1. **阅读笔记**
2. **优化**
3. 优化的好处：优化能产生更加高效紧凑的目标代码，能有效减小目标程序的大小。
4. llvm本身在生成IR时就对代码进行了优化，包括整数收缩等优化，正常情况下整数收缩的实现是在AST表示中发现并优化的。然而llvm生成IR代码时进行的优化只是简单的优化，一些复杂的优化技术需要通过添加llvm包含的各种优化。
5. llvm支持各种小模块的优化并且能选择各模块组合起来对代码整体进行优化。实现方法：生成各个模块并生成FunctionPassManager对这些模块进行管理，在各抽象语法树生成代码中使用FunctionPassManager对模块对应的代码进行优化。
6. **即时编译**

1.对输入的表达式进行计算的实现：类型为KaleidoscopeJIT的全局变量TheJIT并在main函数中初始化，此后设置TheJIT数据布局，添加包含表达式的module到TheJIT，调用findSymbol（）获得表达式生成的目标代码，调用getAddress（）获取函数指针，最后调用函数指针指向的函数获得计算结果。实现表达式计算后可以计算得到表达式和输入的函数的结果，但是再次调用函数时由于包含函数计算的模块被释放而无法再次进行计算。

2.函数进行再次调用解决方法：函数原型存储到全局变量FunctionProtos中，调用函数时通过getFunction（）获取函数。