第一部分 入门篇

1.1 程序员的梦想——中国的比尔·盖茨

作为一个 IT 行业职员，我经历过一个普通程序员成长的过程，同时也接触过许多不同层次的程序员。他们或在我身边匆匆而过，或与我共同工作，或在我可以关注的范围内成长着。他们的喜怒和哀乐、挫折和成功、幻想和现实、希望和失望，无不与我心共振。我知道这个行业从业人员的梦想，也知道这个行业的残酷。无数人怀着希望而来，却抱着无奈离去。我早就有和他们共语的愿望，希望通过这个主题和他们交流程序员所关注的各种问题，希望我的经验有助于他们的成长，同时我也想谈谈 EOM 对程序员的真正价值的影响，以及如何实现“成为比尔·盖茨”这个程序员的最高梦想。

什么是程序员？什么人能称得上是程序员？会编程序的人都是程序员嘛！这个问题看似简单，但仔细想一下，也很难回答。其实在中国，关于程序员的称呼有很多种近似的叫法，例如“开发人员”、“编程人员”、“计算机人员”等。只是现在分工越来越细、专业化程度不断提高的情况下，程序员这个词才逐渐地流行起来。

那么什么人才算是程序员呢？现在看来凡是从计算机专业或相近专业毕业的、以编写程序为职业的人都可算得上是程序员。但是在 20 世纪 80 年代到 90 年代，由于计算机还是新生事物，整个社会对其有种神秘的、高贵的、不可触及的印象。加之当时计算机人才少之又少，除了计算机专业从事开发工作之外，很多非计算机专业的学生，甚至初、高中生也加入到计算机开发队伍之中。他们充满激情，敢于学习，勇于探索，其中有许多人很快就成为开发队伍中的主力军，成为编程人员中的佼佼者。有的时候，专业的程序员还不如业余的程序员，程序设计语言比较单调，技术书籍更是少之又少。记得当时只能把单位印制的 8086、Z80 等汇编程序设计资料当做教材，用 debug 把操作系统中的代码打印成厚厚书籍来阅读。由于当时我的单位是生产（组装）计算机的，因此，使用计算机还是比较方便的，但是，不像现在，绝不可能在家里使用计算机的。

由于那时程序员可以触及计算机，可以看明白别人不懂的代码，可以让计算机执行自己的指令，这让许多外行甚至内行人很羡慕。

程序员一般只掌握单一的程序设计语言，比如编写汇编程序的程序人员，编写 C 语言程序的程序人员，编写 Unix、XENIX、AIX、SCO、HP unix 等 Unix 类的 shell 程序的程序人员，编写与数据库打交道的 proc c 程序人员，编写 C++程序的程序人员，编写面向对象的 VFP、VB、Delphi、PB 的程序人员，编写 Web 程序

的 HTML、CGI、ASP、PHP 的程序人员，编写 C#、Java 的程序人员。这些程序语言有些保留了下来，有的则被时代无情地淘汰了。同样是程序员，如果不能适应语言的发展也避免不了被淘汰的命运。

我自己认为的程序员与其他职业人员之间的区别：

1）因为比尔·盖茨是编程序的，所以似乎每个程序设计人员都有一个“比尔·盖茨”梦想：比尔·盖茨能做的，我也许能做到，即使做不到，做到一半也是不错的。这个潜在的意识是程序员最大的财富，许多程序人成了为理想、有抱负的人。我想很多人选择 IT 大都是与此有关吧。

2）工作成果完全由自己把握，随时编随时运行随时出结果。这种自我感觉是很多职业都不具备的。因此，程序员有很强的自信心。而且这种自信心往往可以使得程序员产生自己开公司的念头。

3）有很高的预期价值。软件通过使用创造了价值，程序员通过制作产生了软件。因此，程序员往往把软件的价值看做自己的价值，例如一个软件卖了 5 万元，程序员就会把自己的劳动价值估算在 5 万元以上；如果这个软件有 100 个潜在的市场，那么程序员就会把这个价值升值到 5×100=500 万以上。所以在程序员这个群体中，很多人都相信自己未来能够获得更多的收入。

当然，程序员也有其他一些特点，例如：有的程序员喜欢晚上干活，白天睡觉；有的喜欢钻研，连续加班；有的头脑灵活，动手能力强；有的喜欢追逐最新技术，变成别人公司的代言人等。

我注意到了有这样一点，那就是现在的程序员已经失去了神秘感，以往给人以仰目而视的形象正逐步走下神坛。

这是程序员职业发展的必由之路，抑或是再正常不过的社会现象？

1.2 谈谈程序员的基础知识

对于程序员需要具备哪些最基础的知识和技能这个问题，不少刚从大专院校毕业出来的新入职员工，甚至是从事过一段时间编程工作的程序员，都是比较模糊的。只有认真掌握一些基础的知识和技能，才能走上程序员这条大道。

说句实在话，我在从事编程工作很长一段时间内都没有关注这个问题，基本上是边编边学，边学边编，从学习中积累，从编程中积累。除了和同事进行工作上交流之外，学习材料很少，基本上是一个人在战斗，也不知道自己是不是成为了合格的程序员。也许是受那个年代所限，当时的程序员人数很少，因此同行间没有什么竞争，有了位置就不怕失去。要是放在现在，真的很后怕。

回到正题，我认为程序员在最初阶段要从流程、语法、调用三个层次要求自己。

1.流程

这是对程序员最基本的要求，这个层次就是要求程序员能够把一个最简单的程序编辑、编译、运行成功，强调的是掌握编程的环境和流程。

在这个层次上又分三个方面的要求：

（1）计算机基础知识

我见过许多程序员新手，他们都自称学过计算机基础知识。但实际情况是，学而不致用，学的概念太多，自己却理不出头绪，和实际工作对不上号。有的甚至认为自己忘了，什么都不知道。但是我要提醒新手的是，尽管基础知识十分丰富，但是有关编程的一些基础知识和概念是必须掌握的。

1）操作系统

什么是操作系统？你所编写的程序在什么操作系统上运行？目前主要有 Windows 类、Unix 类、Linux 类操作系统。每种操作系统对编程的影响是不同的。

2）计算机、内存、硬盘

这些概念对编程来说也是最基础的，例如计算机分为 PC 机、小型机、大型机。在 PC 机上编程和小型机上编程是有差别的。程序设计语言安装时也要注意内存大小和硬盘大小。

3）目录、文件

这些是最基础的概念了！一定要掌握和理解。因为你编写的程序就是一种文件，而且要放置在指定目录下。

4）程序设计语言、程序、编辑、源程序、编译、可执行程序、运行

这些概念也是最基础的。不同的程序设计语言对编程具有很大的影响。

目前主流的程序设计语言有 Java、C#、C 语言等。

（2）编程环境

作为程序员一定要知道自己的编程环境是什么：

1）了解所要使用的计算机

2）操作系统安装

3）程序设计语言安装要求环境

4）程序设计语言安装盘

5）安装程序设计语言

PC 机和小型机有很大的不同，使用 PC 机的程序员一定对 5 个部分要全部掌握，使用小型机的程序员只需要了解程序设计语言是否安装好了。

（3）编程流程

PC 机开发程序的一般流程：

运行开发环境→建立项目→建立源程序→编写源程序→编译项目（源程序）执行可执行程序→查看运行结果。

小型机开发程序的一般流程（以 C 语言为例）

编写源程序→编译→运行→查看结果。

程序员要验证自己是否掌握了这个层次，建议编写一个最简单的显示“hello world！”的程序。如果程序员能白手起家，能运行自己编写的程序并能显示“hello world!”则可以通过了。

2.语法

这是对程序员的基础要求。这个层次建立在上一层的基础上，应该掌握程序中最基本的语法、运算、基本功能。

主要掌握：进入程序参数、退出程序命令、赋值语句、条件语句、循环语句、引用、字符串操作、算术运算、文件操作等。

对于人机交互程序主要掌握：窗口、标示控件、文本框控件、命令按钮控件、列表控件、下拉框控件、表格控件等。掌握对控件属性赋值、对属性的读取、增加事件、对事件的调用等。

3. 调用

这是对程序员的一般要求。这个层次建立在上一层次的基础上，应该掌握程序中对数据库、库函数、动态链接库等外部环境的调用。相应的概念也要掌握。

另外，程序员还要具备查询语法和寻求帮助的技能。

这里还谈不上程序员水平问题，需要强调的是，以上是程序员必须掌握的，是最低的要求。由于开发环境不断在变化，程序设计语言也不断在升级，作为程序员就应该扎实地掌握这些方法，做到心中有数，以不变应万变。

1.3 语言选择与就业方向

程序设计语言与就业之间的关系现在变得非常强烈和敏感，这是我始料未及的。记得当年自己在学校里学习程序设计语言，自己从没有选择程序语言的概念。基本上是学校规定学什么，自己就学什么，从未考虑到这些学习将对今后分配工作产生的影响。这可能是由于在当时的社会环境下，不愁就业的状况所造成的。没有了就业的压力，学生可以安心学习一些基础性的课程，可以学习很多经典的程序设计语言，看到语言发展的轨迹，从而对一种程序语言打下良好基础。当然，很多学生并没有把程序语言学好的主观意识，只是想做学习的“奴隶”，而不是为自己真正掌握生存技能服务的。若跟这样的学生交流好好学习，那就是浪费时间。

而今，社会已经发生翻天覆地的变化。程序员从高高在上的“稀缺人才”地位开始下降。这期间的变化令人感慨和无奈。不得不承认，我们已经进入了市场经济社会，而且是一个充满竞争的市场经济社会。面对市场和竞争，我们必须拿出满足市场需求的商品：我们的编程能力。编程能力有两个方面的含义，一是编程语言，二是编程本身能力。所以，如何选择我们的编程语言是首要问题。而编程技术的提高则需要时间积累和项目积累。

当我们能够树立市场观念和就业意识时，我们对程序设计语言的选择就变得相对简单了。就这个问题我想提出自己的一些建议供各位参考。

1.就业方向

程序员就业范围应该很广。有的是软件工厂式的编程工作，有的是项目开发的编程工作，有的是用户单位维护类的开发工作，有的是后台编程工作，有的是前端编程工作，有的是编制网站工作，有的是软件培训工作，有的是与硬件相关的汇编级编程工作，有的是数据库类编程和管理工作。随着软件专业化分工的加

快，分工会越来越细，就业种类也会越来越多。面对各种就业种类如何选择，这是一个仁者见仁智者见智的问题。只有先定下就业方向，然后再根据就业方向所要求的必须掌握的语言来确定自己要学的语言。

例如，你想去做后台的编程工作，你就可能会选择 C 语言、数据库等。

如果你想去编写网站，你就会在 C#、Java、PHP 中进行选择。

如果你想去做底层与硬件有关的编程，你就可以学习汇编和专业的单片机语言。

2.市场状况

当你自己不知如何选择的时候，可以从“唯利是图”角度出发，看看在程序员这个行业当中，做什么职业收入最高。你可以通过百度搜索的方式反复比较，获知收入相对较高的职业，然后根据这个职业要求选择所需的编程语言。

你当然可以从“技术至上”的角度出发，你可以通过百度搜索的方式了解当前哪种语言最流行，因为流行的程序往往体现出这种语言未来可持续发展，当你选择了流行的语言的时候，这就意味着软件市场对这种语言的程序员需求是比较大的，因此，就业的概率相对较高。但是，正是需求增加，在供过于求的情况下，竞争就会加剧，如果你没有表现出更多的能力的话，就很难在竞争中胜出。

当然，你也可以从“反向思维”选择相对冷门的职业，由于是相对冷门的职业，竞争就会有所减弱，就业的概率同样会大大增加。

3.社会资源

就已经参加工作的人来说，就业还取决于个人的社会资源，假如你有足够的社会资源，你就可能想到哪里就到哪里。这种个别的现象不在我讨论的范围之内。其他的个人的社会资源如学校、家庭、朋友、朋友的朋友，其他求职中介和求职渠道，都是你求职的重要因素。例如，你的朋友正好在一家软件公司，而这家软件公司正好要招聘你这种程序员，求职往往会很快成功。因此，将目光放在你的社会资源上，看看他们能够触及的单位，然后去了解这些单位对程序设计语言的要求，再进行语言的准备，我感到这样做还是非常有必要的。

4.个人基础

当然，你通过各方面考量最终选择了就业方向，也就确定了你的程序设计语言。你可能在学校学过这个语言，也可能没有学过。对于你来说都要有一个重新学习的阶段。这个学习阶段和大学里无忧无虑的学习有

着本质区别，这可是关系到你能否被用人单位录用，关系到你今后的生存，关系到你今后的职业生涯规划。因此，你不但要把用人单位所需要的程序设计语言学好，而且要把相关的其他语言学好，以及相关的计算机基础知识学好。这样才能把你的个人编程基础夯实。可以说基础是必需的。

另外要有的放矢，了解用人单位所采用的语言、开发项目等情况，只有这样，自己在应聘的时候才会更有把握。

我们可能因为语言而获得就业，我们也可能因为语言失去工作。关键是我们每时每刻要注重语言的发展趋势，注重用人单位的语言发展趋势。语言的学习可以伴随程序员终身。从我的经验来看，要想进入程序员这个行业，主流的程序语言都是应该了解和掌握的，至于掌握的深度可以根据就业的要求深浅不一。因为语言的掌握是无止境的，人们不可能为掌握而花费太多的时间和精力。就目前而言，学习.NET、Java，学习任何可用于网络应用软件开发的程序语言都是非常有前途的，因为基于网络、互联网、手机（包括移动设备）的软件将是未来软件的主流。

现在我们因为语言而生存，未来能否让语言因为我们而骄傲？

1.4 新手如何学习一门新的语言

学习程序设计语言是程序员的看家功夫。许多程序员边编边学，没有止境，可以说是“活到老学到老”。在语言面前，程序员永远是学生。然而，程序设计语言是一项不断发展的技术，从机器语言到汇编语言，从低级语言到高级语言，从面向过程语言到面向对象语言，从 C/S 语言到 B/S 语言，从非跨平台语言到跨平台跨系统语言，从纯语言到开发平台工具，真的是目不暇接。这些革命性的技术浪潮推动着程序员不断学习新的语言以适应其发展，否则程序员将被淘汰。

学习新的语言有两类人员，一类是从来没有程序设计语言基础的、没有编过程序的人；另一类是已经掌握了一门或一门以上语言，正想要学习更高版本语言或新的语言的人。对于第一类的人我且称为新手。

新手要学习一门程序语言，若是将新手比作一张白纸，可以画出最新最美的图画，同样也可以画得乱七八糟没有美感。新手的第一门语言对其今后的影响是巨大的，如果学得好，则自信心大增，对今后的发展非常有利；如果学得不好，学不下去，则对其职业生涯打击很大。新手面对众多的语言往往无从下手，第一，不知道学哪种语言为好；第二，不知道如何学习；第三，不知道能不能学好！

