Mathematica程序设计

如果要让Mathematica为你完成复杂的数学计算问题，那就需要利用Mathematica编写程序或者函数。本节将简要介绍Mathematica程序设计的基本功能。

1、全局变量与局部变量

如果不使用Clear[]等命令删除的话，全局变量在整个程序中都存在，而使用Module[]或者Block[]定义的变量称为局部变量(称为模块)，它只在所定义的模块内是可见的。实际上，模块就是其它计算机语言中的函数或者子程序。

- **Module[{x,y,…},body]** 建立模块，并且申请局部变量x,y,…
- **Module[{x=x0,y=y0,…},body]** 同上，但已经给局部变量赋初值
- **Block[{x,y,…},body]** 建立模块，并且申请局部变量x,y,…
- **Block[{x=x0,y=y0,…},body]** 同上，但已经给局部变量赋初值

其中body中可含有多个语句，除最后一个语句外，各语句间以分号结尾，可以多个语句占用一行，也可一个语句占用多行。但这两个命令略有差别，当Module[]申请的局部变量与全局变量重名时，它会在内存中重新建立一个新的变量，Module[]运行完毕，这个新的局部变量也会从内存中消失，而Block[]此时不会建立新的变量，它将重名的全局变量的值存起来，然后使用全局变量作为局部变量，当Block[]运行完毕后，再恢复全局变量的值。

另外，如果在Module[]或Block[]中有Return[expr]命令，则程序执行到Return[expr]后，将会跳出模块，并返回expr的值；则模块中无Return[]命令，则返回模块中最后一个语句的计算结果(注：最后一个语句不能用分号结束，否则将返回Null，即空信息)。

下面这段程序是用Module编写的,它不需要输入任何信息,也不返回任何信息,但运行此程序,由打印出程序的计算计算结果.

```
In[4]:= mmm := Module[{x, y, z}, x = 1; y = 2;
                    z = x + y; Print[z];
                  ];

In[5]:= mmm

        3
```

此段程序与程序:

x=1;y=2;z=x+y;Print[z];

的运行结果相同,但上面的程序中的x,y,z是局部变量,而后面的程序中是全局变量.

如果将刚才的程序变为如下形式:

```
In[6]:= f[x_, y_] := Module[{z}, z = x + y; Return[z];];

In[7]:= f[1, 2]

Out[7]= 3

In[8]:= f[a + 1, b + 2]

Out[8]= 3 + a + b
```

则它就是一个即有输入又有输出的子程序,其中的f[x_,y_]中的下划线是必不可少的,如果你对其它计算机语言很熟悉,你一定要问一个问题:

程序中的参数x,y是什么类型的变量?

实际上,它是mathematica中任一合法表达式.

看看下面对此程序的一个小小改动:

```
In[11]:= f[x_, y_] := Module[{xx, yy, z}, xx = Sin[x];
                                yy = Expand[y];
                                z = xx + yy; Return[z];];

In[12]:= f[t^2, (t + 1)^2]

Out[12]= 1 + 2 t + t^2 + Sin[t^2]
```

在某些情况下,我们可能需要输入参数x,y是某种特殊类型的表达式,可以这样写:

```
In[16]:= g[x_Integer, y_Real] := Module[{z}, z = x + y; Return[z];]

In[17]:= g[3, 4.5]

Out[17]= 7.5

In[18]:= G[3, 4]

Out[18]= G[3, 4]
```

2、 输入与输出

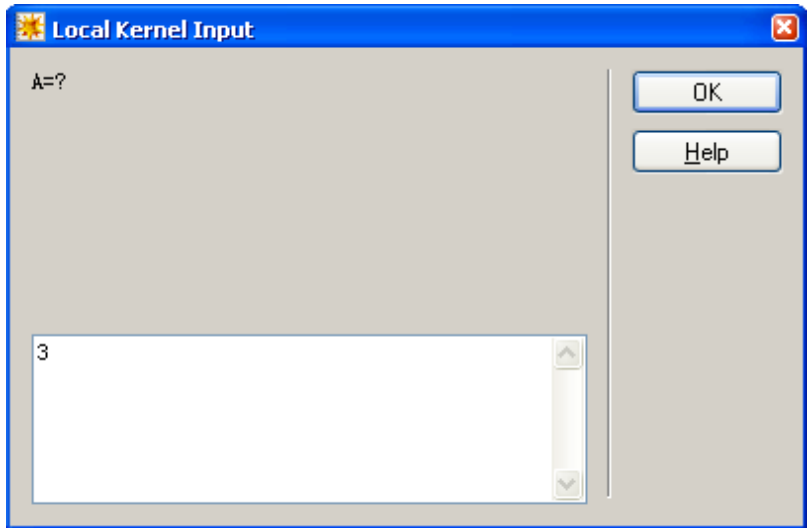
Mathematica中，有两个在Notebook中不常用到的函数，对于程序设计来说很方便。

