数学和编程

好些人来信问我，要成为一个好的程序员，数学基础要达到什么样的程度？十八年前，当我成为大学计算机系新生的时候，也为同样的问题所困扰。面对学数学，物理等学科的同学，我感到自卑。经常有人说那些专业的知识更加精华一些，难度更高一些，那些专业的人毕业之后如果做编程工作，水平其实比计算机系毕业的还要高。直到深入研究程序语言之后，对这个问题我才得到了答案和解脱。由于好多编程新手遇到同样的困扰，所以我想在这里把这个问题详细的阐述一下。

数学并不是计算机科学的基础

很多人都盲目的认为，计算机科学是数学的一个分支，数学是计算机科学的基础，数学是更加博大精深的科学。这些人以为只要学会了数学，编程的事情全都不在话下，然而事实却并非如此。

事实其实是这样的：

- 计算机科学根本不是数学，它只不过借用了非常少，非常基础的数学，比高中数学还要容易。
- 所谓“高等数学”，并不是研究计算机科学必须的。你可以用计算机来做微积分计算，可是这时候你其实是在做数学工作，用计算机作为工具。你研究的并不是计算机科学。这就像你可以用计算机来设计建筑，但建筑学却不是计算机科学的基础。
- 计算机是比数学更加基础的工具，就像纸和笔一样。计算机可以用来解决数学的问题，也可以用来解决不是数学的问题，比如工程的问题，艺术的问题，经济的问题，社会的问题等等。
- 计算机科学是完全独立的学科。学习了数学和物理，并不能代替对计算机科学的学习。你必须针对计算机科学进行学习，才有可能成为好的程序员。
- 数学家所用的语言，比起常见的程序语言（比如C++，Java）来说，其实是非常落后而蹩脚的设计。所谓“数学的美感”，其实大部分是夜郎自大。
- 99% 的数学家都写不出像样的代码。

数学是异常糟糕的语言

这并不是危言耸听。如果你深入研究过程序语言的理论，就会发现其实数学家们使用的那些符号，其实是一种非常糟糕的程序语言。数学的理论很多是有用的，然而数学家门用于描述这些理论所用的语言，却是纷繁复杂，缺乏一致性，可组合性（composability），简单性，可用性。这也就是为什么大部分人看到数学就头痛。这不是他们不够聪明，而是数学语言的“设计”有问题。人们学习数学的时候，其实只有少部分时间在思考它的精髓，而大部分时间是在折腾它的语法。

举一个非常简单的例子。如果你说 $\cos^2\theta$ 表示 $(\cos\theta)^2$，那么理所当然，$\cos^{-1}\theta$ 就应该表示 $1/(\cos\theta)$ 了？可它偏偏不是！别被数学老师们的教条和借口欺骗啦，他们总是告诉你：“你应该记住这些！”可是你想过吗：凭什么？$\cos^2\theta$ 表示 $(\cos\theta)^2$，而 $\cos^{-1}\theta$，明明是一模一样的形式，表示的却是 $\arccos\theta$。一个是求幂，一个是调用反函数，风马不及，却写成一个样子。这样的语言设计混淆不堪，却喜欢以“约定俗成”作为借口。

如果你再多看一些数学书，就会发现这只是数学语言几百年累积下来的糟粕的冰山一角。数学书里尽是各种上标下标，带括号的上标下标，x，y，z，a，b，c，f，g，h，各种扭来扭去的希腊字母，希伯来字母…… 斜体，黑体，花体，双影体，……用不同的字体来表示不同的“类型”。很多符号的含义，在不同的子领域里面都不一样。有些人上一门数学课，到最后还没明白那些符号是什么意思。

直到今天，数学家们写书仍然非常不严谨。他们常犯的一个错误是把 x^2 这样的东西叫做“函数”（function）。其实 x^2 不是一个函数，它只是一个表达式。你必须同时指明“x 是参数”，加上 x^2，才会成为一个函数。所以正确的函数写法其实看起来像这样：$f(x) = x^2$。或者如果你不想给它一个名字，可以借用 lambda calculus 的写法，写成：$\lambda x.x^2$。

