

# 词法分析实验报告

17341190 叶盛源 数据科学与计算机学院

## Tiny+语言词法

### 一、关键词(KEY)

Tiny+中可能出现的关键词包括以下这些

```
// 定义语法中所有可能出现的关键词
bool Token::is_KEY(string& str) {
    return str == "true" || str == "false" || str == "or"
    ||
        str == "and" || str == "not" || str == "int" ||
        str == "bool" || str == "string" || str == "while"
    ||
        str == "do" || str == "if" || str == "then" ||
        str == "else" || str == "end" || str == "repeat"
        || str == "until" || str == "read" || str ==
"write";
}
```

### 二、特殊符号(SYM)

常见特殊符号包括以下这些，其中注释用大括号包括。

```
// 判断是否为特殊符号
bool LexicalAnalysis::is_special_symbol(char c) {
    return c == ':' || c == ',' || c == ';' ||
        c == '<' || c == '>' || c == '=' ||
        c == '+' || c == '-' || c == '*' ||
        c == '/' || c == '(' || c == ')' ||
        c == '{' || c == '}';
}
```

其中单引号包括起来的是字符串，需要特殊考虑。在处理单引号包括的内容时，我们不做任何处理，并维持当前单引号状态

### 三、标识符(ID)

标识符由非数字开头的字符串，用来代表变量名等。

### 四、空白(blank)

Tiny中常见空白包括：空格（0x20）、水平制表（0x09）、垂直制表（0x0B）、换页（0x0C）、回车（0x0D）和换行符（0x0A）

## 词法分析

### 一、基本工作流程

编译器中只有该部分直接接触源代码，其他的部分都是通过使用之前的工作成果来间接接触源代码。词法分析器要进行的工作包括：

1. 去掉注释、空格一类的无用部分；
2. 处理和平台有关的输入，比如文件结束符的不同表示；
3. **根据模式识别记号，交给语法分析器；**
4. 调用符号表管理器/出错处理器，进行相关处理。

工作方式：

1. 词法分析器单独进行扫描，生成记号流。再将整个记号流交给语法分析器；
2. 词法分析器作为语法分析器的子程序进行工作，**语法分析器调用词法分析器去读源程序，得到词法分析器返回的记号就拿来构造语法树。**然后用掉了这个记号就再去调用词法分析器读新的记号，如此重复；
3. 词法、语法分析器并行工作。两者有一个共享的记号流，前者不停读程序、把记号放入记号流，后者不停取记号流来构造语法树。

## 二、状态转移矩阵

在Tiny语言中，有很多可能出现的状态转移。我们可以通过对词法规则的设计，得到基本的状态转移矩阵，帮助我们编写代码。这里拿数字转移成各种状态简单举例：

	blank	num	char	single_quote	special_char
num	blank	num	error	single_quote	special_char

在数字模式下，如果遇到字符，就会出现error，这也是为什么我们的标识符不能用数字开头的原因。

## 三、异常处理

程序中定义了error类用来处理错误的信息。比如在进入数字状态后，如果遇到字母，我们就会判定为非法，并进入非法状态，并记录错误信息：


```
...
case CharType::c_letter:
    errorMsgs.emplace_back(ErrorMsg(currLine, currColumn -
word.length()+1, "数字和字母是非法组合"));
    word = "";
    status = Status::error;
    break;
```

## 四、实验结果

我们使用下列的简单程序进行测试：

```
int A,B; {define int}
string C; {define string}
bool D; {define bool}
C:='hello world'; {assign string}
A:=1;
B:=2;
if A<=B do
    D:=true;
end
```

实验结果为：

 C:\Users\YSY\Desktop\lexical\_analysis\Debug\lexical\_analysis.exe

```
词法分析结果为：
(KEY int)
(ID A)
(SYM ,)
(ID B)
(SYM ;)
(KEY string)
(ID C)
(SYM ;)
(KEY bool)
(ID D)
(SYM ;)
(ID C)
(SYM :=)
(STR hello world)
(SYM ;)
(ID A)
(SYM :=)
(NUM 1)
(SYM ;)
(ID B)
(SYM :=)
(NUM 2)
(SYM ;)
(KEY if)
(ID A)
(SYM <=)
(ID B)
(KEY do)
(ID D)
(SYM :=)
(KEY true)
(SYM ;)
(KEY end)
请按任意键继续. . .
```

可以看到，源代码中的字符串都被转化为了token。