# 跟vczh看实例学编译原理——一：Tinymoe的设计哲学

自从《序》胡扯了快一个月之后，终于迎来了正片。之所以系列文章叫《看实例学编译原理》，是因为整个系列会通过带大家一步一步实现Tinymoe的过程，来介绍编译原理的一些知识点。

但是第一个系列还没到开始处理Tinymoe源代码的时候，首先的跟大家讲一讲我设计Tinymoe的故事。为什么这种东西要等到现在才讲呢，因为之前没有文档，将了也是白讲啊。Tinymoe在github的wiki分为两部分，一部分是介绍语法的，另一部分是介绍一个最小的标准库是如何实现出来的，地址在 <https://github.com/vczh/tinymoe/wiki> 不带问号的那些都是写完了的。

## 系列文章的目标

在介绍Tinymoe之前，先说一下这个系列文章的目标。Ideally，只要一个人看完了这个系列，他就可以在下面这些地方得到**入门**：

1. 词法分析
2. 语法分析
3. 语义分析
4. 符号表
5. 全文CPS变换
6. 编译生成高效的其他语言的代码
7. 编译生成自己的指令集
8. 带GC的虚拟机
9. 类型推导（intersection type，union type，concept mapping）
10. 跨过程分析（inter-procedural analysis）

## Tinymoe设计的目标

处理Tinymoe需要的知识点多，用于编译原理教学

容易嵌入其他语言的程序，可以调用宿主程序提供的功能

语法优化于描述复杂的逻辑，而不是优化与复杂的数据结构和算法（虽然也可以）

动态类型语言

## Tinymoe的设计哲学

小规模的语言核心+大规模的标准库

扩展后的东西跟原生的东西外观一致

## Tinymoe的实现难点

需要三次Parsing操作，动态生成文法，但是仍然按照手写递归下降的方法来写

CPS变换后没有特殊处理直接执行容易导致callstack积累的没用的东西过多

动态类型语言的类型推导

跨过程分析和对程序的控制流的化简（譬如continuation转状态机等）