实验一 词法扫描器设计

**一 实验目的**

**通过设计调试词法分析程序，实现从源程序中分出各种单词的方法；加深对课堂教学的理解；提高词法分析方法的实践能力。**

**二 实验内容**

**设计一个简单的类C语言的词法扫描器。**

**三 实验要求**

**（一） 程序设计要求**

1. **根据附录给定的文法，从输入的类C语言源程序中，识别出各个具有独立意义的单词，即关键字、标识符、常数、运算符、分隔符五大类；文法见最后附录。**
2. **提供源程序输入界面；**
3. **词法分析后可查看符号表和TOKEN串表；**
4. **保存符号表和TOKEN串表（如：文本文件）；**
5. **遇到错误时可显示提示信息，然后跳过错误部分继续进行分析。**

**（二）实验报告撰写要求**

1. **系统功能分析设计（包括各个子功能模块的功能说明）；**
2. **设计方案；**
3. **功能模块结构图；**
4. **主程序流程图；**
5. **主要子程序的流程图（若有必要）；**
6. **主要数据结构：符号表、TOKEN串表等。**
7. **开发平台（操作系统、设计语言）；**
8. **具体实现（包括主控程序、各个功能模块的具体实现，给出主要函数部分即可，不用粘贴全部代码）。**
9. **实验总结**

**四 实验报告**

1. **系统功能分析**

词法扫描器的功能是将输入的源程序代码转换为TOKEN串，并在过程中生成符号表和TOKEN串表，供后续过程使用。一个词法分析器应该包含源程序扫描模块、词素分析模块、符号表生成模块和输入输出模块等。

