SHANGHAI UNIVERSITY

**<编译原理>实验报告**

|  |  |
| --- | --- |
| **学 院** | **计算机工程与科学学院** |
| **组 号** | **A04** |
| **实验题号** | **实验五 中间代码生成** |
| **姓名** | **刘洋** |
| **学号** | **16121368** |
| **日期** | **2018.5.23** |

|  |  |
| --- | --- |
| 学号 | 姓名 |
| 16121348 | 卿波 |
| 16121368 | 刘洋 |
| 16121391 | 陈杰 |
| 16122804 | 刘锦 |

## 实验目的

* 通过上机实习，加深对语法制导翻译原理的理解，掌握将语法分析所识别的语法范畴变换为某种中间代码的语义翻译方法。
* 掌握目前普遍采用的语义分析方法──语法制导翻译技术。
* 给出 PL/0 文法规范，要求在语法分析程序中添加语义处理，对于语法正确的表达式，输出其中间代码。

## 实验内容：

已给 PL/0 语言文法，在实验三的表达式语法分析程序里，添加语义处理部分输出表达式的中间代码，用四元式序列表示。

## 实验要求：

* 在实验三“语法分析器”的里面添加 PL/0 语言“表达式”部分的语义处理，输出表达式的中间代码。
* 中间代码用四元式序列表示。
* 准备至少 10 组测试用例，每组测试用例包括：输入文件和输出结果。

## 算法描述：

本实验采用递归子程序法进行中间代码生成，建立在前三次实验的基础之上，输入的为PL/0 表达式，经过词法分析后使用词法分析的结果进行表达式的语法分析，进行中间代码的生成，以四元式的形式表示。

首先将词法分析的结果对应存在文件result中，再一个个按顺序读取，利用advance()函数获取下一个单词，单词与系统名称存在全局变量sym,sym2中，每执行一次advance(),更新sym和sym2。定义四元式的结构体，利用emit()函数生成结构体数组存放所有产生的四元式，利用newtemp()函数对字符串进行处理，分别按照表达式、项、因子的产生式，进行处理。与语法分析不同的点在于，表达式中需要记录加减运算符两侧的项的操作数，项中需要记录乘除运算符两侧因子的操作数，因子中需判断为标识符还是无符号整数还是表达式，每执行一次表达式或因子的判别函数都将生成一个四元式，按顺序生成中间代码。

