编译原理实验报告

实验名称： 编写语法分析程序

实验类型： 设计型实验

实验成绩：

日期： 2017年 5月 4日

实验二 编写语法分析程序

**一、实验目的**

通过设计、编写、调试一个LL（1）语法分析程序，实现对词法分析程序所提 供的单词序列进行语法检查和结 构分析，掌握LL（1）语法分析方法。理论上验证 编译原理，并获得深刻和系统的理解，在实践上，掌握使用实验环境的仅能技巧 以及程序的调试方法。

**二、实验设计**

1. 简述LL(1)分析的基本原理.
2. 递归下降分析首先是对文法的理解与改造，去掉左递归，左公因子使文法成为LL(1)文法，或为保持文法的可读性或其他原因保留一些不符合LL( 1)文法的正则表达式,但程序中需特殊处理。
3. 递归下降分析包含很多子程序，每个子程序对应相应的正则表达式，以判断当前状态下语法分析程序程序状态的走向。
4. 构造LL（1）分析表，分析每一个表达式的FIRST FOLLOW集，对FIRST(A)的每个终极符a pop()状态集 加入该表达式到状态集；若时，对每个DOLLOW(A)的每个符号b（终极符或#号）对状态集pop();对于终结符状态与终结符匹配则直接出状态，若为#号匹配时则接受。
5. 对于当前状态下若对于分析字符没有走向则表示分析出现了错误！
6. 简述LL(1)程序设计的基本过程
7. 改写文法，消除左递归，提取左公因子（对没有处理的地方做超前读取或标记的方法解决）
8. 求FIRST集，FOLLOW集 仅仅求取first集合中有ε的非终结符号的follow集合。
9. 检查文法是不是LL(1)文法，对不是的地方设计特殊处理，检查FIRST,FOLLOW集
10. 根据FIRST,FOLLOW集构造LL(1)分析表。
11. 根据LL(1)分析表构造程序。
12. 运行，检测，修改程序。
13. 本实验的设计过程：改写文法，右部符号串的FIRST集和相关非终结符的FOLLOW集，产生式的特殊处理方法及理由。

文法改写部分：

1. <declaration\_list>::=<declaration\_list><declaration\_stat> | ε

改：消除左递归。

<declaration\_list>::=<declaration\_list1>

<declaration\_list1>::=<declaration\_stat><declaration\_list1>| ε

总结为：<declaration\_list>::= <declaration\_stat><declaration\_list>|ε

1. <statement\_list>::=<statement\_list><statement>| ε

改：消除左递归

<statement\_list>::=<statement\_list1>

<statement\_list1>::=<statement> <statement\_list1>| ε

总结为：

<statement\_list>::=<statement><statement\_list>| ε

1. <if\_stat>::=if (<expression>) <statement >| if (<expression>) <statement >else < statement >

不能提取左公因子因为

假设提取 <if\_stat>::= if (<expression>) <statement ><if\_stat1>

<if\_stat1>::=ε|| else < statement >

其中FOLLOW(if\_stal1)={if,while,for,write,read,{,;,ID,NUM,(,}，else }，而FIRST(<if\_stat1>)={ε, else }出现交集else不满足LL(1)文法。

所以对起做特殊处理：用一个标记flag代替else < statement > 当读到该标记时如果为终结符else表示应走if (<expression>) <statement >else < statement > pop()状态集 加入<statement> NEXT 否则直接pop()状态集。

1. <bool\_expr>::=<additive\_expr> |<dditive\_expr >(>|<|>=|<=|==|!=)<additive\_expr >

改：提取公共左公因子。

<bool\_expr>::=<additive\_expr><bool\_expr1>

<bool\_expr1>::=ε|(>|<|>=|<=|==|!=)<additive\_expr>

1. < additive\_expr>→< additive\_expr>+<term>|< additive\_expr>-<term>|< term >

改：消除左递归。

<additive\_expr>::=<term><additive\_expr1>

<additive\_expr1>::=(+|-)< term >< additive\_expr1>|ε

1. < term >→< term >\*<factor>|< term >/<factor>|< factor >

改：消除左递归

<term>::=<factor><term1>

<term1>::=(\*| /)<factor><term1>|ε

1. <expression>::= ID=<bool\_expr>|<bool\_expr>

存在隐式左公因子ID;

采用超前读一个符号来解决，避免对该规则进行改写，保持文法的可读性.

