**实验（实习）报告**

一、 实验目的

设计一个LL(1)语法分析器，利用语法分析器对符号串的识别，加深对语法分析原理的理解。

二、 实验内容

设计并实现一个LL(1)语法分析器，实现对算术文法G[E]:E->E+T|T T->T\*F|F F->(E)|i所定义的符号串进行识别，例如符号串abc+age+80为文法所定义的句子，符号串(string2.txt) (abc-80(\*s5)不是文法所定义的句子。

三、 实验要求

本次实验可分两个同学完成，两人分工合作，交互讨论，最后报告内的姓名栏可写2个人的名字。最后，每个人以自己的学号+姓名作为实验报告后缀，提交到学习通。

1、学习基础知识.docx文档，阅读parser\_my.cpp中的代码，理解程序的结构：包括消除间接和直接左递归，求解First集和Follow集，构建LL(1)分析表等。了解哪些功能有待实现，所有需要实现代码部分有以下提示：

/\*请编程实现以下功能 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/。

2、检测左递归，如果有则进行消除，实现remove\_left\_recursion()函数；消除直接左递归（已实现）。

3、求解FIRST集和FOLLOW集，分别实现getFirst()和getFollow()函数；

4、理解构建LL(1)分析表getTable()中的代码，实现格式打印分析表的功能（显示Table部分）；

5、对于输入符号串，实现自顶向下的LL(1)分析，打印出分析过程：AnalyzePredict函数。

四、 运行结果

测试源文件（根据parse\_test1.txt或parse\_test2.txt或自行增加内容）：

E->T|E+T;

T->F|T\*F;

F->i|(E);

部分代码：

char cur\_s = \*it;

                /\*请编程实现以下功能

                \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                cur\_s->cur\_string[0]

                    a加到A的FIRST集

                cur\_s->cur\_string[0]

                    B的FITRST集加到A的FIRST集

                \*/

                for (auto it0 = P[cur\_s].begin(); it0 != P[cur\_s].end(); it0++)

                {

                    string left = \*it0;

                    if(!isupper(left[0]))

                    {

                        FIRST[cur\_s].insert(left[0]);

                    }

                    else

                    {

                        for (auto it1 = FIRST[left[0]].begin(); it1 != FIRST[left[0]].end();it1++)

                        {

                            FIRST[cur\_s].insert(\*it1);

                        }

                    }

                }

/\*请编程实现以下功能

                \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

                cur\_s->cur\_string[0]

                a加到A的FIRST集

                cur\_s->cur\_string[0]

                B的FITRST集加到A的FIRST集

                \*/

                for (set<string>::iterator it1 = P[cur\_s].begin(); it1 != P[cur\_s].end(); it1++)

                {

                    string cur\_string = \*it1;

                    for (int i = 0; i < cur\_string.length(); i++)

                    {

                        if(!isupper(cur\_string[i]))

                        {

                            continue;

                        }

                        if(i == cur\_string.length()-1)

                        {

                            /\* 第四步：

                                AC/ACK为最后两个或者三个

                                B->AC

                                B->ACK(K的first集含有@)

                                将B的follow集加入到C的follow集

                            \*/

                            for (auto it2 = FOLLOW[cur\_s].begin(); it2 != FOLLOW[cur\_s].end();it2++)

                            {

                                FOLLOW[cur\_string[i]].insert(\*it2);

                            }

                        }

                        else

                        {

                            int j;

                            for (j = i + 1; j < cur\_string.length(); j++)

                            {

                                char next = cur\_string[i + 1];

                                /\*第一步：

                                    B->Ac

                                    将c加到A的follow集

                                \*/

                                if (!isupper(next))

                                {

                                    FOLLOW[cur\_string[i]].insert(next);

                                    break;

                                }

                                /\*第二步：

                                    B->AC

                                    将C的first集加到A的follow集

                                \*/

                                else if(FIRST[next].count('@') == 0)

                                {

                                    for (auto it2 = FIRST[next].begin(); it2 != FIRST[next].end(); it2++)

                                    {

                                        FOLLOW[cur\_string[i]].insert(\*it2);

                                    }

                                    break;

