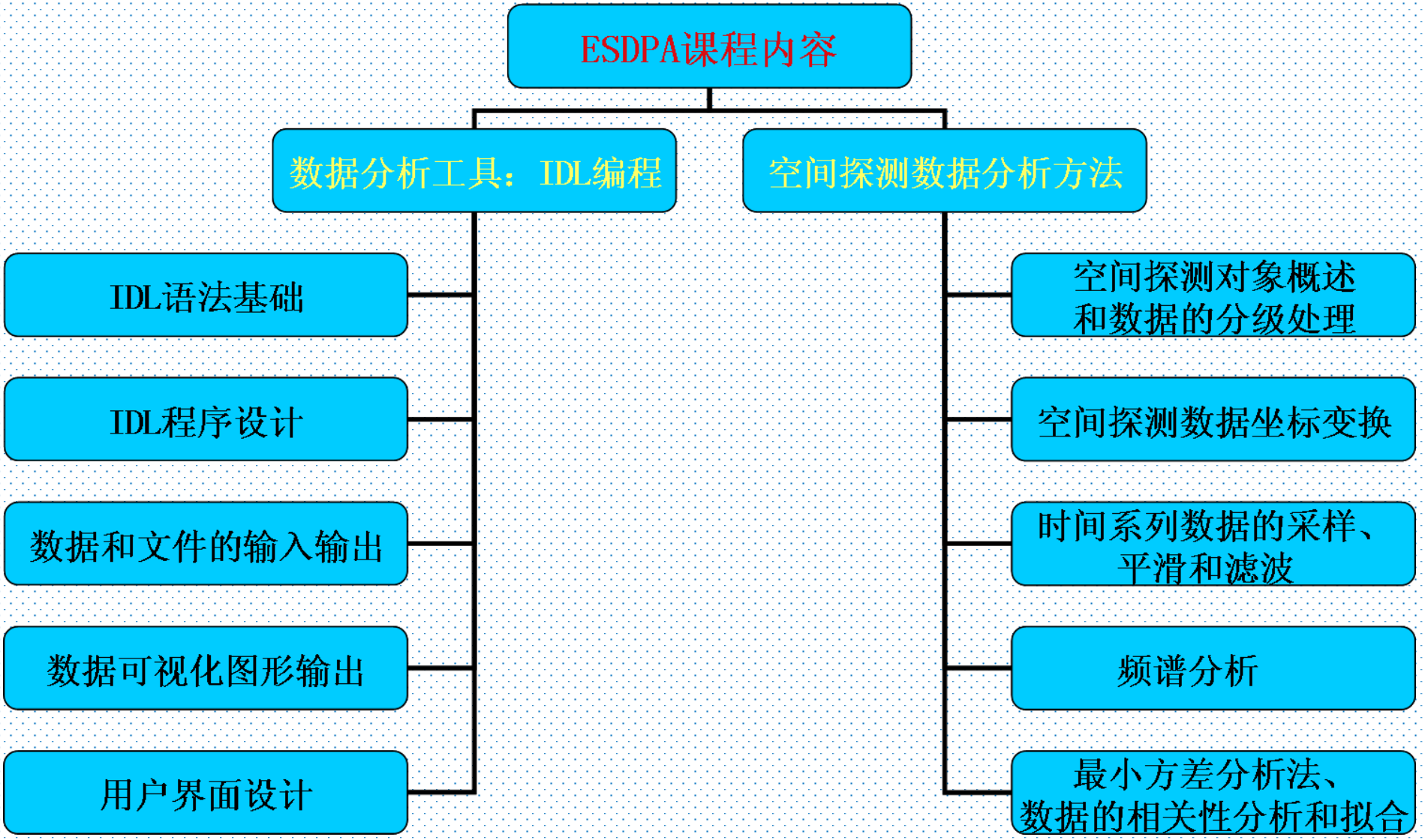
平时成绩（40%）：考勤（10%）+ 大作业/实验报告（30%）

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



一、IDL简介

ENVI *VS* IDL

- **IDL**
- 全称交互式数据语言（**Interactive Data Language**）
- 语法简单、功能强大的可视化语言
- **NASA**