若下一个符号式’=’表示走用前一个式子否则走后面个式子。

右部FIRST集：

1. FIRST({<declaration\_list><statement\_list>})={ { }
2. FIRST(**<declaration\_stat><declaration\_list>**)=First(**<declaration\_stat>)**={int}

FIRST(**ε**)={ε}

1. FIRST(int ID;)={int}
2. FIRST(**<statement><statement\_list>**)=FIRST(<statement>)=

{if,while,for,write,read,{,;,ID,NUM,(}

FIRST(ε)={ε}

1. FIRST(<if\_stat>|<while\_stat>|<for\_stat>|<read\_stat> |<write\_stat>|<command\_stat> |<expression\_stat>)=FIRST(<if\_stat>)+FIRST(<while\_stat>)+FIRST(<for\_stat>)+FIRST(<read\_stat>)+FIRST(<write\_stat>)+FIRST(<command\_stat>)+FIRST(<expression\_stat>)={if,while,for,write,read,{,;,ID,NUM,(}

因为太多和在一起了

1. FIRST(**if(<expression>)<statement >**)={if}

FIRST(if (<expression>) <statement >else < statement >)={if}

1. FIRST( while(<expression>)< statement >)={while}
2. FIRST(for(<expression>;<expression>;<expression>)<statement>)={for}
3. FIRST(write <expression>;)={write}
4. FIRST(read ID;)={read}
5. FIRST({<statement\_list>})={{}
6. FIRST(< expression >;)=FIRST{< expression >}={ID,(,NUM}

FIRST(;)={;}

1. FIRST(ID=<bool\_expr>)={ID}={ID}

FIRST(<bool\_expr>)={ID,(,NUM}

1. FIRST(**<additive\_expr><bool\_expr1>**)=FIRST(**<additive\_expr>**)={ID,(,NUM}
2. FIRST(**ε|(>|<|>=|<=|==|!=)<additive\_expr>**)={>,<,>=,<=,==,!=,ε}

因为多和在一起表示，LL购建表时分开

1. **FIRST(<term><additive\_expr1>)=FIRST(<term>)=**{ID,(,NUM}
2. **FIRST((+< term >< additive\_expr1>)={+}**

**FIRST(-< term >< additive\_expr1>)={-}**

**FIRST（ε）={ε}**

1. **FIRST(<factor><term1>)=FIRST(factor)=**{ID,(,NUM}
2. **FIRST（/<factor><term1>）={/}**

**FIRST(\*<factor><term1>)={\*}**

**FIRST(ε)={ε}**

1. **FIRST(NUM)={NUM}**

**FIRST(**(**< expression >))={(}**

**FIRST(ID)={ID}**

**需要的FOLLOW集：**

1.FOLLOW(declaration\_list)={if,while,for,write,read,{,;,ID,NUM,(,}}

2.FOLLOW(<statement\_list>)={}}

4.FOLLOW( <bool\_expr1>)={ ; ,)};

5.FOLLOW(additive\_expr1)=FOLLOW(<additive\_expr>) U FOLLOW(<bool\_expr>) ={>, <, >=, <=, ==, !=, ; , )}

6.FOLLOW(<term\_1>)=FOLLOW(<term>)=FIRST(<additive\_expr\_1>)U FOLLOW(< additive\_expr\_1>)={+, -, >, <, >=, <=, ==, !=, ; , )}