## 实验代码：

C++源程序如下：

|  |
| --- |
| #include<bits/stdc++.h>  using namespace std;  //result 词法分析结果 analysis 原表达式  ifstream infile("result.txt");  ifstream infile2("in/analysis0.txt");  ofstream outfile("result.txt");  map<string,string> word;  std::map<string,string>::iterator it;  string str;//读入的字符串  string sym;//类型符号  string sym2;//值符号  int count1=0,k=0,flag=0,conterr=0,lpnum=0;  string expressionAnalysis();  string termAnalysis();  string factorAnalysis();  struct quad{//定义四元式  string result;  string arg1;  string arg2;  string op;  };  struct quad quad[50];  void map\_init(){//对应关系进行初始化，如下只包括了表达式的相关符号  word["+"]="plus";  word["-"]="minus";  word["\*"]="times";  word["/"]="slash";  word["="]="eql";  word["("]="lparen";  word[")"]="rparen";  }  void lexanalysis(){//词法分析  char ch;  char a;  string word1;//string变量识别单词  string str;//string变量进行字符识别  ostringstream buf;  while(buf&&infile2.get(ch)) buf.put(ch);//将文件中的字符读出来  str= buf.str();//将得到的字符储存到string类型变量中  int csize=str.length();  for(int i=0;i<csize;i++){//对整个字符串进行遍历  while(str[i]==' '||str[i]=='\n') i++;//若最开始为空格或换行符，则将指针的位置往后移  if(isalpha(str[i])){//对标识符和基本字进行识别,调用库函数isalpha()  word1=str[i++];  while(isalpha(str[i])||isdigit(str[i])){  word1+=str[i++];  }  it=word.find(word1);  if(it!=word.end()){//判断是不是基本字，若为基本字则进行输出  outfile<<"("<<word[word1]<<","<<word1<<")"<<endl;  }  else{//否则直接输出  outfile<<"(ident"<<","<<word1<<")"<<endl;  }  i--;  }  else if(isdigit(str[i])){//判断是不是常数，调用库函数isdigit()  word1=str[i++];  while(isdigit(str[i])){  word1+=str[i++];  }  if(isalpha(str[i])){  outfile<<"error"<<endl;  break;  }  else{  outfile<<"(number"<<","<<word1<<")"<<endl;  }  i--;  }else{//对其他的基本字依次进行判断  word1=str[i];  it=word.find(word1);  if(it!=word.end()){  outfile<<"("<<word[word1]<<","<<word1<<")"<<endl;  }else{  outfile<<"error"<<endl;  break;  }  }  }  infile2.close();  }  int advance(){//分析result文件词法分析的结果 用来读入下一个单词  int found1,found2;  if(!getline(infile,str)){  return 0;  }  found1=str.find(',',0);  if(found1==-1){  conterr++;  cout<<"语法错误 识别字符错误"<<endl;  return -1;  }  found2=str.length();  sym=str.substr(1,found1-1);  sym2=str.substr(found1+1,found2-found1-2);  return 1;  }  //中间代码生成  string newtemp(){//产生新变量名t1,t2等  char \*p;  char m[12];  p=(char\*)malloc(12);  k++;  itoa(k,m,10); //将整数k转换成字符串m 十进制转换  strcpy(p+1,m); // 将m放在指针后一个  p[0]='t';  string s;  s=p;  return s;  }  //产生四元式用于显示，并记录次数  void emit(string op,string arg1,string arg2,string result){  quad[count1].op=op;  quad[count1].arg1=arg1;  quad[count1].arg2=arg2;  quad[count1].result=result;  count1++;  return;  }  string expressionAnalysis(){//表达式的递归下降分析程序  string op,arg1,arg2,result;  if(conterr){  return NULL;      }      arg1=termAnalysis();//通过项分析得到第一个参数的值      if(conterr){  return NULL;      }      //每执行一次advance() 全局变量sym sym2都会修改一次，变成二元式      while((sym=="plus")||(sym=="minus")){  op=sym2;          flag=advance();          if(conterr){  return NULL;          }          if(flag==0){  cout<<"语法错误 <加法运算符>后缺项"<<endl;  conterr++;              return NULL;          }          arg2=termAnalysis();//通过项分析得到第二个参数的值          if(conterr){  return NULL;          }          result=newtemp();//产生中间变量名，相当于对结果进行存储          emit(op,arg1,arg2,result);//产生四元式，相当于进行加法或减法运算，得出结果          arg1=result;//保存得到的结果      }      return arg1;//返回表达式最终得到的值  }  string termAnalysis(){//项的递归下降分析程序  string op,arg1,arg2,result;  arg1=factorAnalysis();//通过因子分析得到第一个参数的值  if(conterr){  return NULL;  }      while((sym=="times")||(sym=="slash")){  op=sym2;          flag=advance();          if(conterr){  return NULL;          }          if(flag==0){              conterr++;              cout<<"语法错误 <乘法运算符>后缺因子"<<endl;              return NULL;          }          if(conterr){  return NULL;          }          arg2=factorAnalysis();//通过因子分析得到第二个参数的值          if(conterr){  return NULL;          }          result=newtemp();//产生中间变量名，相当于对结果进行存储          emit(op,arg1,arg2,result);//产生四元式，相当于进行乘法或除法运算，得出结果          arg1=result;//保存得到的结果      }      return arg1;//返回项最终得到的值  }  string factorAnalysis(){  string arg;  if(sym=="ident"){//如果是标识符，最终返回标识符的符号  arg=sym2;  flag=advance();  if(conterr){  return NULL;          }          if(lpnum==0&&sym=="rparen"){  conterr++;              cout<<"语法错误 ')'不匹配"<<endl;              return NULL;  }  }  else if(sym=="number"){//如果是无符号整数，最终返回相应的整数  arg=sym2;  flag=advance();  if(conterr){  return NULL;          }          if(lpnum==0&&sym=="rparen"){  conterr++;              cout<<"语法错误 ')'不匹配"<<endl;              return NULL;  }  }  else if(sym=="lparen"){//如果是左括号，则要进行右括号匹配，并判断中间是不是表达式，并得出表达式的值进行返回  lpnum++;  flag=advance();  if(conterr){  return NULL;          }          if(flag==0){              conterr++;              cout<<"语法错误 '('后缺少表达式"<<endl;              return NULL;          }  arg=expressionAnalysis();  if(conterr){  return NULL;          }  if(flag==0||sym!="rparen"){              conterr++;              cout<<"语法错误 表达式后面缺少')'"<<endl;              return " ";          }else{           lpnum--;  flag=advance();  if(conterr){  return NULL;  }  if(flag==0){  return arg;  }          }  }else{          cout<<"语法错误 因子首部不为<标识符>|<无符号整数>|'('"<<endl;          conterr++;          return " ";      }      return arg;  }  int main(){  int i=0,num,result;  //开始词法分析  map\_init();  lexanalysis();  cout<<"PL/0算术表达式的中间代码生成结果："<<endl<<endl;  flag=advance();  if(flag){  expressionAnalysis();//开始进行表达式分析  }  if(flag!=-1&&!conterr){//若表达式正确则输出表达式的中间代码表示  for(int i=0;i<count1;i++){  cout<<'('<<quad[i].op<<','<<quad[i].arg1<<','<<quad[i].arg2<<','<<quad[i].result<<')'<<endl;;  }  }  infile.close();  return 0;  } |

