关于本实验，以下表格列出在本实验中使用频率较高的单词对应的单词种别码。

|  |  |
| --- | --- |
| 单词 | 单词种别码 |
| begin | 1 |
| end | 6 |
| # | 0 |
| 回车 | 50 |
| 数字 | 20 |
| 标识符 | 10 |
| 赋值号 := | 18 |
| 分隔符 ; | 34 |
| +、- | 22、23 |
| \*、/ | 24、25 |

以下是代码中主要函数的说明

WORD \* scanner();//扫描,获取到每个单词的种别码；

int Irparser();//程序的判断；

int yucu();//语句串的分析；

int statement();//语句的分析；

int expression();//表达式的判断；

int term();//项的分析；

int factor();//因子的分析；

void check\_enter()//检查到是回车，则将row++，取下一个字符；

void error\_enter()//出错之后检查回车直接跳到下一行，并且把行数对应加1；

待分析的语法结构用扩充的BNF表示如下：

（1）<程序>：：=begin<语句串> end

（2）<语句串>：：=<语句>{；<语句>}

（3）<语句>：：=<赋值语句>

（4）<赋值语句>：：=ID：=<表达式>

（5）<表达式>：：=<项>{+<项>|-<项>}

（6）<项>：：=<因子>{\*<因子>|/<因子>}

（7）<因子>：：=ID | NUM | {<表达式>}

在本实验中，输入单词串，以“#”结束，如果输入的每一行都没有错误，则输出“Success”，如果某一行出错，打印出“第X行，XXX错误”，并且在分析完毕之后打印出总共输入的行数，以及错误的个数。本实验采用的是从文本读取程序并且和分析结果一同显示在控制台上，在代码文件中放置了两个txt文档，一个是出错的程序（error\_data.txt），一个是正确的程序（right\_data.txt），通过在代码的main函数中设置读取的文件名，就可以读取到相关的文件了。

在error\_data.txt中基本上囊括了所有错误，两个文件分别读取之后的结果如下：

|  |
| --- |
| 读取error\_data.txt，如下：  a:=4  do:=5;  b=7;  c:=a\*((a+b)));  d:=(a+((a\*(b+c))  # |
| 预期结果：开始处：缺少‘begin’  第1行：缺少分隔符  第2行：变量名错误  第3行：赋值号错误  第4行：缺少左括号  第5行：缺少右括号  第6行：缺少‘end’  总共6行，其中有7个错误 |
| 实际显示结果：  C:\Users\asus01\AppData\Local\Temp\1525963386(1).png |

|  |
| --- |
| 读取right\_data.txt，如下：  begin  a:=4;  e:=5;  b:=7;  c:=a\*(a+b);  d:=(a+(a\*(b+c)))  end# |
| 预期结果：Success  总共7行，其中0个错误 |
| 实际显示结果：  C:\Users\asus01\AppData\Local\Temp\1525963606(1).png |