 

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** | **实验四 语义分析** |
| **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

法正确的算术表达式，输出其计算值。

## 实验内容：

已给 PL/0 语言文法，在表达式的语法分析程序里，添加语义处理部分。

## 实验要求：

语义分析对象重点考虑经过语法分析后已是正确的语法范畴，实习

重点是语义子程序。

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。

计算表达式的语义值。

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：

算法描述如下：首先通过词法分析器，将输入的表达式转换成二元组的形式。使用递归调用，对项、因子、表达式进行分析。对于一个表达式，可以是一个单独一个项，也可是表达式+项或表达式-项，对于项的成分可以分为因子或者项\*因子或者项/因子。最后因子可以是带左右括号的表达式，或者标识符或者无符号整数。当需要对其某一个部分进行分析时调用递归分析的子程序。在分析过程中，在进行分析的过程中保存因子、项或者表达式的值，最后递归结束时，就可以最终计算出表达式的语义值。

## 实验代码：

C++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;  int expressionAnalysis2();//表达式分析  int termAnalysis2();//项分析  int factorAnalysis2();//因子分析  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-输出结果至result 二元式序列  {  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() //用来读入下一个符号  {  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;  }  //表达式的语义计算  int expressionAnalysis2() //表达式  {  string op;  int arg1,arg2,result;  if(conterr)  {  return 0;  }  arg1=termAnalysis2();//通过项分析得到第一个参数的值  if(conterr)  {  return 0;  }  while((sym=="plus")||(sym=="minus"))  {  op=sym2;  flag=advance();  if(conterr)  {  return 0;  }  if(flag==0)  {  cout<<"语法错误 <加法运算符>后缺项"<<endl;  conterr++;  return 0;  }  arg2=termAnalysis2();//通过项分析得到第二个参数的值  if(conterr)  {  return 0;  }  if(op=="+") //若是加法符号则进行加法运算，并对得到的结果进行保存  {  result=arg1+arg2;  arg1=result;  }  else //若是减法符号则进行加法运算，并对得到的结果进行保存,均为整数运算  {  result=arg1-arg2;  arg1=result;  }  }  return arg1;//返回该表达式所代表的值  }  int termAnalysis2() //项  {  string op;  int arg1,arg2,result;  arg1=factorAnalysis2();//通过因子分析得到第一个参数的值  if(conterr)  {  return 0;  }  while((sym=="times")||(sym=="slash"))  {  op=sym2;  flag=advance();  if(conterr)  {  return 0;  }  if(flag==0)  {  conterr++;  cout<<"语法错误 <乘法运算符>后缺因子"<<endl;  return 0;  }  if(conterr)  {  return 0;  }  arg2=factorAnalysis2();//通过因子分析得到第二个参数的值  if(conterr)  {  return 0;  }  if(op=="\*") //若是乘法符号则进行加法运算，并对得到的结果进行保存  {  result=arg1\*arg2;  arg1=result;  }  else //若是除法符号则进行加法运算，并对得到的结果进行保存  {  if(arg2==0)  {  conterr++;  cout<<"除数不能为0"<<endl;  return 0;  }  result=arg1/arg2;  arg1=result;  }  }  return arg1;//返回该项所代表的值  }  int factorAnalysis2()  {  int arg;  if(sym=="ident") //算数表达式中不含有字母，否则无法进行运算  {  cout<<"算术表达式中含有字母"<<endl;  conterr++;  return 0;  }  else if(sym=="number") //若果是数字，则返回相应的值  {  arg=atoi(sym2.c\_str());  flag=advance();  if(conterr)  {  return 0;  }  if(lpnum==0&&sym=="rparen")  {  conterr++;  cout<<"语法错误 ')'不匹配"<<endl;  return 0;  }  }  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 0;  }  if(flag==0)  {  conterr++;  cout<<"语法错误 '('后缺少表达式"<<endl;  return 0;  }  arg=expressionAnalysis2();//返回表达式的值  if(conterr)  {  return 0;  }  if(flag==0||sym!="rparen")  {  conterr++;  cout<<"语法错误 表达式后面缺少')'"<<endl;  return 0;  }  else  {  lpnum--;  flag=advance();  if(conterr)  {  return 0;  }  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 0;  }  return arg;//返回该因子所代表的值  }  int main()  {  int i=0,num,result;  //开始词法分析  map\_init();  lexanalysis();  cout<<"PL/0算术表达式的语义计算结果："<<endl<<endl;  flag=advance();  if(flag)  {  result=expressionAnalysis2();//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值  {  cout<<result<<endl;  }  }  else  {  cout<<"error!"<<endl;  return 0;  }  infile.close();  return 0;  } |

## 实验结果：

1、测试样例1：

输入：

12-2\*7/2+6

输出：11

2、测试样例2：

输入：

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

输出：

14

3、测试样例3：

输入：

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

输出：

28

4、测试样例4：

输入：

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

输出：

60

5、测试样例5：

输入：

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

输出：

除数不能为0

6、测试样例6：

输入：

2+3/

输出：

语法错误 <乘法运算符>后缺因子

7、测试样例7：

输入：

2+

输出：

语法错误 <加法运算符>后缺项

8、测试样例8：

输入：

(2+4\*5

输出：

语法错误 表达式后面缺少')'

9、测试样例9：

输入：

a\*a/6+3

输出：

算术表达式中含有字母

10、测试样例10：

输入：

-7+5\*6)

输出：

语法错误 因子首部不为<标识符>|<无符号整数>|'('

## 实验体会：