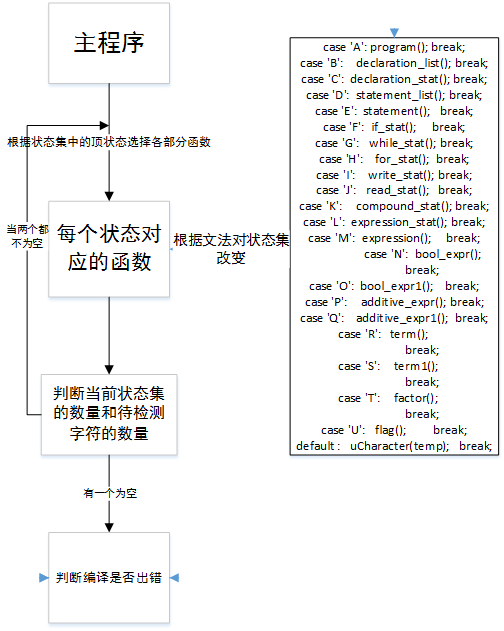
LL（1）分析表： 见附录（2）

1. 分析程序的设计框架，错误信息的输出等

设计框架：首先由于对终结符与非终结符状态，如果用本身来表示状态有的会过长，所以给超过一个字符的用一个标记符来表示。

主要就是对各个状态按照LL(1)分析表对状态集进行改变，终结符状态一个函数，其他状态各一个函数

分析结束后按照过程是否出错，状态集是否为空判断语法分析是否成功。



4.1程序大致结构框图。

错误信息的输出：

Line 4 2 error!

原因int后的应该式ID

Line 6 for error!

Line 6 ( error!

Line 6 i error!

Line 6 = error!

Line 6 1 error!

Line 6 i error!

Line 6 = error!

Line 6 i error!

Line 6 ) error!

Line 7 { error!

5行少了；号所以后期都入‘；’状态就会判错。

Line 10 b error!

Line 10 = error!

Line 10 b error!

10行缺少‘；’号引起的。

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

主要式因为while(x<=)

C=a+b+(x\*y;

两个错误缺少<=后的式子和（x\*y缺少‘）’引起的。

**三、实验过程**

1、完成整个实验的先后步骤

1. 观察式子改写文法，提取左公因子，消除左递归。
2. 对没有提左公因子，消除左递归却又存在的分析做特殊处理方式。
3. 写右部各式子的FIRST集，对FIRST中包含ε的，对左部非终结符求FOLLOW集。
4. 根据FIRST集FOLLOW集设计LL(1)分析表，
5. 根据LL(1)分析表做代码实现。
6. 找多组测试用例进行分析，对输出做分析判断程序正确性，修改程序。
7. 撰写报告。

2、实验调试记录（问题表现，分析原因，解决方案，解决结果）

（1）问题表现：对于正确的句子会出现初错误的分析，当出现比较句子比较后会出错。

分析原因:有可能状态集在转换在某出出现问题。

解决方案：将状态变换输出观察。

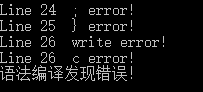
解决结果：发现不是代码写错而是文法在开始时不小心写错了

<bool\_expr1>::=ε|(>|<|>=|<=|==|!=)<additive\_expr>

写成了<bool\_expr1>::=ε|(>|<|>=|<=|==|!=)<additive\_expr1>

更改后分析正确。

（2）问题表现：对于最后一个字符没有分析到导致出现错误。



缺少‘}’

分析原因: 可能是对词的空判断提前了，最后一个字符没有做分析。

解决方案： 将对词的空判断放到while后的第一条语句；

解决结果： 语法分析完整进行。

**四、实验结果**

列出实验结果并进行分析（含分步测试结果）。

{

int a;

int i;

int 2b;

int c

for (i=1; i <= 10 i=i+1)

{

;

a=a+i

b=b\*i;

{

c=a+b;

}

while(x<=)

{

C=a+b+(x\*y;

if (a>b)

{

read a;

}

else

{

write b;

}

write c

}

报错1：

Line 4 2 error!

Line 6 for error!

Line 6 ( error!

Line 6 i error!

