图片包含 徽标

描述已自动生成

**编 译 原 理 实 验 报 告**

**实验五 中间代码生成器**

姓 名 李波

学 号 22920202204570

院 系 信息学院

专 业 计算机科学与技术

指导教师 陈怡疆

# 一、实验目的

掌握中间代码生成器的构造原理和编程方法。

# 二、实验内容

(1) 声明语句的翻译

声明相关的文法如下：

decls -> decls decl | ε

decl -> type id;

type -> basic // 四种基本数据类型 int | float | char | bool

语法制导定义：对设置两个综合属性 type 、 width

type ：存放类型

width ：存放该类型占用存储单元的字节数

这里basic的type属性由词法分析器直接得到，因此width不需要通过设置属性传递，直接在插入变量表前判断类型即可。声明语句翻译并不会生成三地址码，而是将变量名、变量类型、存储地址记录到变量表，所以在对非终结符 decl 进行归约时，根据 type 记录的变量类型以及相应的域宽大小和由 id 分析得到的变量名可向变量表中插入一项。在对非终结符decls 进行归约时不需要额外的翻译动作。

文本

描述已自动生成

一个代码块内部声明的变量，只能被代码块内部的语句使用，所以需要将产生式b1ock -> {

decls stmts }进行改写：

在对非终结符b1ockM1进行归约时，意味着马上要进入一个新的代码块，因此在此时创建一个本作用域的变量表。在对非终结符b1ockM2进行归约时，意味着该代码块马上结束，将变量表的内容输出到终端显示，并将变量表删除。翻译方案如下：

文本

描述已自动生成

产生式type -> basic翻译方案如下：

文本

描述已自动生成

(2) 顺序的赋值语句的翻译

1、类型检查

类型转换发生有以下两种情况：一种是产生式stmt -> id = expr;中，变量id的数据类型与expr类型不同，此时将expr的类型转 换为变量id的类型。 第二种是四则运算中，两个不同类型的操作数进行运算时。

对变量赋值语句进行类型转换时，转换类型直接由变量id类型决定，因此不一定是拓宽类型。思路是生成指令将待转换的数转换到临时变量，然后之后的操作由该临时变量替代。

这里需要编写typeWiden函数，将类型type转换为类型wideType，如果两个类型相同则直接返回地址，否则生成相应的三地址指令进行类型转换并返回转换生成的临时变量地址。

typeWiden函数如下：

文本

描述已自动生成

2、语法制导翻译方案

变量复制语句部分代码如下：

文本

描述已自动生成

运算语句如下，这里以加法为例：

文本

描述已自动生成

图形用户界面

中度可信度描述已自动生成

unary和factor代码如下：

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

(3) 控制流语句的翻译

需要对文法进行修改，增加M属性、N属性作为辅助，并使用回填技术。

if语句和if-else语句使用M标志当布尔表达式为真或为假时跳转的指令地址，将其相应回填到truelist和falselist中。在if语句中，当bool表达式为真时执行代码段stmt1，因此M1.instr回填到布尔表达式truelist；当bool表达式为假时执行代码段stmt2，因此M2.instr回填到布尔表达式falselist。之后将stmt1和stmt2的nextlist合并到整个语句的nextlist。

文本

描述已自动生成

while语句当布尔表达式为真时，执行stmt代码段，否则执行下一段程序，因此M.instr回填到布尔表达式truelist，nextlist回填到布尔表达式truelist。do while语句和while语句类似。

