**编译原理实验一实验报告**

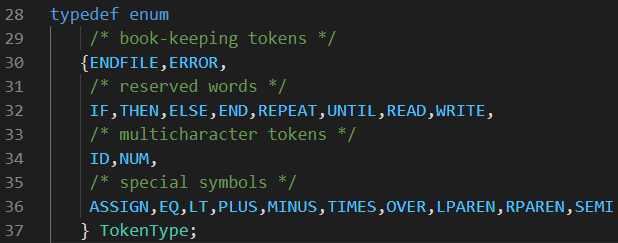
计科 MisakaCJQ

1. **实验目的**

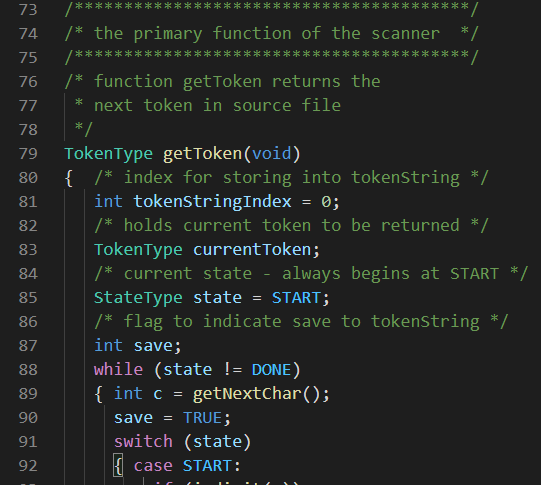
学习和掌握词法分析程序手工构造状态图及其代码实现方法。

1. **实验任务**
2. 阅读已有编译器的经典词法分析源程序；
3. 用C或C++语言编写一门语言的词法分析器。
4. **实验内容**
5. TINY编译器词法分析程序的阅读

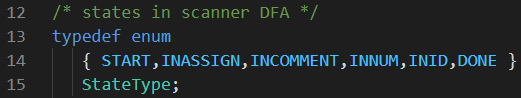
分析器关键部分说明：



在global.h文件中定义了TINY语言中的词法单元类型，其中包括保留的文件终止和错误的枚举，保留字的枚举，多字符的枚举ID和数字，以及其他的特殊符号如赋值号:=，等于号=等。



在文件scan.c中定义的函数getToken为词法分析程序的核心部分。在这一部分函数在未到达终态的每一轮循环中读入一个字符，并使用一个switch语句判断当前的状态并根据当前状态的不同来对输入字符进行不同的处理。在每一轮到达DONE状态后会将读取的词素和词法单元的类型保存下来，从而实现词法分析操作。

