专题2 递归下降语法分析设计原理与实现

13281193 田容先

# 程序功能

此程序为语法分析程序，可以根据词法分析程序所得结果，对字符串用文法规则判断，看是否满足当前文法的要求。

# 程序结构

## 主要数据结构

char current;

存放当前读入的字符

int NextSym = 0;

指向下一个将要读入的字符的指针，一开始指向首位置

char input[30];

输入字符串，为专题一的输出结果，（关键字用标号代替，非关键字给出）

## 函数定义

程序包含以下几个自定义函数：

bool equal(char\* id);

判断当前输入的字符串是否与id相等。

void rollback(char\* id);

将当前的输入指针位置向前移id的长度。

char\* iden(char\* num);

将标识符转化为真实符号。

void read();

读入专题一的输出结果，将标识符转化为真实的语句。

bool E();

bool E\_();

bool T();

bool T\_();

bool F();

bool A();

bool M();

以上函数均为递归向下分析中用到的，判断该字符是否正确的函数。（’用\_代替）

## 设计方法

此实验与第四章一个附加实验非常相似，所以思路上没有太难，而且在课件上已经有流程图供参考，所以只需按照流程图的步骤，一步一步走下来即可。只是此实验与第四章附加实验有个别地方不同。

首先，这个实验的文法与之前的文法不一样，它有非终结符号出现在头部，经过调试发现，我只能在每个equal 函数中加入rollback，也就是说在判断相等之后再把指针送回去。而这也是实际上标准的方法。原来的附加实验中，我没有那么设置，所以省掉了一些nextsym操作，而在这里就必须按照流程图来做。

另外，我在做专题一的时候，输出结果无论是否为标志位都是带有单词的，在这里我改了专题一的程序，让它在输出if、while等关键字时只输出数字标识符。而原来的个位数标识符1,2,3 我也改为了01 02 03，方便读入。

## 设计难点

本程序不是很难，重要在于按照流程图来画，对于我来说，难点主要在于将专题一的结果读入。当时我犹豫是要把专题一合在一起，不进行文件操作，还是进行文件操作，后来根据要求选择了带有中间文件的，这也符合实际的情况。

void read(){

FILE\* fp2;

char num[2], defined[20];

char \*reserved;

int i, c;

if ((fp2 = fopen("out.txt", "r")) == NULL)//打开操作不成功

{

printf("open error");

exit(1);//结束程序的执行

}

fseek(fp2, 1, SEEK\_CUR);

while (1){

if (!fgets(num, 3, fp2))break;

if (iden(num) == NULL){

fseek(fp2, 1, SEEK\_CUR);

for (i = 0;; i++){

c = fgetc(fp2);

if (c == 32){ fseek(fp2, -1, SEEK\_CUR); break; }/\*space ascii:32\*/

defined[i] = c;

}

defined[i] = '\0';

strcat(input, defined);

}

else{

reserved = iden(num);

strcat(input, reserved);

}

fseek(fp2, 1, SEEK\_CUR);

}

}

这是我写的读入函数。为了读入方便，配合每次循环的空格，我在输入的第一个位置和最后一个位置也加上了空格。有个问题现在还没有完全解决，我用feof函数无法判断文件结束，导致一直读入，我在这个地方卡了很长时间。后来我用if (!fgets(num, 3, fp2))break;来解决文件读完的操作。

# 程序测试

## 测试用例

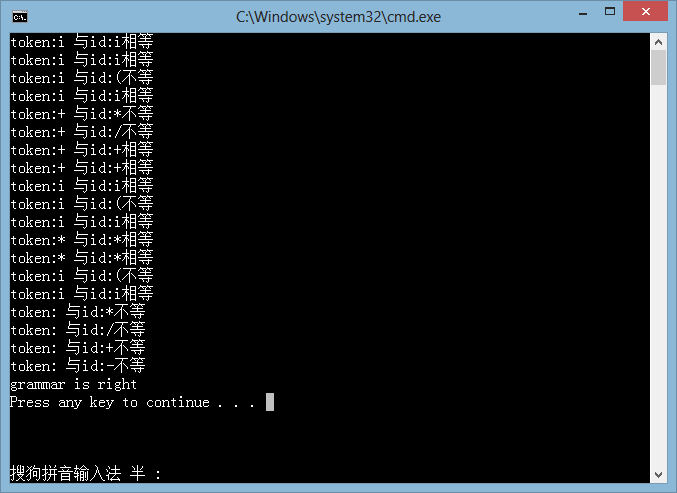
out.txt

10 i 12 10 i 14 10 i

//前面有个空格，最后有空格

其中10表示标识符，12表示+ 14表示\*。

## 测试结果



在每处equal判断出做一个输出，这样可以方便监视到哪里出了问题。

# 实验总结

本次实验学习了递归向下的语法分析程序。这种算法在我看来比较简单，便于理解，方便编程。不过文法比较固定，如果选用新的文法，这个逻辑结构都要改变，所以不是很灵活。在实验中加深了对文件操作的使用。总的来说有了上次做的选做实验的基础，这个实验不是很难。