可是数学家们灰常的喜欢“约定俗成”。他们定了一些不成文的规矩是这样：凡是叫“x”的，都是函数的参数，凡是叫“y”的，都可能是一个函数…… 所以你写 x^2 就可以表示 $\lambda x.x^2$，而不需要显式的写出“λx”。殊不知这些约定俗成，看起来貌似可以让你少写几个字，却造成了许许多多的混淆和麻烦。比如，你在 Mathematica 里面可以对 $x^2 + y$ 求关于x的导数，而且会得到 `y'(x) + 2x` 这样蹊跷的结果，因为它认为 `y` 可能是一个函数。更奇怪的是，如果你在后面多加一个 `a`，也就是对 $x^2 + y + a$ 求导，你会得到 `2x`！那么 `y'(x)` 到哪里去了？莫名其妙……

相对而言，程序语言就严谨很多，所有的程序语言都要求你必须指出函数的参数叫什么名字。像 x^2 这样的东西，在程序语言里面不是一个函数（function），而只是一个表达式（expression）。即使 JavaScript 这样毛病众多的语言都是这样。比如，你必须写：

```
function (x) { return x * x }
```

那个括号里的`(x)`，显式的声明了变量的名字，避免了可能出现的混淆。我不是第一个指出这些问题的人。其实现代逻

辑学的鼻祖 Gottlob Frege 在一百多年以前就在他的论文“Function and Concept”里批评了数学家们的这种做法。可是数学界的表达方式直到今天还是一样的混乱。

很多人学习微积分都觉得困难，其实问题不在他们，而在于莱布尼兹（Leibniz）。莱布尼兹设计来描述微积分的语言（∫, dx, dy, …），从现代语言设计的角度来看，其实非常之糟糕，可以说是一塌糊涂。我不能怪莱布尼兹，他毕竟是几百年前的人了，他不知道我们现在知道的很多东西。然而古人的设计，现在还不考虑改进，反而当成教条灌输给学生，那就是不思进取了。

数学的语言不像程序语言，它的历史太久，没有经过系统的，考虑周全的，统一的设计。各种数学符号的出现，往往是历史上某个数学家有天在黑板上随手画出一些古怪的符号，说这代表什么，那代表什么，…… 然后就定下来了。很多数学家只关心自己那块狭窄的子领域，为自己的理论随便设计出一套符号，完全不管这些是否跟其它子领域的符号相冲突。这就是为什么不同的数学子领域里写出同样的符号，却可以表示完全不同的涵义。在这种意义上，数学的语言跟 Perl（一种非常糟糕的程序语言）有些类似。Perl 把各种人需要的各种功能，不加选择地加进了语言里面，造成语言繁复不堪，甚至连Perl的创造者自己都不能理解它所有的功能。

数学的证明，使用的其实也是极其不严格的语言——古怪的符号，加上含糊不清，容易误解的人类语言。如果你知道什么是 Curry-Howard Correspondence 就会明白，其实每一个数学证明都不过是一段代码。同样的定理，可以有许多不同版本的证明（代码）。这些证明有的简短优雅，有的却冗长繁复，像面条一样绕来绕去，没法看懂。你经常在数学证明里面看到“未定义的变量”，证明的逻辑也包含着各种隐含知识，思维跳跃，非常难以理解。很多数学证明，从程序的观点来看，连编译都不会通过，就别提运行了。

数学家们往往不在乎证明的优雅性。他们认为只要能证明出定理，你管我的证明简不简单，容不容易看懂呢。你越是看不懂，就越是觉得我高深莫测，越是感觉你自己笨！这种思潮到了编程的时候就显出弊端了。数学家写代码，往往忽视代码的优雅性，简单性，模块化，可读性，性能，数据结构等重要因素，认为代码只要能算出结果就行。他们把代码当成跟证明一样，一次性的东西，所以他们的代码往往不能满足实际工程的严格要求。

