C 程序设计从零开始之何谓 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/133/2021_2022_C___E7_A8_8B_E5_BA_8F_E8_c97_133815.htm 程序，即过程的顺序，准确地说应该是顺序排列的多个过程，其是方法的描述。比如吃菜，先用筷子夹起菜，再用筷子将菜送入嘴中，最后咀嚼并吞下。其中的夹、送、咀嚼和吞下就被称作命令，而菜则是资源，其状态（如形状、位置等）随着命令的执行而不断发生变化。上面就是吃菜这个方法的描述，也就是吃菜的程序。 任何方法都是为了改变某些资源的状态而存在，因此任何方法的描述，也就是程序，也都一定有命令这个东西以及其所作用的资源。命令是由程序的执行者来实现的，比如上面的吃菜，其中的夹、送等都是由吃菜的人来实现的，而资源则一定是执行者可以改变的东西，而命令只是告诉执行者如何改变而已。 电脑程序和上面一样，是方法的描述，而这些方法就是人期望电脑能做的事（注意不是电脑要做的事，这经常一直混淆着许多人），当人需要做这些事时，人再给出某些资源以期电脑能对其做正确的改变。如计算圆周率的程序，其只是方法的描述，本身是不能发生任何效用的，直到它被执行，人为给定它一块内存（关于内存，请参考《C 从零开始（三）》），告诉它计算结果的精度及计算结果的存放位置后，其才改变人为给定的这块内存的状态以表现出计算结果。 因此，对于电脑程序，命令就是CPU的指令，而执行者也就由于是CPU的指令而必须是CPU了，而最后的资源则就是CPU可以改变其状态的内存（当然不止，如端口等，不过一般应用程序都大量使用内存罢了）。所以，电脑程序

就是电脑如何改变给定资源（一般是内存，也可以是其他硬件资源）的描述，注意是描述，本身没有任何意义，除非被执行。 何谓编程 编程就是编写程序，即制订方法。为什么要有方法？方法是为了说明。而之所以要有说明就有很多原因了，但电脑编程的根本原因是因为语言不同，且不仅不同，连概念都不相通。 人类的语言五花八门，但都可以通过翻译得到正解，因为人类生存在同一个四维物理空间中，具有相同或类似的感知。而电脑程序执行时的CPU所能感受到的空间和物理空间严重不同，所以是不可能将电脑程序翻译成人类语言的描述的。这很重要，其导致了大部分程序员编写出的拙劣代码，因为人想的和电脑想的没有共性，所以他们在编写程序时就随机地无目的地编写，进而导致了拙劣却可以执行的代码。电脑的语言就是CPU的指令，因为CPU就这一个感知途径（准确地说还有内存定位、中断响应等感知途径），不像人类还能有肢体语言，所以电脑编程就是将人类语言书写的方法翻译成相应的电脑语言，是一个翻译过程。这完全不同于一般的翻译，由于前面的红字，所以是不可能翻译的。 既然不可能翻译，那电脑编程到底是干甚？考虑一个木匠，我是客人。我对木匠说我要一把摇椅，躺着很舒服的那种。然后木匠开始刨木头，按照一个特殊的曲线制作摇椅下面的曲木以保证我摇的时候重心始终不变以感觉很舒服。这里我编了个简单的程序，只有一条指令做一把摇着很舒服的摇椅。而木匠则将我的程序翻译成了刨木头、设计特定的曲木等一系列我看不懂的程序。之所以会这样，在这里就是因为我生活的空间和木工（是木工工艺，不是木匠）没有共性。这里木匠就相当于电脑程序员兼CPU（因为最后由木匠

来制作摇椅），而木匠的手艺就是CPU的指令定义，而木匠就将我的程序翻译成了木工的一些规程，由木匠通过其手艺来实现这些规程，也就是执行程序。上面由于我生活的空间和木工（指木工工艺，不是工人）没有共性，所以是不可能翻译的，但上面翻译成功了，实际是没有翻译的。在木工眼中，那个摇椅只是一些直木和曲木的拼接而已，因为木工空间中根本没有摇椅的概念，只是我要把那堆木头当作摇椅，进而使用。如果我把那堆木头当作凶器，则它就是凶器，不是什么摇椅了。“ 废话加荒谬加放屁！”，也许你会这么大叫，但电脑编程就是这么一回事。CPU只能感知指令和改变内存的状态（不考虑其他的硬件资源及响应），如果我们编写了一个计算圆周率的程序，给出了一块内存，并执行，完成后就看见电脑的屏幕显示正确的结果。但一定注意，这里电脑实际只是将一些内存的数值复制、加减、乘除而已，电脑并不知道那是圆周率，而如果执行程序的人不把它说成是圆周率那么那个结果也就不是圆周率了，可能是一个随机数或其他什么的，只是运气极好地和圆周率惊人地相似。上面的东西我将其称为语义，即语言的意义，其不仅仅可应用在电脑编程方面，实际上许多技术，如机械、电子、数学等都有自己的语言，而那些设计师则负责将客户的简单程序翻译成相应语言描述的程序。作为一个程序员是极其有必要了解到语义的重要性的（我在我的另一篇文章《语义的需要》中对代码级的语义做过较详细的阐述，有兴趣可以参考之），在后续的文章中我还将提到语义以及其对编程的影响，如果你还没有理解编程是什么意思，随着后续文章的阅读应该能够越来越明了。 电脑编程的基础知识编译器和连接器 我从没