Line 6 = error!

Line 6 1 error!

Line 6 i error!

Line 6 = error!

Line 6 i error!

Line 6 ) error!

Line 7 { error!

Line 10 b error!

Line 10 = error!

Line 10 b error!

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

Line 27 缺少 }

改 4 行 int 2b;->int b;后

报错2：

Line 6 for error!

Line 6 ( error!

Line 6 i error!

Line 6 = error!

Line 6 1 error!

Line 6 i error!

Line 6 = error!

Line 6 i error!

Line 6 ) error!

Line 7 { error!

Line 10 b error!

Line 10 = error!

Line 10 b error!

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

Line 27 缺少 }

改5行int c->int c;后

报错3：

Line 6 i error!

Line 6 = error!

Line 6 i error!

Line 6 ) error!

Line 7 { error!

Line 10 b error!

Line 10 = error!

Line 10 b error!

Line 10 ; error!

Line 11 { error!

Line 12 c error!

Line 12 = error!

Line 12 a error!

Line 12 + error!

Line 12 b error!

Line 12 ; error!

Line 13 } error!

Line 14 while error!

Line 14 ( error!

Line 14 x error!

Line 14 <= error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 22 else error!

Line 27 } error!

Line 27 缺少 }

改6行i <= 10->i<=10；后

报错4：

Line 10 b error!

Line 10 = error!

Line 10 b error!

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

Line 27 缺少 }

改9行a=a+i->a=a+i;后

报错5：

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

Line 27 缺少 }

改14行while(x<=)->while(x<10), C=a+b+(x\*y;-> C=a+b+(x\*y);

报错6：

Line 27 } error!

Line 27 缺少 }

改26行write c-> write c;

报错7：

Line 27 缺少 }

改28行加上‘}’后

报错8：

Line 28 缺少 }

改29行加上‘}’后



分析

Line 4 2 error!

原因int后的应该式ID

Line 6 for error!

Line 6 ( error!

Line 6 i error!

Line 6 = error!

Line 6 1 error!

Line 6 i error!

Line 6 = error!

Line 6 i error!

Line 6 ) error!

Line 7 { error!

5行少了；号所以后期都入‘；’状态就会判错哈维=和第六行i <= 10后少‘；’所导致。

Line 10 b error!

Line 10 = error!

Line 10 b error!

10行a=a+i缺少‘；’号引起的。

Line 14 ) error!

Line 15 { error!

Line 16 = error!

Line 16 a error!

Line 16 ; error!

Line 18 if error!

Line 18 ( error!

Line 18 a error!

Line 18 > error!

Line 18 b error!

Line 19 { error!

Line 20 read error!

Line 20 a error!

Line 20 ; error!

Line 21 } error!

Line 22 else error!

Line 23 { error!

Line 24 write error!

Line 24 b error!

Line 24 ; error!

Line 25 } error!

Line 26 write error!

Line 26 c error!

Line 27 } error!

主要式因为while(x<=)

C=a+b+(x\*y;导致

两个错误缺少<=后的式子和（x\*y缺少‘）’引起的。

Line 27 缺少 }

由代码缺少两个‘}’所引起。

**五、讨论与分析**

（主要部分：通过实验对课程知识点的理解；回答实验指导书的实验思考提出的问题等）

1. TEST语言语法规则中哪些不满足无回溯的递归下降分析条件？你是怎么处理的？给出你的处理方案。

答：1 **< expression >→ ID=<bool\_expr>|<bool\_expr>** 通过超前读入一个符号 来提前判断，避免回溯。

2 <if\_stat>::=if (<expression>) <statement >| if (<expression>) <statement >else < statement > 把式子转换为<if\_stat>::=if (<expression>) <statement >flag 做一个flag标记当为flag标记是直接判断当前是否为else 如果是就加入<statment>读取下一个字符，否则直接去除该状态。