我认为，就目前而言，先学习 Java、C#，今后再想学习应用于后台的 C 语言都是不错的选择。

针对第三点，我认为只要想学，有职业压力，没有学不好的，只有学不到最好的。

关于第二点，我的建议应特别注重两个要点：一是基础，二是动手能力。学语言首先要看书，然后要动手。那些认为光看书就能学会语言的想法是很幼稚的，而那种光注重编程而不注重读书的人，将来一定是动手能力强而编程水平低的。

1）新手不要急于求成，要把时间放长一点，先把基础知识学好，基础越扎实，今后编程水平就越有可能提高。看书至少要花三个月时间。

主要选择计算机原理、程序设计原理以及所学语言相关（最好是入门类，不要刻意地选择哪本，对初学者来说，任意一本都是好的）的三类书籍。

2）由于看书过程中肯定会遇到很多不懂的概念，而且各种概念之间的关系也不容易理解和掌握，所以许多新手望而生畏，坚持不下去。这个时候一定要学会坚持，坚持读下去，反复读下去，对实在不懂的概念要注意收集，将其牢记在心。这个时候最好不要找人去解答，而是把所有的书籍反复看完 3 遍之后，再去找人解答，这样就可以加深对这个问题的理解，而且解答者也愿意回答。如果一有问题就去问，一是解答者容易失去耐心，二是自己对问题没有印象，容易产生依赖性。

3）有了一定基础知识之后，就要自己想办法安装编程环境。安装编程环境的要点可参照 1.2 节中有关“流程”的阐述，这里就不再赘述。

4）开发环境安装完成后，新手就可以编写显示“hello world！”程序了。

编写这个程序目的主要是学习主程序的作用、主程序的参数入口、简单的赋值语句、显示功能调用和退出程序语句。通过完成这个程序，新手就可以对编写程序流程有一个切身体会。

5）接下来可以学习算术运算编程。试一试算术表达式编写，可以简单地编写一个计算器示例；也可以编写一个显示日历的复杂程序（若输入年份，则显示这个年份的日历）。在这个阶段主要是学习函数以及函数的调用、算术运算、条件语句、循环语句、显示功能等，这些都是编程的基础。

6）完成算术运算的学习后，可以编写更复杂的完整的程序了。例如可以编写一个学生信息管理程序。其功能为：接受一个学生信息（例如，学号、学生姓名、班级、年龄）并把它保存在计算机中，同时提供增加、删除、修改、查询功能。信息保存形式可以是文本文件，也可以是数据库。

这个阶段主要是学习变量、数据存放、文件操作、数据库操作、程序完整性等，这也是编程的基础。

7）完成上述学习之后，程序员要学会回头梳理自己编写的程序，梳理自己已经学过的概念。可以对自己以前的程序进行修改，培养不断提高自己编程水平的意识。

8）在这些过程中若遇到问题，先看帮助。帮助不行，最好是找懂行的人询问，不要自己钻牛角尖，浪费时间。上网查询也可以，但是不如询问别人直接。看书是不能解决问题的，切记！

9）在这些都完成之后，程序员可以有目的地针对自己将要开发的内容进行相应的技术学习和准备了。

10）编写程序的时间估计需要 2~3 个月。一般而言，一个新手学习一门语言需要半年左右的时间。有的可能要少一点，有的可能会多一点。但是，无论如何，最终的结果是程序员对语言有了初步的了解，可以用语言编写简单的程序了。

1.5 理性看待考证热

在当今这个社会，经常会出现各种所谓的热门现象。这说明目前的社会相对浮躁，而 IT 行业也同样不能避免。IT 证书可谓名目繁多，应该有几十种。IT 证书（认证证书）大体可分两类：一类是各大企业的认证，如微软、IBM、SUN、CISCO 等认证考试，这类考试主要是考核考生的专业技能和特殊技术的水平；另一类是国家认可的考试认证：如全国计算机等级考试（NCRE）和全国计算机应用技术等级考试（NIT），考核的是考生综合的或某一领域的标准要求达到的程度。其实几乎所有的程序员都知道，程序员的能力是一种综合能力，其动手能力的成效是其能力的主要方面。若认为单靠考试就可以反映出考生的实际能力，那简直就是一个笑话。

证书的出现说明社会对任职资格的追求，反映出社会对无序现象的一种规范要求，反映出对“持证上岗”的认同。社会资格的本质是对人们之间竞争的一种限制。为了突破这种限制，获得在竞争中的优势，人们便千方百计地获取这种社会资格。于是社会上便出现了三类群体，一类是发放证书的，一类是获取证书的，一类是招聘中认可证书的。由于这三类群体都能从中获得直接的或间接的证书经济利益，于是，证书不火也难。当证书火到一定程度的时候，其原本内在的职能就慢慢地被异化了。人们不是为证明证书持有人的能力而设立考试，而是为了获取考试带来的经济利益而设立各种证书，于是证书的价值就贬值了。

但是，我们很多考生并没有深入地看到证书背后真正的内容，他们认为用人单位需要证书，没有证书就不能被录取。社会上大量流传着未经证实的证书和高薪之间的关系，更使某些证书神乎其神。因此，他们在证书方面花了大量的时间和金钱，为了证书而证书，能考多少证书就考多少证书。为的是在应聘时递交的简历上可以附上厚厚的各种证书。

就我本人而言，我好像没有考过什么证书，工作依然很顺利。这虽然和我成长所处的年代有关，但更重要的是我可能更关注自己能力的提高。如果自己的能力水平能够达到用人单位满意的要求，我相信即使你没有那些证书，用人单位也会考虑录取你的。我也相信有些用人单位的人力资源部门会按规定办事，没有证

书不予录取状况也是存在的。这个时候，你自己就要权衡了，是进入按能力看人的企业求得发展呢，还是进入凭证书看人的企业求得发展呢？

我对证书方面的建议：

1）如果不占用自己大量的时间和金钱，能考多少证书就考多少证书。把考证书当做消遣和对知识的复习。

2）如果需要对国家证书和企业证书进行选择时，在没有针对性的情况下，选择国家证书。

3）如果自己对某类公司或某类岗位有意向的时候，最好打听其是否有刚性的证书要求，若有，则要不惜时间和金钱去获得这个证书。若没有刚性的要求，仅仅是一种参考，那就要根据个人情况，例如时间和金钱方面的条件，进行取舍。

4）在已经有足够证书的情况下，不要见一个证书就去考一个证书。尽量把时间放在自己的学习上和能力提升上。把时间和金钱节约下来做更重要的事。

5）不要和别人攀比证书多与少。因为每个人就业方向和机会都是不相同的。

6）没事的时候，可花点时间跟踪流行证书的情况，以便自己及时掌握证书最新情况，早作打算。

7）工作之后，也有可能因为你考了相关证书，可能会对你的薪水、岗位产生有利的影响；也可能对你跳槽有所帮助。这些在现实生活中也是可能出现的情况。

总之，我们要理性地对待考证热，我们不赞成为考证而考证，我们也不赞成能考证而不考证。我们赞成那种实用主义的态度：当我们刚性地需要考证的时候，我们就准备考试，争取获得证书。当证书仅作参考的时候，我们能考就考，不考则把时间和精力花在自己的能力提升上面。

在国内，证书的泛滥早已使证书失去了原有的价值。但是，我们应学会内外兼修，也不妨在提升自己内在能力的同时学会打扮自己，让自己更加漂亮一些，让别人更欣赏自己。

1.6 选择大公司还是小公司

很多人在第一次求职的时候几乎都会遇到“到大公司还是到小公司”的问题。他们认为大公司的薪水高、工作稳定、技术水平高、升迁机会多，但是，大公司要求高，竞争激烈，自己怕进不了；而小公司薪水相对低一些，工作稳定性较差，技术水平参差不齐，升迁机会不多，但是，录取率相对较高。这可能是各种

求职中的一个常见话题。这个选择应该是因人而异，几乎没有什么正确答案。所以，我们看到过选择大公司的人获得了成功，也看到过选择小公司的人获得了成功，相反的情况我们同样也见到过。

作为程序员的求职，我想还是具有职业特殊性的。程序员职业和一般的职业有很大的不同，程序员职业有其鲜明的特点：1）个人劳动；2）产品可复制。这两个特点注定了程序员有很大个人发展的空间。

1.个人劳动

表明其工作主要和劳动者自身有关。无论是程序员独自承担一个编程项目，还是在一个项目中承担部分编程工作，都可以归结为个人劳动。不像有的职业需要昂贵的劳动设备，依赖昂贵的设备，例如：炼钢工人必须要有炼钢厂的炼钢炉；也不像有的职业需要其他人联合劳动，例如，流水线上的装配工；也不像其他职业有严格操作规范和工作流程，例如，制药厂必须按照制药的配方生产药品。

程序员只要有台计算机（其价格可以忽略不计）就可以工作了，而且在工作时是一个人在工作（工作前后以及项目的衔接可以忽略不计）。其编程过程完全取决于程序员个人的技术水平发挥。

2.产品可复制

绝大部分企业生产出的是实物产品，每个产品都需要投入一定的原材料，价值很大限度上与其原材料的成本有关。劳动者生产一件产品，其最大的价值就限制在这个产品的售价之内了。

而程序员生产的产品是软件，软件最大的特点是可复制性，而且可复制得不计其数。因此，程序员生产软件的价值就是软件拷贝数。当拷贝数不断增加的时候，程序员生产的软件价值就在不断增大。一个软件最大的价值等于其单价乘上可能的用户数。如果这个用户数是成千上万的时候，程序员的一个劳动价值可达到一个天文数字。所以，世界上成功的软件企业，正是这种可复制性的受益者。

程序员的这两个职业特点表明程序员的发展空间很大。程序员不管是在大公司还是在小公司都能获得成功。只要这个公司能生产和销售出可复制的软件，程序员都能够获得比职业高得多的收益。关键是程序员是否具备编制这些软件的技术，这些软件能否被大量复制（定制软件价格高也行）。

比尔·盖茨正是成功通过编制软件而将微软发展成为一个软件帝国的。在现实生活中，也有一些成功的程序员自己当上了软件公司的老板。不成功的也有很多，但是，其中的原因不在此，我会在后面的内容中加以分析。

通过以上分析，当程序员面对“选择大公司还是小公司”的时候，我建议：

1）在程序员趋向于安稳工作的情况下，而且具有大公司所要求的学历、经验、证书、能力的时候，应该首选大公司。这些大公司指的是国内外著名企业，有的甚至是地区内著名企业。进入大公司后，可以保证收入高，岗位稳定。

2）在程序员技术水平一般的情况下，想进大公司也可能是一个梦想，那只好选择小公司了。小公司的好处是起点低、机会多，缺点是收入低、开发不规范。

3）对于有理想和抱负的程序员，我建议先进小公司，然后再进大公司。小公司专业化分工比较粗糙，有的甚至一个人就会负责一个项目，对人的锻炼机会很多，程序员既可以学到编程，又可以学到设计和项目管理，往往会成为一个“全能型”的程序员，这对程序员以后的发展有很大的好处。但是，程序员在完成编程积累之后，应该转向到大公司发展，学习大公司的软件开发流程、团队意识、大项目的开发经验、规范和管理、企业间的合作以及技术交流和运用等。

4）我们知道软件能否复用是软件小公司发展的关键。因此，我们在面对软件小公司的招聘的时候，我们应当对公司在软件行业应用的范围，以及软件的客户数有所了解，以判断小公司的成长性。当小公司成长性很高的时候，你的选择应该是正确的，当小公司成长性很低的时候，你若选择，就有可能不正确。

5）那些缺乏远大理想的程序员，只是把软件当做普通工作，求得平均工资和正常收入的程序员，最好能找到一些工作较为稳定的部门，以保证有一个稳定收入。这些人即使进入大公司、小公司，都可能面临解雇的危险。最好的情况就是处在一个岗位多年不动，工资多年不涨的局面。

对于有理想的程序员来说，无论进入大公司还是小公司，都是一个积累过程，都是一个锻炼自己的天地。对于普通的程序员来说，进入大公司则实现梦想更容易些，进入小公司则可能存在诸多困难。只是前者取决于自己，而后者取决于别人。而关注公司成长性则是两者都要重视的，毕竟这和自己的当前和未来收入密切相关。

总之，我想说程序员是一个很特殊的职业，它给每个程序员造就了一飞冲天的可能。关键是程序员自己是否有这个想法，能不能在进入各种公司之后，积累和提高自己的技术水平，为一飞冲天做好准备。

1.7 新手面试常见问题和对策

招聘面试是一个用人单位（面试官）与招聘者之间的博弈，不同的应聘者、不同的用人单位、不同的面试官会产生不同的面试过程和不同的面试结果。因此，如何应对面试是很难有什么正确答案的。但是，大量面试沉淀下来的各种成功的经验和失败的教训却是我们可以参考的。

在日常生活当中，常常有朋友和同事找到我，向我询问他们学计算机专业的孩子在面试时要注意些什么。也有一些软件公司因为和我比较熟悉，请我在业余的时候为他们招聘人员把把关，提提意见。在单位的时候，有时候也会带上实习生，也会面临培养新员工的问题。因此，我见到新手的机会相对比较多一些，加之自己也是从新手走过来的，所以对新手相对了解一些。可以说我身兼了被招聘者和招聘者两种角色。客观了解和分析这两者的面试心理可以让我们面试时心态更从容和平和一点。

作为新手，主要是一些刚毕业的大学生，他们很少有编程经历，很少有人能把学的东西与现实对上号。但是他们求职心切，急需通过录用来证明自己的社会价值，证明自己的独立生活能力，因此，他们在应聘中一直处于弱势地位。

作为面试官，肩负着企业招聘员工的重任，知道企业招人的急迫性和招人的标准，当应聘者众多的时候，其招聘要求应更加严格；当应聘者不多的时候，要求就可能放宽。他们拥有招与不招的大权。在招聘中处于强势地位。

新手在面试时常见有以下几个问题：

1.简历灌水

新手在求职时，往往会投寄和递交简历。这些简历对用人单位了解应聘者的意义重大，这是用人单位对其第一印象。现实中有很多简历言过其实，例如：“精通 C#、Java 语言”，其实远远达不到精通的水平。“开发过某某项目”，实际上仅仅是参与过这个项目，在项目中作用微乎其微。这样言过其实的部分往往会在“掌握语言”、“编程水平”、“英语水平”、“项目能力”、“团队精神”、“学习经历”、“学习成绩”有所体现。有的人明明没有学过 C#，但是他敢于写上自己精通 C#。

灌水的直接结果就是应聘者在面试时，显得很尴尬，显得很不诚实。例如用人单位急聘 C#程序员，看到简历上精通 C#之后，必然会在面试时问及 C#编程情况，在某些情况下，还会出一些 C#编程试题。如果这个新手并不精通或根本不会 C#，其结果就可想而知了。