- **Print[expr1,expr2,…] 打印表达式**
- **Input[“string”] 通过键盘输入表达式，其中string为提示字符串**

请看看下面的例子：

```
In[19]:= A = Input["A=?"];
         Print["A^2=", A^2];

         A^2=9
```



3、条件语句

Mathematica提供了多种设置条件的方法，对于编程来说很方便。

- lhs=rhs/；test 当test为True时，执行lhs=rhs
- If[test,then,else] 若test为True，执行then，否则执行else
- Which[test1,value1,test2,value2,…] 返回首个testi为True时的valuei值
- Switch[expr,form1,value1,form2,value2,…] 先计算expr的值，然后依次与formi比较，返回首个与formi匹配的valuei的值，如果没有匹配项则返回Null
- Switch[expr,form1,value1,form2,value2,…,def] 同上，但如果没有匹配则返回def

下面以一个分段函数的定义为例来说明条件语句的使用方法:

```
f[x_] := If[Abs[x] >= 2, 2, If[(Abs[x] >= 1) && (Abs[x] < 2), Abs[x], x^2]]
```

此外，利用Which[]也可写成

```
f[x_] := Which[Abs[x] >= 2, 2, (Abs[x] >= 1) && (Abs[x] < 2), Abs[x],
   (Abs[x] < 1), x^2]
```

还可以写成如下形式

```
f[x_] := 2 /; Abs[x] >= 2
f[x_] := Abs[x] /; (Abs[x] >= 1) && (Abs[x] < 2)
f[x_] := x^2 /; Abs[x] < 1
```

在Mathematica中，用于条件判断的逻辑运算符与C语言是一致的，它们是>、>=、<、<=、==、!=、&&、||。

4、循环语句

Mathematica提供了多种循环方式，例如Nest[]、FixedPoint[]等等，而我们下面所介绍的是与其它计算机语言相似的三种循环结构，即Do循环、For循环和While循环。

- Do[expr,{n}] 将表达式重复计算n次
- Do[expr,{i,imax}] 计算expr，i从1到imax，步长为1
- Do[expr,{i,imin,imax}] 计算expr，i从imin到imax,步长为1
- Do[expr,{i,imin,imax,di}] 同上，但步长为di
- For[start,test,incr,body] 以start为初值，重复计算body，当test为False时，循环终止。其中incr一般为循环计数器
- While[test,body] 只要test为True时就重复执行body

Do循环是三种循环结构中用法最简单的一个，下面是两个例子:

```
S = 0; Do[S = S + 1 / k^2, {k, 1, 1000}]; Print[N[Pi^2 / 6 - S]]
0.0009995
S = {}; Do[AppendTo[S, {x, x^2}], {x, -1, 1, 0.2}]; S
{{-1, 1}, {-0.8, 0.64}, {-0.6, 0.36}, {-0.4, 0.16}, {-0.2, 0.04},
 {0., 0.}, {0.2, 0.04}, {0.4, 0.16}, {0.6, 0.36}, {0.8, 0.64}, {1., 1.}}
```

For循环与C语言中的for(;;)语句用法一样，只不过现在变成了For[,,,]的形式，下面是有关For循环的例子:

```
For[i = 1, i ≤ 10, i++,
          a = i^2; Print["a=", a];
   ]
a=1
a=4
.....
a=81
a=100
```

其中i++的用法与C语言一样，你也可以用i=i+1代替。再如

```
a = {}; For[x = 0, x ≤ 1, x = x + 0.1, AppendTo[a, x^2]]; a
{0, 0.01, 0.04, 0.09, 0.16, 0.25, 0.36, 0.49, 0.64, 0.81, 1.}
```