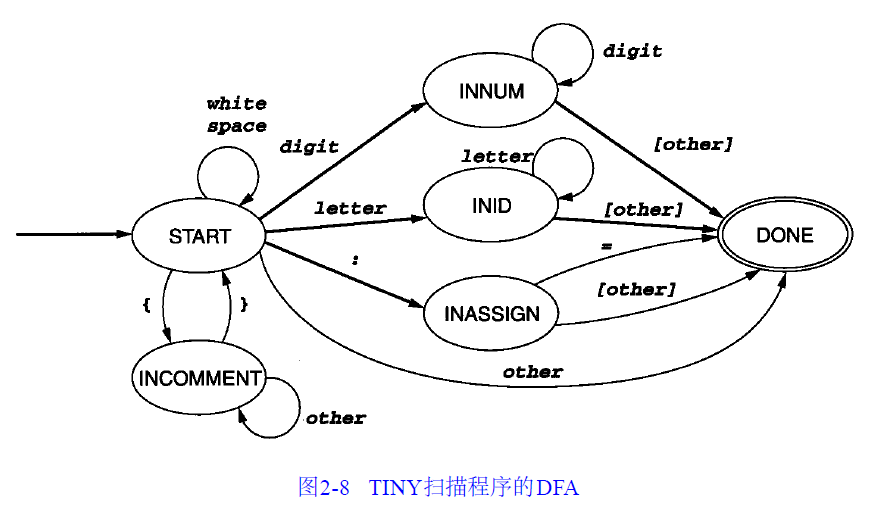


在scan.c文件中定义的6种状态枚举，便于在getToken词法分析函数当中进行状态的转换，提高可读性。

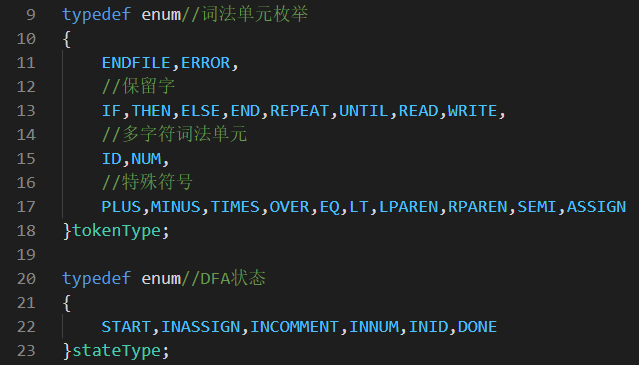
1. 编写TINY语言语法分析器

这一次的语法分析器选择TINY语言来进行实现。TINY语言的定义和词法分析DFA如下：

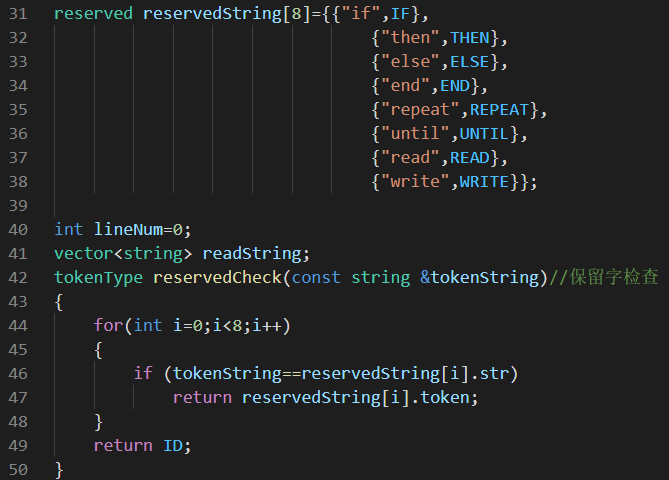




词法单元和DFA状态枚举定义：



通过一个表以及reserveCheck函数来进行保留字的检查，如果确认是保留字则返回对应的保留字词法单元类型，否则通过检查默认返回ID。

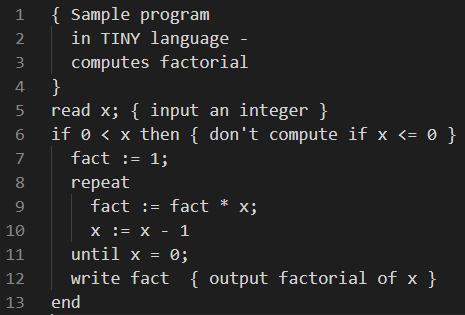


词法分析程序的核心部分getToken函数读入一个string字符串temp，这个字符串包含了输入的所有代码，含有换行符。通过一个定义的数组下下标tempIndex来对其进行遍历，从而实现getNextChar的操作。再部分状态当中读取的字符需要回退，这种情况下直接将下标-1即可。剩下的部分则是和示例的词法分析程序一样使用一个

switch语句来对不同的状态实现不同的处理和转换，以及对词素和词法单元的判断和保存。

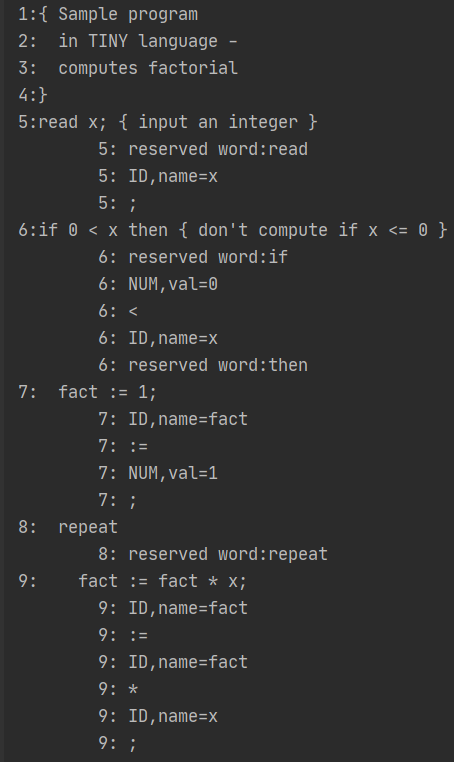
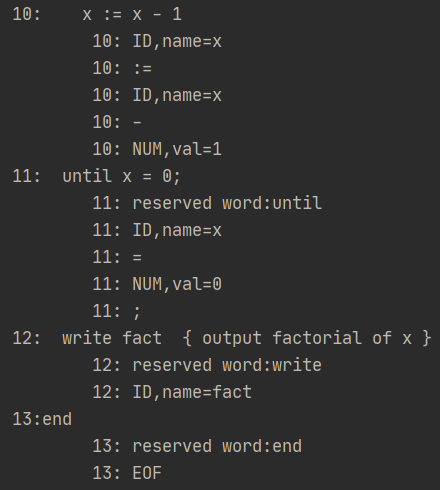