                                }

                                /\*第三步：

                                    遍历C的first去除@，加到A的follow集

                                \*/

                                else

                                {

                                    for (auto it2 = FIRST[next].begin(); it2 != FIRST[next].end();it2++)

                                    {

                                        if((\*it2)!='@')

                                        {

                                            FOLLOW[cur\_string[i]].insert(\*it2);

                                        }

                                    }

                                }

                            }

                            if(j == cur\_string.length())

                            {

                                for (auto it2 = FOLLOW[cur\_s].begin(); it2 != FOLLOW[cur\_s].end();it2++)

                                {

                                    FOLLOW[cur\_string[i]].insert(\*it2);

                                }

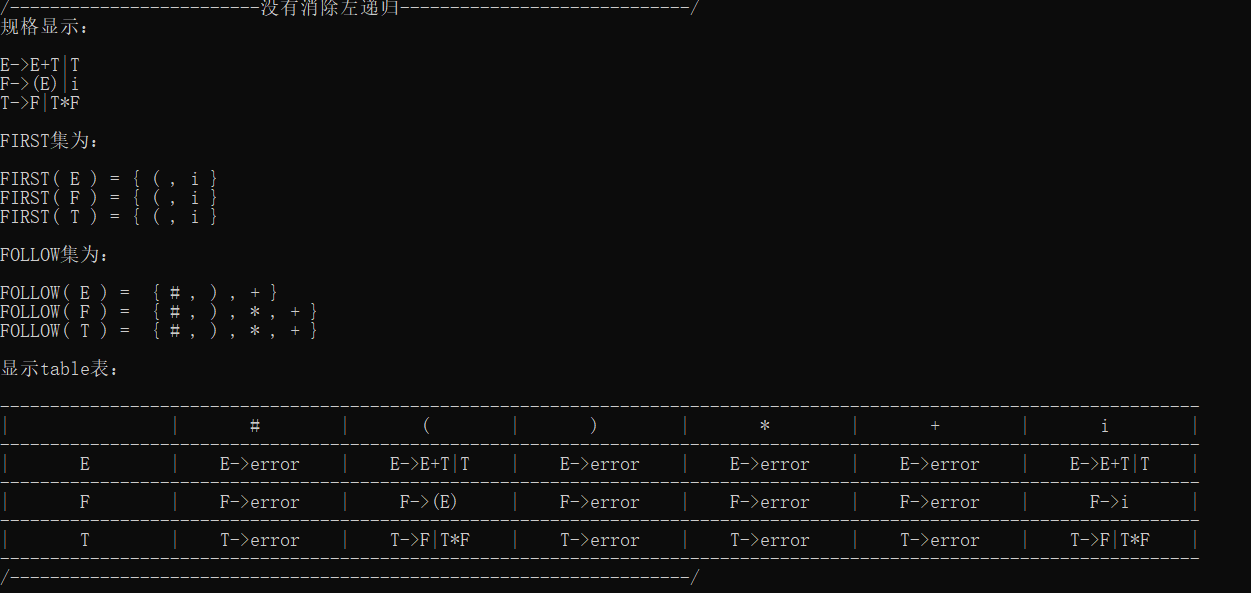
                            }

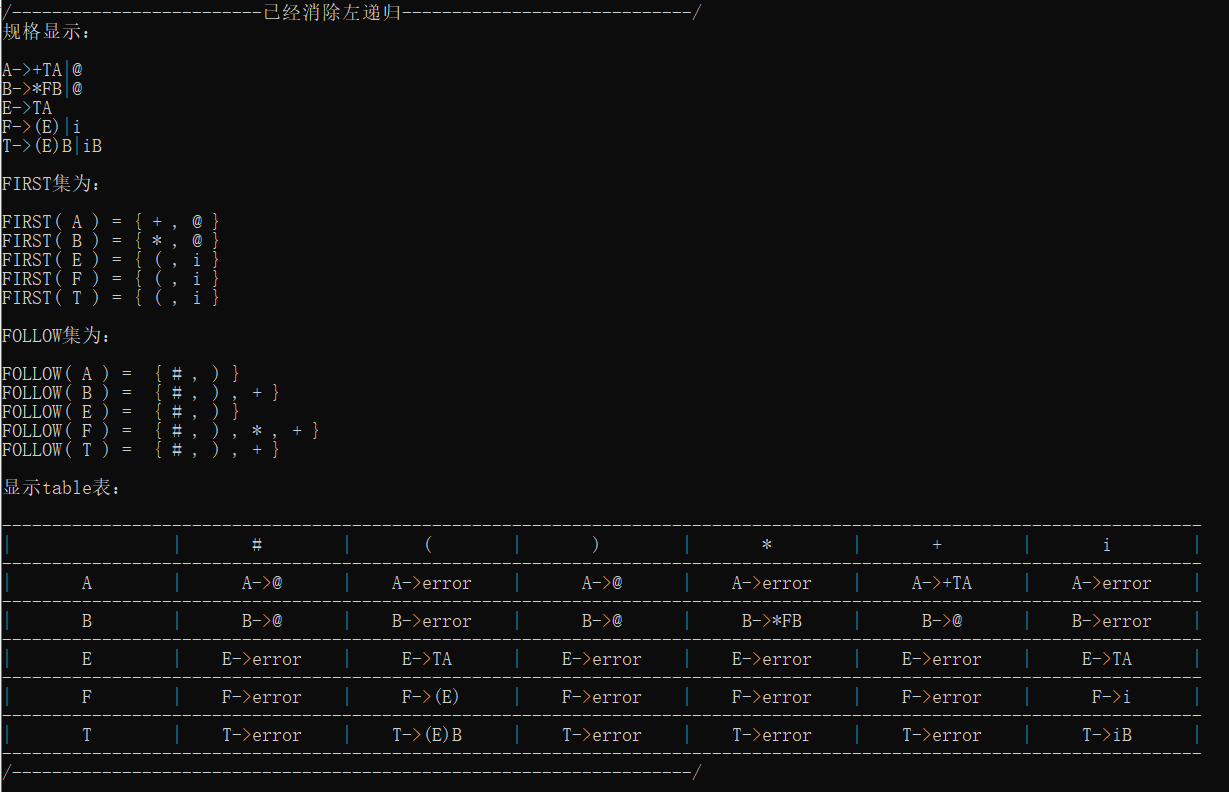
                        }

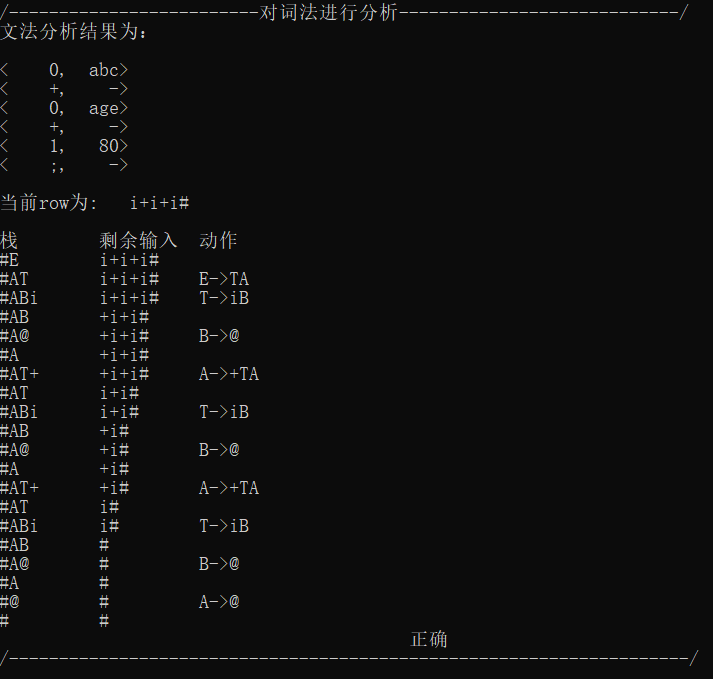
                    }

                }

分析结果（截图）：







附－实现代码文件和结果文件（可直接将cpp文件及实验结果文件拖至下方位置）：



五、一些思考和总结

通过本次实验，我更好的了解掌握领悟了消除左递归、计算FIRST集、计算FOLLOW集、计算SELECTED集、LL(1)文法的算法逻辑，为今后的学习和实验奠定了坚实的基础。