基于C0文法的编译器V3.0

10011603 2016302731 冯宇

**一.主要功能**

相较于初步分析V2.0添加以下功能：

1.在定义变量的同时赋予初值

2.实现while循环

**二.代码实现**

1.在变量定义函数void varDef()中添加赋值部分，代码如下：

string infixString = "";//存储数字赋值

if (sy == becomes) { // 读取“=”，进行变量的赋值

if (idIndex != -1) {

if ((idTable[idIndex].cls != vars && idTable[idIndex].cls != params)

|| idTable[idIndex].length != 0) {

error(IdKindNotMatch); 变量名匹配

}

}

insymbol();

expression(rightToEqual);获得“=”右边表达式的值

insertInfix("ASSIGN", " ", rightToEqual, leftToEqual);//进行变量赋值

2.while循环

该循环主要参照for循环书写完成，由于四元式中无新添加的语句，所以我们主要修改的是main.h、lexical.cpp以及praser.cpp中的代码

首先，在main.h中的枚举类symbol中定义while关键字，将关键字的总个数keywordCount加一

在词法分析中进行读取判断

在语法分析中

首先在<语句>中添加对于while的选择：

case whilesy:

whileState();

break;

添加语法函数：

<while> ::= while ‘(’ <条件>’)’{语句}

void whileState(){

string judgeLabel = "";//判断条件LABEL

string bodyLabel = "";//循环体LABEL

string endLabel = "";//while循环外部语句开头LABEL

首先读取while关键字

定义循环体的LABEL bodyLabel = labelStr + intToString(labelIndex);

定义判断条件的LABEL judgeLabel = labelStr + intToString(labelIndex);并插入infixTable

进行条件的判断judgement(judgeResult)

insertInfix("BNE", judgeResult, "0", bodyLabel); // 如果judgeResult != 0(为真) 转入bodyLabel

endLabel = labelStr + intToString(labelIndex);//定义endLABEL

labelIndex++;

insertInfix("BEQ", judgeResult, "0", endLabel); // 如果judgeResult = 0(为假) 跳转到 endLabel，即while循环外部

读取语句statement();

insertInfix("JMP", " ", " ", judgeLabel);//跳转到判断体的LABEL

insertInfix("LABEL", " ", " ", endLabel);//在此处插入while循环外部的LABEL

}

**三.测试结果**

|  |  |
| --- | --- |
| Input code（变量定义时初始化并实现while） | 四元式 |
| int main(){  int ax =2;  int i=0;  while(i <=3){  i=i+1;  }  ax=ax+5;  return;  } | int main()  int ax  ax = 2  int i  i = 0  LABEL1  #t0 = i <= 3  goto LABEL0, if #t0 != 0  goto LABEL2, if #t0 == 0  LABEL0  #t1 = i + 1  i = #t1  jmp LABEL1  LABEL2  #t2 = ax + 5  ax = #t2  return |

**四.Debug以及C++知识点**

**1.缺少“)”报错**

在修改程序时误删一个“）”导致程序在编译的时候出现大量缺少“;”的报错，由于数量太多导致一开始对于错误原因毫无头绪。后续通过一步一步汇编最终发现是在修改时误删一个“）”导致的

**2.whie语句设计报错**

在设计while语法的时候，由于思路不清晰，导致词法分析器读取没能与预先设计的内容相匹配，后面通过一步一步调试，不断修改代码，最终得到对应的while语法