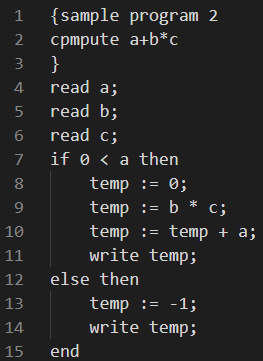
1. 样例输出

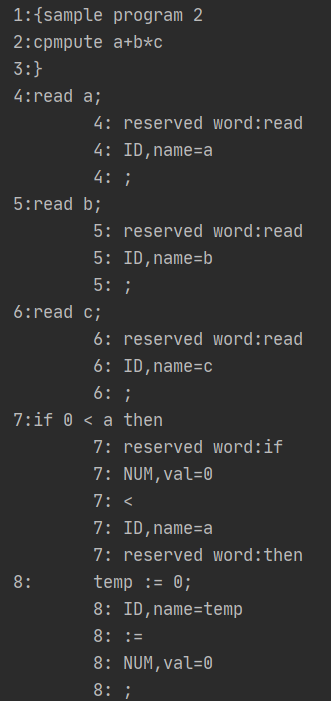
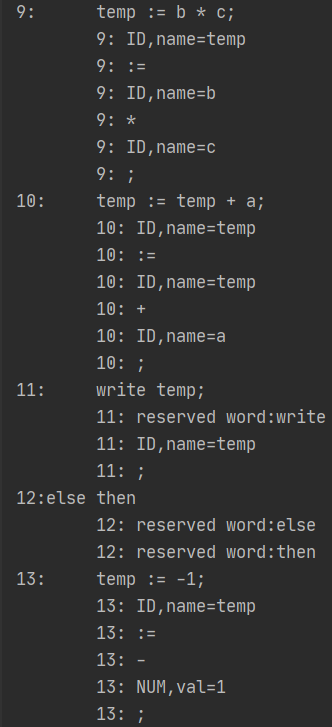
样例程序1：

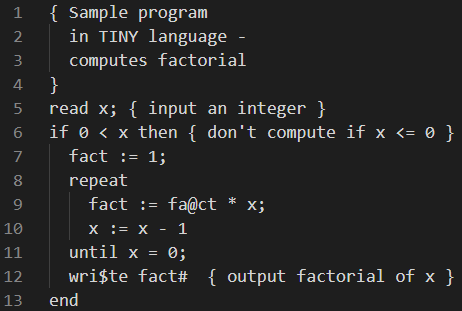
输出：

逐行地输出了对以上TINY语言样例文件的词法分析，逐个输出词法单元。

样例程序2：

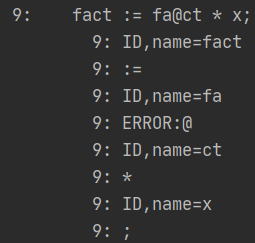
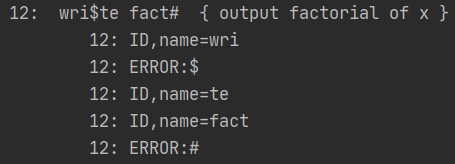
输出： 

样例程序3：

在样例程序1的基础上加入一些非法字符形成认为的词法错误，在第9行插入@符号，在第12行插入$和#号。

输出：

输出大体上与样例程序1一致，唯一有区别的是插入了非法字符的两行：

两行的词法分析均在非法字符位置检测到了错误并将其标记为ERROR。其中被非法字符分割的保留字成为了两个id，因此也证明词法分析程序的正确性。

1. **实验总结**

这一次实验的难点主要在于理解词法分析器的核心程序getToken函数，并要能学会通过switch语句来进行状态的转换和处理，从而实现TINY语言词法分析DFA。总之通过这一次实验我收获很多。