很多新手认为，你不写夸大一点，你可能连面试的机会都得不到。但是，到了面试，一切都会暴露的。我认为，在简历上诚实地写清自己的各种情况和水平，这可能更容易得到用人单位的认可。如果自己掌握的东西不够的话，还是应该把应该掌握的东西在应聘之前多掌握一些，把真实的自己展示在用人单位面前。

大家知道，在招聘程序员的时候，面试官中一定会有一个懂程序的高手（用人单位的高手），而这些人自信心很强，具有好斗的潜意识。你说强，他就要让你把强说出来，好让他表现自己。所以这个职业特点注定你不能太夸大自己。

2.恐惧心理

由于新手处于弱势地位，其恐惧心理很严重，有的大学生社会实践比较少，连起码的见面礼节和打招呼都忘了。有的人在介绍自己的时候话不成句，声音极小，连面试官都听不清楚，尤其是被问到程序方面问题的时候，更是紧张得词不达意。几乎所有新人都有恐惧心理，只是恐惧程度有高有低罢了。恐惧心理使应聘者留给面试官的印象不好，他会认为你社会经验比较少，沟通力差，还是一个学生。

可以通过时间和经历来克服恐惧心理，新手平时要注意这方面的改进，讲话声音尽量大一些，说话要尽可能多，话与话之间尽量连贯，在说话时，眼光要尽量看着听者。新手可以找一些同学、老师、家人当做面试官练练兵，一次不行二次，二次不行三次，多练几次就会好得多。

3.表现欲强

程序员中也有一些表现欲强的人，这类人动手能力相对强一些，对新技术有追逐的爱好，有的也做过一些项目，自己也赚过一些钱，有的认为自己已经会编程了，有的认为自己已经是高手了，他们在面试时表现出了那种过于自信的谈吐。有的甚至会问面试官知道不知道某个东西，这往往让面试官感到不快。因为在企业工作的程序员或项目经理都知道技术水平的提高是没有止境的，否则，他们也不会来招聘新人。他们既要看到应聘者的能力，又要看到应聘者表现能力的方式。他们认为，那些过高看待自己的人往往在团队合作上容易出现问题。而且夸夸其谈的人往往和那些没有真实能力的人画上等号。

面试时，缺乏能力的时候，要表现出能力；缺乏风度的时候，要表现出风度，应聘者，切记不要班门弄斧，弄斧可能伤到自己的脚，如果想要表现自己的话，可以在同学、家人、陌生人面前尽情的表现，虽然这些表现可能不会影响你的应聘。

不过，有的企业不把表现欲很强的人录用为程序员，而是把其录用为软件销售人员，这种情况在现实生活中也真的出现过。

4.准备不足

很多企业和单位在招聘的时候对招聘岗位和人员要求都是很明确的。但是，有的应聘者往往对此没有针对性、重点性的准备。例如，某用人单位要招聘有金融软件开发经验、熟练使用 C#语言的程序员，很显然用人单位是要做金融方面程序的，而且缺少 C#程序员。应聘者应该针对金融方面的知识和 C#语言两大方面多做准备，如果你有金融软件开发经历，那就要在面试时突出这个方面的内容，如果没有，你最好多准备金融方面的知识，谈谈对金融业务的理解和自己的看法，表明你对金融软件的了解程度。同理，在 C#方面，

也要针对 C#在金融软件方面常用到的技术做些准备。这样有的放矢地回答面试官的问题，总比摆出一副叫我做什么我就做什么的架势要好得多。

许多新手对用人单位的性质、员工人数、工资状况、同业中排名、产品方向、单位特点、发展趋势、可能笔试、面试的内容都无准备，便匆匆应聘，其结果是可想而知的。机会总是留给那些有准备的人。

5.性格内向

由于编程是一种个人劳动，很多学程序的学生很自然地沉浸于个人世界里，与外界交往和交流并不主动和积极。表现在与人打交道上很内向，少言寡语，给人看上去很“老实”的感觉。这些人往往对自己很自信，却又埋怨别人不理解自己的能力。在面试时，往往和面试官形成一问一答那种很机械的场景，严重缺乏主动性，从而给面试官留下一种很不自信的印象。除了女人找对象要找“老实人”之外，很少有软件企业要招那种性格极端内向的程序员的。

在当今项目规模越来越大，项目内部越来越需要协调、交流和合作，软件开发更需要团队精神来支持。无论是交流意识、交流方式都是优秀程序员必须具备的。所以在面试的时候，要避免一问一答，要主动地在重点问题上谈谈自己的看法和想法，主动和面试官进行沟通，这样面试官自然会对你另眼相看，我们不排除的用人单位喜欢雇用性格内向的程序员，但那毕竟是少数。

6.逻辑不清

程序员最重要特征之一就是逻辑性，凡事要讲逻辑、讲条理，有条理才能成为程序。但是，面试时也常常遇到一些应聘者答非所问。如果你要问他多高，他绝不会回答他 1 米 70，而是可能回答“今天上午我才赶来面试的”。你要问他常用的排序方法有几个，他绝不会回答有 3 个，第 1 个是什么，第 2 个是什么，第 3 个是什么，而是回答让你不知道他在说些什么，也许其中有那么个排序的名词。对于那些没有什么正确答案的问题，那更是能回答得云里雾里，听不出其中的头绪。可以说这种人说话和思维缺乏逻辑，能把一个简单事情复杂化，能把一个真实事情虚无化。

这种人在理解别人的话的能力方面，在处理问题的能力方面，在逻辑方面都很欠缺。如果不加以注意和改进，那么进入了程序员这个行业后，自己痛苦，同时别人也会跟着痛苦。了解这些后，应聘者在面试时，一定要集中精力，要充分理解面试官提问的真正意图，回答简明而有条理，不懂也不要胡乱说。只有这样面试官才会认可你是做程序员的材料。

7.潜力不明

用人单位招收程序员一般有两种类型，一类是招来即用的，一类是培训后使用的。对于前者，用人单位招的是和自己项目开发最接近的程序员，比如之前做过此类项目最好（挖同行的人员），招人的标准很明确，能干则要，不能干就不要。针对第二类，主要是看应聘者的各方面素质，看看是否通过培训后能逐步成为单位的有用之才，所以标准比较宽泛，无论是应聘者还是招聘者都有很大的选择范围。很多应聘者并不了解这些，在面试的时候，没有主动地表现自己各个方面的素质，表明自己是一个有素质和潜质的员工，以获得面试官的认可。

有一点要特别说明，应聘者千万不要过度地表现自己的学习欲望，说一些“到单位之后，我会好好学习的”之类的话，认为用人单位是一个学校，只要当一个好学生就可以了。用人单位要的不是一个学生，不会提供一个免费培训基地，而是让你为它创造财富的。面试官特别不喜欢那些已经走出校门而思想还没有走出校门的应聘者。你不如说“我到了单位后，将多做工作，为单位创造财富贡献自己微薄之力”之类的话，这反而更能让面试官认同。

应聘者的语音语调、着装打扮、递交材料方式、等候面试、介绍自己的分寸、对用人单位的关注度、对自己未来的企盼、对自己可成长性阐述、对自己不懂问题的回答等面试中的细节，都可能是面试官对你产生印象的一个因素。这些因素的综合形成了面试官对你的素质的判断。

面试官可以理解你现在编程技术达不到所要求的水平，但是不能容忍今后你达不到所要求的水平。关键是让面试官看到你的潜质。

总之，新手在面试时，要准备充分，要端正自己的心态，即不要过分地表现自己，也不要恐惧犹豫，说话要条理清楚，注意面试中的各个细节，尽可能地向面试官展示自己良好的职业素质，展示自己未来的发展潜力，这样面试官就可能对你有一个良好的印象，有助于获得这个就业机会。

面试只是应聘者素质的集中反映。素质的形成可以是在面试之前，也可以是在面试之后。只要我们注重培养和提高自己的素质，我们就可以在任何时候面对任何的面试。

1.8 薪水的苦恼

上班了！拿工资了!从学生转变成社会的劳动者，这是人生阶段的一个重要转折标志。当拿到自己的薪水之后，是喜悦、是满足、是苦恼、是无奈，每个人的感觉都是不一样的，我记得自己第一次领到上班的工资不到 10 元钱，兴奋异常，忙着请客庆贺。而今一个新手拿着 2000 多元的薪水估计也不会欣喜到哪里去。

程序员注重自己的收入与其他所有劳动者一样，注重自己的收入是天经地义的事情。只要是为了谋生而工作的（为了其他目的，如爱好、兴趣、自我实现等而工作的，不在我们谈论之列），都会注重自己的收入。几乎没有一个人会认为自己的收入高，所有的人都希望自己的收入越高越好。这个原因主要是人对金钱的追求是无止境的。

现实中程序员对自己的薪水还是有很多苦恼的。

1.从宏观上看收入

从宏观上看，由于程序员所处的行业或企业不同，其收入的差距确实存在。所以，当程序员看到在其他行业工作的同学所拿到的工资时，就会感叹自己拿的工资太少。我初步估算了一下，新进软件公司的大学生的工资收入一般是当地最低工资的 2 倍左右。

2.从企业内部看收入

在一个企业内部，由于各个部门或职位分工不同，其工资也存在较大差异，而且在不同的行业中这种差距也比较明显。

作为软件公司的程序员一般是公司收入最低的阶层，而作为用户单位的程序员的收入一般处于单位平均工资的中等水平。总之程序员的平均收入水平是不高的。而程序员本身也因工龄、技术能力、项目的不同而收入不同，有的程序员升任到了设计师、项目经理，其收入也能是普通程序员的 5~10 倍。这种眼前工资的差异也会让新进公司的程序员心中有所不平，尤其是看到和自己一样工作，甚至能力不如自己的同事，拿着比自己高的工资，这种心态尤其强烈。

3.从话语权看收入

程序员除了感到收入低之外，还会对自己的薪水制定没有话语权感到沮丧，而且也会为奖金有无，分配问题激动，还会为收入不能满足各种生活支出而感到无奈。大凡与收入挂钩的事都会让人苦恼不已。

现实，现实，还是现实。程序员不但要学习技术，还要学习适应社会，增加自己的社会知识和经验。我们无法掌控薪水的多少，薪水不是靠我们想出来的，就像发财不是靠做梦就能实现的道理一样。所以我们要学会“自己不能左右的事不要多想”，多做些自己能把控的事，例如，有想象的时间不如把自己的工作做好，把自己的学习忙好，把自己周围的人际关系搞好。

从另一个角度来看，新进公司的程序员至少在 3 个月到半年时间内还是处于学习和适应阶段，还没有能力为公司创造利润。从市场经济的“等价交换”原则来看，程序员此时对公司来说是负效益的。而那种只要上班就必须给我工资的想法是一种很天真的想法。在这段时期，如果程序员能想通这个道理，就不会为薪水烦恼了。

建议程序员换一种阿 Q 的心态，毕竟自己要比那些没找到工作的强呀。

有了好的心态，我们可能更加现实，可能更加有利于我们在这个社会的成长。今天的薪水也许很低，但是必须承认我们的水平也很低；我们眼前虽然有收入比我们高得多的程序员，但是总有一天我们也会达到他们的收入水平。如果我们能认识到这点，抓住一切时间去学习、工作、提高自己的工作能力和技术能力，提高自己的社会生存能力，我们可能就会缩短自己的成熟时间，我们也可能缩短新手的低薪水的时间，而增加自己的高薪水时间。

程序员的能力是最重要的，而能力需要得到程序员的理想支持。在程序员这个职业中，心有多高，薪水就会有多高。

1.9 求书、求网还是求人

我们已经知道了新手如何学习一门新的语言，那么对于已经掌握一种或一种以上的程序员如何学习一门新的语言呢？由于程序员已经有一定语言基础，形成了自己的一套学习方法和思维定式，所以这个问题还是和新手学语言有些差别的。

在现实生活中，我发现不少程序员因为工作的需要而学习新的编程语言。他们心怀恐惧，不知道自己能否学好，而且学习方法非常传统，像新手一样：首先去买本这方面的书（或者借本书，或者网上阅读），看完以后再动手。但是，我不知道他们是怎么看书的，是从头到尾地看？还是挑选着看？他们关注书中的什么内容？第二种情况是程序员动手能力特别强，先把语言安装起来，先编起来再说，但是他们不知道这个过程有多长，何时是终点。无论哪一类程序员，当你向他讨教如何学习一门新语言时，大都是仁者见仁，智者见智，而且几乎没有人能从头到尾说清楚。若不信，读者可以自己回答这个问题。

其实，程序员学新语言是有讲究的。学习与自己现在使用的不同类型的语言难度最大，相同类型的语言难度就小些。当掌握过程语言（如 C 语言）的人去新学面向对象的语言（如 C++）时，难度就非常大，因为这两种语言的思想是完全不同的，用过去的定势去思考新的语言，那种痛苦是难以言表的。例如 VB 程序员去学 VFP、PB 等同是 C/S 类开发程序相对就容易得多。又例如，VB 程序员去学 ASP 就很困难，因为一个是 C/S，一个是 B/S，虽然语法上相差不大，但是架构差距很大，很难马上适应。所以，学习新的语言，第一

个要分析这个语言的类型自己是否已经掌握，如果已经掌握，那么学习的时间会很短，一般在一个星期到半个月就行了；如果没有学过，则学习时间会很长，一般要 2~3 个月或者更长，而且非常痛苦。

对程序员而言，学习新的语言，第一，建议不买什么资料，买了资料也不要花时间一字一句地去看。主要的学习手段就是动手编程序，通过在编写范例程序的过程中学习！第二，一定要找到一个懂这种语言的人，如果能够找到人问的话，一定要找人去问。这个时候和新手找人问是完全不同的，因为新手会听不懂别人所说的一些基本概念，这会引起高手的不耐烦。第三，程序员千万不要自己去“刻苦钻研”，因为，程序员有一定的语言基础，心中只要记住“流程”、“语法”、“调用”（见 1.2 节）就可以了。

程序员在掌握编程的流程之后，可以通过编写以下三个程序：

1）显示“hello world！”程序。

2）打印日历程序。

3）“学生学籍信息”处理程序。

来掌握新语言中的语法和调用方法。当然程序员自己也可以选择编写自己的程序以达到学习目的。

与新手不同的是：

1）要特别注意语言的语法差别。一般来说，语言的语法差别很小，但是很令人讨厌。程序员学过之后往往会混淆不同语言的语法。例如，有的语句是以分号为结束符的，有的语言中的语句是以回车符为结束符的。如果混淆了，你就会一会儿加分号，一会儿不加分号的。尤其当程序员对原先语言特别精通时，这种操作惯性是很大的，因此也特别烦人。

