小组信息：

2016302580100 杨欢

2016302580093 黄山

2016302580097 林航

万花筒语法分析器设计文档

1. 错误信息打印：

Value \*LogErrorV(const char \*str); 在函数中调用解析器中的LogError()

1. 表达式代码生成：
2. 数字值代码生成

Value \*NumberExprAST::codegen()；

在LLVM IR中，数字常量用ConstantFP类表示 ，这个函数返回一个ConstantFP

1. 变量代码生成：

Value \*VariableExprAST::codegen();

检查指定的名称是否在映射中（如果没有，则引用未知变量）并返回其值

1. 二元表达式代码生成：

Value \*BinaryExprAST::codegen();

递归的选出操作符左右两侧操作数，然后计算二元表达式的结果。

然后创建指令，用IRBuilder在指定的地方插入新创建的指令。

1. 函数调用代码生成：

Value \*CallExprAST::codegen();

首先在LLVM模块的符号表中执行函数名称查找，因为LLVM模块通过为每个函数指定与用户指定的名称相同的名称，使得我们可以使用LLVM符号表来解析我们的函数名称。

1. 函数代码生成：
2. 原型代码生成：

Function \*PrototypeExprAST::codegen();

“原型”实际上是关于函数的外部接口

由于VSL中的所有函数参数都是int类型，因此首先创建一个LLVM int类型的向量

它通过使用Functiontype::get方法创建一个函数类型，返回一个int

创建与Prototype相对应的IR功能，表示要使用的类型，链接和名称，以及要插入的模块

1. 函数代码生成：

Function \*FunctionExprAST::codegen();

在符号表中搜索此函数的现有版本，以防已经使用'extern'语句创建了一个。如果不存在先前版本，我们将从Prototype中编译一个。

创建一个BasicBlock，向其中插入函数指针，再将函数参数添加到NamedValues映射

构建函数后，我们调用由LLVM提供的verifyFunction，此函数对生成的代码执行各种一致性检查，以确定我们的编译器是否正在执行所有操作