1. 所有文法都可以改写为满足递归下降分析条件的文法吗？如果不能，请给出一个反例。

答:不能，如 S->Ap|Bq A->Ap|d B->Bq|e 。

3、改写文法有什么弊端？

答：1.在对文法进行拓展时候，会增加程序函数的数量，代码量和程序的复杂性也 会相应提高。

2.些该文法可能会附加更多的文法产生式，增加first和follow集合，也有可能 产生一些隐含错误（与原文法的表意不同）

3.改写文法会降低文法的可读性。

**六、附录：关键代码（给出适当注释，可读性高）**

**#include<stdio.h>**

**#include<string.h>**

**#include<vector>**

**#include<iostream>**

**std::vector< char >status;**

**char chID[20],ch[20]; //chID当前字符标记 ,ch当前字符**

**int line; //当前的函数**

**int f;**

**FILE \* istream;**

**bool identify;**

**void error()**

**{**

**identify=false;**

**printf("Line %d %s error!\n",line,ch);**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**void program()**

**{**

**if(!strcmp(chID,"{"))**

**{**

**status.pop\_back();**

**status.push\_back('}');**

**status.push\_back('D');**

**status.push\_back('B');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**return;**

**}**

**void declaration\_list()**

**{**

**if(!strcmp(chID,"int"))**

**{**

**status.pop\_back();**

**status.push\_back('B');**

**status.push\_back('C');**

**}**

**else if(strcmp(chID, "if") == 0 ||**

**strcmp(chID, "while") == 0 ||**

**strcmp(chID, "for") == 0 ||**

**strcmp(chID, "read") == 0 ||**

**strcmp(chID, "write") == 0 ||**

**strcmp(chID, "{") == 0 ||**

**strcmp(chID, ";") == 0 ||**

**strcmp(chID, "ID") == 0 ||**

**strcmp(chID, "(") == 0||**

**strcmp(chID, "NUM") == 0 ||**

**strcmp(chID, "}") == 0)**

**{**

**status.pop\_back();**

**}**

**else**

**{**

**error();**

**}**

**return ;**

**}**

**void declaration\_stat()**

**{**

**if(strcmp(chID, "int") == 0)**

**{**

**status.pop\_back();**

**status.push\_back(';');**

**status.push\_back('g');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**return ;**

**}**

**void statement\_list()**

**{**

**if(strcmp(chID, "}") == 0)**

**{**

**status.pop\_back();**

**}**

**else if( strcmp(chID, "if") == 0 || strcmp(chID, "while") == 0 ||**

**strcmp(chID, "for") == 0 || strcmp(chID, "read") == 0 ||**

**strcmp(chID, "write") == 0 || strcmp(chID, "ID") == 0 ||**

**strcmp(chID, "{") == 0 || strcmp(chID, ";") == 0 ||**

**strcmp(chID, "NUM") == 0 || strcmp(chID, "(") == 0 )**

**{**

**status.pop\_back();**

**status.push\_back('D');**

**status.push\_back('E');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void statement()**

**{**

**if(strcmp(chID, "{") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('K');**

**}**

**else if(strcmp(chID, "if") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('F');**

**}**

**else if(strcmp(chID, "while") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('G');**

**}**

**else if(strcmp(chID, "for") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('H');**

**}**

**else if(strcmp(chID, "write") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('I');**

**}**

**else if(strcmp(chID, "read") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('J');**

**}**

**else if(strcmp(chID, "ID") == 0 ||**

**strcmp(chID, "NUM") == 0 ||**

**strcmp(chID, "(") == 0||**

**strcmp(chID, ";") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('L');**

**}**

**else {**

**error();**

**}**

**return ;**

**}**

**void if\_stat()**

**{**

**if(strcmp(chID, "if") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('U');**

**status.push\_back('E');**

**status.push\_back(')');**

**status.push\_back('M');**

**status.push\_back('(');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void while\_stat()**

**{**

**if(strcmp(chID, "while") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('E');**

**status.push\_back(')');**

**status.push\_back('M');**

**status.push\_back('(');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void for\_stat()**

**{**

**if(strcmp(chID, "for") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('E');**

**status.push\_back(')');**

