GitHub网址：

https://github.com/Interpretion/Interpreter.git

扩展语言&控制流程

这一部分主要是将先前的内容进行扩展，从而具有if/then/else表达式和简单“for”循环，增加控制流，可以使代码有条件分支，性能可以进一步提升。

1. if/then/else

要让编译器支持if/then/else，则需要向lexer, parser, AST，LLVM code emitter添加支持。if/then/else表达式需要像其他表达式一样有返回值，所以我们在评估条件后，根据条件的解决方式返回then或else值。

①Lexer拓展：首先为token添加枚举值，然后就可以进一步识别lexer的新关键词了。

②AST拓展：添加新的AST节点，而AST节点仅具有指向各种子表达式的指针。

③Parser拓展：在进行了lexer标记后，创建了AST节点，然后定义新的解析函数作为一个主要表达式连接起来。

④LLVM IR：在完成了上面的工作后，就添加LLVM代码生成支持。此间如果要可视化控制流程图，可以使用LLVM提供的“OPT”工具，而LLVM本身具有多种可以画图形的优良特性。要良好的执行if/then/else，需要使用Phi操作，该操作可以获取对应块控件的值，它的操作对应相对输入的控制块的值。

⑤代码生成：为了生成代码，为IfExprAST实现了codegen方法，首先列出该条件的表达式，然后将值与零进行比较，得到bool值的真值，将此应用到if/then/else语句相关的块中。

2、“for”循环表达式

和if/then/else类似，需要添加对for循环的支持。

①Lexer拓展：首先为token添加枚举值，然后就可以进一步识别lexer的新关键词了。

②AST拓展：添加新的AST节点，来捕获节点中的变量名和组成表达式。

③Parser拓展：在进行了lexer标记后，创建了AST节点，然后定义新的解析函数作为一个主要表达式连接起来。这里需要处理可选的步长值，而这可以通过检查第二个逗号是否存在来处理。

④LLVM IR： 这个循环依旧是包括了Phi节点，几个表达式和一些基本块。

⑤代码生成：codegen方法的第一部分输出循环值的开始表达式，然后为循环体的开始设置LLVM基本块，此时要创建Phi节点来记住进行循环的块，然后就可以进行循环了。首先在插入点为回路创建Phi节点，完成主体后设置退出循环的值来确定循环是否退出，随后创建退出后的块，在退出值的基础上创建一个条件分支，在再次执行循环和退出之间抉择。