

### 编译原理实验三

**中间代码生成**

学 院 计算机科学与技术

学　　 号

班　　 级

学 生

日 期 2022年4月23日

# 编写文件列表

在实验二的基础上添加语义分析函数头文件和文件：

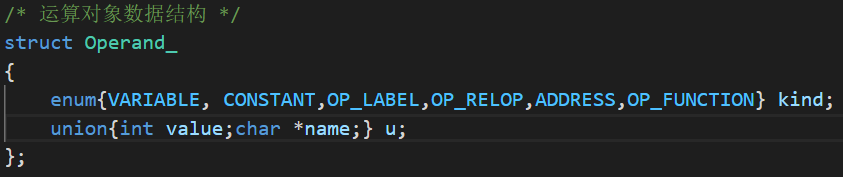
interCode.h：定义中间代码链表结点数据结构，声明全局辅助变量，声明辅助数据结构操作函数和翻译模式。

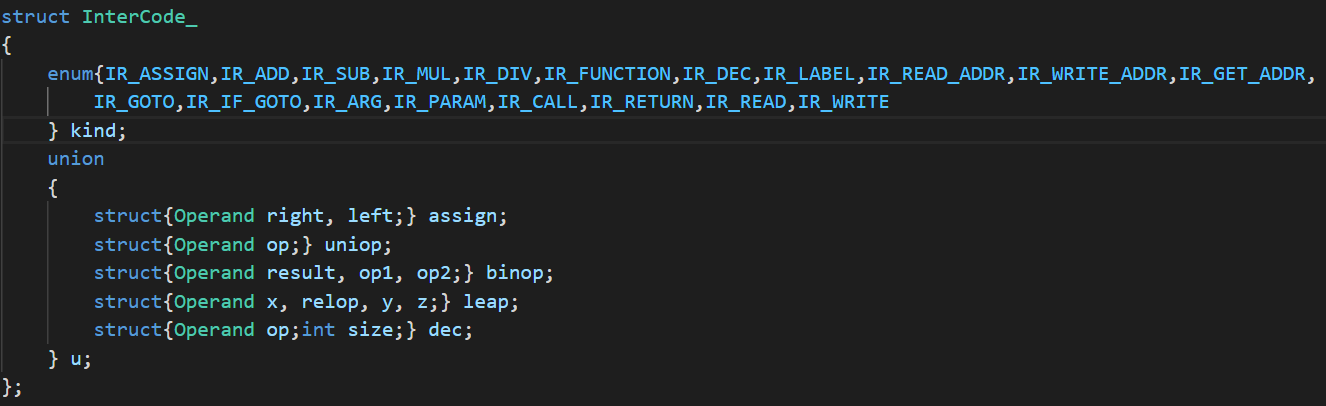
interCode.c：定义头文件中声明的函数。

# 程序实现功能

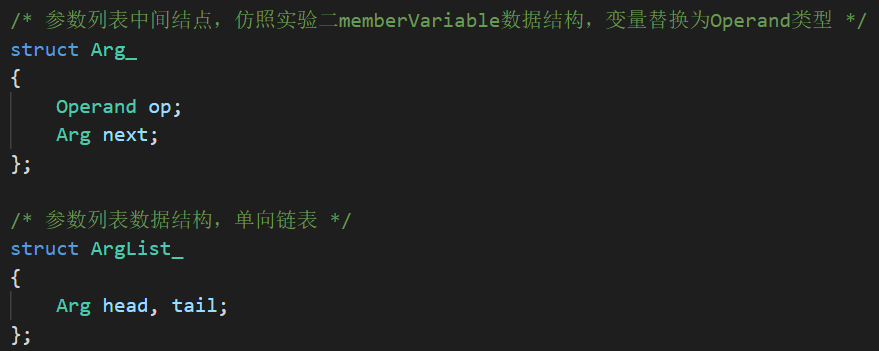
## 2.1线型双向循环链表

本人决定使用线型双向循环链表表示中间代码行。参照实验指导书，在interCode.h中定义中间代码的各数据结构，根据程序需要完成的功能补充操作数和中间代码行的枚举类型(kind)中的枚举元素。





编写辅助数据结构参数列表ArgList，用于在遇到函数调用语句时，保存参数的操作数，便于生成中间代码时在调用函数前声明函数参数。



## 2.2修改代码

修改实验二中的analyse.h中哈希表项的结构，添加成员变量isArg。由于实验二中将所有函数参数也插入符号表，需要添加一个变量用来标记符号是否是参数。



## 2.3关键函数

int getSize(struct type \*type):根据符号类型不同，返回该类型所占字节的大小。简单类型直接返回大小，对于复杂的类型递归地调用自身，累加获得该复杂类型的大小。

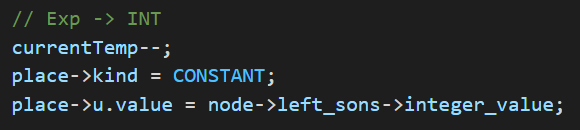
void genInterCode(int kind, ...):根据第一个参数kind（中间代码行的类型）的不同，使用变长参数获得对应的操作数，并产生kind对应的语句，将生成的语句插入链表中；当类型复杂时，可能需要产生新的temp或label并递归调用自身生成新的中间代码。

## 2.4优化

没有进行遍历中间代码链表进一步优化，而是只在产生中间代码过程中做了简单优化。

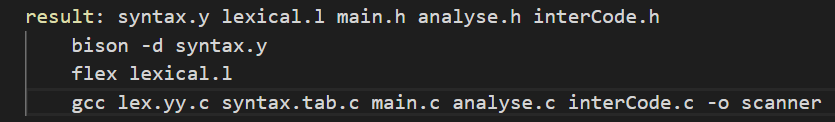
程序维护两个个全局变量：下一个产生的temp和label的下标currentTemp和currentLabel，当程序需要产生新的temp和label时，可以分别调用函数createTemp和createLabel产生新的临时变量和标号。

当我们在程序产生临时变量时，若该符号是立即数或ID，则我们可以放弃使用这个临时变量，并令“currentTemp -1”。于是通过这种方法，我们可以在中间代码中省略一个临时变量。



# 编译和使用方法

使用Makefile进行编译，进入目录后在终端输入make，按照Makefile文件内进行编译和链接后，生成可执行文件scanner。



使用方法：./scanner [待分析文件名] [输出结果文件名]

例： ./scanner input1.cmm output.txt

程序可正确生成必做的2个测试样例,和第一个选做样例的中间代码，并且可以通过小程序的测试。