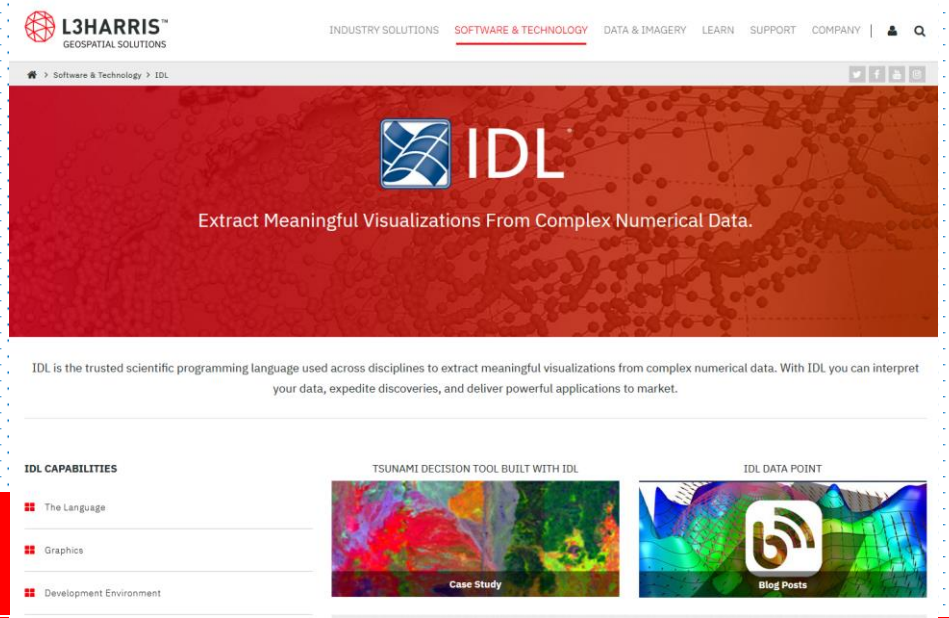
  ESA、NOAA、LOS ALAMOS国家实验室、LAWRENCE LIVERMORE国家实验室、IBM、SIEMENS、GE Medical、Army Corps of Engineers、……

  清华、中科院、浙大等等
- 海量数据处理
- 可视化
  - 计算可视化(计算机图形学)

    三维数据转换为图像（矢量、张量、流场可视化，数值模拟，并行计算等等）

    -地震勘探
    -气象预报
    -流体力学
    -天体物理
    -海洋观察
    -地理信息
    -环境保护等等
  - 信息可视化

    -计算机网络应用



https://www.harrisgeospatial.com/Software-Technology/IDL

一、IDL简介

IDL(交互式数据语言）是由美国ITT VIS（ITT Visual Information Solutions）公司推出的面向矩阵的且用于数据可视化研究与开发的第四代可视化交互数据语言。广泛应用于航天、军事、医学、地球科学（地震勘察、气象预报海洋观察、地理信息、大气物理和环境保护等）和天文学等各个领域。