2）要特别注重程序的调试方法。因为除语言本身之外，如何调试也可能不太相同。调试对于程序员来说太重要了，必须加以关注。

3）要特别注重程序运行环境。程序编出来了，生成 exe 可执行文件了，但这并不意味着程序就能执行了。例如，用 C 语言编写的程序一般是不需要额外环境的，直接运行就可以了；用 VFP 编写的程序，则需要系统中安装 VFP 系统 DLL，因此，VFP 程序要做安装盘；C#则需要更多的环境才能执行等。

4）要特别注重程序中的各种调用。由于新学的语言往往功能更强，涉及调用功能更多，程序员要关注新语言如何调用的问题，调用也是今后程序编写的最重要的内容。

程序员按照这种方法去学习新的语言，刚开始一定会感到寸步难行，可能会遇到很多困难。但是，我相信只要程序员知道整个学习的流程，知道下一步工作的目标，他的进步会越来越快。我最不希望看到程序员在学习上表现出自发、盲目、恐惧的精神状态。

1.10 新手看高手

入门之前，很多程序员心里有一个高手情结。通过书籍、媒体、传说渲染，他们认为 IT 行业是一个高手林立的行业，好像这些高手创造了这个行业的奇迹。这些高手可能是国外的，也可能是中国的。这个高手可能是一个具体的人，也可能抽象于某些著名软件背后看不见的程序员。只知其名，不闻其声，这个时候的高手是一种无所不能的神，一种虚幻，是令程序员崇拜的偶像。

到了工作岗位之后，这种高手情结更加严重，由于新手发现自身技术水平有限，而内心想尽快摆脱这种状况，使得很多新手对高手感觉更加恐惧和渴望。我发现在 IT 行业中，中国程序员认为外国的程序员是高手；网下的程序员认为网上的是高手；搞硬件的人认为搞软件的是高手，搞软件的认为搞硬件是高手；搞应用程序的认为搞系统的是高手；搞界面设计的认为程序员是高手，程序员认为搞界面设计的是高手；年轻的人认为年长的是高手，年长的认为年轻的是高手；企业内的程序员认为企业的外部程序员是高手；客户单位内部的程序员认为软件公司是高手，软件公司的程序员认为客户单位内部的程序员是高手等。所有这些高手都是建立在程序员本身对某个领域无知或掌握不精的基础上。因此，高手的多少是和程序员的水平成反比的，如果程序员水平高，那他眼中的高手就会少；如果水平低，那他的眼中的高手就会多。所以，不同程序员对待高手的标准应该是不同的。

高手情结是新手的正常心理现象，它反映了程序员对未知领域的向往，说明程序员还有远大的理想，还有激情，还有奋斗的目标。关于高手，我想新手应该关注以下几个方面。

1.向身边的高手学习

身边的高手，一般是新手的第一个师傅，也就是第一个培养他的那个人。这个人对新手来说是一个决定其技术命运的人。程序员很多人可以忘记，但是第一份工作，第一次带他的人是很难忘记的。假定这个师傅带他的时间有一年以上，我想从这个程序员身上一定能够看到其师傅的某些影子。比如说，这个师傅喜欢的语言，这个师傅喜欢的编程格式等，都能在新手身上找到。

作为新手来说，除了要向他的师傅学习之外，也要把身边的同事看做高手，要向他们学习，只要单位或团队范围内有比自己强的，不管这个人是早于自己进入这个单位一天，哪怕只是一个函数、一个语句、一个调用、一个算法都值得去讨教和学习。这个时候学习是很实在的，是能解决具体问题的。

向身边的高手学习，就是要充分利用身边的优势，可以面对面地接受别人的技术传授。新手千万不要顾及面子，或不好意思，或怕麻烦别人，能请教的就赶快请教，学习是硬道理。向别人请教不仅仅是去弄懂某个问题或解决某个问题，更重要的是培养与人沟通和建立良好人际关系的习惯。

新手千万不要有那种有人会主动帮助自己的幻想，一方面是大家工作都很忙，另一方面很多程序员不习惯于主动帮助新手。

要注意身边的高手并不是所有问题都能解答的，遇到这种情况，新手不要吃惊，很多“伪高手”就是在这个时候被揭开其伪装的。对于他们，新手要怀有宽容的心态，应坦然面对，不要深究，这个人不能解答，那就去请教另一个人。新手要记住，你请教的不是高手，而是解决问题。因此，向其请教的人是否是高手并不重要。

新手千万不要太迷信不可触及的高手，因为迷信了也没有用，而且也不会对你的实际工作有什么帮助，而应该把对那些高手崇拜的时间和精力用在身边的工作上，这可能更有成效。只有当自己有了一定进步之后，确定了自己发展方向之后，我们才有本钱和资格向外部高手学习。

2.学习高手的技术

（1）软件制作流程

我记得很多新手并不会安装操作系统、不会安装开发环境、不会配置网络、不会新建项目、不会新建程序、不会编写程序、不会运行程序、不会调试程序、不会制作安装盘等。即使会也只是知其然，而不知其所以然。而一般的程序员对此驾轻就熟，而且没有太多变化，相对机械，新手死记硬背就行了。因此，新手首先要把编程的流程搞清楚。把流程搞清楚之后，我们就可以把精力放在编写程序上了。

（2）语言的掌握

在程序的编写方面，新手常常遇到的就是语法问题，如语句使用不当、变量问题、格式问题、备注问题、命名问题、函数调用问题、参数问题等，这些问题就不那么简单了，不同的程序员会给出不同的解决方案。当解决这些问题之后，只能说是新手刚刚学会编程的工具而已，接下来就要学习如何运用这些工具来开发项目了。

（3）开发项目

这个时候新手就会遇到需求问题、功能问题、处理流程问题、数据结构问题、算法问题、可靠性问题、边界处理问题等，针对这些问题，不同的程序员更是有不同的观点和看法，所以，新手要多听不同的解决方案，通过比较来加深对这些问题的认识。

（4）高级技术

以上就是满足开发的基本要求了，但是，能运行的程序并不是一个好程序，只是站在一个项目的角度上来看技术，也不是最好的技术。因此，新手要更进一步提高自己的技术水平，还要在程序的架构、接口、参数、共享、安全、效率、交互等与程序密切相关的问题上进行探讨和研究，而在这方面有所建树的程序员已经不多见了。所以，新手要多留意，自己也要多积累，在这些方面多下工夫。

3.学习高手的方法

具体的问题是永远解决不完的，尤其是编程。遇到的问题可以说是千奇百怪，无所不有，因此，新手不能把全部的精力放在学习解决问题上。新手要在学习具体问题的解决方法前提下，更要学习高手们是如何解决这类问题的。通过方法的学习，可以使新手有很大的进步。例如，高手向新手讲解一个具体程序的调试方法。新手除了掌握这个程序的调试方法之外，还要学会跳出具体程序，掌握好调试基本流程、主要调试命令和注意事项。只有这样，当新手再次遇到调试问题的时候，就可以不用再找高手指导了，可以用这种方法自己来处理了。

高手的另一特点就是站得高，看得远，一般新手的问题，他们都很轻松摆平，所以，新手也要在平时看问题时，尽量地跳到具体问题之外，站在更高的层次上看待具体问题。

4.学习高手的激情

高手除了技术水平高、工作方法好之外，还有一点是新手要学习的，那就是他们的激情、坚持、专注。高手之所以是高手，一定付出比常人更多的劳动和辛苦。而这种付出并不像工作那样是强制的，往往是他们心甘情愿的，而且乐在其中。新手可能对老程序员没日没夜地连续工作感到不可思议，但是一旦他们也这样做了，他们同样也会感受到那种付出后获得成功的喜悦。

在遇到困难的时候，新手要向高手学习那种永不放弃的精神，不战胜困难就绝不罢休。高手之所以是一个高手，那是在某个领域，他能比别人有更大发言权，因为他比别人更了解这个领域的客观规律，这个规律

是他长期专注和研究的结果。因此，新手遇到各种问题时，要沉下心来，注意观察问题中的细节，不放过任何疑点。只有这样，新手对问题的理解就会比别人更加深入。

5.不迷信高手

其实，现实中并不存在完美的高手，高手总会存在这样或那样的缺陷。新手不要对高手一味盲从，编程好，不一定界面设计好；编程好，不一定设计好。所以，新手要有自己判断是非的能力。例如，高手让新手按照某种要求编写程序，而这种要求本身就可能存在问题。当新手发现这个问题时，就应该停止编程，找出问题的原因所在，最好能找出解决方法，向高手说明情况，争取高手的理解和支持。这样高手就能看到新手的进步。而新手也可以在自己发现的新方法的过程中找到自我价值。

虽然我们是新手，但是，面对高手我们无须恐惧，我们会发现，随着时间推移，自己看到的高手会越来越少。这说明自己的技术水平在不断提高，当有一天有人开始请教你问题的时候，尽管你不处在高手的最前列，但是，你已经步入了高手行列。其实，面对高手就是面对未来的自己。

1.11 新手应该具备的基本素质

新手走过的路，我们都走过了，别人走过的路，我们也看过了。回首往事，我们有过很多经验和教训，但是，几乎没有人会刻意告知我们应该怎么成长，更没有人专门对我们进行素质上的要求和引导。直到今天我们才知道我们的成长是自发的。它完全取决于我们的工作环境和内容，取决于我们遇到的人和开发的项目。假如我们能够回到从前，我们一定会在年轻的时候更加注重培养自己的素质，让我们成长得更好更快。我将与程序员有关的素质方面的要求进行了一个系统性的归纳，希望这些会对程序员有所启发。

1.激情

激情是程序员的职业标志，在很多情况下，激情具有递减规律。也就是说新手的激情最高，越到最后激情越低，甚至麻木和消失。而成功的程序员几乎都是充满激情的，他们能将激情始终保留在身边。我们要对年轻的程序员说，请保持你们的激情，无论你们在工作中遇到多大的不满、委屈、挫折、失望都不要丧失你们的激情，只要你们有了激情，你们才能东山再起，才能勇往直前，才能达到事业的顶峰。

2.学习

无论是新手还是优秀程序员，无论是程序员还是其他职业人员都应该注重学习，人只有在学习中才能增加自己的知识，才能将更多知识用于自己的工作。针对程序员这个职业，由于其涉及软件技术、项目管理、用户的业务知识等方方面面，而且这些方面的知识还在不停地变化和更新，所以只有学习才跟得上职业发

展的要求。刚开始的时候，我们发现若不学习，那就什么都不会，我们只好学习。后来，我们发现即使学习了，还是有很多东西不会。当我们把学习看做一种常态的时候，我们就会不断获得新知识，这样才会适应职业要求。

3.基础

“万丈高楼平地起”，这充分说明了基础的重要性。程序员在开始的时候，并没有感觉到基础的重要性，但是随着程序员不断成长，这种基础的制约现象就会很明显。很多程序员总是感觉自己的进步不大，进步不快，不知道自己的问题出在什么地方。其实，很多最根本的问题是大学生的基础问题。因为很多大学生在学校的时候，并不知道所学的基础知识用在什么地方，有什么看得见的重要性。很少有人会对基础知识有真正的理解。到了工作单位之后，一些程序员在编程的时候，复制网上的程序，还是知其然不知其所以然，基础越来越差。不单是知识基础，其他基础也很重要，例如，一些新手开始不愿意学习盲打，不肯在开始学打字的时候练习盲打，基础没打好，结果若干年后，只会一个手指头按键盘，被人们笑谈为“一指残”。程序员要有基础意识，要把现有的取得的成绩当做基础，只有把现有的基础打牢，这样才能向上更好地发展。

4.好胜

没有一个程序员没有好胜之心的。但是，好胜有强有弱，有的程序员大有舍我其谁的气概，有的程序员则不露声色暗下工夫比高低，有的则看不出来是否好胜，一副好坏与己无关的样子。有的因能力而好胜，有的因面子而好胜。作为程序员好胜是必需的，因为程序员是一种智慧劳动，要比就是比智慧。只有通过智慧的竞争，智慧才能精彩，程序才能更加漂亮，软件才能向前发展。好胜意味着要比同们做得更好，好胜意味着遇到困难必须克服，好胜意味着自己要比自己原先做得更好。我们要的是好胜的结果，我们不需要的是好胜的表现。其实，不要刻意表现你的能力，你的成果足以表现你的能力了。

5.动脑

软件本质上是一种智力的产物，这个智力并非天生就有的，它和程序员的技术能力、理论基础、思维方式、知识范围、周围影响等因素密切相关。只有通过对这些因素进行收集、存储、加工、处理，进行各种各样排列组合，形成各种解决方案，然后在这些方案中进行择优判断，才能得出最后的解决问题的办法。程序员除了动脑还要通过打键盘编写程序，所以程序员还有一个动手的工作。很多程序员习惯于边想边编，久而久之就养成了动手习惯。从成长的眼光来看，程序员应更加注重开动脑筋，而且要把其和动手编程分离开。这样养成动脑的习惯之后，对编写程序有很大的帮助。一般而言，动脑的时间越长，动手的时间越短，程序员技术水平就可能越高。

6.外向

程序员性格对成长有很大的影响，无论是原来外向还是内向性格的程序员，只要从事这个职业，只要和计算机打交道，其性格都会有点向内向方面转变的趋势，我们看到不少程序员不善于和另人交谈，怕见人，甚至从内心里不愿意和别人交谈。这样的自我封闭其实对自己并无益处，我认为程序员性格要外向一些，要乐于和别人交流，要主动和另人交流，和别人交流并不一定要限于技术，用户的需求、公司成长情况甚至对方个人生活等方方面面都可以进行交流。这些外向的性格可以让程序员见识更多的人，见识更多的领导，见识更多的高手，见识过动去不敢见识的人。这样可以积累更多的社会关系。

7.技能

程序员的技术能力是程序员生存下来的基础，而技能从某些方面来说是首要的。可以说程序员就是技术的代名词。有了好的技能你就能更好地开展自己的工作，有了好的技能你就有了和别人交流的内容，有了好的技能就可能让自己获得更高一层的技能。因此，程序员要重视自己的技能学习和提高，要在技术上体现自己的能力，要通过技术能力去影响或帮助自己的同伴。只有这样，程序员的最基本的价值才能真正体现出来。很多程序员到了一定阶段放松了对技术的追求，技术平平，只能满足现有工作，这就愧对程序员这个称号了。

8.团队

程序员可以一个人编程序，但是，一个人只能编写一些程序或小规模的程序。有一些程序员一个人完成了整个系统开发，其技术水平固然值得称贺，但是，如果他认为软件可以一个人摆平，这就会影响到其向更高水平发展。随着软件业发展，软件的规模也在变大，软件制作的专业化程度变得越来越高，一个人即使有这个能力开发一个完整的系统，最好也不要一个人去完成，只有通过团队的分工协作，软件制作才能走向正道。如果程序员一开始就能注重团队意识，一开始就认为自己仅仅是团队的一分子，一开始就注重与其他团员的沟通和协作，这样程序员一定能融入团队中，而团队的巨大作用是每个程序员个人作用所不可比拟的。程序员不要只是享受自己独自编程的快乐，而是要享受整个团队编程的快乐。