1. **设计方案**
   * 1. **功能模块结构图**

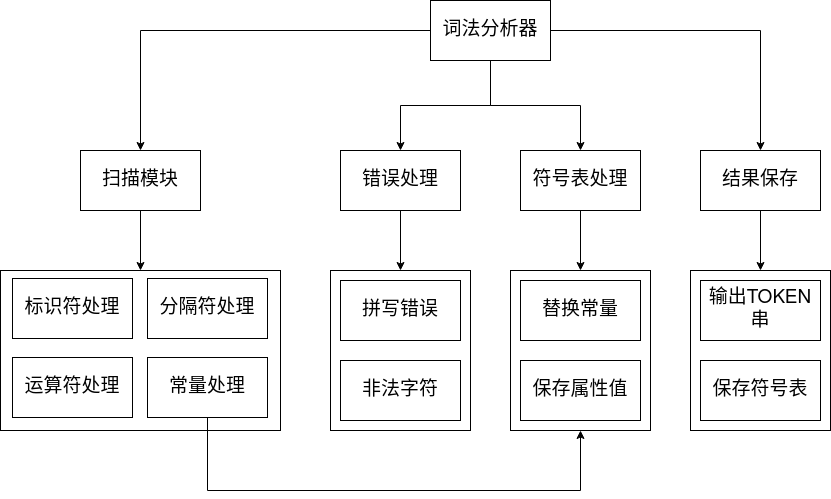
****

图 1 功能模块结构图

* + 1. **主程序流程图**

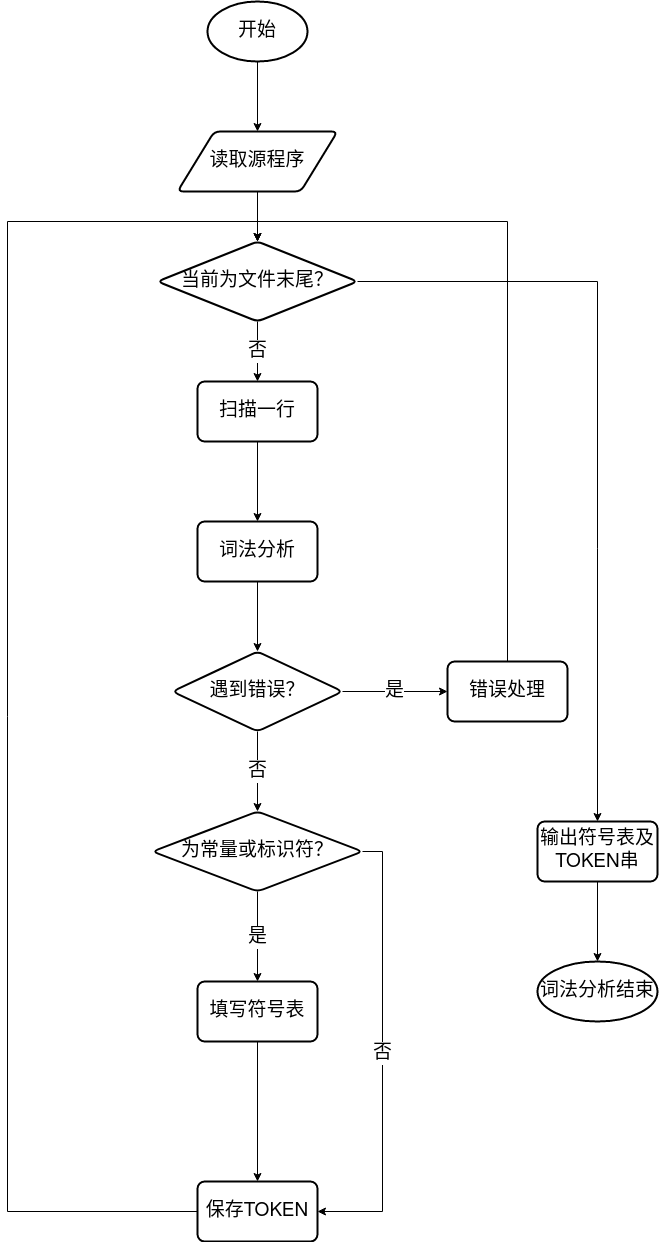
****

图 2 主程序流程图

* + 1. **扫描子程序流程图**

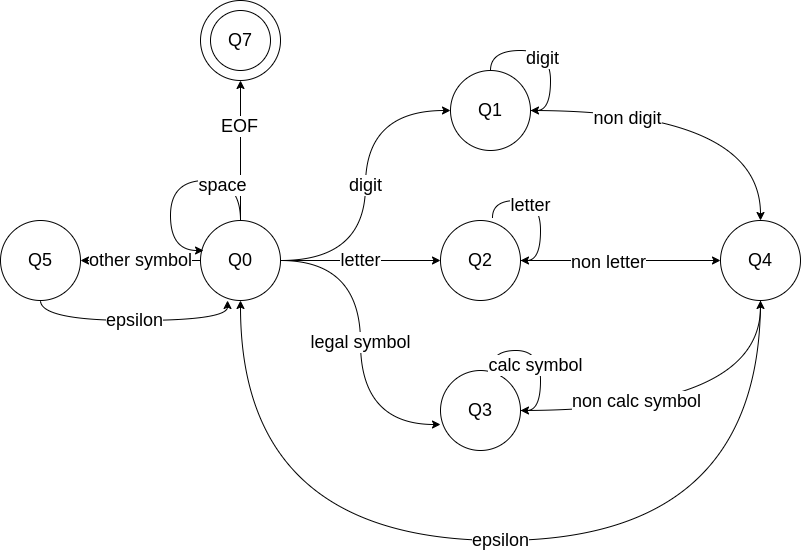
****

图 3 扫描子程序原理图

扫描子程序基于有穷状态自动机实现，根据当前所处的不同状态和下一个输入的字符判断下一步动作。自动机结构如上图，其中，Q0-Q5各状态定义如下

|  |  |
| --- | --- |
| **名称** | **描述** |
| Q0 | 开始状态 |
| Q1 | 处理数字 |
| Q2 | 处理单词 |
| Q3 | 处理运算符 |
| Q4 | 保存TOKEN，填写符号表 |
| Q5 | 错误处理 |
| Q6 | 结束 |

表 1 状态定义

* + 1. **主要数据结构**

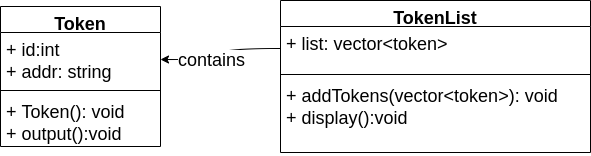
****

图 4 主要数据结构UML图

* + - * 1. **TOKEN类**

包含两个属性，种别码和value地址。对于关键字和运算符，value地址为空；对于常量（字符、数字）以及标识符，value地址为符号表中对应指针。Token()函数为类构造函数，output()函数用与将Token输出到控制台。

* + - * 1. **TOKENLIST类**

包含一个属性，Token列表。包含所有扫描到的Token对象。addToken()函数将一个Token对象添加到列表末端，display()函数调用输出所有Token对象的output()方法，将列表中所有Token对输出到控制台。

* + 1. **开发平台**
       - 1. **操作系统**

Ubuntu 20.04 LTS

* + - * 1. **设计语言**

C++

1. **程序实现**
   * 1. **主控程序**

int main*() {* TokenList tokenList;  
 **String** filename;  
 std::ifstream file;  
 **Vector***<***String***>* source;  
 while *(*true*) {* std::cout << "请输入目标文件名：" << std::endl;  
 std::cin >> filename;  
 file.open*(*filename*)*;  
 if *(*!file.is\_open*()) {* std::cout << "文件不存在或文件名不正确！" << std::endl;  
 *}* else break;  
 *}* std::cout << "#################################" << std::endl;  
 std::cout << "源代码:" << std::endl;  
 **String** s;  
 int count = 1;  
 while *(*getline*(*file, s*)) {* std::cout <<count<<" "<< s << std::endl;  
 count ++;  
 source.push\_back*(*s*)*;  
 *}* outputSymbol*()*;  
 for *(***Vector***<***String***>*::iterator it = source.begin*()*;it<source.end*()*;it++*) {* **String** t = \*it;  
 **Vector***<*Token*>* temp = scan*(*t,it-source.begin*())*;  
 tokenList.addTokens*(*temp*)*;  
  
 *}* std::cout << "#################################" << std::endl;  
 std::cout << "token串表" << std::endl;  
 tokenList.display*()*;  
 std::ofstream ofile*(*"token.txt"*)*;  
 **Vector**<std::pair<int,char\*>> otplist;  
 for *(*auto it = tokenList.list.begin*()*;it!=tokenList.list.end*()*;it++*){* ofile<<it->id<<' '<<it->token<<std::endl;  
 *}* if*(*!ofile.is\_open*()){* std::cout<<"fail"<<std::endl;  
 exit*(*1*)*;  
 *}* ofile.close*()*;  
 return 0;  
*}*

