Mathematica 的最新版本是 Mathematica 8，常用版本是 Mathematica 6。下载并安装后会直接得到一个可执行的程序，不需要做其他配置。需要破解。

Mathematica 是一个函数式的语言。函数的参数用方括号括起来，中间用逗号分割参数。系统函数、常数是大写字母开头的，用户自己定义的函数和变量用小写字母开头。

list 是最基本的数据结构，集合、矩阵、数组、表格、图片、动画都是 list。list 的表示方法是，用花括号括起来，中间用逗号分割。list 可以嵌套，得到多维度的 list。

Table 是最常用的生成 list 的函数。组合数学中经常用 Tuples、Subsets、Permutations、RandomChoice、RandomSample 等函数来生成 list。常用于 list 的函数：First、Last、Take、Drop、Select、Delete、Append、Prepend、Reverse、Length、Position、MemberQ、Count、Flatten、Join、Sort、Tally、Total、Map、Apply 等。他们的用法都可以在帮助文档中找到。

在 Mathematica 中，数都是用符号精确表示出来的，1/2+1/3 可以直接等于 5/6。查看一个数的小数近似，可以用函数 N。赋值用=，等号用==，不等于用!=，乘方用^，圆周率用 Pi，自然底数用 E，无穷用 Infinity。这些数学符号也可以通过点击工具栏的按钮直接输入。

Mathematica 的字符串加双引号表示，用\转义。字符串函数和 list 的函数很像，只是函数名加了一个 String。例如 StringTake、StringDrop、StringLength 等等。

在 Mathematica 中，执行语句用 Shift 加回车。%表示上一个输出，%%表示上上个输出，%12 表示序号为 12 的输出。

Mathematica 内置了大量函数，帮助文档中介绍得非常详细。嵌套使用更有神奇的效果，举例如下：

列出前 100 个质数：Table[Prime[i], {i, 1, 100}]

解方程：Solve[x^2 + x == 1, x]

代数式展开：Expand[(1 + x)^3]

求极限：Limit[Sin[x]/x, x -> 0]

求导数：D[x^2, x]

求积分：Integrate[x^2, x]

求无穷级数的和：Sum[1/i^2, {i, 1, Infinity}]

因式分解：Factor[x^2 - 1]

质因数分解：FactorInteger[1001]

找出一组勾股数：FindInstance[x^2 + y^2 == z^2 && x > 0 && y > 0 && z > 0, {x, y, z}, Integers]

画函数图像：Plot[x^2, {x, -2, 2}]

画方程：ContourPlot[x^2 + y^2 == 1, {x, -2, 2}, {y, -2, 2}]

画不等式：RegionPlot[x^2 + y^2 < 1, {x, -2, 2}, {y, -2, 2}]

画三维图像：Plot3D[Sin[x] Cos[y], {x, -3, 3}, {y, -3, 3}]

播放平方函数的“声音”：Play[Sin[5000 t^2], {t, -1, 1}]

列出首尾三个（及以上）的字母完全相同的单词：DictionaryLookup[RegularExpression["([a-z]{3,})[a-z]*\1"]]

查询单词的意义：WordData["football"]

字频统计：Tally[Characters["上海自来水来自海上"]]

列出所有 GBK 一级汉字：Drop[Flatten[Table[FromCharacterCode[{i, j}, "CP936"], {i, 160 + 16, 160 + 55}, {j, 161, 254}]], -5]

按像素密度排序:SortBy[Characters["按照字符的像素多少对这句话中的所有字重新排序"], Count[Flatten[ImageData[Binarize[Rasterize[Style[#, FontSize -> 20]]]]], 0] &]

Mathematica 支持 For、While 等循环的方式。下面这个句子可以计算从 1 加到 100 的和（当然只是举例，其实可以用 Total[Range[1, 100]]直接算出来）：

```
s = 0;
For[i = 1, i <= 100, i++, s = s + i];
Print[s];
```

写程序时经常用分号结尾，这不但起到连接语句、排版规范的作用，还可以防止 Mathematica 把运行时函数返回的结果自动打印出来。如果真的要打印东西，用函数 Print。
Mathematica 的变量不用事先声明，s = 0 就立即让 s 有了值。这意味着，下文的 s 就不能乱用了，比如

```
s = 0;
Solve[s^2 + s == 1, s]
```

就会出错，因为 Mathematica 将以为我们要解方程 0^2 + 0 == 1 中的 0，于是返回 Solve[False,

0]，并提示 0 不是未知量。如果想把 s 还原为未定义的状态，可以执行 Remove[s]。

下面是传统的 3x+1 问题模拟程序，取初始值 n = 27。

```
n = 27; Print[n];
While[n != 1,
  If[EvenQ[n], n = n/2, n = 3 n + 1];
  Print[n];
  ];
```

在 Mathematica 中，我们倾向于把结果整理成 list，以便今后完成作图之类的操作：

```
n = 27; l = {n};
While[n != 1,
  If[EvenQ[n], n = n/2, n = 3 n + 1];
  l = Append[l, n];
  ];
Print[l];
```

当然，其实上述过程也有一个专门的函数来实现。我们可以利用 NestWhileList 加上一些临时定义的没有名字的函数(Mathematica 中这叫做 Pure Functions)一句话完成刚才的操作：

```
NestWhileList[If[EvenQ[#], #/2, 3 # + 1] &, 27, # != 1 &]
```

Mathematica 可以定义函数。下面定义一个“自然数连接”函数

```
f[x_, y_] := FromDigits[Join[IntegerDigits[x], IntegerDigits[y]]];
f[123, 4567]
```

局部变量用 Module，这里不细讲。

函数可以递归定义，例如 Fibonacci 函数可以这么写（只是举例而已，其实 Mathematica 自带了 Fibonacci 这个函数）：

```
f[x_] := f[x - 1] + f[x - 2];
f[1] := 1;
f[2] := 2;
Table[f[i], {i, 1, 10}]
```

但如果输出前 100 个 Fibonacci 数就很慢，因为我们没有让它记住算过的值。实现记忆化的方法如下：

```
f[x_] := f[x] = f[x - 1] + f[x - 2];
f[1] := 1;
```

```
f[2] := 2;
Table[f[i], {i, 1, 100}]
```

讲到这里就差不多了，Mathematica 的帮助文档非常详细，有大量的示例，我基本上是看帮助文档学习的。
有问题可以给我发邮件：gs.matrix67@gmail.com
希望大家能利用 Mathematica 搞一些无聊的小研究。