9.兴趣

程序员的工作内容看起来似乎只是一行行代码。代码的编写来自于程序员头脑的指令，而程序员头脑中的指令并不是来源于程序设计语言，而是来源于各种需求、各种问题的分析方法和处理方法。因此，程序员要编好程序，不仅要对程序语言感兴趣，而且要对所有和编程序相关的事物感兴趣，甚至对看上去与编程无关的东西也要感兴趣。我们成长后才知道各种事物都是相通的，有些东西会潜移默化地影响到你的分析

方法和处理方式中。所以，程序员不要只关心编程，这样只会成为“书呆子”。我们日常生活中看电视、听新闻、上网聊天、社交活动等都应放在兴趣之中。只有广泛的兴趣才能体会到你最爱的兴趣。

10.谦和

我喜欢程序员有一种谦和的精神，尤其是那些有本领骄傲的程序员的谦和。我们常常看到一些得意于自己的程序、自己的项目、自己的收入、自己的职位而忘形的程序员，他们无视别人的意见和建议，有的甚至自傲，看不起其他程序员。但是，要知道艺无止境，假定软件领域拥有顶峰，而谦和正是通向这个顶峰的一个阶梯。更何况软件领域没有顶峰，更需要我们的谦和来表明自己不拘泥现有的成就，我们还有更高更远的理想。

11.求新

我们常说软件技术发展太快，跟不上就会被淘汰。回顾几十年来的软件技术发展就可以明白这一点。求新本质就是注重时代的变化、跟上时代的变化。因此，对程序员来说，要有求新的意识，不要排斥新生事物。要通过网络和各种媒体注意各种新技术的产生，对于与自己工作相关或感兴趣的技术要花点时间进行跟踪，要了解这些新东西的新方面，要学会新旧对比，对自己必须掌握的新技术要毫不犹豫地花时间拿下。求新会让程序员时刻感到压力，但是求新又让程序员能看到自己未来成长的方向。

12.主动

主动绝对是程序员的一个良好素质。我和许多年轻的程序员打过交道，有主动型的与被动型之分，大凡学习上主动，工作上主动，甚至是劳动上主动的人，大都会得到同事的喜爱，都会得到更多的回报。而那些虽然听话，但是“不说不动”的人，因为缺乏主动意识，只能被动听从别人的安排。别人给什么，自己就吃什么，就像一只填鸭，当哪天没有人给你填食的时候，结果就可想而知了。“会哭的孩子有奶吃。”这是很有道理的一句话。

13.吃苦

现在的程序员和我们那个年代的程序员有很大的不同，他们很多都是独生子女，家庭娇惯比我们那个年代多了很多，而且绝对的自我。因此，他们眼高手低，怕吃苦，不能吃苦。对于程序员来说，连续工作是一种很常态的事。有的吃不了程序员的苦，因苦而选择离开。我们不能说吃苦是一件好事，但是，一个人能吃苦说明这个人不怕困难，有坚韧不拔的意志。这对程序员成长是很有帮助的。想想当年，自己要是怕吃

苦，可能也不会有今天的成果。当一个人没有苦吃的时候，说明这个人真的没有发展机会了。而且吃苦的人更能感到甜的味道。

一个人的素质是其成长的基础，良好的素质一定是后天培养的，是后天自我约束和完善形成的，这种约束和完善的内在要求使得他在做任何事的时候趋于合情、合理，容易获得进步和成功。同时，一个人良好的素质可以给外界一个良好的形象，而外界因此会给这个人更多的关注和鼓励，外界的影响反过来又促使这个人进一步提高素质。相同的时间和相同的环境，不同的人因其素质的不同会有很大的不同。因此，如果我们在入门阶段能够重视自己的素质的培养，知道自己在什么方面需要改进和提高，程序员一定会少走弯路，成长更加顺利，而其中的良好素质将影响程序员的一生。

很多程序员对以上方面也有所了解，也知道应该怎么做。但是，涉及自身时，说归说，做归做。如果是这样的话，缺乏自己约束的程序员也只能“自发”地成长了。

第二部分 成长篇

2.1 加班，加班，加班

在从事程序员工作之前，虽然工作时偶然有过加班之说，但那仅仅是偶尔而为之，并不会太在意。后来，无论是从自己的亲身经历，还是从周围同事的经历，或者是从亲眼所见的各个软件公司的开发过程，更或者是从可听、可读关于程序员加班的传言，一切的一切都证实了：对程序员这个职业来说，加班是一种常态。如果你去问程序员有哪个没有加过班的，举手的一定是第一天参加工作的程序员。

度过了入门阶段的程序员，由于他们具备了编程能力（尽管这个能力并不是太高，技术并非很强），大量的工作或者说是超负荷的工作，就成了这种能力提高的“体能”训练。有的程序员加班实属不情愿，很无奈。因此，他们对加班叫苦连天，埋怨很多。所以，很多人认为程序员是吃青春饭的职业，年龄一大就承受不了加班的压力了。

加班有两种类型：一种是单位因工作而要求的加班，是强制性的，称为被动加班；另一种是程序员自身习惯性延续工作所产生的加班，称为主动加班。

程序员加班发展趋势是：开始阶段是主动加班时间不断增加；第二阶段是单位强制性的被动加班时间不断增加，主动加班时间减少；第三阶段是被动加班时间逐步减少，主动加班时间逐步增加；第四阶段是自己和单位都不要求加班。

第一阶段：程序员刚进入程序员这个行业，需要大量的自学时间，以提高自己的编程能力。无论是看书还是编程的热情和压力都很大，而且程序员的思维惯性很强，不容易突然中断，加之程序员是个人劳动，没有外人的约束，所以很多程序员喜欢在无人干扰的夜深人静的时候开始工作，这就造成了晚睡晚起，工作没有规律，养出来职业“加班”习惯。这个时候的加班特点是主动自愿的。

第二阶段：当程序员具备了一定的工作能力后，单位完成开发任务的时间一般要求都比较急，所以，一般在正常工作时间是完不成的，程序员因而被要求加班。不同的单位因开发项目的多少，项目要求时间的松紧，会要求不同程度的加班。这种加班一般会伴随着程序员成长的大部分时间。许多人因加班而抱怨程序员工作太苦而转行。在这个阶段中，由于工作的疲惫，个人主动加班的时间会大大减少，但是，当单位要求某个工作在某个时间点完成的时候，尽管不明确要求加班，个人还是会为完成工作而加班。

要特别指出的是，这个时候的程序员已经很少有精力和兴趣像刚开始时那样去学习和研究自己感兴趣的编程问题了。

第三阶段：当程序员经过成熟期之后，其资历和能力有了提高，很多人转向了软件设计和项目经理角色，“被加班”情况大大减少，往往是要求其他程序员加班，大部分时间是“陪加班”，加班的工作强度大大弱于以前了。

所以，这个阶段有的程序员会由于自身能力提高的需要，又开始增加自身加班的时间了，但是，其程度和开始阶段相比要弱了很多。

第四阶段：程序员发展到最后，要么转行，要么跳槽，要么升职，自己学习的激情消失殆尽，无论是工作还是自身都不需要加班了。

当我们严肃地面对加班问题的时候，我们也会思考加班带来的东西。

1.加班很自我、很享受

当主动加班的时候，是程序员最自我的时候。夜深人静，没人干扰，思绪和编程效率很高，尤其是把一个问题看懂的时候，把一段程序编写出来的时候，把一个出错原因查出来的时候，程序员会很有成就感。此时加班令人兴奋和享受，令人忘却了加班带来的疲劳。

2.加班环境很宽松

当整个项目组一起加班的时候，因管理要比白日相对宽松，项目组成员会在加班的时候交流、吃夜餐、听音乐、闲聊、等待、观看、学习、小睡等，大家更加随意，比平时人与人之间的关系更容易相处。很多时候，人与人之间的信任就是这个时候产生的，而且这时更容易感觉到团队的精神和力量，当一个项目完成的时候，最能让人难忘的是那些加班时发生的故事。

3.程序员水平低

某些时候，加班并非都是项目经理和项目本身造成的。有的时候是由于程序员自身水平造成的。有些程序员水平不高，一个高手几分钟就能解决的问题，他们可能要一天或是几天才能完成。所以，从宏观层面来看，如果程序员的水平能提高的话，其加班时间也会相应缩短。

4.鞭打快牛

说句笑话，也许你能力越强加班就会越多。在任何单位，“鞭打快牛”都是很普遍的。但是这种打其实是很舒服的，只是当时有点疼罢了。当过了一段时间之后，没有人鞭策你的时候，你就可能享受不了那种“痛感”了。因为你的“价值”没有了，人家不“利用”了。在这个商品社会，当你有价值的时候，别人才会和你进行“等价交换”。所以，让自己价值最大化才是员工职场的最大目标。

5.加班费

加班中最让人关心的事情就是加班费的问题。有些单位在加班费问题上总是不尽如人意。程序员能反映一定要反映，能争取一定要去争取，不能争取到钱也要争取到假期。说起来容易，做起来不易。程序员对自己无力改变的事，只能放宽心态，努力提高自己的技术水平，只有这样我们才有可能避免“免费加班”。

总之，加班是程序员这个职业的一个普遍现象，它主要由编程工作的不可中断性所决定。加班说明我们有软件需要开发，加班越多说明我们的软件市场越大，加班越多说明我们的软件开发竞争激烈程度越高。当有一天我们不加班的时候，我们就要怀疑我们的饭碗还能否保住，程序员这个职业是否还会存在下去。而我们主动加班却代表了我们对程序员职业的一种兴趣、爱好和追求。我们在加班中付出了劳动，我们就要在劳动中寻找我们的快乐，去寻找和延续我们的梦想。

2.2 大量编程带来的快乐和烦恼

程序员成长阶段面临的工作就是编程，而且是大量编程。这和以往自娱自乐式的编程是完全不同的。这个时候的编程是职业，是和薪水以及被用户认可的软件联系在一起的。编程少了则无法深入体会各种快乐和烦恼。

程序是程序员创造出来的产品。当然程序越多，说明程序员生产能力越强。据统计，1~3 年之间，程序员一般可能编写 5 万~20 万行代码（包括编写后删除的）。很多程序员都不太清楚自己从业后到底编写了多少行代码，如果真的有心统计，自己绝对会被吓一跳。从代码行数来说明程序的工作量还只是机械的统计方法，很多程序代码很少，但是花的时间却很多，主要是含金量很高。

程序员马不停蹄、加班加点编制了大量的程序，他们快乐吗？如果没有快乐，难道他们真的只是为了薪水而这样不快乐地工作吗？如果有快乐，难道他们真的以乐为苦，故意喊着工作劳累以博取别人的同情吗？

以我几十年的编程经验来看，编程是快乐和苦恼相互交织的一种工作，而且是一种挑战性的、超越自己的工作。对于我来说，其快乐要远大于苦恼，否则，我早就弃它而去了。

编程会给程序员带来快乐。这种快乐有时只能意会，是无法用言语、文字表达的。那么编程会给程序员带来怎样的快乐呢？

1.成就感

我问过很多程序员同样的话题：“编程的最大快乐是什么？”得到的回答几乎都是“成就感”。成就感意味着觉得自己做了一件了不起的事，做了一件非常有用的事，做了一件有价值的事，做了一件别人做不了的事。程序编多了，无论是编程的结果还是编程的过程，都会让人产生这种感觉。尽管有的软件项目是分拆给好几个程序员的，但是，就其工作性质而言还是属于个人劳动的范畴。程序员的很多工作在某种程度上都是个人劳动。编写一段代码、一个函数、一个模块、一个软件都是程序员自我实现的过程。每当程序员完成这种过程后都感到如释重负，有一种“我终于成功了”的感觉。

2.被认同感

程序员原来对程序具有无知、恐惧的心理，而通过大量的编程将逐渐克服这些问题。程序员的自信心也会逐步强大起来，而周围的同事往往比他自己先一步看到这种进步，从而率先对他进行认同。尤其是自己原本初来乍到，水平、能力不能充分展示，自己内心也很着急，但是同事并不当回事，对自己不温不火的。随着工作的开展，自己的能力逐渐显现出来，同事也开始转变对自己的看法，从各从方面或明或暗地表现

出对自己的认同，这种认同往往会让程序员内心涌出一种满足感。尤其当程序员的上级甚至老板表扬自己工作成果的时候，这种被认同的感觉让人有一种归宿感，甚至用户对自己的认可都会让程序员倍感高兴。

3.团队氛围

程序员在成长中，一定会和其他程序员以及项目经理打交道。每个程序员和每个项目经理由于个性、能力、经历的不同而与之交往的方式和结果也会不同。随着时间的推移，程序员在这种不断交往的过程中，增强了团队的意识，增加了软件开发中团队的凝聚力。程序员在团队中一方面能够获得团队成员的帮助和支持，另一方面作为团队的一分子，也在为团队整体作出贡献。每当一个项目在千辛万苦的工作之后完工时，那种团队集体相拥的开心是难以言表的，有的男女甚至因此而结缘。也有个别程序员不能处理好与其他同事的关系，那工作起来就会感到很别扭。

4.技能熟练

在编程初期，程序员编起程序起来可以用“一步一个跟头”来形容，编程速度慢得不可想象。随着大量积累编程经验，程序员逐步找到编程的工作流程和窍门，编程速度会大大加快。到后来他们几乎到了“兵来将挡，水来土掩”的境界。原来要好几天才能编好的程序，现在只要几分钟就摆平了。有时这种熟练程度连自己都会不敢相信。

5.学生变老师

程序员开始工作的时候绝对是一个学生，在工作中慢慢由学生变成了老师，而在其后进入职场的则当起了学生。当学生们问起自己曾经问过上一任老师的问题的时候，那种老师的优越感不由你不产生，不由你不认真去解答。有的甚至有主动教学的冲动。

6.扩大朋友圈

编程多了，项目自然就多了；项目多了，接触的人也就多了；接触人多了，程序员交友的机会就多了。程序员在这个过程中，无论是和程序员同行、软件设计师、项目经理、上级主管、公司老板、用户、合作伙伴甚至是网友都会有所接触。许多程序员因工作需要经常在用户单位进行开发和维护，这样与用户打交道的机会很多，因此，有可能会结交用户朋友。在 IT 人员稀缺的年代，有时用户会对看中的程序员挖墙脚，在项目验收后，程序员由乙方变成了甲方。

7.当然编程也会带来烦恼，而且这个烦恼因人而异，各不相同。

1.遇到问题

程序员最大的烦恼就是会遇到问题。编程遇到的问题可以说是千奇百怪。常见的问题就是不会编、编不好、调不通、运行错、查不出错、效率慢等。尤其遇到那些无从下手，查不出问题，同时又找不到懂行的人来帮助的时候，最为心急和苦恼。这些问题困扰越深，一旦问题解决后程序员就越兴奋。