**status.push\_back('M');**

**status.push\_back(';');**

**status.push\_back('M');**

**status.push\_back(';');**

**status.push\_back('M');**

**status.push\_back('(');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void write\_stat()**

**{**

**if(strcmp(chID,"write")==0)**

**{**

**status.pop\_back();**

**status.push\_back(';');**

**status.push\_back('M');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void read\_stat()**

**{**

**if(strcmp(chID,"read")==0)**

**{**

**status.pop\_back();**

**status.push\_back(';');**

**status.push\_back('g');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void compound\_stat()**

**{**

**if(strcmp(chID,"{")==0)**

**{**

**status.pop\_back();**

**status.push\_back('}');**

**status.push\_back('D');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**void expression\_stat()**

**{**

**if(strcmp(chID,";")==0)**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"ID")==0||**

**strcmp(chID,"NUM")==0||**

**strcmp(chID,"(")==0)**

**{**

**status.pop\_back();**

**status.push\_back(';');**

**status.push\_back('M');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void expression()**

**{**

**char chID1[20],ch1[20];**

**int line1,filep;**

**if(strcmp(chID,"ID")==0)**

**{**

**filep=ftell(istream);**

**fscanf(istream, "%d%s%s\n", &line1, &chID1, &ch1);**

**if(strcmp(chID1,"=")==0)**

**{**

**status.pop\_back();**

**status.push\_back('N');**

**fscanf(istream, "%d%s%s\n", &line, &chID, &ch);**

**}**

**else**

**{**

**fseek(istream, filep, 0); //返回原位。**

**status.pop\_back();**

**status.push\_back('N');**

**}**

**}**

**else if(strcmp(chID,"NUM")==0|| strcmp(chID,"(")==0)**

**{**

**status.pop\_back();**

**status.push\_back('N');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void bool\_expr()**

**{**

**if(strcmp(chID,"ID")==0||**

**strcmp(chID,"NUM")==0||**

**strcmp(chID,"(")==0)**

**{**

**status.pop\_back();**

**status.push\_back('O');**

**status.push\_back('P');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void bool\_expr1()**

**{**

**if( strcmp(chID, ">") == 0 ||**

**strcmp(chID, ">=") == 0 ||**

**strcmp(chID, "<") == 0 ||**

**strcmp(chID, "<=") == 0 ||**

**strcmp(chID, "==") == 0 ||**

**strcmp(chID, "!=") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('P');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID, ";") == 0||strcmp(chID, ")") == 0)**

**{**

**status.pop\_back();**

**}**

**}**

**void additive\_expr()**

**{**

**if(strcmp(chID, "ID") == 0||**

**strcmp(chID, "NUM") == 0 ||**

**strcmp(chID, "(") == 0 )**

**{**

**status.pop\_back();**

**status.push\_back('Q');**

**status.push\_back('R');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void additive\_expr1()**

**{**

**//printf("1");**

**if (strcmp(chID, "+") == 0 || strcmp(chID, "-") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('Q');**

**status.push\_back('R');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if( strcmp(chID, ";") == 0 || strcmp(chID, ")") == 0 ||**

**strcmp(chID, ">") == 0 || strcmp(chID, ">=") == 0 ||**

**strcmp(chID, "<") == 0 || strcmp(chID, "<=") == 0 ||**

**strcmp(chID, "==") == 0 || strcmp(chID, "!=") == 0)**

**{**

**status.pop\_back();**

**}**

**else**

**{**

**error();**

**}**

**}**

**void term()**

**{**

**if (strcmp(chID, "ID") == 0 || strcmp(chID, "NUM") == 0 || strcmp(chID, "(") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('S');**

**status.push\_back('T');**

**}**

**else**

**{**

**error();**

**}**

**}**

**void term1()**

**{**

**if (strcmp(chID, "\*") == 0 || strcmp(chID, "/") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('S');**

**status.push\_back('T');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if( strcmp(chID, ";") == 0 || strcmp(chID, ")") == 0 ||**