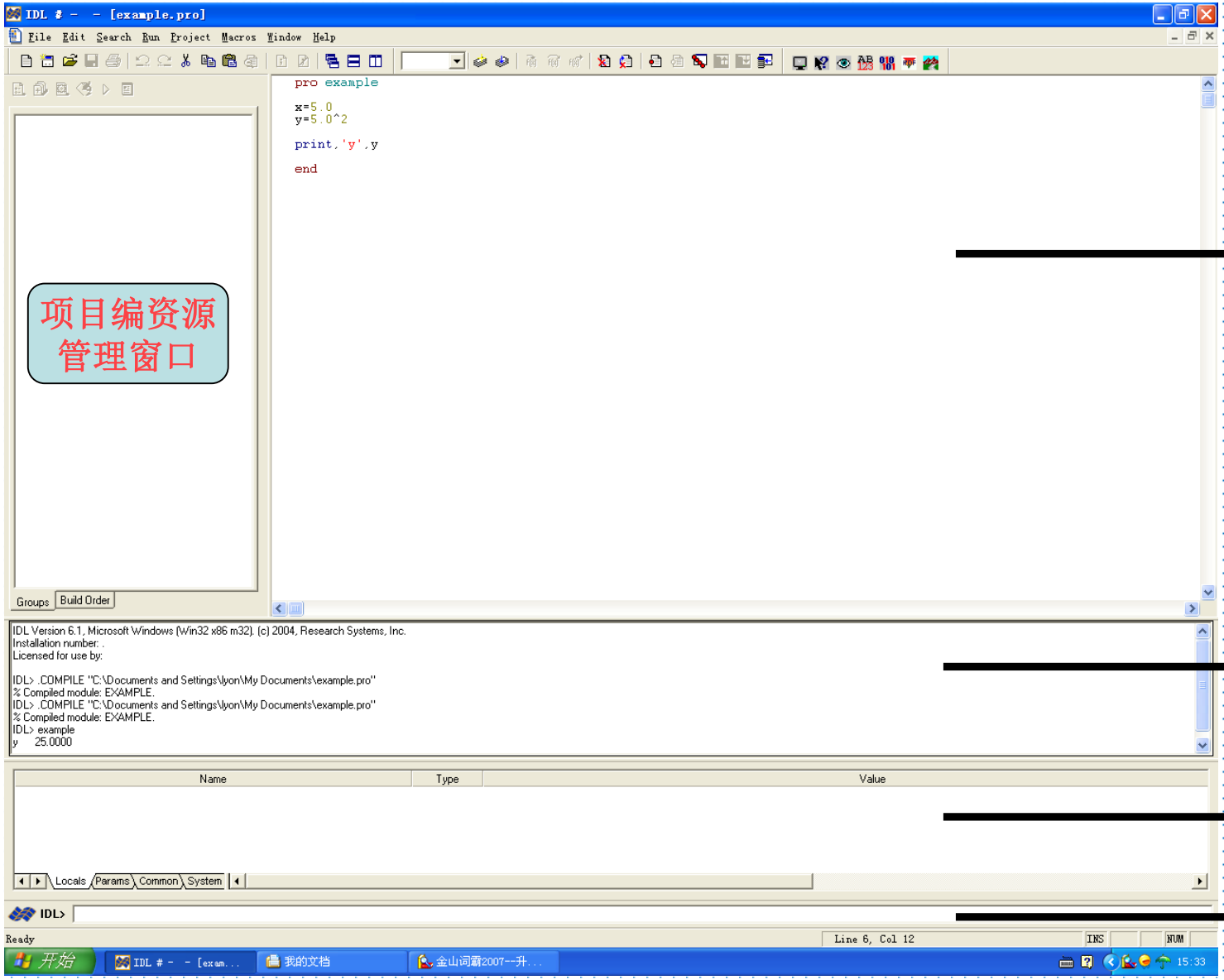
运行平台：Windows、MacOS和UNIX（包括Linux）等。

主要功能：（1）能够读入几乎所有数据格式，如二进制、十进制数等；
（2）数据快速可视化；
（3）图像处理。

特 点：（1）“交互式”和“编译”模式 （C和Fortran只提供编译模式）；
（2）强大的数组和矩阵运算功能 （C和Fortran最适合标量操作）；
（3）IDL可以在任何时候创建和重新定义变量的类型、大小和值；
（C和Fortran只能在执行时重定义变量的大小和值）
（4）IDL包括可视化、数据分析及图形用户界面等内建的程序包；
（C和Fortran要求外部的库）
（5）强大的外部语言接口。支持COM/ActiveX组件，可以将IDL应用开发集成到COM兼容的环境中，从Visual Basic、Visual C++等访问IDL，还可以通过动态链接库和COM组件方式在IDL程序中调用C或Fortran等；
（6）IDL变量名不区分大小写；
（7）能够像MATLAB进行快速的矩阵运算，又比MATLAB具有较强的图像可视化功能。

交互式或编译模式均可
Help命令：跟踪

一、IDL简介



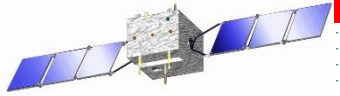
项目编资源管理窗口

程序文本编辑窗口

编译和运行结果输出窗口

变量检测窗口

命令输入窗口



一、IDL简介

- 界面编辑和命令操作

1、点击“FILE”→“PREFERENCES”对话框来改变显示的默认界面；

2、*IDL>DEMO* 可以查看IDL功能范例；

3、*IDL>?* 或者直接点击“HELP”→”CONTENTS…” 可以查看在线帮助；

4、*IDL> ?PLOT*（任意过程/命令特征字符） 可以查看该过程/命令的使用方法；

5、创建和保存一个IDL程序 (IDL程序设计部分重点讲）；

6、创建和保存一个IDL项目 (IDL程序设计部分重点讲）；

7、其他常用菜单功能介绍 （自己试用）。

一、IDL简介

IDL命令格式

命令结构

命令结构：命令动词 + 命令选项

命令动词：表示该命令所要完成的操作

命令选项：表示对所执行命令进行具体限制

命令动词[, 参数1, …, 参数n][, 关键字1, …, 关键字n]