* + 1. **词法扫描子程序**

**Vector***<*Token*>* scan*(***String** target,int line*) {* **Vector***<*Token*>* tempV;  
 **String**::iterator it = target.begin*()*;  
 **String** t;  
 while *(*it != target.end*()) {* if *(*\*it == ' '*) {* it++;  
 *}* else *{* if *(*isDigit*(*\*it*)) {* while *(*isDigit*(*\*it*)) {* t += \*it;  
 it++;  
 *}* int id = keyword\_map.at*(*"digit"*)*;  
 Token token*(*id, t*)*;  
 tempV.push\_back*(*token*)*;  
 t.clear*()*;  
  
 *}* else if *(*isLetter*(*\*it*)) {* int id;  
 while *(*isLetter*(*\*it*)* || isDigit*(*\*it*)) {* t += \*it;  
 it++;  
 *}* if *(*keyword\_map.find*(*t*)* != keyword\_map.end*()) {* id = keyword\_map.at*(*t*)*;  
 *}* else *{* id = OP\_map.at*(*"id"*)*;  
 *}* Token token*(*id, t*)*;  
 tempV.push\_back*(*token*)*;  
 t.clear*()*;  
 *}* else *{* int id;  
 t = \*it;  
 it++;  
 if *(*OP\_map.find*(*t*)*!=OP\_map.end*()) {* id = OP\_map.at*(*t*)*;  
 *}* else *{* int pos = it - target.begin*()*;  
 std::cout<<"[InvalidSymbolError] illegal character at "<<line<<":"<<pos<<std::endl;  
 std::cout<<target<<std::endl;  
 **String** foo = **String***(*pos,' '*)* + "^";  
 std::cout<<foo<<std::endl;  
 id = 0;  
 *}* Token token*(*id, t*)*;  
 tempV.push\_back*(*token*)*;  
 t.clear*()*;  
 *}  
 }  
 }* return tempV;  
*}*

* + 1. **错误处理子程序**

int pos = it - target.begin*()*;  
 std::cout<<"[InvalidSymbolError] illegal character at "<<line<<":"<<pos<<std::endl;  
 std::cout<<target<<std::endl;  
 **String** foo = **String***(*pos,' '*)* + "^";  
 std::cout<<foo<<std::endl;  
 id = 0;  
 *}* Token token*(*id, t*)*;  
 tempV.push\_back*(*token*)*;  
 t.clear*()*;

1. **实验总结**

词法分析是编译器的第一个环节，实际上在此阶段完成的功能相对简单，但词法分析为后续的语法和语义分析提供了重要的基础，Token结构的合理构造和存储能够为语法分析提供极大的便利。

在词法分析过程中，遇到非法字符、拼写错误等问题时，通常采用恐慌模式进行处理。

实验二 LR语法分析器设计

**一 实验目的**

**通过设计调试LR语法分析程序，实现根据词法分析的TOKEN字，进行文法的语法分析；加深对课堂教学的理解；提高语法分析方法的实践能力。**

**二 实验内容**

**使用附录中的文法，可以对类似下面的程序语句进行语法分析：**

**int a;**

**int b;**

**int c;**

**a=2;**

**b=1;**

**if （a>b）**

**c=a+b;**

**else**

**c=a-b;**

**三 实验要求**

**（一） 程序设计要求（1）给出主要数据结构：分析栈、状态等；（2）将词法扫描器作为一个子程序，每次调用返回一个TOKEN；**

**（3）程序界面：基本要求包含源程序输入功能、语法分析的结果表示（文件或者图形**

**方式）；其他过程表示形式可以自行设计。**

**（二）实验报告撰写要求**

1. **系统功能分析设计（包括各个子功能模块的功能说明）；**
2. **设计方案；**
3. **功能模块结构图；**
4. **主程序流程图；**
5. **主要子程序的流程图（若有必要）；**
6. **主要数据结构：分析栈、状态、分析表等。**
7. **开发平台（操作系统、设计语言）；**
8. **具体实现（包括主控程序、各个功能模块的具体实现，给出主要函数部分即可，不用粘贴全部代码）。**
9. **实验总结（程序调试结果以及实验心得体会等）**

**四 实验报告**

1. **系统功能分析设计**

语法分析器主要包括状态生成处理、分析表构造、分析栈处理、主控程序、错误处理等模块。其中，状态生成处理模块的主要任务是将用户输入的语法规则进行预处理，生成相应的产生式和LR(0)项；分析表构造模块包括计算GO函数、计算项目闭包、构建FIRST集和FOLLOW集、构建SLR分析表等功能；分析栈处理模块根据当前栈顶状态、字符和SLR分析表决定下一步动作；主控程序调用词法分析程序产生的TOKEN串表进行语法分析。