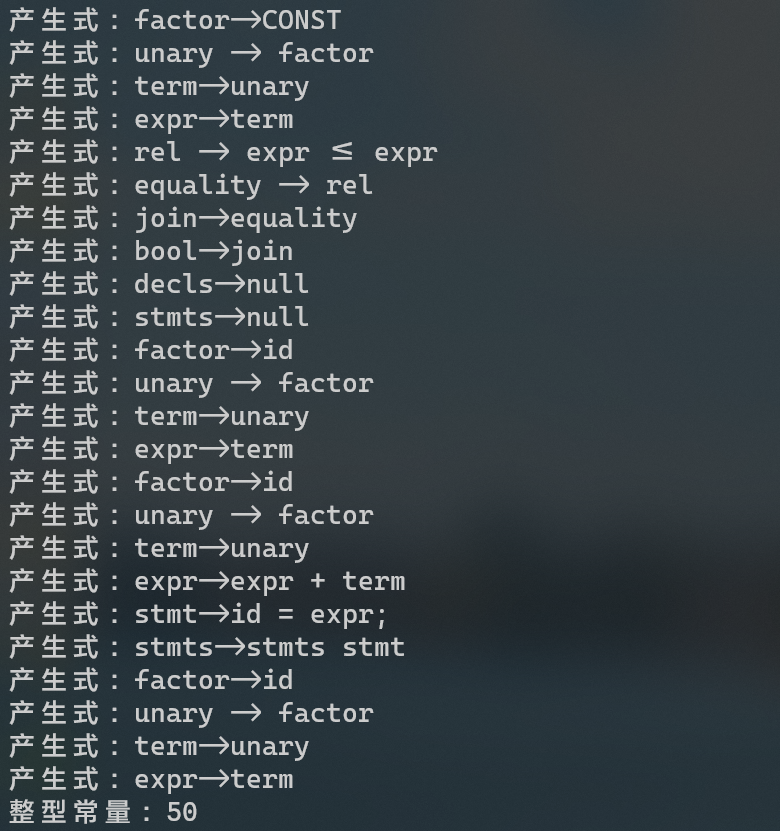
文本

描述已自动生成

# 三、实验结果

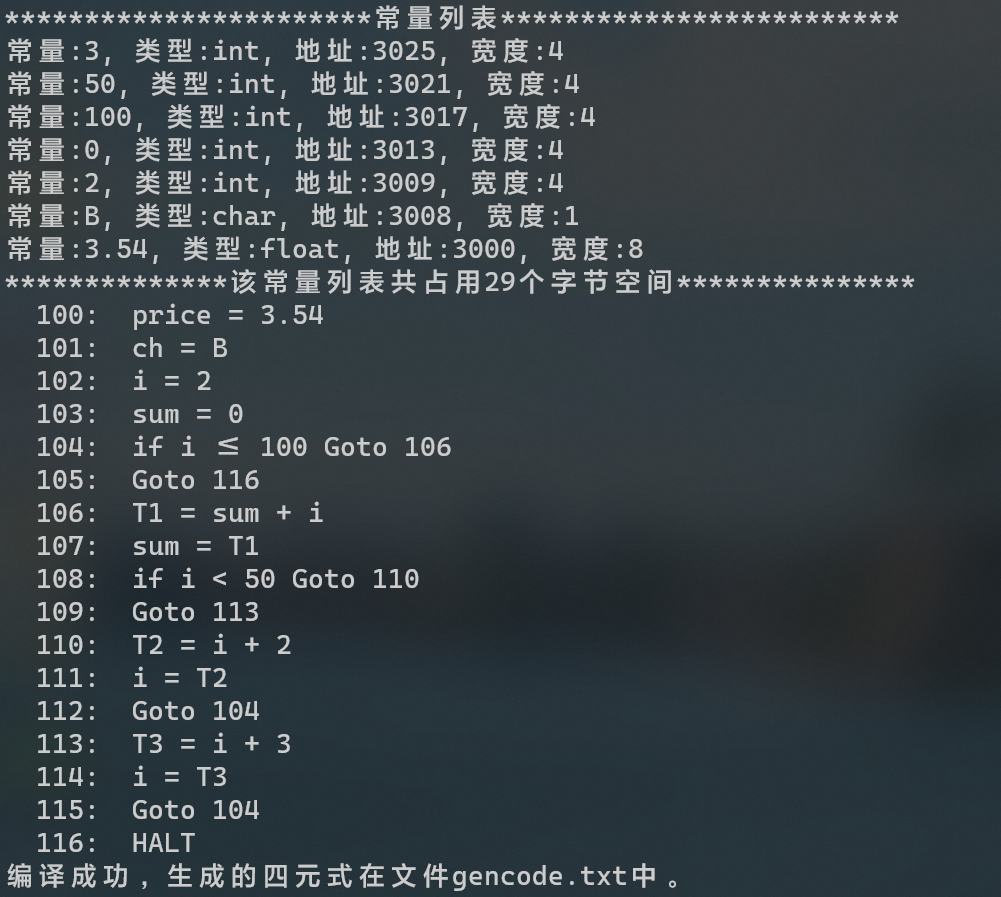










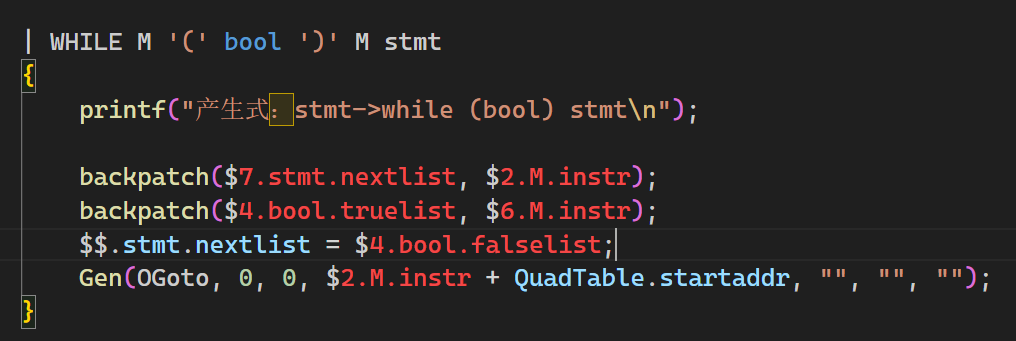


图形用户界面, 文本, 应用程序

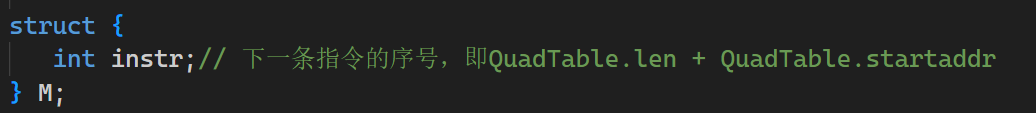
描述已自动生成

# 四、实验遇到的困难

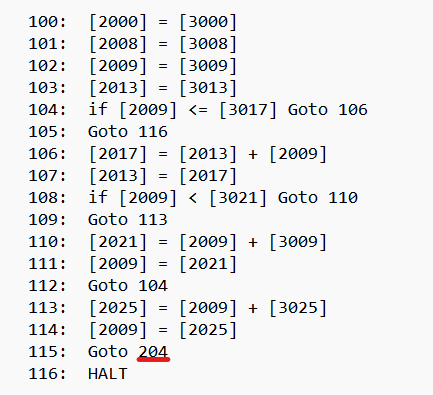
本次实验非常复杂，代码量较大，极易发生代码编写方面的粗心错误，导致错误难以发现，例如：



这里最后一句代码错误地将QuadTable.startaddr加到instr上面导致结果错误，因为instr属性中已经包含QuadTable.startaddr。



以上代码错误导致以下的结果错误，115号四元式中goto后面的地址发生错误，实际应为104。



经过仔细观察，将错误更改后即得到正确结果。

