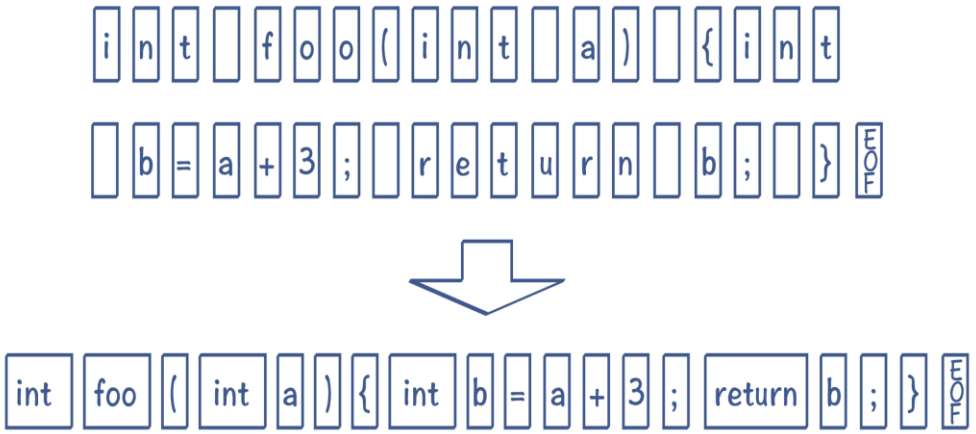
基于C/Arm 结构的编译器

代码框架：LeeOrange-is-me/2024NKUCS-Compilers-Lab (github.com)

词法分析将字符转化为token

输入：经过预处理的程序
过程： 逐一扫描，识别token。
输出：(种类，值) 二元组形式的符号表



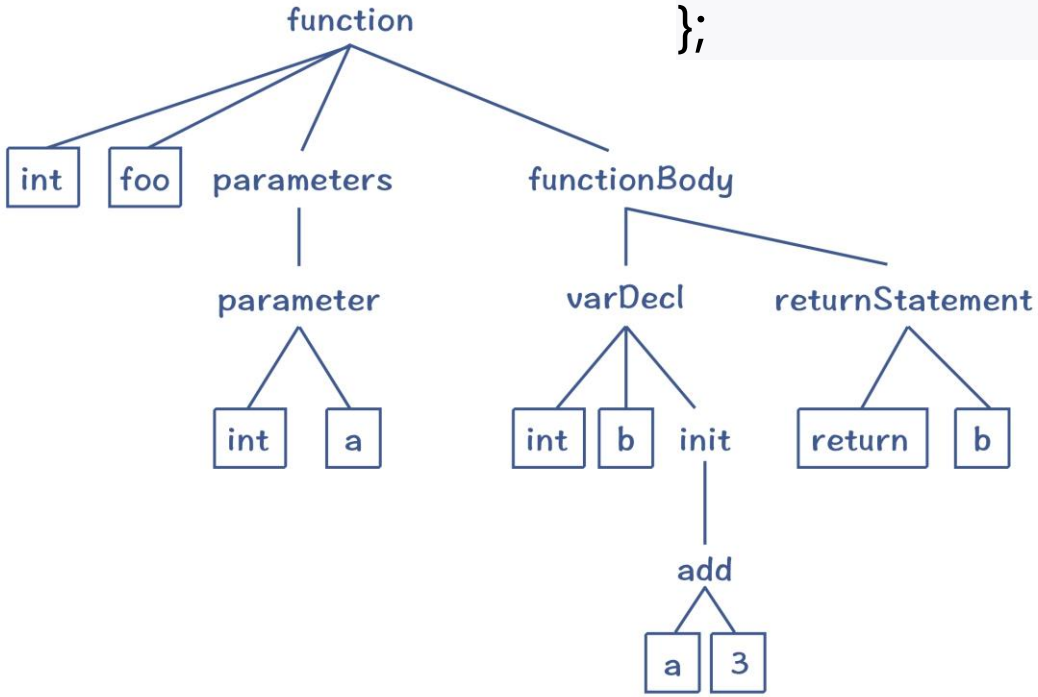
具体工作：
使用flex&Lex来编译定义 编写的lexer.l文件
1. 定义token的通配符
2. 定义每匹配到一次这样的token要做什么