1. **设计方案**
   * 1. **功能模块结构图**

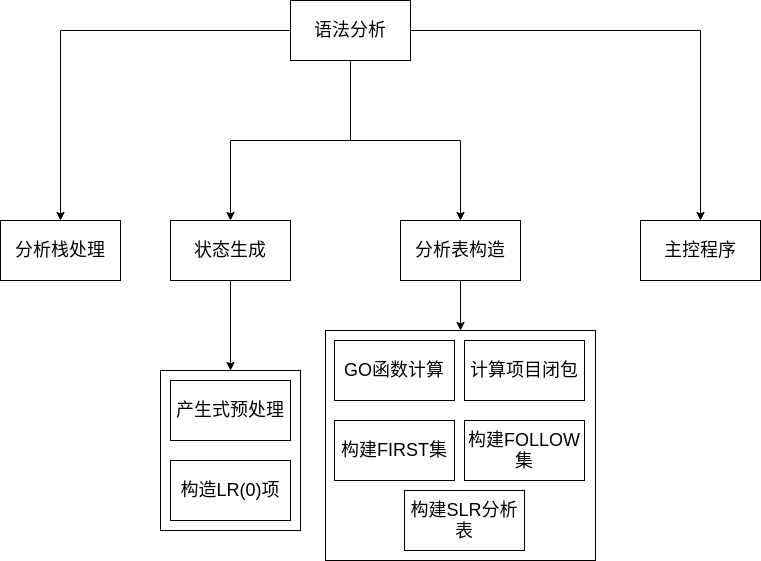
****

图 5 功能模块结构

* + 1. **主程序流程图**

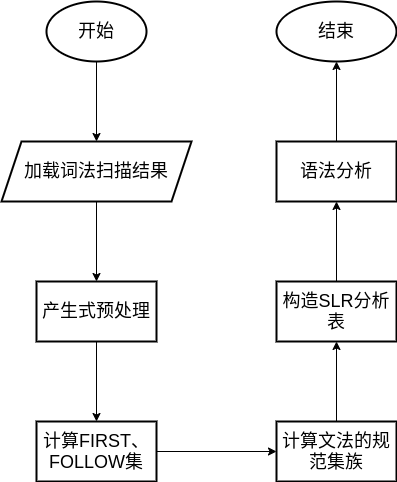
****

图 6 主程序流程图

* + 1. **主要数据结构**

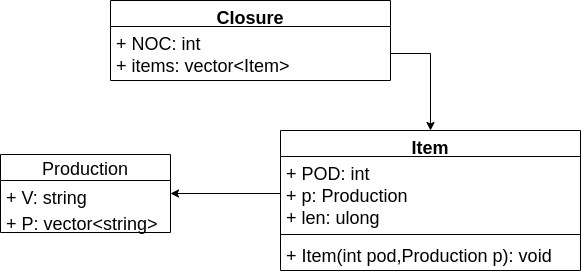
****

图 7 UML类图

* + - * 1. **First Set 和Follow Set**

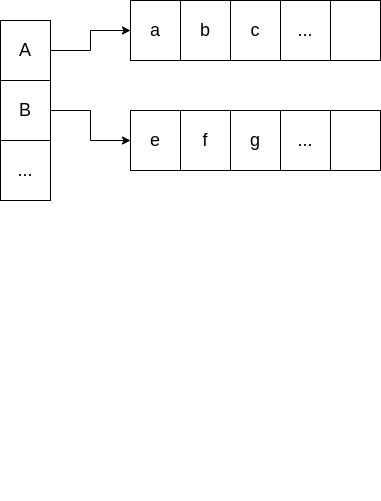
****

图 8 First集和Follow集

First集和Follow集采用字典的形式存储

* + - * 1. **Action Table 和 Goto Table**

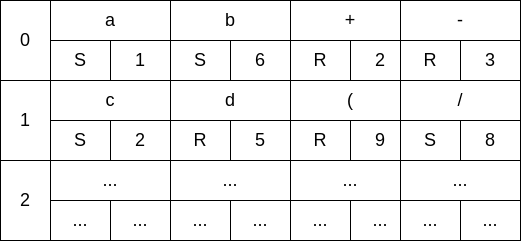
****

图 9 Action表和Goto表

Action表和Goto表使用列表+字典的形式进行存储，列表的每一项对应每个状态，字典元素对应该状态下接受key时进行的语法动作。

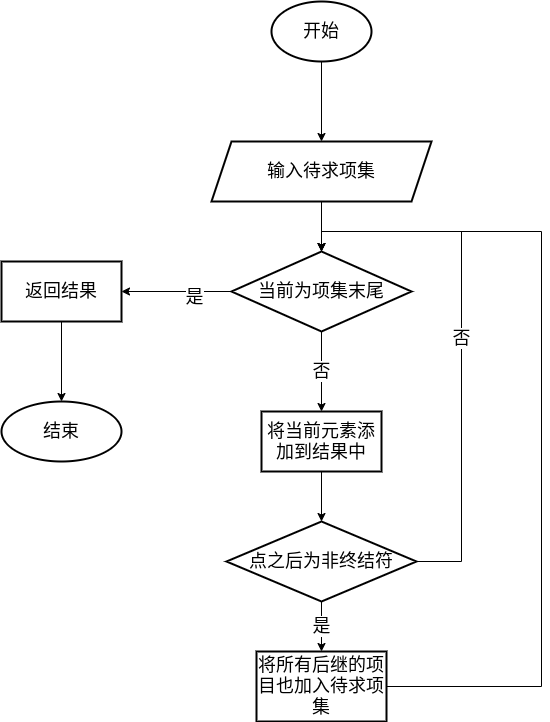
* + - * 1. **Closure、Item、Production**

Production结构体用于存储产生式，其中V表示产生式的左部，P表示产生是的右部。

Item类是LR(0)项的实现，POD表示项目中点的位置，p是项目对应的产生式，len是产生式右部的长度。

Closure类是项集闭包的实现，NOC表示Closure的编号，items是项集包含所有项目的列表。

* + - * 1. **状态栈、符号栈**
    1. **主要子程序流程图**
       - 1. **计算项集闭包**

****图 10 计算项集闭包

* + 1. **开发环境**
       - 1. **操作系统**

Ubuntu 20.04 LTS

* + - * 1. **设计语言**

C++%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-400%22%20y%3D%22-200%22%20width%3D%22470%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22470%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%225%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%226%22%20value%3D%22a%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%227%22%20value%3D%22S%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%228%22%20value%3D%221%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%229%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22120%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2210%22%20value%3D%22b%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2211%22%20value%3D%22S%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2212%22%20value%3D%226%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2213%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22240%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2214%22%20value%3D%22%2B%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2215%22%20value%3D%22R%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2216%22%20value%3D%222%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2217%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22350%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2218%22%20value%3D%22-%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2217%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2219%22%20value%3D%22R%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2217%22%3E%3CmxGeometry%20y%3CmxGraphModel%3E%3Croot%3E%3CmxCell%20id%3D%220%22%2F%3E%3CmxCell%20id%3D%221%22%20parent%3D%220%22%2F%3E%3CmxCell%20id%3D%222%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%221%22%3E%3CmxGeometry%20x%3D%22-400%22%20y%3D%22-200%22%20width%3D%22470%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%223%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22470%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%224%22%20value%3D%22%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%225%