**LL(1)语法分析程序实验报告**

1. **文法**

原文法：

E->E+T|E-T|T

T->T\*F|T/F|F

F->id|(E)|num

其中： id: a-f, A-F，num:0-9

消左递归：

E->TA A->+TA A->-TA A->e

T->FB B->\*FB B->/FB B->e

F->i F->(E) F->n

其中：i:id, n:num, e:epsilon

1. **FIRST集和FOLLOW集**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | TA | +TA | -TA | e | FB | \*FB | /FB | e | i | (E) | n |
| FIRST | i,(,n | + | - | e | i,(,n | \* | / | e | i | ( | n |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | E | A | T | B | F |
| FOLLOW | $,) | $,) | +,-,$,) | +,-,$,) | \*,/,+,-,$,) |

1. **预测分析表**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i | n | + | - | \* | / | ( | ) | $ |
| E | E->TA | E->TA |  |  |  |  | E->TA | synch | synch |
| A |  |  | A->+TA | A->-TA |  |  |  | A->e | A->e |
| T | T->FB | T->FB | synch | synch |  |  | T->FB | synch | synch |
| B |  |  | B->e | B->e | B->\*FB | B->/FB |  | B->e | B->e |
| F | F->i | F->n | synch | synch | synch | synch | F->(E) | synch | synch |

1. **运行环境**

CodeBlocks-13.12 with GCC compiler from TDM-GCC (4.7.1, 32 bit)

1. **输入输出设计**

输入：文件“fin.txt”输入待分析串

输出：命令行界面输出预测分析表，LL(1)分析过程输出至“fout.txt”

1. **主要数据结构**

vector<vector<string>> table(5,vector<string>(9)) //预测分析表

vector<string> G //消除左递归后的文法产生式

map<char, int> index //文法符号到下标的转换字典

string terminal("in+-\*/()$") //终结符

string nonTerminal("EATBF") //非终结符

vector<string> First // 产生式右部符号串的first集

vector<string> Follow //非终结符的follow集

1. **核心算法**
2. int main()

{

for(文法G每个产生式itG，itFirst为其右部符号串的first集){

x = itG左部非终结符号的下标；

for(itFirst中的每个终结符号first){

y = 终结符号first的下标；

把itG加入分析表表G[x][y]；

}

if(终结符号first == epsilon)

for(Follow集中的每个符号follow){

y = follow的下标；

把itG加入分析表G[x][y];

}

}

for（所有非终结符号的Follow集）

if（对应表项为空）

写入synch；

将分析表输出到命令行界面；

return analysis();

}

1. int analysis(void)

{

从文件fin.txt读取待分析串到s；

s末尾加‘$’；

分析栈vector<char> analyStack;

向栈压入‘$’、‘E’；

ip指向s的第一个字符；

do{

top是栈顶符号；

cur是ip所指向的输入符号；

if（cur是字母） cur = ‘i’；

if（cur是数字） cur = ‘n’；

if（top是终结符号或‘$’）{

if（top == cur）{从栈顶弹出cur；ip前移一个位置；}

else error；

}

else{

x = top对应下标； y = cur对应下标；

产生式production = table[x][y];

if（production非空）{

栈顶弹出cur；

把production右部逆序压栈；

输出production；

}

else error；

}

while（top != ‘$’）；

}