编译原理 MiniDecaf 编译器实验
Step 9 实验报告

李祥泽
2018011331
lixiangz18@mails.tsinghua.edu.cn

实验内容

实现函数调用

大致按实验指导书的指导完成, 为了方便和语法正确做了修改.

具体地说, 有以下两点:

1. 实验指导书中的语法将参数列表 (形参实参皆然) 单独提出作为一个语法单元, 这对分析函数的参数类型, 数量等造成了麻烦 (必须借助中间变量来传递参数信息, 而且该信息还是可变的). 我参照助教的实现, 将参数列表直接合并在函数声明, 定义和调用语句的语法中, 直接在一个 visitor 函数中进行处理.
2. 实验指导书中的函数体是一个 `compoundStatement`. 但是, 在我的实现中, 每次进入这样一个语法单元时, 都将开启一级新的作用域. 这使得在函数体内定义与参数同名的变量时不报错. 为此, 我参照助教的实现, 将这里改为 `'{' blockItem* '}'` (就是 `compoundStatement` 的右部), 也即忽略了这一对大括号. 而作用域交由进入函数时创建.

思考题

MiniDecaf 的函数调用时参数求值的顺序是未定义行为. 试写出一段 MiniDecaf 代码, 使得不同的参数求值顺序会导致不同的返回结果.

```
int foo(int a, int b)
{
    return a * b;
}

int main()
{
    int a = 0;
    int b = 5;
    foo(a=b+1, b=a-1);
}
```

以上代码中, 如果 `foo` 从左边开始求值, 传入的参数为 6 和 4, 返回值是 24; 反之, 传入的参数为 0 和 -1, 返回值是 0.

Honor Code

主要参考实验指导书实现. 参考了助教的示例代码.