**strcmp(chID, "+") == 0 || strcmp(chID, "-") == 0 ||**

**strcmp(chID, ">") == 0 || strcmp(chID, ">=") == 0 ||**

**strcmp(chID, "<") == 0 || strcmp(chID, "<=") == 0 ||**

**strcmp(chID, "==") == 0 || strcmp(chID, "!=") == 0)**

**{**

**status.pop\_back();**

**}**

**else**

**{**

**error();**

**}**

**}**

**void factor()**

**{**

**if(strcmp(chID, "(") == 0)**

**{**

**status.pop\_back();**

**status.push\_back(')');**

**status.push\_back('M');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if (strcmp(chID, "ID") == 0 || strcmp(chID, "NUM") == 0)**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**}**

**void flag()**

**{**

**if(strcmp(chID, "else") == 0)**

**{**

**status.pop\_back();**

**status.push\_back('E');**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID, "#") == 0)**

**{**

**error();**

**}**

**else**

**{**

**status.pop\_back();**

**}**

**}**

**void uCharacter(char temp)**

**{ //printf("%s",ch);**

**if(strcmp(chID,"if")==0&&temp=='b')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"while")==0&&temp=='c')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"for")==0&&temp=='d')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"write")==0&&temp=='e')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"read")==0&&temp=='f')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"ID")==0&&temp=='g')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"NUM")==0&&temp=='h')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(strcmp(chID,"else")==0&&temp=='j')**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else if(chID[0]==temp&&strlen(chID)==1)**

**{**

**status.pop\_back();**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**}**

**else**

**{**

**error();**

**}**

**}**

**int main()**

**{**

**if((istream=fopen("lex.txt","r"))==NULL)**

**{**

**printf("文件打开错误！\n");**

**return -1;**

**}**

**status.clear();**

**//status.push\_back('#');**

**status.push\_back('A'); //初始压栈**

**fscanf(istream,"%d%s%s", &line,chID,ch);**

**f=1;**

**identify=true;**

**while(!status.empty()&& f)**

**{**

**if(feof(istream)) //字符已经读完。**

**{**

**f=0;**

**continue;**

**}**

**char temp=status.back();**

**// printf("%s\n" ,ch);**

**switch(temp)**

**{**

**case 'A': program(); break;**

**case 'B': declaration\_list(); break;**

**case 'C': declaration\_stat(); break;**

**case 'D': statement\_list(); break;**

**case 'E': statement(); break;**

**case 'F': if\_stat(); break;**

**case 'G': while\_stat(); break;**

**case 'H': for\_stat(); break;**

**case 'I': write\_stat(); break;**

**case 'J': read\_stat(); break;**

**case 'K': compound\_stat(); break;**

**case 'L': expression\_stat(); break;**

**case 'M': expression(); break;**

**case 'N': bool\_expr(); break;**

**case 'O': bool\_expr1(); break;**

**case 'P': additive\_expr(); break;**

**case 'Q': additive\_expr1(); break;**

**case 'R': term(); break;**

**case 'S': term1(); break;**

**case 'T': factor(); break;**

**case 'U': flag(); break;**

**default : uCharacter(temp); break;**

**}**

**}**

**if(!status.empty())**

**{**

**identify=false;**

**}**

**if(identify )**

**{**

**printf("语法编译成功未发现错误!\n");**

**}**

**else**

**{**

**printf("语法编译发现错误！\n");**

**}**

**return 0;**

**}**

**七、实验者自评（主要从实验态度、方法、效果上给一个客观公正的自我评价）**

**本次实验前期理论工作实际比代码生成工作时间较长，对FIRST,FOLLOW集求**

**多次验证，对依据文法做语法分析有了更深刻的理解，对编译器的报错方式有了了解**

**通过本次实验验证了编译原理的理论加深了我对他们的理解，培养了我坚持做事，耐心做事的心境。**