编译原理

实

验

报

告

**实验名称：实验三 典型语句的语义分析及中间代码生成**

**姓名： 方澳阳**

**学号： 180110115**

**学院： 计算机科学与技术**

**专业： 计算机类**

# 实验目的与方法

（1）加深对自顶向下语法制导翻译技术的理解与掌握

（2）加深对自底下上语法制导翻译技术的理解与掌握

（3）巩固对语义分析的基本功能和原理的认识，理解中间代码生产的作用

实现语言: C++

IDE: Clion2020.2

平台: windows10、linux

# 实验总体流程与函数功能描述

本次实验建立在语法分析的基础上。由于使用的是LR1文法，较为适合自底向上的语法分析。

在语法分析的规约过程中，每当规约出一个产生式，就可以将其翻译出对应的中间代码，通过翻译后可以输出对应的三地址码。

本次实验主要添加了以下三个函数：

// 将把产生式序列转化成带非终结符的三地址码

bool Grammar\_Analyzer::translate(deque<attributeTable> &pro)；

// 将多余的三地址码序列消除

void Grammar\_Analyzer::clearUselessProducer()；

// 判断是否为非终结符

bool Grammar\_Analyzer::notEnd(const string &input, map<string, int>

NoEndIndex)；

# 实验内容

（1）针对自底向上分析法中所使用的文法，在语法分析的基础上为语法正确的单词串其设计翻译方案，完成语法制导翻译。

（2）利用该翻译方案，对所给程序段，进行分析，输出生成中间代码序列和符号表，并保存在相应文件中。

（3）中间代码选择三地址码

（4）完成了常见赋值语句的语义分析与中间代码生产。

# 实验结果与分析

输入：

f = 2\*10+2\*4#

c = f-2#

d = c/10#

a = 10/d+3

输出的三地址码：

A = 2 \* 10

A = 2 \* 4

f = A + A

c = f - 2

d = c / 10

A = 10 / d

a = 3 + A

输出的符号表:

symbol: token

value: a

intValue:8

type: 0

symbol: token

value: c

intValue:28

type: 0

symbol: token

value: d

intValue:2

type: 0

symbol: token

value: f

intValue:28

type: 0

**中间代码的存储结构为：**

vector<vector<string>> midCodeOut;

即由二维字符串数组构成。一个字符串表示一个操作符或参数。

例如value1 = 10\*value2

midCodeOut[0][0]表示“value1”,midCodeOut[0][1]表示“10”，midCodeOut[0][1]表示“\*”,midCodeOut[0][3]表示“value2”。

**分析**

由于在实验二时没有意识到需要入栈、出栈时的那些元素需要用到其对应的属性，如type、value以及原来的字面值等。导致当初符号栈只存放了字符串类型的数据。在入栈时将token串中的ID、常数替换后，在出栈时就丢失了原来的值。

于是开始添加token串携带的信息，将符号栈也修改为能够携带更多的信息的栈类型。在translate()函数中对每个产生式进行判断，产生对应的翻译代码。

在做这次实验的时候存在两个问题。第一，没有对实验的要求有清晰的认识，导致自己在之前的数据结构设置错误，从而导致了本次实验在一个错误的起点中；第二，时间较为紧张，在第一个原因的影响下，自己没有足够的时间去重构代码，只能在原有的代码上缝缝补补，将错误方向的代码结果外面再套一层代码纠正过来，导致代码量巨大，实现的功能却十分有限，并且可读性不佳，思路不够清晰。

希望自己未来能够整理一下代码，理清思路。

# 实现的拓展功能

在实验二中实现了for、函数以及两者互相嵌套的结构；实现了浮点数的判断。

在本实验中，添加了上述功能的产生式判断框架，但由于时间关系没有实现具体代码，只是单纯地设置为exit（-1），表示暂时不接受该输入形式。