可选项，省略即默认

同类项允许重复

例如：*IDL>PRINT,2019,FORMAT='(F7.2)'*

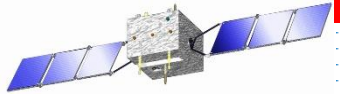
命令规则

- 参数，命令动词右边，一般有顺序和类型，且一般为必选项
- 关键字，命令动词右边，顺序任意，一般为可选项
- 命令太长，“$”换行（253个字符→任意长度）
- 不区分大小写（系统保留字和命令动词一般大写）
- 一行多命令，“&”续命令符
- 注释语句，“；”分号注释符，不编译不执行

执行模式

- 交互式
- 编译式
- 菜单操作式

```
PRO HELLOSPACE
PRINT,'HELLO SPACE 2019 !'
END
```



二、IDL语法基础：程序格式和特殊字符

2.1 程序格式和特殊字符

- IDL程序扩展名.PRO。以PRO开始和END结束程序。

  例如：*PRO EXAMPLE1*

  A=10.0

  PRINT, 'A=',A

  END

- 输出命令：*PRINT, 'HELLO' ;输出到编译和运行结果输出窗口。*

  PRINTF,LUN,X ;输出到OPENW指定的数据文件中，LUN是文件通道号。

  例如：

  OPENW,LUN,FILEPATH('FGM_C2_MINUTE.DAT',ROOT_DIR='D:\IDLBENCH\IDLLESSON\MYDEMO\'),/GET_LUN

  A=5.0

  PRINTF,LUN,A

  FREE_LUN,LUN

- 特殊字符：
  - 分号“；”表示注释内容开始；
  - 美元符号“$”表示续行标志。“$”前面的语句和下一行语句构成一个完整的命令语句。

    例如：*IDL> PRINT,'CD', $*

    IDL> 'UT'

    CDUT
  - “&”符号分开同一行中的多个语句。

    例如：

    X0=0.12 & X1=0.9 & Y1=0.96 & DY=0.18 & Y0=Y1-DY

2.2 十二种基本数据类型

一、数字类数据类型

数据类型	描述	组成	位/字节	范围
byte	字节型	数字	8/1	0~255
int	有符号整型	正负号和数字	16/2	-32768~+32767
uint	无符号整形	数字	16/2	0~65535
long	有符号长整型	正负号和数字	32/4	-2^{31}~$2^{31}-1$
ulong	无符号长整型	数字	32/4	0~$2^{32}-1$
long64	64位长整型	正负号和数字	64/8	-2^{63}~$2^{63}-1$
ulong64	无符号64位长整型	数字	64/8	0~$2^{64}-1$
float	浮点数	数字、小数点、正负号和浮点数标记E	32/4	-10^{38}~10^{38}
double	双精度浮点数	数字、小数点、正负号和浮点数标记E	64/8	-10^{308}~10^{308}
complex	复数	数字、小数点、正负号和浮点数标记E和复数标示i	64/8	-10^{38}~10^{38}
dcomplex	双精度复数	数字、小数点、正负号和浮点数标记E和复数标示i	128/16	-10^{308}~10^{308}
string	字符串	汉字、数字、字母及ASCII码中可打印的其他专用字符	每个字符占一个字节	0~32767个字符

二、IDL语法基础：组合数据类型

2.3 四种组合数据类型

（1）数组（ARRAY):由一个或多个相同类型的数据组成的组合数据类型，是一组有序的数据的集合。数组是由一组名字相同，而下标不同的多个元素组成。一般用于存储一组类型相同的数据。

（2）结构体（STRUCTURE):由一个或多个不同的数据类型组成的组合数据类型。结构体又可
分为匿名和署名结构体两种。一般用于存储一组类型不同的数据。

（3）对象（OBJECT)：对象类型与结构体十分相似，指一种具有特定类的组合数据类型，而类描述了一个或者多个不同的数据类型以及这些数据类型定义的一系列操作或函数。每一个对象都有自己的属性、事件和方法，占4个字节。