最后举一个While循环的例子，利用乘幂法求矩阵A按模最大的特征值λ，其计算方法说明如下：首先选定一个初始向量z_0，然后反复用公式$y_k=Az_{k-1}$，$m_k=\max|y_k|$，$z_k=y_k/m_k(k=1,2,...)$，最后得$\lambda=\lim\ m_k$。

```
Clear[A, y, z, err, m, n, k];
A = {{1, 2, 3}, {2, 3, 4}, {3, 4, 5}}; err = 10^(-6); y = {1, 1, 1};
z = {2, 2, 2}; m = k = 0; n = 1;
While[Abs[m - n] > err, y = A.z // N; n = m; k++; m = Max[y] // N;
                        z = y / m // N; If[k == 200, Break[]];
      ];
Print["k=", k, "          m=", m];
k=8          m=9.62348
```

5、程序的流程控制

- **Break[] 退出最近一层的循环结构**
- **Continue[] 忽略Continue[]后面的语句，进入下一次循环**
- **Return[expr] 用于函数中，返回expr的值**
- **Label[name] 定义一个名字为name的标号**
- **Goto[name] 直接跳转到当前过程中的name标号处**
- **Break[]只能用于循环结构之中，它退出离它最近的一层循环结构。**

```
Do[a = i^2; If[a > 80, Break[]], {k, 1, 10}]; Print[k]
```

```
8
```

Continue[]也与Break[]一样，用于循环语句中，当程序执行到此语句后，将不会执行当前循环中Continue[]后面的语句，而是继续下一次循环。

```
For[i = 1, i ≤ 4, i++, If[i == 2, Continue[]]; Print[i];]
```

```
1
3
4
```

Label[]与Goto[]的用法与BASIC语言相同，下面是一个例子:

```
Clear[i]; i = 1; Label[one]; Print["i=", i]; i = i + 1;
If[i ≤ 3, Goto[one], Goto[two]]; Print["**********"];
Label[two];
```

```
i=1
i=2
i=3
```

在使用Block[]用Module[]编写函数子程序时，如果你没有使用Return[]语句，则程序返回最后一个表达式的值，并且此时最后一个表达式不能以分号结尾，否则程序将返回Null，即返回一个空信息。但是，我们可以使用Return[]语句在任何地方从程序中跳出，将某个表达式返回给函数。Return[]语句可以返回任何一个合法的Mathematica表达式。下面的例子是利用四阶龙格-库塔法求一阶微分方程$y'=f(x,y),y(x_0)=y_0$的数值解，其中：x1为终止计算时的x值，h为计算步长，函数y[x]将返回满足方程$y'=f(x,y),y(x_0)=y_0$的积分曲线上在区间[x0,x]上间隔为h的每一点上的值，并将这些数据存入data之中，我们最后用ListPlot[]画出了方程解的函数曲线。

```
Clear[f, "x*", "y*", h, data]; f[x_, y_] = Input["f[x,y]="] // N;
x0 = Input["init x0="] // N; y0 = Input["init y0="] // N;
x1 = Input["end  y="] // N; h = Input["step length="] // N;
h = Abs[h]; If[x1 < x0, h = -h]; data = {};
y[x_] := Block[{k1, k2, k3, k4, nowx, nowy, end, hh}, end = "n";
  hh = h; nowx = x0; nowy = y0;
  While[end == "n", AppendTo[data, {nowx, nowy}];
   If[Abs[nowx + hh] > Abs[x], hh = x - nowx; nowx = x; end = "y",
    nowx = nowx + hh];
   k1 = hh * f[nowx, nowy]; k2 = hh * f[nowx + h / 2, nowy + k1 / 2];
   k3 = hh * f[nowx + h / 2, nowy + k2 / 2]; k4 = hh * f[nowx + hh, nowy + k3];
   nowy = nowy + (k1 + 2 * k2 + 2 * k3 + k4) / 6 // N;];];
y[x1];
ListPlot[data];
```

6、数据的存取

Mathematica提供了一系列基本输入与输出函数用于文件的读写，在Mathematica中，你可以将运算结果存入文件供其它应用程序调用，也可以从磁盘上读入其它应用程序生成的数据供Mathematica使用。下面是两个常用的数据读取语句。