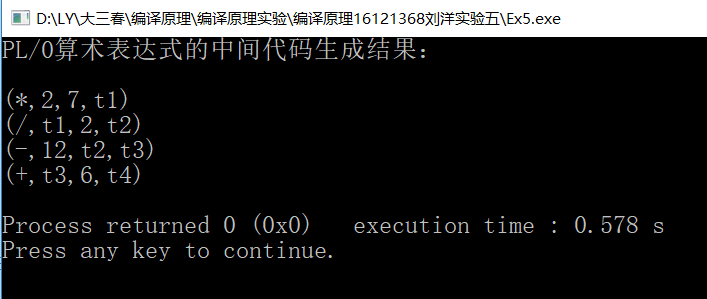
## 实验结果：

**1、测试样例1：**

**输入：**

12-2\*7/2+6

**输出：**

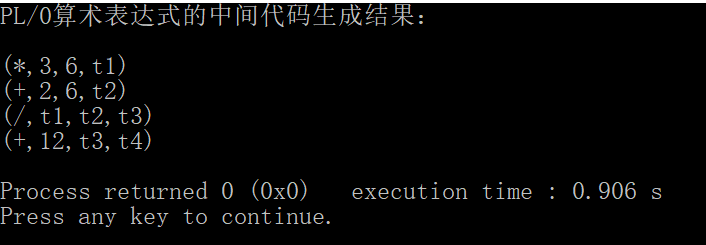


**2、测试样例2：**

**输入：**

12+3\*6/(2+6)

**输出：**

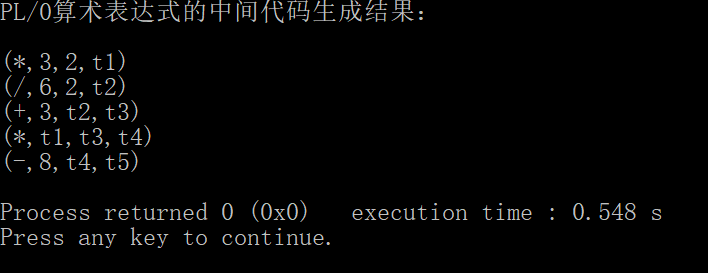


**3、测试样例3：**

**输入：**

(8-3\*2\*(3+6/2))

**输出：**

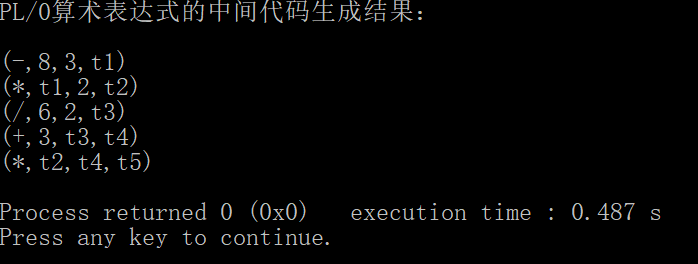


**4、测试样例4：**

**输入：**

(8-3)\*2\*(3+6/2)