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%226%22%20value%3D%22a%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%227%22%20value%3D%22S%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%228%22%20value%3D%221%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%225%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%229%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22120%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2210%22%20value%3D%22b%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2211%22%20value%3D%22S%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2212%22%20value%3D%226%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%229%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2213%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22240%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2214%22%20value%3D%22%2B%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20width%3D%22120%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2215%22%20value%3D%22R%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2216%22%20value%3D%222%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2213%22%3E%3CmxGeometry%20x%3D%2260%22%20y%3D%2240%22%20width%3D%2260%22%20height%3D%2240%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2217%22%20value%3D%22%22%20style%3D%22group%22%20vertex%3D%221%22%20connectable%3D%220%22%20parent%3D%222%22%3E%3CmxGeometry%20x%3D%22350%22%20width%3D%22120%22%20height%3D%2280%22%20as%3D%22geometry%22%2F%3E%3C%2FmxCell%3E%3CmxCell%20id%3D%2218%22%20value%3D%22-%22%20style%3D%22rounded%3D0%3BwhiteSpace%3Dwrap%3Bhtml%3D1%3BfontSize%3D18%3B%22%20vertex%3D%221%22%20parent%3D%2217%22%3E%3CmxGeometry%20width%

1. **程序实现**
   * 1. **主控程序**

int main*() {* vector*<*string*>* symbol\_table;  
 vector*<*Production*>* p;  
 vector*<*Item*>* i;  
  
 vector<pair<int, string >> tokenlist;  
 tokenlist = LoadTokenList*()*;  
  
 ifstream file;  
 file.open*(*"production.txt"*)*;  
 if *(*!file.is\_open*()) {* cout << "error";  
 exit*(*0*)*;  
 *}* p = Production\_Pre\_Process*(*&file*)*;  
 i = Create\_Item\_List*(*p*)*;  
  
 GenerateSymbolTable*(*symbol\_table*)*;  
 for *(*auto it = NonTerminalMap.begin*()*; it != NonTerminalMap.end*()*; it++*) {* GenerateFirstSet*(*FirstSet, it->first, p*)*;  
 *}* for *(*auto it = NonTerminalMap.begin*()*; it != NonTerminalMap.end*()*; it++*) {* GenerateFollowSet*(*FollowSet, it->first, p*)*;  
 *}* SymbolStack.push*(*"$"*)*;  
 StatusStack.push*(*0*)*;  
 *//Compute Closure* Closure i\_0, i1;  
 list*<*Item*>* ll = *{*i[0]*}*;  
 i\_0 = Compute\_Closure\_I*(*closure\_cnt, ll, i*)*;  
 list*<*Closure*>* g;  
 g.push\_back*(*i\_0*)*;  
 for *(*auto it = g.begin*()*; it != g.end*()*; it++*) {* for *(*auto it\_ = symbol\_table.begin*()*; it\_ != symbol\_table.end*()*; it\_++*) {* Go*(*\*it, \*it\_, i, g*)*;  
 *}  
 }  
 //SLR Table* for *(*auto iter = g.begin*()*; iter != g.end*()*; iter++*) {* GenerateAnalyzeTable*(*\*iter, i, p*)*;  
 *}*