# 五、实验心得

在进行中间代码生成器实验的过程中，我深入学习了中间代码生成的构造原理和编程方法，并在此基础上编写了语法制导翻译方案，生成了四元式。

实验的第一部分是编写声明和顺序的赋值语句的翻译规则。这一部分需要进行类型检查，确保变量的声明和赋值操作符的类型匹配。我发现这一过程需要仔细考虑不同数据类型的处理方式，并遵循语言规范中的类型转换规则。在实现过程中，我注意到细节的重要性，因为疏忽可能导致错误的类型转换或类型不匹配，影响中间代码的生成结果。

实验的第二部分涉及控制流语句的翻译。这一部分包括条件语句和循环语句的翻译规则。我发现在处理条件语句时，需要考虑条件表达式的求值和跳转指令的生成。而在处理循环语句时，需要注意循环的起始和结束条件，以及生成适当的跳转指令来实现循环控制。这一部分的挑战在于理解源代码的控制流逻辑并将其转化为中间代码的等效表示。

整个实验的难点在于思路的复杂性和可能的疏忽。由于中间代码生成涉及多个步骤和不同的翻译规则，我必须仔细考虑每个步骤，并确保没有遗漏任何关键细节。此外，代码量较大，我需要保持良好的组织和结构，以便于维护和调试。

通过完成这个实验，我获得了许多宝贵的经验和收获。我深入了解了中间代码生成的原理和技术，对编译器的工作原理有了更深入的理解。