数学里最在乎语言设计的分支，莫过于逻辑学了。很多人（包括很多程序语言专家）都盲目的崇拜逻辑学家，盲目的相信数理逻辑是优雅美好的语言。在程序语言界，数理逻辑已经成为一种灾害，明明很容易就能解释清楚的语义，非得写成一堆稀奇古怪，含义混淆的逻辑公式。殊不知其实数理逻辑也是有很大的历史遗留问题和误区的。研究逻辑学的人经常遇到各种“不可判定”（undecidable）问题和所谓“悖论”（paradox），研究几十年也没搞清楚，而其实那些问题都是他们自己造出来的。你只需要把语言改一下，去掉一些不必要的功能，问题就没了。但逻辑学家们总喜欢跟你说，那是某天才老祖宗想出来的，多么多么的了不起啊，不能改！

用一阶逻辑（first-order logic）这样的东西，你可以写出一些毫无意义的语句。逻辑老师们会告诉你，记住啦，这些是没有意义的，如果写出来这些东西，是你的问题！他们没有意识到，如果一个人可以用一个语言写出毫无意义的东西，那么这问题在于这个语言，而不在于这个人。一阶逻辑号称可以“表达所有数学”，结果事实却是，没有几个数学家真的可以用它表达很有用的知识。到后来，稍微明智一点的逻辑学家们开始研究这些老古董语言到底出了什么毛病，于是他们创造了 Model Theory 这样的理论。写出一些长篇大部头，用于“验证”这些逻辑语言的合理性。这些问题在我看来都是显而易见的，因为很多逻辑的语言根本就不是很好很有用的东西。去研究它们“为什么有毛病”，其实是白费力气。自己另外设计一个更好语言就完事了。

在我看来，除了现代逻辑学的鼻祖 Gottlob Frege 理解了逻辑的精髓，其它逻辑学家基本都是照本宣科，一知半解。他们喜欢把简单的问题搞复杂，制造一些新名词，说得玄乎其玄灵丹妙药似的。如果你想了解逻辑学的精华，建议你看看 Frege 的文集。看了之后你也许会发现，Frege 思想的精华，其实已经融入在几乎所有的程序语言里了。

编程是一门艺术

从上面你也许已经明白了，普通程序员使用的编程语言，就算是 C++ 这样毛病众多的语言，其实也已经比数学家使用的语言好很多。用数学的语言可以写出含糊复杂的证明，在期刊或者学术会议上蒙混过关，用程序语言写出来的代码却无法混过计算机这道严格的关卡。因为计算机不是人，它不会迷迷糊糊的点点头让你混过去，或者因为你是大师就不懂装懂。代码是需要经过现实的检验的。如果你的代码有问题，它迟早会导致出问题。

计算机科学并不是数学的一个分支，它在很大程度上是优于数学，高于数学的。有些数学的基本理论可以被计算机科学所用，然而计算机科学并不是数学的一部分。数学在语言方面带有太多的历史遗留糟粕，它其实是泥菩萨过河，自身难保，它根本解决不了编程中遇到的实际问题。

编程真的是一门艺术，因为它符合艺术的各种特征。艺术可以利用科学提供的工具，然而它却不是科学的一部分，它的地位也并不低于科学。和所有的艺术一样，编程能解决科学没法解决的问题，满足人们新的需求，开拓新的世界。所以亲爱的程序员们，别再为自己不懂很多数学而烦恼了。数学并不能帮助你写出好的程序，然而能写出好程序的人，却能更好的理解数学。我建议你们先学编程，再去看数学。

如果你想了解更多关于数学语言的弊病以及程序语言对它们的改进，我建议你看看这个 Gerald Susman 的讲座。