*Analyze();* return 0;  
*}*

* + 1. **语法分析子程序**

void Analyze(){

auto it = tokenlist.begin*()*;  
 while *(*true*) {* int curstate = StatusStack.top*()*;  
 string input = keyword\_map.at*(*it->first*)*;  
 if *(*ActionTable[curstate][input].first == "S"*) {* StatusStack.push*(*ActionTable[curstate][input].second*)*;  
 SymbolStack.push*(*input*)*;  
 it++;  
 *}* else if *(*ActionTable[curstate][input].first == "R"*) {* Production t = p[ActionTable[curstate][input].second];  
 for *(*int iter = 0; iter < t.P.size*()*; iter++*) {* SymbolStack.pop*()*;  
 StatusStack.pop*()*;  
 *}* curstate = StatusStack.top*()*;  
 int next = SearchCache*(*curstate, t.V*)*;  
 StatusStack.push*(*next*)*;  
 SymbolStack.push*(*t.V*)*;  
 Display*(*t*)*;  
 *}* else if *(*ActionTable[curstate][input].first == "acc"*) {* cout << "accepted!";  
 break;  
 *}* else *{* cout << "err";  
 break;  
 *}  
 }  
}*

* + 1. **加载词法分析结果**

*/\*\*  
 \* @brief load token list from text file  
 \* @return token list in order  
 \*/*vector<pair<int, string>> LoadTokenList*() {* vector<pair<int, string >> data;  
 ifstream ifile*(*"token.txt"*)*;  
 string p;  
 while *(*getline*(*ifile, p*)) {* vector*<*string*>* t;  
 t = split*(*p*)*;  
 pair*<*int, string*>* q*(*atoi*(*t[0].c\_str*())*, t[1]*)*;  
 data.push\_back*(*q*)*;  
 *}* data.push\_back*(*pair*<*int, string*>(*0, "$"*))*;  
 return data;  
*}*

* + 1. **产生式预处理和构造项目集**

*/\*\*  
 \* @brief pre-process the productions, read productions from text file and generate Production list  
 \* @param file production file pointer  
 \* @return list of productions  
 \*/*vector*<*Production*>* Production\_Pre\_Process*(*ifstream \*file*) {* vector*<*Production*>* p\_list;  
 string p;  
 while *(*getline*(*\*file, p*)) {* Production t;  
 t.V = p[0];  
 NonTerminalMap.insert*(*pair*<*string, int*>(*p.substr*(*0, 1*)*, 1*))*;  
 t.P = split*(*p.substr*(*6*))*;  
 for *(*auto it = t.P.begin*()*; it != t.P.end*()*; it++*) {* if *((*\*it*)*[0] > 'Z' || *(*\*it*)*[0] < 'A'*) {* TerminalMap.insert*(*pair*<*string, int*>(*\*it, 1*))*;  
 *}  
 }* p\_list.push\_back*(*t*)*;  
  
 *}* return p\_list;  
*}*

*/\*\*  
 \* @brief generate items of all the productions  
 \* @param p\_list list of productions  
 \* @return list of items  
 \*/*vector*<*Item*>* Create\_Item\_List*(*vector*<*Production*>* p\_list*) {* vector*<*Item*>* item\_list;  
 for *(*auto it = p\_list.begin*()*; it != p\_list.end*()*; it++*) {* for *(*int i = 0; i <= it->P.size*()*; i++*) {* if *(*it->P[0] != "e"*) {* Item t*(*i, \*it*)*;  
 item\_list.push\_back*(*t*)*;  
 *}  
 }  
 }* Production production;  
 production.V = "D";  
 production.P = *{*"e"*}*;  
 Item t*(*0, production*)*;  
 item\_list.push\_back*(*t*)*;  
 return item\_list;  
*}*

* + 1. **计算项目闭包**

*/\*\*  
 \* @brief compute the closure of given set of Items  
 \* @param num\_of\_closure number of the result Closure  
 \* @param target\_list list of target Items  
 \* @param global\_list list of all Items  
 \* @return Closure of given set of Items  
 \*/*Closure Compute\_Closure\_I*(*int num\_of\_closure, list*<*Item*>* target\_list, vector*<*Item*>* &global\_list*) {* Closure c;  
 vector*<*Item*>* t;  
 c.num\_of\_closure = num\_of\_closure;  
 */\*\* through the target\_list of Items \*/* for *(*auto it = target\_list.begin*()*; it != target\_list.end*()*; it++*) {* t.push\_back*(*\*it*)*;  
 */\*\* exclude Items like A->αB. \*/* if *(*it->position\_of\_dot != it->len*) {  
 /\*\* if the symbol X after the dot is Non Terminal\*/* if *(*IsNonTerminal*(*it->p.P[it->position\_of\_dot]*)) {  
 /\*\* search all Items for Items looks like X->α.βγ,no matter where the dot is \*/* for *(*auto it\_ = global\_list.begin*()*; it\_ != global\_list.end*()*; it\_++*) {* if *(*it\_->position\_of\_dot == 0 && it\_->p.V == it->p.P[it->position\_of\_dot]*) {  
 /\*\* if the symbol Y after the dot is Non Terminal \*/* if *(*IsNonTerminal*(*it\_->p.P[it\_->position\_of\_dot]*)) {  
 /\*\* if the Item does not belong to target\_list, add it to the end and keep the computation \*/* if *(*find*(*target\_list.begin*()*, target\_list.end*()*, \*it\_*)* == target\_list.end*())* target\_list.push\_back*(*\*it\_*)*;  
 *}* else if *(*find*(*t.begin*()*, t.end*()*, \*it\_*)* == t.end*())* t.push\_back*(*\*it\_*)*;  
 *}  
 }  
 }  
 }  
 }* c.items = t;  
 return c;  
*}*