代码量300-400行

工具：
Lex 是用来辅助生成词法分析程序的，它可以将大量重复性的工作自动计算，通过相对简单的代码可以生成词法分析程序

语法分析理解语句结构

```
int foo(int a){
    int b=a+3;
    return b
};
```



输入：经过词法分析的符号表

过程：将单词符号序列分解成语法单位，如“表达式”，“语句”，“程序”等。

输出：语法树

工具：Yacc&Bison, yacc 生成的编译器主要是用 C 语言写成的语法分析器（parser），一般需要与 Lex 生成的词法分析器一起使用，两部分生成的 C 程序一并编译。

主要工作：

1. 编写c，定义各种节点
2. 编写parser.y，利用上下文无关文法识别对应的语句,识别语法树的节点,并根据语法规则将各个节点连结起来构造语法树

代码量400-500行

语义分析是结合上下文语法树对代码进行理解，对代码进行检查

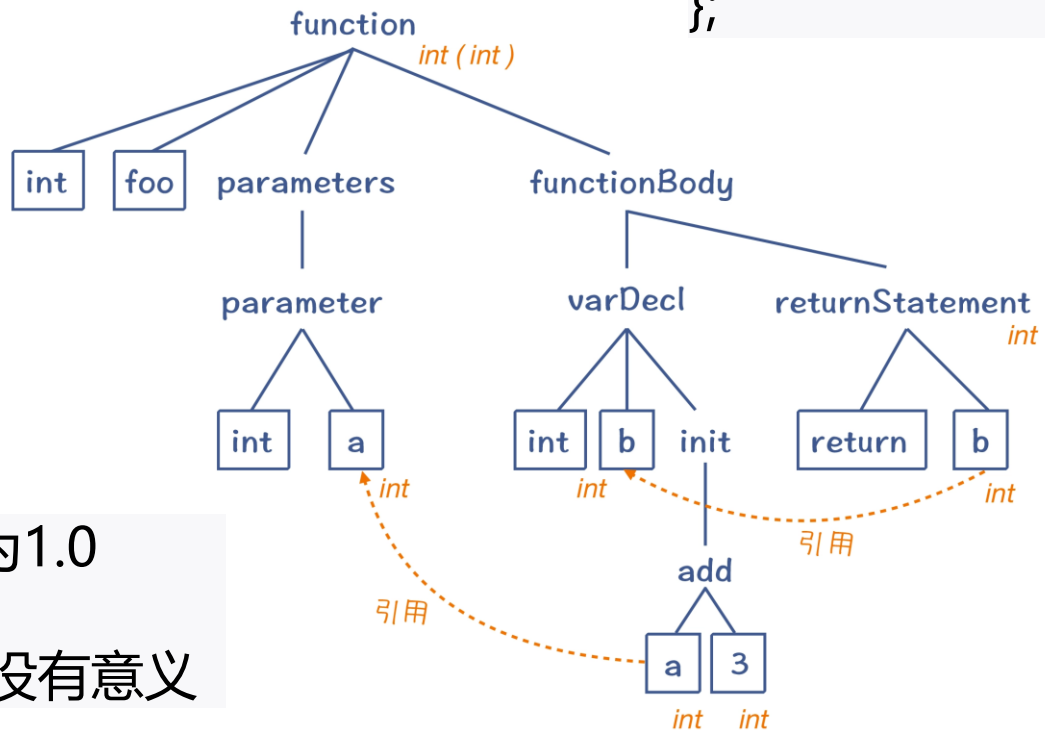
主要任务：通过对所构造的语法树进行分析，检查类型

输入：经过语法分析的语法树

过程：类型转换、类型检查

输出：标识类型了的语法树

```
int foo(int a){
    int b=a+3;
    return b
};
```

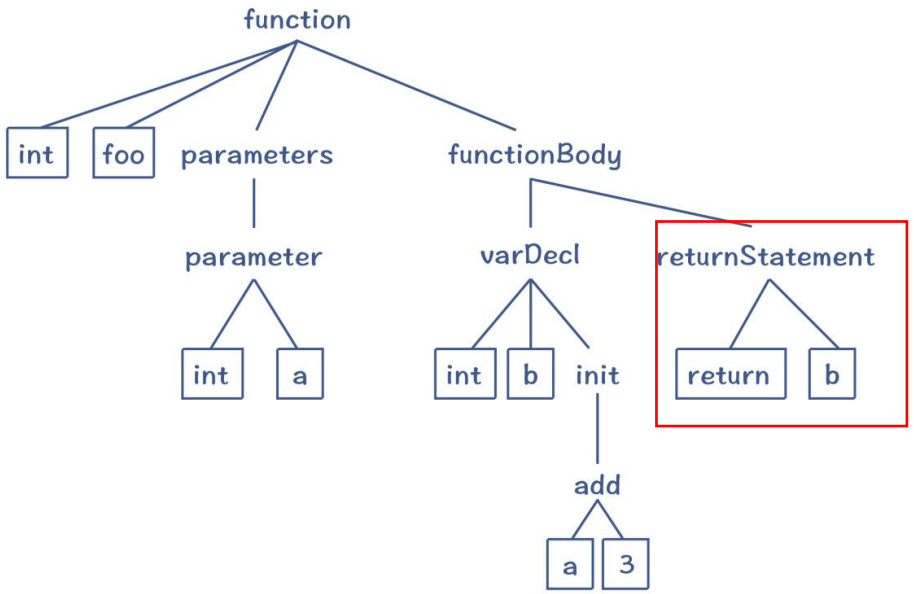



```
float a= 1+1.2; //类型转换，将1隐式转化为1.0
int *b,*c;
Int *d =b+c ;  //类型检查，两个指针相加没有意义
```

中间代码

将二叉语法树转化为形式简单的中间代码

```
int foo(int a){   // define i32 @foo(i32 %a)
    int b=a+3;    //%add = add i32 %a, 3
                  //%b = alloca i32
                  //store i32 %add, i32* %b
    return b      //%b_val = load i32, i32* %b
                  //ret i32 %b_val
};