2.加班劳累

加班是一件很劳累的事，尤其是连续加班更是如此。许多时候，我们都以不愉快的心情在加班，无论是从效率还是从最终结果来看，都不是一件太好的事情。程序员更应该劳逸结合，累的时候，休息一下，保持自己头脑的清醒，这样才能编出好的程序来。

3.编程厌倦感

编程如同开车，刚开始时感到新鲜，有冲动、过瘾，有一种如痴如狂的感觉。过了一段时间后，累了、疲了，竟会有一种不想开的感觉。编程多了也是如此，可能有一段时间你见到程序就觉得疲倦。那真的是编写程序过多了。程序员到了这个阶段，就要特别小心，因为这种厌倦感加上周围发生的其他事，会产生一种合力，让你有一种强烈地要求离开这个职业的愿望。

4.技术水平提高慢

虽然程序编了很多，编程速度也很快，但是，很多人在时间压力下，往往只关心能否编出来，而没有时间关心编写的程序的质量。复制粘贴、复制粘贴，你都没有时间和心情去品味程序的好坏，去思索程序的优劣。很多程序员尽管编了好几年的程序，你若去问他，编过什么让自己或是别人叫好的程序，他一定不会给出直接肯定的回答。时间有序地增长，而程序员的技术水平却不见得在提高。能提高技术水平的程序员一定是有心要提高并追求提高的程序员。那些把编程序纯粹当做混饭吃的程序员，100 个中有 100 个的技术水平不高。

5.收入和劳动不成正比

每一个善良的劳动者都希望自己的劳动和收入成一种正比的关系。但是在现实中，你付出的和你得到的很难成正比。这种情况在程序员工作的初期特别常见。因此，很多程序员都心生怨言。也有少数水平高的程

序员拿着相对高的工资，但是，我想这些程序员过去也是当过几年“媳妇”的，否则，他们也熬不成这个“婆婆”。

面对金钱，我只能说一句话，抓紧时间把自己的能力提高，扩大自己的就业机会，是金子总会闪光的。任何企业都不会拒绝能为他们创造财富的员工。

6.跟不上新技术发展的步伐

从职业生涯的角度来看，程序员最担心的是自己跟不上技术发展的步伐。程序员这个职业的特点就是技术更新快，无论是编程环境（计算机、网络、操作系统、数据库等）还是程序设计语言都在不断地升级和更新。一些环境和语言在不知不觉中就淡出了人们的视野，一些不为人知的名词和概念不时闪出，令人眼花缭乱，目不暇接。而新的东西往往代表着未来，所以，程序员担心自己掌握不了新技术，担心自己的未来，而且这种担心贯穿于整个程序员的职业生涯。程序员编程序多了，没有时间和精力去了解新的技术，不去了解新的技术，只能用原有语言进行编程，如此形成一个非良性循环。每循环一次，程序员的心事都会加重一点儿。有的因为这种担心而放弃了程序员这个职业。

其实，大量编程会给程序员带来大量的快乐和苦恼，我们很难全部列举出这些快乐和苦恼。这个不是你自己愿不愿意的事，而是工作的需要所致。任何一个程序员从入门到成长都必须经历这个阶段，这个阶段将会为今后的成熟阶段和优秀阶段打下一个重要的职业基础。我们面对编程要怀有一颗快乐的心，无论多么烦恼，我们都要坦然面对。只有这样，我们才能真实享受编程中的种种快乐。否则，当烦恼超过快乐的时候，我们所有的快乐将会化为乌有，我们所有以前积累的价值将回归于零。

2.3 需求总是变化，程序总在修改

当程序员学会编程序之后，开始慢慢地进入项目开发的状况。随着项目开发越来越多，编程也变得熟练起来，程序员渐渐地把学习和工作的重心转移到客户的需求上来。通过与用户进行交流，对用户需求有了由浅入深的感觉，并且越来越感觉到了用户需求和编程之间的密切关系后，如果对需求理解比较深而且理解比较正确，程序编起来就感到特别顺畅，反之，编起来总是担心程序编的对不对。程序员慢慢地就形成了由用户来验证程序正确性的习惯了。即使自己编写的程序很正确，心里还是没有什么信心。

这些仅仅是一个开始，让程序员感到头痛的是，明明上午说好了这样做的需求，到了下午又要改成那样做了。这样编好的程序又要修改或是重写了。这种情况在项目的开发阶段程序员还可以忍受，很多情况下项目到了测试阶段甚至是上线阶段的时候，用户还会提出功能变更的要求。另外，项目投产运行后，用户还会有变更维护的需求。

面对用户需求的不断变更，程序员倍感头痛但也无可奈何，抱怨归抱怨，还是只能按照用户的要求去做。因为用户需求始终掌握着程序员程序的生杀大权。很多软件公司、项目经理、程序员面对不断变化的用户需求，想尽各种办法，希望需求能够稳定下来，需求能够不变。他们想通过用户需求书，通过用户对需求书的确认，通过增加需求变更手续环节和增加需求变更的费用来约束用户对需求的变更。但是，实际收效甚微，很难有不变的用户需求书。用户需求不断改变不是什么个案，而是一个普遍现象。

1.用户需求不断变化是一种客观规律

很多程序员想当然地认为，用户需求是不应该变的。如果这种观念正确的话，那么在现实中，程序员看到过有多少不变的用户需求？可以说，项目越大，用户需求变化的概率就越大。至于很小的程序、不复杂的程序，用户需求不变是有可能的。但是，用户需求变化是很正常的事情。

用户需求不断变更应该是一种规律，它是和企业经营不断发展而导致企业信息化发展相一致的。由于市场竞争和企业发展的需要，企业经营必须随时进行调整和变革，以适应这种变化。而企业信息化则是这些调整和变革的集中反映。例如，2003 年中国人民银行提出反洗钱的要求，作为金融机构的银行必须接受央行的监管，就必须开展反洗钱工作，从而导致银行业各种反洗钱系统的开发。如果一个企业的经营方式没有太大的变化，这表明企业发展已经很正常、很健康了，而这种现状一定是原先信息化带来的结果，而当初的信息化一定是在一种不断修正中成熟的，由此也可以想象其用户需求变化之多、之快了；而到目前阶段，企业经营走向正常那就可能不用再进行软件开发了。如果一个企业的经营方式发生了变化，那就必须采用信息化的形式来实现这个变化，而这个变化是一个过程，信息化一定不会等到这个变化完成之后才开始。比如，当初银行有网上银行的设想时，银行不可能一下就能考虑到这种方式的方方面面，它需要走一步看一步，通过前进来修正前进中的问题。所以银行业的网上系统是一个不断完善的过程。我们可以回头看看，当初网上银行是什么样，然后看看现在网上银行是什么样，那就会知道，网上银行业一定是一个不断完善的过程。因此，在企业信息化过程中就会面对用户需求的不断变更。

2.程序员要调整心态面对这些变化

既然用户需求不断变化是一种客观规律，那么程序员在心态上就要进行调整，将用户需求不能变化调整为用户需求一定会变化，树立起“变正常，不变不正常”的观点。有了这样的心态，我们就不会去抱怨用户需求了，就会对用户需求的变更有更多的理解。有时候，我们希望用户需求变更，如果现在不变更将来还是要变更的，而往后变更的代价要比现在变更的代价大得多。举例来说，如果一个用户在系统开发期内提出一个需求变更，程序员就可以开发期内解决；如果用户在系统上线后提出同样的需求变更，那这个变更可能对其他程序产生重大的影响，甚至现在系统架构产生影响。更有可能发生的是，当初编这种程序的程

序员已经离开了，现在又要有一个新人来实现这个需求，可想而知，后一种变更的代价的确太大了，所以，用户提出需求变更过程对程序员来说不是一个坏事，而一件好事。

3.程序员要积极主动地面对需求变化

（1）理解用户需求

程序员编写程序时，理解用户的需求很重要。有的程序员凭需求书或自己的想象去理解用户需求，然后就开始编写程序了，编写完之后，经用户检查才发现需求理解错误，程序只好重新修改。这种情况不在少数，推翻原来程序重新编写的事情常有发生。因此，程序员要能正确理解客户需求，少走弯路。我的经验是增加一个验证需求的环节，以避免这种现象的出现。比如，拿到需求书后，可以向项目组的其他程序员讲述自己的理解，听听别人的看法，如果别人不能认同，就有可能说明自己的理解有问题。又比如，我们可以把自己的理解和用户当面进行交流，得到确认后，再去编程。

（2）正确对待需求书

很多人对用户需求书比较迷信，认为那就是需求的最终版本，不可更改，一切要按需求书来编程。这种一是一、二是二的看法，想法太理想、太天真，可以说这样的程序员还处于“学生气”状态。需求书固然是用户需求的书面表示，是开发项目程序编写的依据，但是，要知道需求书也是人写的，只要是人写的，这个需求书就会反映出作者对业务的理解能力和业务水平以及表达能力。而这些往往和最终理想的用户需求是有偏差的。另外，需求书的编写也是需要时间的，有的需求书写得匆忙，论证不够，不妥之处必定存在。在现实中，用户只是提出一个大概的业务功能要求，具体用户需求书都是由程序员来完成，最后由用户确认。这说明需求书并不一个绝对的东西。在现实中，最终的系统功能与最初的需求书中的功能往往相差很大，最终的系统功能要大大多于需求书的功能。我的意思是，程序员不可迷信需求书，要根据项目的设计目标来分析需求书中的逻辑性。

（3）不要急于编程

程序员一般性格比较急，用户一提出修改，往往马上会按用户的要求去改程序。虽然一般情况下用户要求的时间比较紧，但程序员还是不要急于编程为好。因为用户提出的修改意见不一定全是想得比较周全的。在现实中，程序员刚刚按用户需求改好了，这个时候用户又提出新的要求了，这种现象并不少见。因此，我的建议是，用户提出需求变更以后，要先和其沟通，看看是否真的需要变更，如果真的需要变更，则要和用户交流一下变更后可能出现的各种情况，让用户认可。此时可以建议用户再想一想，再看一看，还有什么要改动的地方。等到用户实在提不出新的要求了，再考虑编程。还有一种情况，那就是将用户变更的

需求放到一边，等 到若干个变更需求都确定的时候，再去编程。这些做的好处在于避免单个需求编程和急促编程带来的重新编程的后果，可以整体上考虑编程的效果。

（4）用技术应付需求变化

很多情况下，程序员抱怨用户需求变化太快，自己跟不上他们的变化，而且前面的编程工作白费，令人沮丧。但是，程序员是否考虑过用户需求变更一定要更改程序吗？如果程序员回答是，那说明程序员的技术水平还有待提高。如果项目一开始就对用户需求分析比较充分，软件架构设计比较合理，功能模块设计比较清晰，那么到了用户提出需求变更时，这种变更也是小范围的和有限度的，大部分可以通过参数化的方式来解决，也可以通过外挂功能的方式来解决。比如，原来用户提出打印两张报表，现在又提出打印 3 张报表。如果程序员对打印报表有足够经验的话，在开始设计的时候，就已经考虑到报表可能不止两张，已经考虑报表产生可以通过外挂方式来实现主程序的不更改。如果程序员只是一张一张地处理报表，整个程序没有架构，那么增加到的 3 张表将会使程序很长很乱，如果用户增加到 30 张、300 张那又如何处理呢？所以，我们说，不好的技术可以解决一个需求，而好的技术可以解决一类需求。随着程序员开发经验的增加和技术水平的提高，用户需求变更将不再是一个令人头痛的问题。

（5）向用户提出建议

很多程序员也看到，很多时候用户提出的需求都是粗线条的，考虑并不周全，这就需要程序员根据自己的经验提出合理的建议，让需求完善起来，否则，最终用户还会回到这个建议上。例如，用户要求查询数据，并把查询结果给显示出来。其实有关数据查询和结果处理有很多种方案可供选择。但是，用户当时只会考到到查询和显示这些主要的功能。如果程序员能够和用户进行沟通，建议查询时要有权限控制，显示时可以增加导出为 Excel 文件等，我想用户是会很欢迎这些建议的。如果程序员不这样做，到后来，用户肯定会提出增加查询权限、增加 Excel 文件导出功能的。到这个时候再修改程序，肯定不如开始设计的时候就有这些功能省事。为用户着想，也是为自己着想。很多时候，用户是不完美的，程序员也是不完美的，但是，若两者结合，就有可能完美起来。

很多情况下，你不喜欢的东西就可能包含着对你有用的东西。用户需求不断变化，程序不断修改，程序员会感到厌烦和无奈。但是，当程序员静下心来，真正去分析和解决这些问题的时候，程序员可能会发现用户需求背后无限的学习空间；就可能发现用户需求背后的软件价值和市场价值所在；就可能发现自己技术的薄弱之处和改进之处;就可能感觉到技术对解决需求的巨大魅力;就可能感觉到自己的能力和价值所在。程序员通过编写程序走向技术成熟，程序员通过用户需求感受价值获取的市场。当我们不可能要求别人改变的时候，我们只能改变自己以适应别人。

2.4 为什么程序员不愿写文档

一提到文档，肯定会有人向你大谈外国公司、大公司、正规公司是怎么怎么重视文档的，什么 2/3 时间用于写文档，1/3 时间才用来编程序；写文档要按照什么 ISO、什么 CMM、什么标准；不按这些标准写出来的就不是文档，就不是好文档；不重视文档就是不正规等。当问及他自己写过什么文档，编写过什么文档标准的时候，这种人就哑口无言了。

不管怎么说，文档绝对是程序员最大的软肋。一些称为高手的程序员往往可能是文档方面的低能儿。不管这个程序员是在大公司、小公司，不管程序员是写文档的、还是不写文档的，大部分程序员在内心深处都是不愿意写文档的。

程序员一般不愿意写文档，但是程序员却喜欢看别人的文档。即使写文档，程序员一般不会把所有功能都写入文档，却抱怨别人的文档中有的功能没有说明。即使愿意写某段文档，程序员一般不会把文档写得很详细，却抱怨别人写的文档不够详细。文档绝对是摆在程序员面前的一个矛盾，如果让程序员选择是喜欢写文档，还是喜欢看文档，我估计大多数程序员都会选择后者。

那么程序员为什么不愿意写文档呢？其中的原因很多，我自己归纳了几点：

1.怕烦

程序员从入门之日起，就在心里埋藏了一颗编程的种子，认为程序员就是编程序的，就是和计算机打交道的，程序就是程序员的全部。无论是在编程之前写文档，还是在编程之后写文档。他们都认为写文档很烦人。

1）文档种类太多。越正规越多。一想到要写那么多的文档，程序员的头都要大了。

2）文档写作要求不低。有格式要求、内容要求，还需要画各种流程图、示意图、关系图、界面图和填写各种表格说明，以及要收集各种资料。虽然没有技术含量，但是要花的时间比编程序多，而且也不一定能写好。

