南京大学本科生实验报告

课程名称：**编译原理**

学院	计算机科学与技术	专业（方向）	计算机科学与技术
学号	**201220215**	姓名	**田馥雯**
Email	1259067849@qq.com	实验名称	**实验三：中间代码生成**

实验内容

已实现全部必做及选做内容，在词法分析、语法分析和语义分析程序的基础上，将C--源代码翻译为中间代码

通过`Makefile`生成可执行文件

```
make clean
make
make test
```

最终的中间代码输出到指定的`ir`文件中，并可以在虚拟机小程序中运行，得到正确的结果

核心代码

`tool.c/tool.h`

如实验手册所说，新增中间代码创建所需要的`Operand_`操作符号、`InterCode`中间代码、`InterCodes_`中间代码双向链表这三个**结构体**，其中`Operand_`中有类型（变量、常量、临时变量、函数、标号）/（值、地址）以及标号数、数值、变量名/函数名的内容；`InterCode`中定义了十九种中间代码类型以及`singleOP/doubleOP/tripleOP/ifgotoOP`四种操作格式所需要存储的op、result等值。

因此新增的**函数**有`createInterCodes`创建双向链表成员、`insertInterTableList`中间代码双向链表插入结点、`createOperand`创建符号、`printOperand`打印符号、`printInterCodes`打印中间代码、`getTypeSize`获取类型大小等。

其中`createInterCodes`创建双向链表成员和`insertInterTableList`中间代码双向链表插入结点用到了**可变参数**这一技巧，以满足不同的传参需求。

`intermediate.c/intermediate.h`

包含中间代码生成的主题函数部分，与实验二一致，也是一个对语法树的遍历过程，这个遍历过程用到了实验二所建立的符号表，通过建立双向链表来得到中间代码序列，从Program结点入手，开启整个遍历过程。

```c
void translateProgram(AST *root, FILE *file)
{
    /* 初始化双向链表 */
    interCodeListHead = (InterCodes)malloc(sizeof(struct InterCodes_));
    interCodeListHead->next = NULL;
    interCodeListHead->prev = NULL;
    interCodeListTail = interCodeListHead;
    /*
    Program -> ExfDefList
    */
    translateExtDefList(getChild(root, 0));
    printInterCodes(file); //打印中间代码
}
```

其余类似的函数还有`translateExtDefList`、`translateExtDef`、`translateExtDecList`、`translateVarDec`等等，按照手册一个一个来即可。

中间代码优化

`IF LP Exp RP Stmt1`、`IF LP Exp RP Stmt1 ELSE Stmt2`、`WHILE LP Exp RP Stmt1`等减少`label`数量，例如必做内容样例1中的if-else if-else语句中仅用了4个label，比原先冗余的6个label少

```
IF v1 <= #0 GOTO label1
WRITE #1
t2 := #0
GOTO label2
LABEL label1 :
IF v1 >= #0 GOTO label3
t3 := #0 - #1
WRITE t3
t4 := #0
GOTO label4
LABEL label3 :
WRITE #0
t5 := #0
LABEL label4 :
LABEL label2 :
```

其他新增内容

为满足read和write函数的需求，在语义分析前新增`createRead`和`createWrite`函数用于创建这两个函数

总结与感想

代码错误及解决方法

lab1遗留错误

RELOP没存符号是啥，导致`ifgoto`语句中的判断条件一直是null，回头看才发现没存.....

解决方法：lab1里面存上`yytext`

```
{RELOP} {yylval.node =
createNewAbstractSyntaxTreeNode("RELOP",yytext,2,yylineno);return RELOP;}
```

段错误

这个问题真的是数不胜数，到处都在用`null`的指针，就是利用`printf`逐步调试找到他在哪，最离谱的是再test过程中不小心在`printf`里面想要输出`op1->varName`，但是根本没管op1可能是null，导致test又给程序添bug

粗心大意

比如判断一个东西不等于写成了等于、传参的中间代码输出忘写了导致一直没传参这种，很离谱……

感悟

终于没有特别赶deadline，稳扎稳打把这个实验做完了，调试起来挺难的，会暴露之前的错误。