（4）指针（POINTER):用于表示或者存储各种类型数据的存储地址的数据类型。指针数据一般指数据在内存中的存储地址，即指针就是地址，占4个字节。

二、非数字类数据类型

数据类型	描 述	数据类型	描 述
string	字符串（0~32767个字符）	pointer	指针变量，指向地址变动的变量
struct	一个或多个变量的组合	objref	对象结构体的引用

二、IDL语法基础：数据类型操作注意事项

- 12种基本数据类型可以归纳为整形、浮点型和字符串型三类，使用应注意以下事项：
  - 检查整数溢出，如在执行250*300时，正确的答案是75000，但最大的无符号整形是65535，那么所得到的答案可能是75000-65535=9464。

    例如：
    ```
    IDL> Y=250*300
    IDL> PRINT,Y
         9464
    IDL> Y=250*300L
    IDL> PRINT,Y
         75000
    ```

  - 避免数量级相差巨大的浮点数之间的加减运算。（排序后从小到大相加）

    例如：*1000000.00+0.01有可能等于1000000.00。*
  - 避免浮点数的等量运算。（计算机存储的浮点数并不精确）

    例如：*100个0.05加在一起有可能不等于5.0。*
  - 避免不同类型的数据进行比较。

    例如：*不要将整形数据与浮点型数据进行比较。*

2.4 常量

在IDL程序运行过程中，其数值始终不变的量为常量。

- 字符型常量：指两个界定符（单引号或双引号）里边的字符或数字，又称字符串。

  例如：*“IDL6.1”*、*‘IDL6.1’*、*‘ “ ’*、*“1’m a boy”*、*“ ”（空间符）*、*“ ”（空格串）*，*“2019.00”*等

  - 配对使用　　*‘Hi there’*（√）　*‘Hi there”*（×）
  - 定界符号是字符串一部分，用另一个　　*“I’m happy”*
  - 中英文字母区分大小写　　*“WELCOME TO CDUT!”*

- 数值型常量：由数字、小数点、正负号、浮点数标识**E**或**D**和复数标识**i**组成的可以进行算术运算的整数、浮点数或双精度浮点数和复数。

  其中，复数**COMPLEX**(实部，虚部)或者**COMPLEX**(实部)，可为浮点数或双精度浮点数。

```
IDL> X=COMPLEX(10.2,2)
IDL> PRINT,X
      (10.2000, 2.00000)
IDL> X=COMPLEX(10.2)
IDL> PRINT,X
      (10.2000, 0.000000)
```

表 2.3　浮点数和双精度浮点数常量格式表

浮　点　数		双精度浮点数	
n.	102.	n.	102.
. n	. 102	. n	. 102
n. n	10. 2	n. n	10. 2
nE	10E	nD	10D
nEsx	10E5	nDsx	10D5
n. Esx	10. E−3	n. Dsx	10. D−3
. nEsx	. 1E+12	. nDsx	. 1D+12
n. nEsx	2. 3E12	n. nDsx	2. 3D12

二、IDL语法基础：常量

- 逻辑型常量：用来表示逻辑判断的真（1表示）或假（0表示）的数据。

逻辑型常量真是指逻辑判断结果为真的数据，它包括以下几种情况：

(1)字节型、整型和长整型的奇数；

(2)非零的浮点型、双精度型和复数类型(包括单精度和双精度)；

(3)非空的字符串类型。

逻辑型常量假(非真)是指非真的数据，它包括以下几种情况：

(1)字节型、整型和长整型的零和偶数；

(2)浮点型、双精度型和复数类型(包括单精度和双精度)的零；

(3)空的字符串类型。

二、IDL语法基础：变量命名规则

2.5 变量名

在程序运行过程中其值可以发生变化的数据称作变量。变量名通常用英文字母（a~z）、数字（0~9）、下划线和符号$组成。

- 且必须以英文字母开头。
- 长度不能超过256个字符，中间不能有空格。
- 变量名没有大小写区别。
- IDL中的保留字不能作为变量名称。
- 变量名要能反映变量的特征，譬如ORBIT_PARAMETERS、ION_DENSITY
- IDL允许用户可以在任何时候创建新的变量或重新定义已有的变量。