```



- 独立于机器，复杂性界于源语言和目标语言之间的特点，能够使编译程序的结构
- 逻辑上更为简单明确，便于进行与机器无关的代码优化工作，更易于移植。

代码优化

基本要求：

1）消除不可达基本块；主要涉及控制流图的分析和遍历，以及标记和删除不可达的基本块，大概估计100行

2) 实现load和store在同一基本块时的mem2reg，删除没有load的store语句
大概估计250行

进阶要求：

1)完整的SSA形式转换（完整的mem2reg）（2分）

2)自由发挥（3分）：完成mem2reg后，在如下优化中任选其二完成或者自由发挥（难度需要大于等于下列优化）（多完成酌情额外加分）

a)稀疏条件常量传播
b)标量运算的循环不变量外提
c)标量运算的公共子表达式删除
d)函数内联
e)激进的死代码消除（基于控制依赖图，需要删除dead-loop)

代码量：350 +700 行

生成最终汇编代码(arm)

```
int foo(int a){
    int b=a+3;
    return b
};
```

```
.global foo

foo:
        push {lr} // 保存链接寄存器
        add r0, r0, #3 // r0 是参数 a，计算 a + 3，并将结果存储在 r0 中
        pop {lr} // 恢复链接寄存器
        bx lr // 返回调用者
```

代码量：1200+行

生成最终汇编代码(arm)

优势

- 历届学长学姐可能已经积累了**大量的经验和代码资源**
- 相比于RISC-V框架，ARM框架可能**更容易扩展到完整的C语言**处理，因为ARM架构在设计时考虑了更广泛的应用场景

劣势

- ARM架构，尤其是ARMv7，包含了一些**复杂特性**，如全局变量处理。主要体现在其多样化的指令集模式、复杂的寄存器管理、多样的寻址模式、**严格的调用约定**和丰富的扩展集。这些特性使得ARM汇编代码的生成过程更为复杂，需要编译器处理更多的边界情况和特殊要求
- ARM后端的框架可能需要同学们**自行探索部分优化内容**，增加了学习和开发的工作量。相比之下，RISC-V框架已经对这部分内容提供了支持。
- 需要同学们**探索ARM的其他指令**以完成作业要求