3）在正规的开发公司中一般一个变更就要编写一系列的变更文档。当不断变更时，只有最后的变更文档是最重要的。当我们不知道当前是不是最后一个文档时，我们就不愿意写可能被后面的文档替代的文档。

4）当写了文档之后，就会不断有使用文档的人来询问细节，这些询问往往会让人崩溃。这是因为很难写出让每个人都提不出问题的文档。

2.没空

在很多情况下，程序员都处于一种“时间紧、任务重”状态。在急于得出编程结果的驱使下，程序员一心扑在编程上，恨不得一分钟一个变化，哪有心思和时间先把文档写好再去编程呢？即使他们拥有先把程序编出来，然后再补写文档的想法，但是一旦他们完成一段程序之后，就会立即扑向第二段程序。如此下来，编写文档只能放在项目开发的后期了。你要是真的写文档，那写文档的时间早已将编程时间给挤占了，你的编程工作就完不成了。

3.没有用

文档的重要性其实对不同对象是不一样的。如果你说文档没有用，立即会有人用唾沫把你淹死，他们立即会搬出哪个哪个说文档是重要的，文档是不可缺的。唯独不敢说“我自己认为文档是重要的”。这也反过来说明文档的有用性程度对不同人是不一样的。对于程序员来说，只要能把程序编出来就行了。很多程序员不写文档照样编出程序来，在他们的观念中文档不写也罢。如果按文档编程序，那就要确保文档的正确性、不可更改性。而实际上，文档不如编程快，编程不如变化快，不断变化的需求和代码让文档如同一张过时的废纸一样。

在现实中，有的文档变成了项目开发后的总结，对开发本身并不起作用，只是保留一个存在的形式，以应付各种各样规范的需要。在这种情况下，文档没有任何实用价值，所以即使程序员写了，也没有什么作用。尤其是项目投产后，几经升级，最初的文档早已和现实的情况对不上号了，文档更新和系统的一致性更是难于控制。

4.没好处

既然认为编程序是正道，那么程序员写文档就是一种额外和辅助的工作，做了就做了，对于程序员来说没有任何好处。

当然，文档的好处更多地体现在软件公司、单位、软件用户、后续程序员身上，它是一种“前人栽树，后人乘凉”的好事。所以程序员不愿意写文档。

文档有多么重要呢？有人把它上升到“程序员头脑的拷贝”的高度，有的人甚至说，程序员走了后，只要有了文档，软件公司可以再招新人，公司照样运转起来。如此说来，天性聪明的程序员会把文档写得清清楚楚吗？

5.不会写

从写作本身来看，写好一篇文档不是一件容易的事。文档有文档的格式和写作要求。现在的人只喜欢动嘴说事，没有多少人在平时会动笔写字，更不用说写文章了。所以，写文档从理论上来说也是需要学习和训练的，需要在平时进行写作积累。当要求一些程序员写文档的时候，他往往会回答你：“怎么写呀？我不会。”你不会写可以不怪你，你不去学，反而理直气壮地说不会，则是你的不对了。难道那些会写文档的人都是“呆子”吗？难道他们不懂干活辛苦的道理吗？

当一个男子向一个自己喜欢的女子示爱，需要用情书来表白的时候，难道他不会写情书吗？即使他不会写，他也一定会克服千难万险把情书写出来的，这是他的主观使然。如果程序员主观上想写文档，怎么会怕烦？怎么会怕没空？怎么会怕文档没有用？怎么会怕不会写？这些怕其实都是借口。真正优秀的程序员应该不单是编程的高手，同时也应该是写文档的高手。

程序员真正不喜欢写文档的原因是：文档是给别人看的，不是给自己看的。如果要使程序员喜欢写文档，那就要提高程序员的意识和境界，或者要给写文档的程序员以奖励。而靠制度、管理让程序员去写文档只能是一种职业上的弊端。当有一天写文档也成为一个专业化的岗位，程序员和文档人员分开时，程序员就会一心一意看文档写代码，文档员就会以此为职业，一心一意写文档拿工资。我们将不再为此进行讨论。这种分工一定是未来的趋势。一些大的公司或管理规范的公司都已经有这样的分工了。只是很多中小软件公司或企事业单位里的 IT 部门还没有专业化到如此程度。程序员不喜欢写文档也许说明他们不愿意承担太多的角色吧。

不管怎么说，作为一个程序员（尤其想成为优秀程序员）一定要学会写文档，一定要学会欣赏文档。无论你是否喜欢，你都应该可以在文档的各种问题面前可进可退。可以这样说，那些既不会写文档，又不会欣赏文档的程序员是没有资格说“我就不喜欢写文档”这种话的。

2.5 为什么编程者总是高估自己低估别人

不知道大家感觉到了没有，一般程序员都很自信。这种自信成了这个职业的一个亮丽的特点。放眼望去，哪个程序员不自信满满？哪个程序员不胸怀比尔·盖茨之志？哪个程序员会对别的程序员发自内心的佩服？所以程序员几乎都是单兵作战，即使在同一个软件公司，同一个开发小组，他们之间也缺乏交流，很难建立那种理想的团队关系。

我认为程序员自信的建立来自于以下几个方面：

1.社会评价

整个社会对 IT 有着很高的评价，凡是 IT 从业人员都沾了光。尽管这些年对 IT 的评价逐年降低，但是普通人对 IT 还是抱有神秘感的。这种社会目光投射到程序员身上，让我们程序员经常有得意的感觉。

2.单位同事的评价

一般的企业或机关部门（除了软件公司），IT 人员只占到 10%以下。人们天生会对自己不懂的东西产生崇拜心理（不管这个东西是不是值得崇拜），大多数人对 IT 行业只是知道些皮毛，而对 IT 程序员这样的专业人士，不流露出羡慕的目光是不可能的。在这种目光下，程序员能不自信吗？

3.同行的评价

由于程序是由个人编写的，程序员之间必然存在竞争。尽管他们会合作共同编写一个大的项目，但是，这种合作是通过接口进行的，每个人还是个体编程者，只要会编程的人都是他的替代人选。但是这种竞争不是显式竞争，而是用一种不言自明水平高低的方式来保持这种平衡。虽然同行认为自己编程水平不行，而且自己的水平确实很差，但是若别人不说，那么自己的自信又重新建立起了。

4.编程的职业特点和程序员的业务水平

编程的职业特点之一是劳动工具昂贵，20 世纪 80 年代一个普通的 PC 机要 4 万~5 万元，即使是现在虽然 PC 机和笔记本便宜了许多，但是，PC 服务器和小型机甚至大型机都是非常昂贵的。这种劳动工具的昂贵也为程序员的自信增添了几分砝码。那些劳动工具低廉的职业是不可能使从业者有自信的。

编程职业的另一特点是程序员的个人劳动、个人聪明和能干能直接反映到编程结果中，而且不受他人的影响。所以，当有成果的时候，程序员无须把成果分享给其他同行。这种独占成果的特点是程序员自信的一个很重要的原因。至于程序编得好坏，由于程序最终都能够通过测试运行，而且即使出现程序上的问题，最终都能消灭，若不消灭则程序运行就不正常，但是没有人去评价编程的水平和质量。这种无人评判、自我欣赏的结果能使程序员不自信吗？

这 4 点是外部环境对程序员的职业评价，职业评价高造成了程序员自信心也高了起来。而最令程序员自信的是其业务水平。程序员有两大类，一类是会说的，一类是会做的。会说的往往精于对计算机发展的新技术、新产品进行跟踪，说起来头头是道，仿佛无所不知、无所不晓，给人以紧跟潮流、水平高超之感。尽管他们所说的对新技术和新产品推广有益，但是这些程序员从本质上来说就是国外新技术和新产品简介的朗读者，他们以别人高水平的光环照亮自己，好像自己也很光亮一样。这些程序员往往说得多，做得少，

水平一般。会做的往往给自己的自信心以有力的支撑，他们靠自己的程序证明自己。大凡做过项目的，无论大小，随着时间延长，这种自信心越发得强，而且这种自信心根植于内心，不容改变。往往工作年限长的会看不起年限短的，项目做得多的会看不起做的少的。虽然很少有人把这种感觉表现出来，但是内心还是客观存在的。

两者的自信心的膨胀都使程序员高估了自己，一个忘记了“光说不练假把式”的谚语，一个忘记了“山外有山”这句至理名言。

程序员要放下身段，脚踏实地，不断地提高编程水平，用自己的编程成果——产品说话，以证明自己的技术价值和社会价值。

2.6 我？还是我们

放眼当今社会，“我”字当头，相当普遍、相当流行。我想、我能、我做、我行，凡事以我为先的状况处处可见。人们有了自我，有了自我实现，突显了与众不同，提升了人的人生价值，这是一件好事。同样，程序员也是“我”字当头的职业，很多程序员以“我”为中心，自信满满。对于普通程序员来说，这是一件好事；但是对于追求优秀的程序员来说，就可能是一种约束。

程序员是一个自信心很强的职业，这主要是因为程序员编写程序是一种个人劳动，而且这种劳动往往能很快产生劳动成果。一段程序编写完成后，立即运行这个程序就得到了预期的结果。这样自己操纵了计算机，成就感就产生了。这个过程一般只要 1 ~ 2 个小时，有的只要几分钟就完成了。如此短暂、频繁的成就感刺激着程序员，想不自信也不行呀。而那些作家写一部小说往往需要几个月甚至几年，这就意味着成就感刺激的频率是很低的，自信远远不如程序员来得迅速和强烈。当然，自信程度还与外界的认可有关，外部认可，自信就可能强一些；外界不认可，自信就可能低一些。

这种成就感造就了很多程序员很自信，当自己比别的程序员编的程序质量更高，用的时间更少的时候；当别人编不出来，自己能够编出来的时候；当自己知道，别人不知道的时候，这种成就感就会油然而生。

程序员这种自信自然而然地会产生很强烈的自我意识，“我想怎么样就怎么样，我认为怎么样就怎么样，我是高手”。加上程序员很少关注外界对自己评价，外界也很少对自己评价，久而久之，不加注意就会形成以“我”为主的意识和工作习惯。

说句实在话，这种自我对满足于为编程而编程的程序员是没有什么问题的，而且这种自我从某种程度上说是程序员各种痛苦的安慰剂，是程序员的心灵鸡汤。我们看到这种自信就会产生一种对程序员的尊敬。

但是，对于一个致力于成为优秀程序员的程序员来说，这种以“我”为中心的思维习惯就会影响他的发展。程序员在成长阶段要特别意识到个人在程序员这个职业中的局限性。要有意识地摆脱个人意识，从“我”过渡到“我们”。当然这个过程很痛苦，需要有战胜自我的坚强意志。

第一， 这个过程首先要求我们把自己的兴奋程度降低，而这是一般人很难做到的。这就意味在自己产生成就感的那一瞬间就要告诫自己，高兴一会儿就可以了，还有更多的事情等待自己去做。这样做可以杜绝自信虚高的情况。

第二， 编写程序时，要时刻想到程序是给别人看的，不要认为只要自己能看懂就行了。

第三， 在编程序前最好将自己的编程思路与其他程序员进行交流和沟通。沟通的目的不仅仅是要获取别人好的建议和思路，更重要的是养成一种做事的方法。即使你已经确定自己的是最好的了，这个思路是不可改变的，但是也要和其他人进行交流。

第四， 当别的程序员提出他们自己的想法的时候，应主动积极地对他们的想法进行分析，并提出看法。千万不要认为“别人的想法跟我没有关系，我没有时间去考虑别人的问题。”要把别人的想法看成自己的想法，以此来扩大自己的技术和业务范围，从而获取更多的有用信息。“三人行，必有我师。”

第五， 无论自己在项目中处于什么角色，都要关心项目的整体进展。有时候，关心项目的整体进展。有时候，关心项目的整体进展要比关心自己的进展更重要。团队意识的强弱是程序员素质高低的一个重要标志。

第六， 要强迫自己写好文档，通过写文档可以把自己的劳动成果记录下来，更重要的是这些文档会给后来者或使用者以更多的帮助和参考。

第七， 不要轻易挑战别人，不要轻易地 PK 这，PK 那。即使 PK 获胜，也说明不了什么，那只是个人的胜利，而不是“我们”的胜利。在程序员这个职业中，无论一个人怎么成功都是微不足道的。因为程序员的成功最终要反映到软件上，反映到软件产品上，反映到软件产品的市场价值上，反映到软件产品的市场占有率上，反映到软件产品对企业经营的影响上。要产生这些反映必须通过“我们”才能实现。

第八， 一定要把用户当做“我们”中的一员，耐心倾听用户的意见和建议。不要把用户看做自己的对立面，一遇到用户提出新的功能、修改需求就心怀不满。要从用户的意见中看到自己设计上和编程上的不足，从而改进程序质量。同时也要把一些技术上的限制用通俗的语言向用户解释，以获得用户的理解。

上面的“我们”只是一个团队、一个部门的“小我们”，当我们把“我们”扩大到我们的用户、同行、外部的各种企业、外部的各种人才等的时候，我们视野才会扩大，这种扩大对于程序员的素质和程序员的技术水平提高都是非常非常有益的。

目前，在中国程序员已经度过了个人英雄主义的阶段，那个阶段的程序规模比较小，一个人就能完成的程序比较多。而现在一方面程序本身的规模越来越大，需要越来越多的程序员分工协作；另一方面，程序之间的关联性越来越多，也从客观上需要程序员之间加强协作。程序员要想获得更大的发展空间以及更多的收入，那就更需要和外界发生联系，去寻找和发现更具有市场价值的软件，以这个软件来寻找更多的“我们”。

总之，程序员在成长阶段，一定要意识到“我”是束缚自己成长的最大紧箍咒。一定要舍小我而求大我，养成以“我们”的角度思考的良好习惯。这个习惯可以很好地支撑着程序员走向成熟阶段和优秀阶段。

2.7 为什么程序员的社会地位在下降

针对这个话题我已经想了很长时间，而且每次想到这个问题心情就特别沉重。今天我从广播中听到一则新闻，北京大学生就业报告出炉，失业或离职者最多的 5 大专业中，计算机科学与技术、信息管理与信息系统两个热门专业名列其中。这也印证了程序员的社会地位在下降这一说法。

大约在 5~6 年前，我就感觉到软件人员的地位和收入过了鼎盛期，开始下降了。我首先感觉到的是，相同规模的项目总价在下降，随后听说软件公司不赚钱了，再后来听说软件人员的收入大幅度下降了，去年听说新招入的大学本科生的月薪只有 1500 元。我今天询问了一般的装潢工人，他们的日收入一般在 100 元以上，好一点的更是 200~300 元。是什么原因造成程序员的地位和收入逐步走低呢？以下情况引起了我的深思。