见过（不过应该有）任何一本C 教材有讲过何谓编译器（Compiler）及连接器（Linker）（倒是在很老的C教材中见过），现在都通过一个类似VC这样的编程环境隐藏了大量东西，将这些封装起来。在此，对它们的理解是非常重要的，本系列后面将大量运用到这两个词汇，其决定了能否理解如声明、定义、外部变量、头文件等非常重要的关键。 前面已经说明了电脑编程就是一个“翻译”过程，要把用户的程序翻译成CPU指令，其实也就是机器代码。所谓的机器代码就是用CPU指令书写的程序，被称作低级语言。而程序员的工作就是编写出机器代码。由于机器代码完全是一些数字组成（CPU感知的一切都是数字，即使是指令，也只是1代表加法、2代表减法这一类的数字和工作的映射），人要记住1是代表加法、2是代表减法将比较困难，并且还要记住第3块内存中放的是圆周率，而第4块内存中放的是有效位数。所以发明了汇编语言，用一些符号表示加法而不再用1了，如用ADD表示加法等。由于使用了汇编语言，人更容易记住了，但是电脑无法理解（其只知道1是加法，不知道ADD是加法，因为电脑只能看见数字），所以必须有个东西将汇编代码翻译成机器代码，也就是所谓的编译器。即编译器是将一种语言翻译成另一种语言的程序。 即使使用了汇编语言，但由于其几乎只是将CPU指令中的数字映射成符号以帮助记忆而已，还是使用的电脑的思考方式进行思考的，不够接近人类的思考习惯，故而出现了纷繁复杂的各种电脑编程语言，如：PASCAL、BASIC、C等，其被称作高级语言，因为比较接近人的思考模式（尤其C 的类的概念的推出），而汇编语言则被称作低级语言（C曾被称作高级的低级语言），因为它们不是很符

合人类的思考模式，人类书写起来比较困难。由于CPU同样不认识这些PASCAL、BASIC等语言定义的符号，所以也同样必须有一个编译器把这些语言编写的代码转成机器代码。对于这里将要讲到的C 语言，则是C 语言编译器（以后的编译器均指C 语言编译器）。 因此，这里所谓的编译器就是将我们书写的C 源代码转换成机器代码。由于编译器执行一个转换过程，所以其可以对我们编写的代码进行一些优化，也就是说其相当于是一个CPU指令程序员，将我们提供的程序翻译成机器代码，不过它的工作要简单一些了，因为从人类的思考方式转成电脑的思考方式这一过程已经由程序员完成了，而编译器只是进行翻译罢了（最多进行一些优化）。 还有一种编译器被称作翻译器（Translator），其和编译器的区别就是其是动态的而编译器是静态的。如前面的BASIC的编译器在早期版本就被称为翻译器，因为其是在运行时期即时进行翻译工作的，而不像编译器一次性将所有代码翻成机器代码。对于这里的“ 动态 ”、“ 静态 ”和“ 运行时期 ”等名词，不用刻意去理解它，随着后续文章的阅读就会了解了。 编译器把编译后（即翻译好的）的代码以一定格式（对于VC，就是COFF通用对象文件格式，扩展名为.obj）存放在文件中，然后再由连接器将编译好的机器代码按一定格式（在Windows操作系统下就是Portable Executable File FormatPE文件格式）存储在文件中，以便以后操作系统执行程序时能按照那个格式找到应该执行的第一条指令或其他东西，如资源等。至于为什么中间还要加一个连接器以及其它细节，在后续文章中将会进一步说明。 也许你还不能了解到上面两个概念的重要性，但在后续的文章中，你将会发现它们是如此的

重要以至于完全有必要在这唠叨一番。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com