•WriteString[“file”,expr1,expr2,…]向文件file中写入表达式。如果file不存在，则WriteString[]就先在磁盘上创建文件file，然后打开它；如果文件存在，则WriteString[]以向file尾部追加方式打开它。如果要向file中输出字符串，可用双引号括起来。另外，WriteString[]是不会自动输出回车换行符的，你可用向file中写入”\n”的方式在file中加上回车换行符。下面的例子将1到15这15个数分三行写入文件abc.dat，数据中间以逗号分隔。

```
f = "abc.dat"; If[Length[FileNames[f]] == 0, DeleteFile[f]]; n = 5;
j = 1; For[i = 1, i ≤ 15, i++,
 If[j < 5, WriteString[f, i, ","]; j = j + 1, WriteString[f, i, "\n"];
   j = 1;];];
Close[f];
```

程序第一行的意思是，如果文件abc.dat存在，就先删除此文件，然后再向文件中写入数据，最后用Close[“file”]语句关闭文件。

```
!! abc.dat
```

1,2,3,4,5
6,7,8,9,10
11,12,13,14,15

在Mathematica中，通过“!!”命令，可以列出文本文件中的的内容，其它命令还有：“<<”读入一个Mathematica程序并运行此程序。

•**ReadList[“file”,type]以指定类型type读入文件中的所有数据并将它赋给一个一维表；**

•**ReadList[“file”,{type1,type2,…}]读入所有数据，并将它赋给一个二维表，如果文件file中的数据不够则用文件结束符EndOfFile补齐；ReadList[“file”,type,n]只读入文件file中的前n个数据，形成一个一维表。其中type的类型为：String(字符串)、Integer(整数)、Real(近似实数)、Number(精确数或者近似数)。另外，文件中的数据与数据间要用空格或者回车换行符分隔开。假设C盘根目录下文件file.dat中存有16个数据，用记事本所看到的数据如下：**

1 2 3 4 5
7 8
9 10 11
12 13 14 15
16

下面是用ReadList[]读取文件file的几个例子，请注意，如果文件名中带有路径，则应该用双反斜杠，如：c:\\dir\\dir1\\abc.dat。

```
Clear[a, b, c, d]; f = "c:\\file.dat"; a = ReadList[f, Number]
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16}
```

```
b = ReadList[f, {Number, Number}]
```

```
{{1, 2}, {3, 4}, {5, 6}, {7, 8}, {9, 10}, {11, 12}, {13, 14}, {15, 16}}
```

```
c = ReadList[f, {Number, Number, Number, Number}]
```

```
{{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16}}
```

在mathematica V4.2以后版本中,提供了两个更为强大的数据输入与输出函数,即Import[]和Export[],它可以输入或者输出数据文件,图像及声音文件,它完全可以代替上面两个函数的功能,下面是简单的例子

假设在C:\有数据文件dat1.txt,格式如下:

1 2 3 4
5 6 7 8
9 10 11 12

则以下命令可将数据读入内存,或者以回车为行分界符的矩阵格式,或者为以空格为分界符的一维数组格式:

```
In[11]:= a = Import["c:\\dat1.txt", "Table"]

Out[11]= {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}

In[12]:= a = Import["c:\\dat1.txt", "List"]

Out[12]= {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

Import[]命令可以读入任意格式的图像文件:JPEG,BMP,TIFF等等,并形成一个图像矩阵

由于此方面内容太烦琐,需要使用这些命令,可以参考mathematica的帮助,下面是一个读入BMP文件的例子,假设在C:\下有文件lena.bmp,它是一个大小是256x256,并且具有256级灰度的BMP格式的图像,下面读入此图像:

```
In[14]:= b = Import["c:\\lena.bmp"];

In[17]:= q = b[[1, 1]];

In[18]:= Dimensions[q]

Out[18]= {256, 256}

In[19]:= ListDensityPlot[q, Mesh → False, Frame → False];
```

其中q是一个256x256的矩阵,q中的每一个元素,代表图像中的一个点,即灰度值.

Import[]命令是从磁盘上向内存中读入文件,而另外的一个命令即Export[]命令则是将内存中的数据或者说变量写入磁盘,其用法与Import[]命令大致相同,关于Export[]命令的使用方法与Import[]命令的进一步用法,可参考mathematica的帮助.