Lab4 中间代码生成1 - 实验报告

本次实验的内容是中间代码生成，具体来说包括 main 函数和表达式的翻译，其中 main 函数中目前只有一条 return 语句，且 return 的值为一个常量表达式。

1. 设计考虑

采用 visitor 模式来完成对 main 函数和表达式的翻译。

我们首先需要考虑的问题是哪些方法应该被覆盖。

需要翻译的表达式种类有:

- 单目运算符：+, -, !
- 双目运算符: +, -, *, /, %
- 单个常量

需要翻译的语句:

- return 语句

因此, 我们需要覆盖的方法就是涉及以上表达式和语句的 visit 方法，此外由于需要翻译函数，我们需要覆盖函数定义这条规则对应的 visit 方法。

由于我们需要进行表达式求值，而表达式具有递归的结构，因此在求一个表达式的值的时候常常需要子表达式的值，因此我们需要 visit 方法返回值类型为 LLVMValueRef.

在遍历的过程中我们需要知道当前在哪个函数里面，此处我采用的做法是用一个栈存储当前正在处理的函数，在进入函数体之前压栈，处理完函数体之后出栈。

2. 实现细节

函数的生成和 return 语句的生成是简单的，此处就不详述。

对于表达式的翻译，采取的方式是先递归计算子表达式的值，再根据运算符调用 LLVM 的 api 进行运算。