IDL保留字（一些逻辑判断或固定语句用词）

And	Begin	Case	Common	Do
Else	End	Endcase	Endelse	EndFor
Endif	Endrep	Eq	For	Function
Ge	Goto	Gt	Ff	Le
Lt	Mod	Ne	Not	Of
On_ioerror	Or	Pro	Repeat	Then
Until	While	Xor		

二、IDL语法基础：变量命名规则

2.5 变量名

在程序运行过程中其值可以发生变化的数据称作变量。变量名通常用英文字母（a~z）、数字（0~9）、下划线和符号$组成。

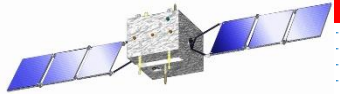
- 且必须以英文字母开头。
- 长度不能超过256个字符，中间不能有空格。
- 变量名没有大小写区别。
- IDL中的保留字不能作为变量名称。
- 变量名要能反映变量的特征，譬如 ORBIT_PARAMETERS、ION_DENSITY
- IDL允许用户可以在任何时候创建新的变量或重新定义已有的变量。

正确：*reade6_$file*，*only_8_bit*，*ComputerType*，*variables*，*data_day_of_year*，*var0_*

错误：*name.last*， *third%file*， *4th_list*，*$temp*

IDL保留字（一些逻辑判断或固定语句用词）

And	Begin	Case	Common	Do
Else	End	Endcase	Endelse	EndFor
Endif	Endrep	Eq	For	Function
Ge	Goto	Gt	Ff	Le
Lt	Mod	Ne	Not	Of
On_ioerror	Or	Pro	Repeat	Then
Until	While	Xor		



二、IDL语法基础：系统变量

2.5.1 系统变量：根据来源分为内部系统变量和自定义系统变量。

■ 内部系统变量：反馈系统运行状态或被程序直接调用。系统变量的名称均由感叹号“！”开头。系统变量根据其读写性能分为只读系统变量和可写系统变量。

■ 自定义系统变量：使用 DEFSYSV

格式：DEFSYSV,‘变量名’，表达式[, READ_ONLY][EXISTS = VARIABLE]

功能：定义一个值等于表达式的值的自定义系统变量。

说明：

A、变量名必须以“！”开头，而且变量名两边的单引号不能省略；

B、选项READ_ONLY是指定义的变量是只读变量，省略它则为读写变量；

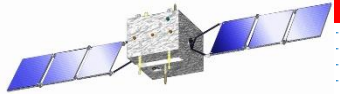
C、自定义系统变量一旦被定义，它的类型和结构不能再改变。

系统变量	意义	备注
!Dtor	度到弧度的转化因子	只读系统变量
!Radeg	弧度到度的转化因子	
!Pi	π（单精度）	
!Dpi	π（双精度）	
!Version.Release	IDL 版本号	
!Version.Os_family	操作平台（‘WinDows’，‘MacOS’，‘unix’或者‘vms’）	
!D.Flags		
!D.N_colors		
!D.Name		
!D.Table_size	颜色表的大小 !D.Table_size=256	
!D.Window		
!P.Multi	将多幅图画在同一页上的控制参数。例如：!p.multi=[0,1,6]	可写系统变量
!P.Font	控制画图字体。例如： !P.Font=1	
!Order		
!Path		
!Except		

二、IDL语法基础：内存变量

2.5.2 内存变量：临时存储于内存中的变量，用来存放操作或程序运行过程中的各种类型的原始数据、中间结果和最终结果。（用时临时定义，不用随时释放！）

数据类型	字节数	创建变量语法例子	数据类型转换函数
字节Byte	1	Var =0B	Var2 =byte(Var)
16位无符号整型 Unsigned Integer	2	Var =0U	Var2 =uint(Var)
16位有符号整型 Integet	2	Var =0	Var2 =fix(Var)
32位有符号整型 Long	4	Var =0L	Var2 =long(Var)
64位有符号整型 64-bit Long	8	Var =0LL	Var2 =long64(Var)
32位无符号整型 Unsigned Long	4	Var =0UL	Var2 =ulong(Var)
64位无符号整型Unsigned Long	8	Var =0ULL	Var2 =ulong64(Var)
浮点型 Floating-point	4	Var =0.0	Var2 =float(Var)
双精度浮点型 Double-precision	8	Var =0.0D	Var2 =double(Var)
复数Complex	8	Var =complex(0.0,0.0)	Var2 =complex(Var)
双精度复数 Double-precision complex	16	Var =complex(0.0D,0.0D)	Var2 =dcomplex(Var)
字符串string	1	Var = ‘’或Var = “”	Var2 =string(Var)
指针 Pointer	4	Var=Ptr_New()	None
对象 Object	4	Var=Obj_New()	None



