# 实验报告

151160055 吴宇昊 地理与海洋科学学院

## 程序如何被编译：

在终端中输入 make 完成编译，make test1，make test2，make test3，make test4 分别完成对 test1，test2，test3，test4 的中间代码生成，生成文件分别储存在 test 里的 out1.ir，out2.ir，out3.ir，out4.ir 中

## 程序功能：

中间代码的生成完全独立于语义分析，并且在 syntax.y 中将语义分析中注释掉了
相关代码置于 intercode.h 与 intercode.c 里
中间代码生成的入口在 syntax.y 的 Program 中，中间代码输出入口在 main.c 中
中间代码的整体框架类似语义分析—使用语法同名的函数并上 translate_
数据结构在实验指导的基础上加以扩充，中间代码的结构为单向链表
没有使用双向链表是认为目前暂时用不上，扩充为双向链表也只需要在 insertCode 中稍作修改
因为完全独立于语义分析，所以单独执行遇见语义错误并不能识别，但是可以通过取消掉注释来在语义分析没有错误的前提下进行中间代码生成，不过目前这样做会出现错误，因为没有编写 write，read 函数
**intercode.h**
struct InterCode_为中间代码的的数据结构，由类型 kind，操作符的 union 与 InterCode next 构成
kind 有运算符=+-*/，关系运算符>=<=!===<>，需要 4 个操作符的 IFGOTO_类型，还有其他类型的中间代码
union 中的结构基本对应了上述四种
还有在高阶数组中使用的 next 指针
struct Operand_为中间代码操作符的数据结构，构成与 struct InterCode_类似
kind 有临时变量，普通变量，常数，地址，标记，函数
union 中 var_no 储存当前临时变量计数，value1 储存常数的数值，value2 是废设，name 储存普通变量/函数的名字
还有在高阶数组中使用的 next 指针
除了与语法同名的函数，还有一些其他函数
insertCode 插入中间代码，printCode 导出中间代码，printOperand 导出操作符，lookup 是获得函数声明，高阶数组的空间列表
同名函数中舍弃了 VarDec 中的普通变量的声明，与类型相关的 Specifier，StructSpecifier
除了 Exp，Args，VarDec，VarList，ParamDec 的返回值为 struct Operand_*，其余皆为 void，Args 是为了获得函数调用时的参数列表，Exp 是后续使用中需要 Exp 的返回值作为操作符，

VarDec 是为了生成高阶数组的空间列表同时 DEC 其所需的所有空间，ParamDec，VarList 是为了函数声明的参数列表，ParamDec

**intercode.c**

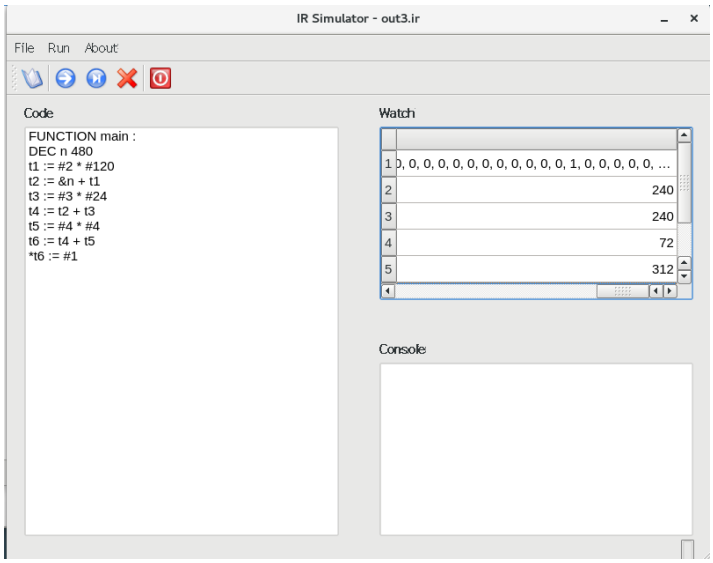
编写代码的顺序为：不涉及 label，function，address 的代码生成，=+-/*等只涉及普通变量的运算->function 的声明与调用->高阶数组的声明与使用->if，while，relop 等涉及 label 的操作+-/*=这类的操作比较简单，步骤类似 malloc 三个 Operand_*，一个 InterCode_*的内存区域，根据操作符的类型进行赋值，并进行临时变量的创建，再将三个/两个操作符填入中间代码中并插入中间代码

Function 的声明与调用除了类似上述的普通代码的操作，还要根据是声明还是调用，在合适的地方进行递归完成 next 的传递，将链表的表头填入中间代码中，便于遍历/使用

高阶数组的声明与使用也是类似的，不过比 function 相关代码复杂的地方在于还要计算内存空间，而且递归调用时还用到了在语义分析，语法分析中均没有用到的 root—窥测上层的状况寻找递归出口

if，while，relop 等涉及 label 的操作一直没有想到很合适的 return 类型，干脆就 return NULL，label 的操作实在 stmt 中生成 label，goto 相关代码，在 exp 中生成 ifgoto，goto 代码

代码每生成一部分进行一次测试，不过没有测试过多种成分复合的情况，只测试了实验指导的示例，不过实验指导的示例倒不是很复杂，感觉有些测试代码比实验指导的还复杂




在测试数组的代码中因为声明的数组太大，找了好些久才找到

没有能对结构型进行翻译，优化也约等于没有

# 总结

这次实验延期了几天，因为有些其他的事情一直没有时间做实验，在周一周二周三三天里做完了实验。看了其他人的代码，都是在语义分析中添加，感觉太过杂乱就全盘推翻。这次实验其实与实验二类似，不过少了很多指针的操作所以写起来不算太难。我对代码的调试也越来越熟练，在代码里加入大量的 printf 寻找错误是十分方便的，而且现在一看到出错位置基本就能知道是哪里的指针发现了错误，也是经验使然吧，在写着写着的时候总感觉脑子不是自己的。总而言之是一次愉快的体验