* + 1. **计算项目规范集族**

*/\*\*  
 \* @brief the goto function, computes the next Closure when the given Closure reads a string x  
 \* @param i current Closure  
 \* @param x input string  
 \* @param global list of all Items  
 \* @param closure\_set set of all Closures  
 \*/*void Go*(*Closure i, const string &x, vector*<*Item*>* &global, list*<*Closure*>* &closure\_set*) {* list*<*Item*>* t;  
 */\*\* find every Item in i that would take x as the next input \*/* for *(*auto it = i.items.begin*()*; it != i.items.end*()*; it++*) {* if *(*it->position\_of\_dot != it->len && it->p.P[it->position\_of\_dot] == x*) {* Item \_t*(*it->position\_of\_dot + 1, it->p*)*;  
 t.push\_back*(*\_t*)*;  
 *}  
 }* if *(*!t.empty*()) {  
 /\*\* compute the Closure of the selected Items \*/* Closure ii = Compute\_Closure\_I*(*++closure\_cnt, t, global*)*;  
 auto it = find*(*closure\_set.begin*()*, closure\_set.end*()*, ii*)*;  
 */\*\* if ii is not in closure\_set, add it to the end \*/* if *(*it == closure\_set.end*()) {* closure\_set.push\_back*(*ii*)*;  
 */\*\* store the state transition \*/* cache.emplace\_back*(*i.num\_of\_closure, pair*<*string, int*>(*x, ii.num\_of\_closure*))*;  
 *}* else *{* closure\_cnt--;  
 cache.emplace\_back*(*i.num\_of\_closure, pair*<*string, int*>(*x, it->num\_of\_closure*))*;  
 *}  
 }  
}*

* + 1. **构造First集和Follow集**

void GenerateFirstSet*(*map<string, vector<string>> &firstset, const string &target, vector*<*Production*>* &plist*) {* vector*<*Production*>* t;  
 vector*<*string*>* s;  
 */\*\* find all the productions like target->αβ \*/* for *(*auto it = plist.begin*()*; it != plist.end*()*; it++*) {* if *(*it->V == target*) {* t.push\_back*(*\*it*)*;  
 *}  
 }* for *(*auto it = t.begin*()*; it != t.end*()*; it++*) {  
 /\*\* if the first symbol is Non Terminal \*/* if *(*IsNonTerminal*(*it->P[0]*)) {* if *(*it->P[0] != target*) {  
 /\*\* recursively compute the First Set of the first symbol \*/  
 /// should've do check for whether the computation is done earlier here,  
 /// but i got kinda lazy so here we are, no check  
 /// however, you don't need to be worried if this could lead to error. the insert method of std::map  
 /// would not cover the original contents if there already exists a specific key. so this would do no harm* GenerateFirstSet*(*firstset, it->P[0], plist*)*;  
 Merge*(*s, firstset[it->P[0]]*)*;  
 *}  
 /// magic trick alert! this is only to deal with the production P->DS, since this is the only production  
 /// that have 2 continues Non Terminal symbols in the right part.* if *(*it->P.size*()* > 1 && IsNonTerminal*(*it->P[1]*)) {* if *(*it->P[1] != target*) {* GenerateFirstSet*(*firstset, it->P[1], plist*)*;  
 Merge*(*s, firstset[it->P[1]]*)*;  
 *}  
 }  
 }* else *{* Merge*(*s, it->P[0]*)*;  
 *}  
 }* firstset.insert*(*pair<string, vector<string>>*(*target, s*))*;  
*}  
/\*\*  
 \* @brief generate the follow set of the given Non Terminal symbol  
 \* @param followset map of all follow sets  
 \* @param target target symbol  
 \* @param plist all the productions  
 \*/*void GenerateFollowSet*(*map<string, vector<string>> &followset, const string &target, vector*<*Production*>* &plist*) {* vector*<*Production*>* target\_productions;  
 vector*<*string*>* results;  
 */// magic trick alert! obviously this is just me being lazy again* if *(*target == "P"*) {* string s = "$";  
 Merge*(*results, s*)*;  
 *}  
 /\*\* find all productions that looks like \*->α target β \*/* for *(*auto it = plist.begin*()*; it != plist.end*()*; it++*) {* auto t\_it = find*(*it->P.begin*()*, it->P.end*()*, target*)*;  
 if *(*t\_it != it->P.end*()) {* target\_productions.push\_back*(*\*it*)*;  
 *}  
 }* for *(*auto it = target\_productions.begin*()*; it != target\_productions.end*()*; it++*) {  
 /\*\* find the position of target in the production \*/* auto t\_it = find*(*it->P.begin*()*, it->P.end*()*, target*)*;  
 */\*\* if the production look like \*->α target β \*/* if *(*t\_it - it->P.begin*()* < it->P.size*()* - 1*) {  
 /\*\* next symbol is a Non Terminal symbol, production like \*->αAB \*/* if *(*IsNonTerminal*(*\**(*t\_it + 1*))) {  
 /\*\* put every item in first(B) into follow A \*/* vector*<*string*>* tmp = FirstSet[\**(*t\_it + 1*)*];  
 */\*\* exclude epsilon \*/* auto tt\_it = find*(*tmp.begin*()*, tmp.end*()*, "e"*)*;  
 if *(*tt\_it != tmp.end*()) {* tmp.erase*(*tt\_it*)*;  
 */\*\* put every item in follow A into follow B \*/* if *(*followset.find*(*it->V*)* == followset.end*())* GenerateFollowSet*(*followset, it->V, plist*)*;  
 Merge*(*results, followset[it->V]*)*;  
 *}* Merge*(*results, tmp*)*;  
 *}* else *{  
 /\*\* next symbol is Terminal \*-αAβ \*/* Merge*(*results, \**(*t\_it + 1*))*;  
 *}  
 }* else *{* if *(*it->V != target*) {* if *(*followset.find*(*it->V*)* == followset.end*()) {* GenerateFollowSet*(*followset, it->V, plist*)*;  
 *}* Merge*(*results, followset[it->V]*)*;  
 *}  
 }  
 }* followset.insert*(*pair<string, vector<string>>*(*target, results*))*;  
*}*