二、IDL语法基础：内存变量

■ 内存变量操作：

内存变量需要时可以临时定义，不用时可以随时释放（注：系统变量不能释放）。每一个内存变量都是相互独立的，一旦退出IDL系统或关掉计算机，其内存变量的值全部丢失。

● 内存变量赋值

格式：变量 = 表达式 （其中，= 为赋值号）

例：

IDL> VALUE0=23.0

IDL> VALUE1=X-11.0

IDL> ISTRING0=“WELCOME TO ”

IDL> ISTRING1=ISTRING0 + “ ” + “CDUT!”

● 通过HELP命令查看变量属性

IDL> VALUE0=23.0

IDL> HELP,VALUE0

VALUE FLOAT=23.0000

二、IDL语法基础：内存变量

内存变量操作：

内存变量需要时可以临时定义，不用时可以随时释放（注：系统变量不能释放）。每一个内存变量都是相互独立的，一旦退出IDL系统或关掉计算机，其内存变量的值全部丢失。

内存变量的引用和输出

PRINT[,表达式1，…，表达式n][,FORMAT=格式]

```
IDL>PRINT,2019,FORMAT='(F7.2)'
      2019.00
```

HELP,表达式1，…，表达式n[，NAMES='通配符'][, OUTPUT=变量][, /SYSTEM_VARIABLES][, /STRUCTURES]

- 省略所有选项时，显示所有内存变量信息（名称、数据类型和值）

  例如： *IDL> VALUE0=23.0*
  IDL> HELP,VALUE0
  VALUE0 FLOAT=23.0000

- **NAMES='通配符'，只显示与通配符相匹配的变量信息**

  例如： *IDL> HELP, NAME='CD*'；只显示CD开头的变量信息*

- **OUTPUT=变量，把变量信息存入变量中**

  例如： *IDL> HELP, VALUE0, OUTPUT=VALUE1 ;将VALUE0的信息存入变量VALUE1*

- */SYSTEM_VARIABLES，系统变量当前信息，其它参数无效*

- **/STRUCTURES，显示结构体变量的详细信息**

  例如： *IDL> HELP, ISTRUCT0,/STRUCTURES ;显示ISTRUCT0的结构体信息*

二、IDL语法基础：内存变量

■ 内存变量操作：

内存变量需要时可以临时定义，不用时可以随时释放（注：系统变量不能释放）。每一个内存变量都是相互独立的，一旦退出IDL系统或关掉计算机，其内存变量的值全部丢失。

● 内存变量的存储：

格式：SAVE[, VAR1, … , VARN][, /ALL][, /COMM,/VARIABLES][,/COMPRESS]
[,FILENAME='FILENAME'][, /ROUTINES][, /SYSTEM_VARIABLES]

其中，

/ALL，将所有变量存储到指定的文件中；
/COMM和/VARIABLES，将公共块及其变量的内容存储到指定的文件中；
/COMPRESS，以压缩的形式存储变量；
FILENAME，指定文件的路径和文件名；
/ROUTINES，将编译的程序和函数存储到指定的文件中；（先编译: .sav）
/SYSTEM_VARIABLES，将所有的系统变量存储到指定的文件中。

例如：*SAVE,A,B,C,FILENAME='D:\IDLBENCH\IDLLESSON\MYDEMO\myvars.dat'*
SAVE,/ROUTINES,'HELLOSPACE',FILENAME=='D:\IDLBENCH\IDLLESSON\MYDEMO\mypro.sav'

● 内存变量的释放：格式.RESET_SESSION 或者.FULL_RESET_SESSION.
关闭所有文件，释放所有用户定义的变量、指针、对象等等，初始化IDL环境，其中 .FULL_RESET_SESSION 将卸载所有动态库。

● 内存变量的恢复：RESTORE [[,“文件名”] | [，FILENAME=“文件名”]]
若省略所有选项，则恢复文件IDLSAVE.DAT中的变量、公共块、程序和函数。