1）程序员人数不断在增加。由于每年大学都要热招计算机专业的学生，以致程序员的存量在不断增加。物以稀为贵，人多了，供需出现了逆转，其价格必然下降。

2）编程技术进步使得编程门槛降低。由于程序设计语言的快速发展，许多复杂的功能都变成控件和库，原来很复杂的界面设计，只要拖拉拽就能实现，原来不懂的要自己去钻研，现在只要会网络搜索，下载调用就行了。所以，现在进行编程，只要会拖拉拽，只要会 Ctrl+C、Ctrl+V，只要会上网搜索，基本上就 OK 了。这样对编程者的技术要求就会降低，程序员价格必然下降。

3）企业竞争十分激烈。企业信息化成了竞争中的利器。一个企业中的每个部门和科室都会提出信息化的具体需求，而且需求必须在极短的时间内实现。这样软件人员根本没有时间去提高自己的编程技能，能把功能实现就是上上策了，根本不管代码重复，质量不高的问题。

4）一般软件企业就专注于一个行业。有的甚至只在一个企业、一个系统中进行开发，这样程序员的业务知识和程序范围就很受局限，程序很难有新意，大部分就是复制了事，在这种情况下，技能有局限性的程序员不可能有太高的价格。

5）企业信息化还在发展初期。软件数量多规模小，而且竞争残酷，价格低，反映在程序员身上的价值自然就很低。例如，一个项目总价为 10 万元，核算为 5 个人月。实际上至少要 10 个人月（竞争中必须降低核算人月数，否则无法获得该项目）。程序员在这种情况下实际价格就会比核算价格低一半。

6）开发方式专业化。程序员脱离了系统设计和项目管理工作内容，专心编程，真正成为了编程工具。一些创造性的工作变成周而复始的机械工作，而且开发只注重结果不注重过程和质量，导致技术高、质量好的程序员得不到额外的鼓励，客观上也未能鼓励优秀的程序员出现。这同样会导致程序员价格下跌。

程序员的社会地位下降，这是整个社会需要反思的问题。很多事我们不可为，但很多事我们必须面对。我们真的要认真思考这个现象，提高程序员工作的含金量，提高程序员的技能水平，还程序员内在价值的本来面目。

2.8 加薪的问题

只要工作就会有薪水。这个问题贯穿一个人的职业生涯，而且是一个基本问题、首要问题。但是薪水高低的决定权并不取决于企业的员工，而是取决于企业的管理者。这就形成了员工与管理者之间的博弈，最激烈的结果是员工跳槽而去。

新手刚参加工作，自然在薪水问题上没什么话语权，更由于求职心切，薪水要求自然就会低了一些。随着时间推移，程序员逐渐进入成长阶段，程序员经过了大量编程实践的历练，自信心有了很大的提升。这时注意力可能并不完全集中于编程，开始分散到工作环境的好坏、周围员工和领导的评价、自己的收入与公司内其他人或公司外其他人的比较等 。这种注意力分散说明程序员步入了一个新的阶段，这些注意力都会有利于程序员成熟起来。

对程序员来说，这些注意力中最纠结的就是加薪问题。其主要原因是：第一，薪水是程序员工作应得的报酬，是必须考虑的问题。第二，由于工作了一段时间后，自己的工作能力有了提高，自我价值增大，需要对这种变化获取价值上的补偿，否则，自己工作再努力、水平再提高，薪水还是没有增加，人可能就会失去工作的积极性。第三，不好意思开口。很多人在内心想要加薪，但是却不表现出来。很多人从表面上表现出对金钱的远离，以示自己不是唯利是图的小人，在想要加薪时也是欲言还休。第四，不知道如何表达。当程序员下定决心要求加薪的时候，却不知道如何正确表达。最怕说了薪水还是没有增加，却给领导留下一个不好的印象，影响今后工作的开展。

作者认为加薪问题是一个相对复杂的问题，每个企业、领导、员工的情况都不一样，这都会给加薪问题带来不同的结果。很难用一个通用的表述来解决。但是，我们也总结了一些要注意的问题，希望可以为程序员提供一些参考。

1.加薪的时机

加薪并不是任何时候都可以提出的，主要还是看时机。一般在年初、年终或年终奖发放后，项目终了，程序员进企业的整数年时间为最好。这很大程度上要看企业本身加薪时间的一般规律。提出加薪的时机不对，会让企业领导们太“重视”你的加薪要求，这样加薪的可能性不大，还不如把握时机，顺势而为。

2.加薪的理由

加薪的理由特别重要，因此程序员要找出足够充分的理由。如果企业是按照程序员工作年限或者资历来确定是否加薪的话，一般情况下，除非自己有特别的理由，否则只能等到满足加薪条件再说。

程序员加薪最重要的理由是为企业创造了巨大的效益，尤其是直接效益更佳。这个效益（扣除了企业成本之后）要远远大于程序员的收入才行。第一，在一般情况下，由于存在竞争，如果软件的利润很低、企业的效益很差，加薪的理由就显得不怎么充分；如果企业效益好，并和自己有很大的关系，这个时候提出加薪理由就很充分了。第二，软件项目开发完成和销售成功主要是自己的原因促成的。自己是企业或团队的骨干，比其他同伴更加努力、出色。 当然，是否是骨干以及是否出色，这个问题程序员不要自己给自己评价，要听听其他人的意见，这样可能更客观一些。第三，程序员在不同的项目中，而不是一个项目中表现出色，这也是加薪的充分理由。这表明程序员可以在未来为企业创造更多的价值。

3.加薪的表达

程序员有了加薪的理由之后，如何表达加薪的要求也很重要。

第一，程序员在内心要树立加薪不仅是为了自己，也是为了企业的思想。不要犹豫不决，要坚定加薪的信念，这样就不会在这个问题上给领导以欲言又止的印象。这种犹豫会影响到你的加薪理由的充分性、必要性。

第二，要找到表达的对象。不同的企业决定员工是否加薪的领导是不一样的。有的是主管起关键作用，有的是部门经理，有的是老板本人。因此，你要去了解你的前辈们加薪时找的是什么人，找错了对象，不仅可能加不了薪，还可能带来负面影响。

第三，表达的方式，可以是书面形式，也可以是面谈，或者是两者结合。根据一般的经验，当面表达可能更好一些，因为面谈能够看到提出加薪请求后领导者的反应。

第四，表达的时候，一定要表明自己非常注重企业的发展，表明是企业发展给自己带来了进步。这些会给领导一个良好的心理暗示。然后很客观地表达自己加薪的请求，请领导考虑。表达时不要过分说明自己的成果，因为这些成果领导应该是知道的。另外更不要夸大自己的成果，把别人的功劳算在自己的身上，在这点上领导也是心知肚明的。

第五，千万不要用威胁的语气提出加薪要求。即使这个企业离开你立即就会倒闭也不要这样去做。有的老板可能会被你威胁成功一次，但是，绝不可能会被你威胁成功第二次的。因为老板可以忍气吞声地答应加薪，但随后必然会找人替代你。

第六，倾听领导者的意见和建议。一般领导者在听到加薪请求之后会表示给予考虑的，但是很多情况下会讲出企业面临各种资金困难的问题。这时候程序员要认真倾听，表示理解企业的困难。如果不是真的很困难，那就再一次表明加薪的请求，让领导者了解你加薪的决心。如果领导者当面拒绝，也不要当面争吵，最好说希望领导能考虑和了解自己的请求。

第七，加薪的请求一次足矣，不要多次找领导交涉。2 次、3 次、4 次，多少次的结果都是一样的。领导定下来的事一般是很难改变的。所以，只要一次，不要多次。多次不仅无用，反而会给领导留下不好的印象。

4.加薪的结果

提出加薪的结果一种是成功了，另一种则是失败了。成功了皆大欢喜，自己要更加努力工作，不待细说。而失败了，千万不要耿耿于怀，要从客观上找找自己的原因，找找企业的原因。把原因找到后，也许自己会对自己有一个全新的认识。

程序员面对失败不要气馁，你只需对自己说，“我已经努力过了”，“加不加是企业的事”，“明年再争取”。这样程序员就可以从加薪的苦恼中走出来，把注意力重新放在工作和技术提高上，为下一次加薪做好充分准备。

说到底，加薪是要靠自己做出成绩，要靠自己争取的。其中做出成绩是正道，争取方式是技巧。两者有机结合就可能达到为自己加薪的目的。如果一切做得很好，但是依然不能加薪，那可能就需要考虑换个环境了。

2.9 门门通还是精通一门

我一直希望计算机只有一种程序设计语言，哪怕是高级程序语言只有一种也行，这样我们就不必为学这种或那种语言而烦恼了；或者我们学习语言不费事，有一种学一种也行，也没有学不全语言的烦恼了。但是，两者都只是一种希望，不知道未来能否实现。

程序员进入了成长期，必然和程序语言打交道，这个时候学语言一定是和具体项目、应用、客户相关。学习语言已经不是一种个人爱好的选择，而是一种工作的选择。就如同扫雪时不仅需要扫帚，而且需要铁锹一样。

有的程序员在单位只负责一个系统的维护和升级工作，这可能只要掌握一种语言就足够了。有的程序员所在的软件公司会接来各种各样的项目，而各个项目应客户要求采用不同的语言开发，有的项目需要用C#开发，有的项目需要用 Java 开发，有的项目需要用 PHP 开发。当公司人手不够时，程序员必须学习多种语言才能满足工作的需要。因此，程序员因工作需要学习语言的数量是不一样的。

程序员不但因为工作需要要学习语言，还会为未来职业的发展而考虑学习语言。未来职业发展需要何种语言，需要掌握多少语言，这一切都是一个未知数。所以，大家潜意识里有一种多多益善的想法，恨不得有一门学一门，一个都不漏过。因此，程序员心理负担极为沉重，无论是在工作时，还是休息时，每每都在想“门门通还是精通一门”这个话题。

当然有一些程序员还没有改变新手时期对语言的认识，还是以个人的兴趣去选择语言和学习语言。就这一点而言，站在实用主义角度，我自己反对因个人兴趣而学习语言，除非个人兴趣和工作需要以及未来职业规划相一致，这种个人兴趣才是值得提倡的。反之，为学语言而学语言且该语言与现有工作以及未来职业无关，仅仅是因为这种语言自己没有学过，或语言很流行就去学，那是没有必要的。因为中国不缺那种不实用的语言，只缺实用的软件。

从我自己的成长经历来看，一个程序员一般至少要精通二门语言。这两门语言一个应该是面向过程的语言，一个是面向对象的语言。至于了解和掌握多少其他的语言，那就要根据工作需要和个人职业规划来定了。一般 3~5 门也就足够了。这也就是说，一个程序员一般要掌握五六种程序设计语言，其中两门必须精通，其他 3~4 门只要求一般掌握，可以利用它们进行编程就行了。

我想对成长期的程序员在语言选择方面、学习方面、实用方面谈一谈自己的看法，以供参考。

1.精通一门到什么程度

虽然很多程序员也知道要精通一门语言，但是他们并不清楚精通一门语言到底有什么标志。有的人会按照程序员掌握语言的时间来确认对语言的掌握程度。例如，如果一个人使用某种语言编了 3 年的程序，他自己或其他人就会认为他对这门语言很精通了。有的人会根据自己对这门语言的驾驭程度来说明自己对这种语言的掌握程度。例如，有的人认为自己编起程序来很顺手，不用查手册，说写就写，很熟练，几乎什么程序都能编出来，以此认为自己对这种语言很精通了。

编程时间的长短和编程的熟练程度仅仅是精通语言表现的一个方面。更重要的是程序员要掌握这门语言的适用范围、整体架构、语法规则、功能分类等基础理论方面的知识，并能利用这方面的知识，以最科学的方法解决现实中各种项目的各个问题。

通俗地说，如果你能当这个语言的老师（反映出对语言的理论和语言架构的掌握程度），又能熟练地用这门语言解决各种问题（反映出对语言的使用能力），那你就可以说你精通这门语言了。

如果有人让你介绍一下某种语言，你不会讲、讲不全、讲不透，说明你对这种语言的理论方面和架构方面的知识掌握得远远不够。有人问你一些这种语言的一些常见问题、一些常用的技巧、一些常见的错误，而你都无法解决和解释，说明你对这种语言的使用能力还不够充分，掌握还很肤浅。这些都说明你没有精通这门语言。

2.最好是精通两门

我感觉精通一门语言还是不够的。程序设计语言一般分两大类，一类是面向过程的语言，一类是面向对象的语言。如果要很扎实地在程序员这个职业中耕耘，最好精通两门语言，一门是面向过程的，另一门是面向对象的。如果怕时间和精力不够，我建议一定要精通一门面向对象的语言，因为在面向对象的语言中也包含了面向过程的编程内容。

3.其他 3~4 门要怎么才算掌握

现实中我们常常会找一本书去学习一种语言，也会用这种语言编写几段小程序，那这样究竟算不算对这种语言已经掌握了呢？怎样才算已经掌握了呢？我认为只要能用这种语言开发一个项目，开发中没有太多问题，那就可以算掌握了。当然，开发的时候，断断续续，走一步、查一步、问一步，问题如同连环绊脚石，这样则不能算掌握了。

4.不要太急于赶潮流

一些程序员往往对技术潮流关注过度，明明自己工作上用的是 C 语言，自己做的是后台维护的工作，但是，听说现在流行 C#、Java，就老想有时间把 C#、Java 学学，不学可能就落伍了。

说实在话，如果不急于应用，最好等这门语言成熟以后，再决定学习也不迟。从各种语言的发展过程来看，有些语言的生命周期也只有短短几年时间而已。

5.语言是互通的

其实各种语言在本质上是相同的，它们有太多的共性，虽然有些个性，但是在实际中个性功能很少用到，用到时现学也不迟。因此，我们可以在精通一门语言的基础上，通过这门语言的结构去学习另外一门语言的结构，对这门的示例采用另外一门语言来编写，这样有对照、有比较地学进步应该会很快的。找出相同点，这是学习多种语言的技巧。

6.语言的无知与有知

现实中经常会有人问你，你知道什么什么语言吗？如果你不知道，千万不要觉得自己无知而羞愧。因为人的价值不在于你知不知道，而在于你知道后做了些什么。常言道“光说不练假把式”，为知道而知道，只会浪费实用的工作时间，影响自己实战能力的提高。在很多情况下，有知和无知可能会发生逆转。有意的无知?扔幸獾挠兄.硐值酶　哟厦鳖?

当然，我们可以关注一些语言方面发展的新闻，就如同我们每天关注国内新闻、国际新闻一样。能给自己留下印象的那就成功了，不能留下印象的那就算没看，传播新闻那可是媒体的事，我们权当休闲就行了。

特别需要提醒的是，对程序员来说，实用主义是比较好的，把现有的实用的语言掌握好，研究深入一些比什么都重要。人们可能都会有“这山望着那山高”的想法，但是只有站在这山上时，才能望到那山。