* + 1. **构造SLR分析表**

*/\*\*  
 \* @param c Closure to compute  
 \* @param global global Item list  
 \* @param productions global production list  
 \*/*void GenerateAnalyzeTable*(*Closure &c, vector*<*Item*>* &global, vector*<*Production*>* &productions*) {* map<string, pair<string, int>> p, q;  
 ActionTable.push\_back*(*p*)*;  
 GotoTable.push\_back*(*q*)*;  
 for *(*auto it = c.items.begin*()*; it != c.items.end*()*; it++*) {* if *(*it->p.V == "P"*) {* ActionTable[c.num\_of\_closure]["$"] = pair*(*"acc", -1*)*;  
 *}* if *(*it->position\_of\_dot != it->len*) {* string tmp = it->p.P[it->position\_of\_dot];  
 int i = SearchCache*(*c.num\_of\_closure, tmp*)*;if *(*!IsNonTerminal*(*tmp*)) {* if *(*tmp == "e"*) {* for *(*auto iterator = TerminalMap.begin*()*; iterator != TerminalMap.end*()*; iterator++*) {* auto iter = find*(*productions.begin*()*, productions.end*()*, it->p*)*;  
 pair*<*string, int*>* pair1*(*"R", iter - productions.begin*()* - 1*)*;  
 ActionTable[c.num\_of\_closure].insert*(* pair<string, pair<string, int>>*(*iterator->first, pair1*))*;  
 *}  
 }* else *{* ActionTable[c.num\_of\_closure][tmp] = pair*(*"S", i*)*;  
 *}  
 }* else *{* GotoTable[c.num\_of\_closure][tmp] = pair*(*"S", i*)*;  
 *}  
 }* else *{* string tmp = it->p.V;  
 for *(*auto iterator = FollowSet[tmp].begin*()*; iterator != FollowSet[tmp].end*()*; iterator++*) {* auto iter = find*(*productions.begin*()*, productions.end*()*, it->p*)*;  
 ActionTable[c.num\_of\_closure][\*iterator] = pair*(*"R", iter - productions.begin*())*;  
 *}  
 }  
 }  
}*

1. **实验总结**

语法分析过程是整个编译过程中最重要的一环，也是难度最大的一环。语法分析的重点是根据输入的语法规则构建语法分析表。在这个过程中，构造First、Follow集、构造SLR分析表、计算规范集族都是不小的挑战。

实验三 语义分析及中间代码生成

**一 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。**

**二 实验内容**

**实现简单的高级语言源程序的语义处理过程。**

**三 实验要求**

**（一） 程序设计要求**

* + 1. **目 标 机：8086及其兼容处理器**
    2. **中间代码：三地址码**
    3. **主要数据结构：三地址码表、符号表**
    4. **语义分析内容要求：**

1. **变量说明语句**
2. **赋值语句**
3. **控制语句(任选一种)**
   * 1. **其它要求：**
4. **将词法分析（扫描器）作为子程序，供语法、语义程序调用；**
5. **使用语法制导的语义翻译方法；**
6. **编程语言自定；**
7. **提供源程序输入界面；**
8. **目标代码生成暂不做；**
9. **编译后，可查看TOKEN串、符号表、三地址码表；**
10. **主要数据结构：产生式表、符号表、三地址码表。**

**所用文法使用实验2中的文法。**

**附录：语义分析源程序范例**

**int a;**

**int b;**

**int c;**

**a=2;**

**b=1;**

**if （a>b）**

**c=a+b;**

**else**

**c=a-b;**

**（二）实验报告撰写要求**

1. **系统功能分析与设计（包括各个子功能模块的功能说明）；**
2. **开发平台（开发软硬件环境）；**
3. **语义翻译中使用的数据结构；**
4. **程序具体设计实现过程（包括主要功能模块的具体实现）。**

**四 实验报告**

1. **系统功能分析与设计**

语义分析是在语法分析的基础上，在进行规约时生成三地址代码，并进行类型和数值检测。在语法分析的基础上，增加中间代码产生、语义检查等功能。

1. **开发平台**
   * 1. **操作系统**

Ubuntu 20.04 LTS

* + 1. **设计语言**

C++

1. **主要数据结构**
   * 1. **中间代码表**

使用中间代码表记录语法分析过程中产生的中间代码，方便进行回填操作。

* + 1. **SDD**

1. **具体实现过程**

**语义分析在语法分析基础上进行，因此主要代码在语法分析部分已经说明，此处不再介绍。**