二、IDL语法基础：数据类型转换

表 2.4　数据类型和转换函数

数据类型	字节数	创建变量	数据类型函数
字节型	1	Var=0B	Var2=Byte(Varl)
有符号整型	2	Var=0	Var2=Fix(Varl)
无符号整型	2	Var=0U	Var2=Ulnt(Varl)
有符号长整型	4	Var=0L	Var2=Long(Varl)
无符号长整型	4	Var=0UL	Var2=ULong(Varl)
64 位有符号整型	8	Var=0LL	Var2=Long64(Varl)
64 位无符号整型	8	Var=0ULL	Var2=Ulong64(Varl)
浮 点 型	4	Var=0.0	Var2=Float(Varl)
双精度浮点型	8	Var=0.0D	Var2=Double(Varl)
复　　数	8	Var=Complex(0.0,0.0)	Var2=Complex(Varl)
双精度复数	16	Var=Dcomplex(0.0D,0.0D)	Var2=DComplex(Varl)
字 符 串	0～3 276	Var="或 Var=""	Var2=String(Varl)

二、IDL语法基础：数据类型转换

- **数据类型转换及注意事项：**
- 通过转换函数将数据从一种类型转换到另外一种类型

```
IDL> X=5.77777        ；X也可以是数组
IDL> HELP,X
      X          FLOAT = 5.77777
IDL> X=FIX(X)
IDL> HELP,X
      X          INT = 5
```

在不同数据类型转换过程中，如果输出变量类型的字节数少于输入变量的字节数，将会发生截断现象，但IDL不会发出警告。如

```
IDL> VALUE=257              ；16位整型（INT）
IDL> HELP,VALUE
      VALUE       INT     =     257
IDL> VALUE=BYTE(VALUE)     ；字节变量（无符号8位整型）
IDL> HELP,VALUE
      VALUE       BYTE    =   1
```

- 当定义一个范围在-32768到+32767之间的整数时，如果不用表示符B、U、L等，IDL默认的整数是16位的有符号整型。

```
IDL> A=-32767
IDL> HELP,A
      A          INT =-32767
```

二、IDL语法基础：数据类型转换

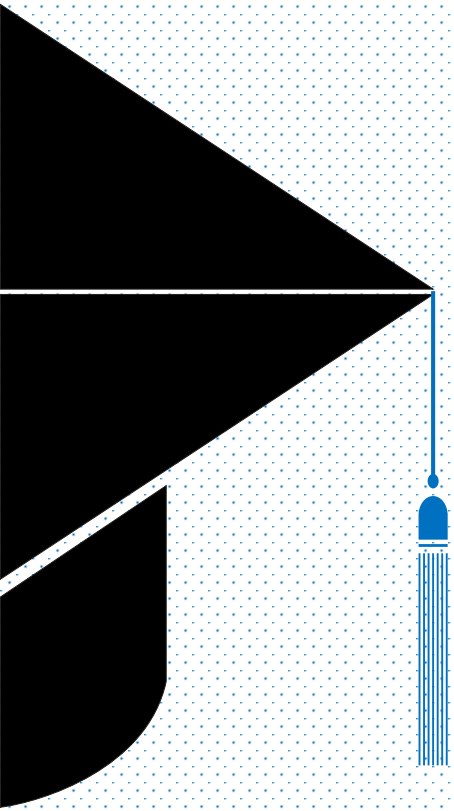
- 浮点数向整型的转化（取整操作）：

通过转换函数将数据从一种类型转换到另外一种类型

```
IDL> ARR=[5.35,9.87,-4.66,-10.0]
IDL> PRINT,FIX(ARR)          ;FIX()转化为整型
       5       9      -4     -10
IDL> PRINT,LONG(ARR)         ;LONG()转化为长整型
       5       9      -4     -10
IDL> PRINT,ROUND(ARR)        ;ROUND()返回一个最接近自变量的长整型整数
       5      10      -5     -10
IDL> PRINT,FLOOR(ARR)        ;FLOOR()返回一个小于或等于自变量，但最接近自变量的长整型整数
       5       9      -5     -10
IDL> PRINT,CEIL(ARR)         ;CEIL()返回一个大于或等于自变量，但最接近自变量的长整型整数
       6      10      -4     -10
```

- 字符串和数字类型之间的转换：用FLOAT(‘ ’)和BYTE(‘ ’) 等

成都理工大學
CHENGDU UNIVERSITY OF TECHNOLOGY



演示完毕

Thank You