**输出：**

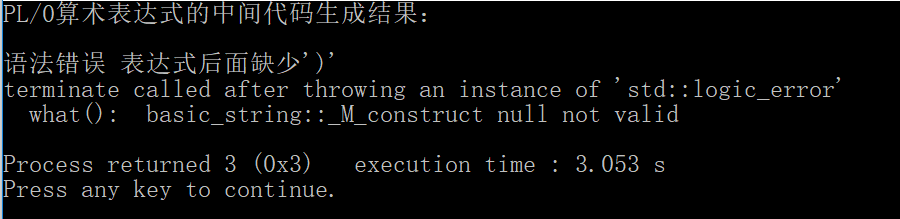


**5、测试样例5：**

**输入：**

(8-3\*2\*(3+6/(3-3))

**输出：**

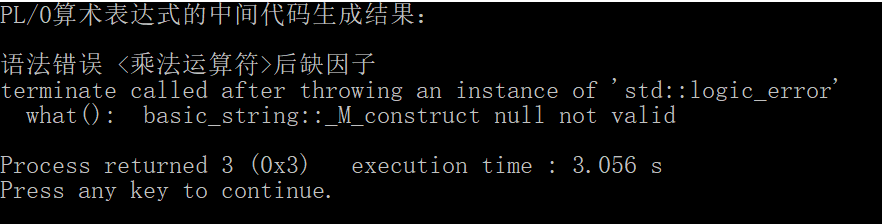


**6、测试样例6：**

**输入：**

2+3/

**输出：**

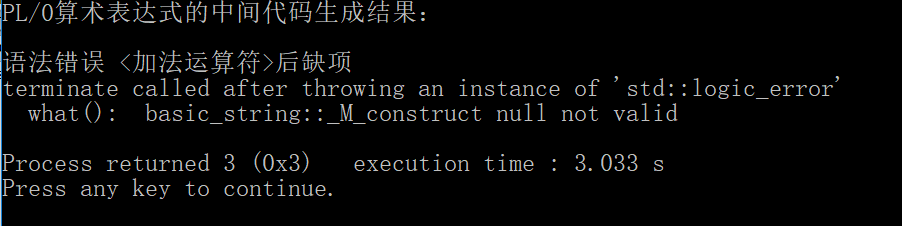


**7、测试样例7：**

**输入：**

2+

**输出：**

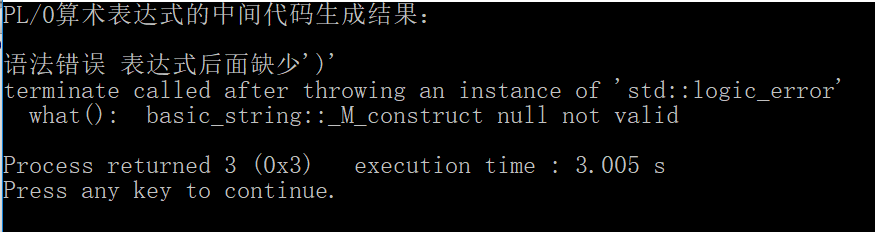


**8、测试样例8：**

**输入：**

(2+4\*5

**输出：**

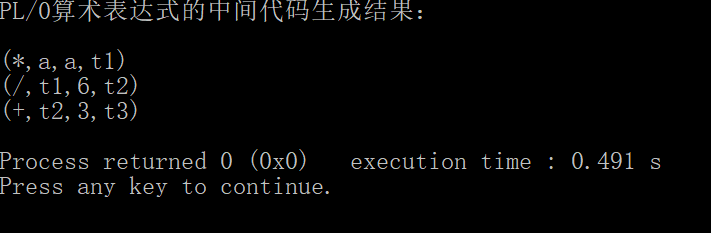


**9、测试样例9：**

**输入：**

a\*a/6+3

**输出：**

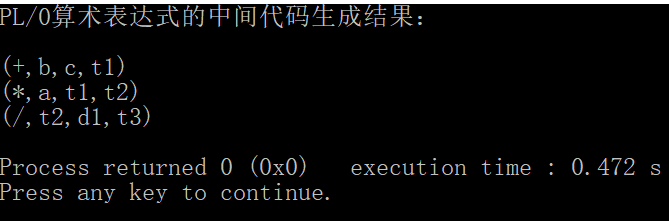


**10、测试样例10：**

**输入：**

a \* (b + c)/d1

**输出：**



## 实验体会：

这次实验是中间代码生成，建立在实验三语法分析的基础上，这次中间代码的实验只要求生成表达式的中间代码，用四元式序列表示，对于程序来说难点在于如何生成四元式，以及