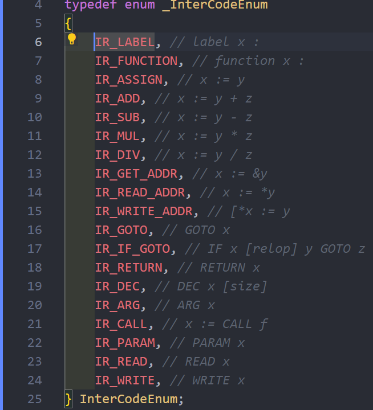
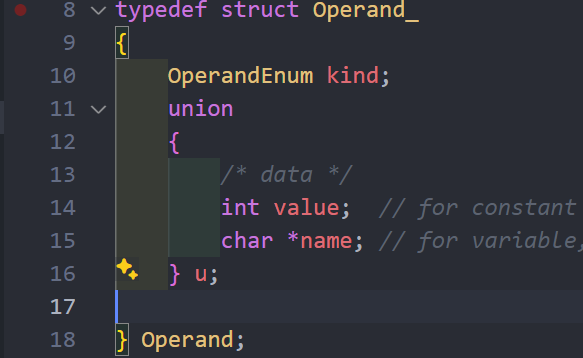
编译原理实验2报告

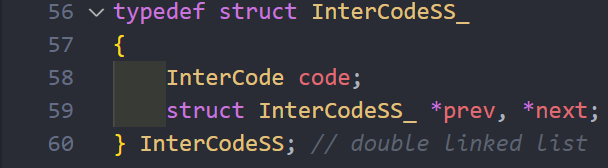
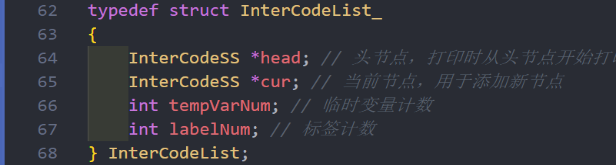
姓名：朱宸慷 学号：2021110908

1. 程序实现的功能
2. 所使用的数据结构

电脑萤幕画面

描述已自动生成首先参照指导书，定义可能出现的操作数类型。在我们的中间代码生成中，操作数可以分为两类：与运算相关的常数、变量以及指针；与控制流相关的标签、函数以及操作符。同样参照指导书，可以定义翻译“一条”中间代码的模板形式，总共可以分为表达式、调用与函数、控制流与跳转三类共计19种种类，各自对应一个中间代码翻译模板。

在确定了一条中间代码的翻译模板之后，还需要确定填充的操作数，将操作数保存为操作数种类和操作数的字面值的结构体Operand，同时根据中间代码对于操作数个数需求的不同，构建中间代码类型和数个操作数的结构体InterCode。

在生成中间代码时，将InterCode连接成一个链表，保存头尾指针，同时保存过程中生成的临时变量和标签的数目，用于生成下一个变量或标签。

1. 中间代码生成的思路

基于实验一构建的语法树，从根节点开始前序遍历整棵语法树。由于实验指导书要求不能使用全局变量，因此我们直接从定义ExtDefList开始进入翻译过程。

在实验三中，总共需要完成的翻译模式有基本表达式、语句、函数调用、数组与结构体。其中结构体不做要求，笔者在实验中实现了没有明确给出的数组翻译模式。

**翻译表达式**分为翻译数值计算和翻译布尔表达式，本质上都是使用一个临时变量接收表达式计算的结果，只是数值计算的结果可以直接返回，而布尔表达式的结果需要经过条件跳转来决定最终返回将接收值的变量赋值为1或0。

**翻译语句**过程中，需要将以下五类语句——表达式语句、复合语句、返回语句、跳转语句和循环语句翻译为对应的中间代码。每种语句出现的位置比较固定，可以检查语法树节点的类型并决定使用何种翻译方法。参考实验指导书可以找到一部分语句对应的翻译模板。

**翻译函数调用**需要完成调用前参数压栈以及函数体内部的参数弹栈过程，这个过程可以借助符号表中存放的函数参数直接完成。

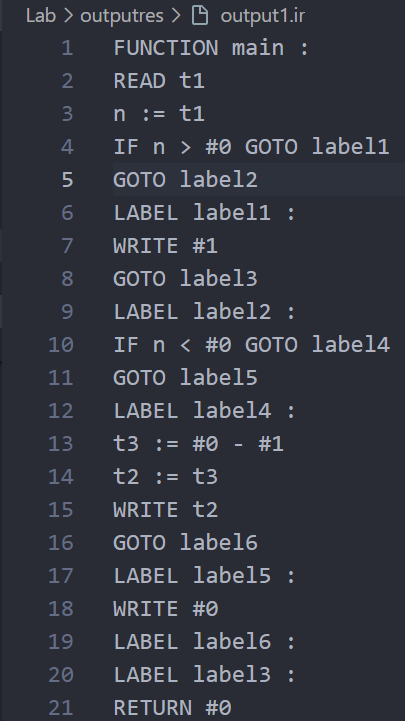
**翻译数组**需要获取数组首地址以及

综上所述，实验三主要完成的就是：针对每种语法树节点的翻译函数，将该节点的子节点全部翻译成一条InterCode链表之后，将该链表插入到整体链表合适的位置，并在

1. Sgdg
2. 程序的编译方法
3. 自动化执行makefile

通过使用makefile自动执行。在code文件夹下先make clean再运行make即可编译本实验的可执行文件。除此之外，笔者还搭配了一个bash脚本用于自动化执行makefile以及输入测试用例进行测试。运行./run即可在控制台打印所有结果。

1. 文本

   描述已自动生成结果展示

如图所示是实验测试用例1、2的中间代码