编译原理实验-词法分析

17341146 王程钥

1 简介

1.1 实验目的

通过扩充已有的样例语言TINY语言的词法分析程序，为扩展TINY语言TINY + 构造词法分析程序，从而掌握词法分析程序的构造方法。

1.2 实验内容

了解样例语言TINY及TINY编译器的实现，了解扩展TINY语言TINY + ，用C语言在已有的TINY词法分析器基础上扩展，构造TINY + 的词法分析程序。

1.3 实验要求

将TINY + 源程序翻译成对应的TOKEN序列，并能检查一定的词法错误。

2 TINY+语法

我对TINY语言进行了一些修改，得到了TINY+语言。因为本次实验的任务是词法分析，所以以下只展示词法方面的修改。

2.1 标识符

TINY+支持的标识符的标准和c语言相同。标识符的开头可以是英文字母或下划线, 不支持数字开头。除开头外，标识符可以由字母、数字、下划线构成。
以下为标识符的EBNF

```
key -> ( Alpha | '_' ) ( Alpha | '_' | Digit )*
String -> '"' (AnyChar)* '"'
Char -> ''' Alpha '''
AnyChar -> Digit | Alpha | '_' | ... | '!' (所有字符，其它省略)
Alpha -> 'a' | 'b' | ... | 'y' | 'z' | 'A' | 'B' | ... | 'Y' | 'Z'
```

2.2 数字

首先，TINY+可以支持整数（如233）和小数（如233.33）的表示。除此之外，TINY+还支持科学计数法的表示（如$1E+7$，$1E-6$）。科学计数法的底数可以是小数（如$2.13E+3$）。另外，TINY+支持数字的前缀正负号，可以表示负数（如-4，$-4.33E-8$）。
以下为注释的EBNF

```
Number -> Signed | Scientific
Scientific -> (Signed | Unisigned) 'E' ( '+' | '-' ) Integer
Signed -> [ '+' | '-' ] UnSigned
Unisigned -> Integer | Decimal
Integer -> NonZeroDigit Digits | '0'
Decimal -> Integer '.' Digits
Digits -> Digit Digits | ϵ
Digit  -> '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
NonZeroDigit = '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
```

2.3 注释

TINY+的注释和$C/C++$相同。提供两种形式，第一种形式是$/**/$，即整段注释。第二种形式是$//$，可以注释当前这一行后面的内容。

2.4 错误类型

根据以上语法规则，我设计了4种词法相关的编译错误。

错误名称	解释	编号
$KeyError$	标识符错误	-1
$AnnError$	注释格式错误	-2
$SynError$	语法错误	-3
$UknError$	出现未知符号	-4

3 词法分析

为了简化实现难度，源代码要求任意两个token之间要有空格或换行符隔开，以方便提取。这样TINY+的词法就好提取很多，也就可以直接通过硬编码实现。对于每个token，先提取第一个字符，然后根据第一个字符的类型进行编码。以下为具体实现思路。

3.1 如果第一个字符是数字 ('0'~'9') 或正负号 ('+', '-') 紧接着数字

如果第一个字符是数字，那么这个token就表示一个数字。

- 如果第一个字符是'0'，则这个数字要么是数字0，要么是整数部分为0的小数。如果都不是，则报错`KeyError`。
- 如果第一个字符是非零数字，则向后找到一个完整的连续字符串。
- 对于小数点'.'，出现的位置之前必须完全由数字组成，出现的位置后要么完全由数字组成，要么出现科学计数法。小数点只能出现一次，否则报错`KeyError`。
- 对于科学计数法的标志'e'或'E'，其左边是一个整数或一个小数，右边是正负号接一个整数。同样地，科学计数法的标志也只能出现一次，否则报错`KeyError`。

另外，如果第一个字符是正负号，并且紧接着一个数字，那么者表示的也是一个数字。从正负号的下一位开始，按照上述方法提取token即可。

相关代码如下。

```
int GetFullNumber(Scanner* Scan){
    int pos = Scan->pos - 1;
    int len = Scan->length;
```

```
    char* buf = Scan->buf;
    if(buf[pos]=='+' || buf[pos]=='-') pos++;
    int st = pos;
    int flag = 0;

    if(buf[pos]=='0' && !(isspace(buf[pos+1]) || buf[pos+1]=='.' ||
buf[pos+1]==';'))
        return -1;

    for(;pos<len;pos++)
    {
        if(isspace(buf[pos])) break;
        if(buf[pos] == ';') break;
        if(isdigit(buf[pos])) continue;
        if(buf[pos]=='.' && flag==0){
            if(!isdigit(buf[pos-1]) || !isdigit(buf[pos+1]))
                return -1;
            flag=1; continue;
        }
        if(flag<2 && (buf[pos]=='e' || buf[pos]=='E')){
            if(!isdigit(buf[pos-1])) return -1;
            if(buf[pos+1]!='+' && buf[pos+1]!='-') return -1;
            if(!isdigit(buf[pos+2])) return -1;
            flag=2; pos+=2; continue;
        }
        return -1;
    }

    Scan->pos = pos;
    int NumLen = pos - st;
    return NumLen;
}
```

3.2 如果第一个字符是字母 ('a'~'z' or 'A'~'Z') 或下划线 ('_')

如果第一个字符是字母或下划线，那么这个token是一个标识符。标识符由任意多个字符，下划线，数字构成。举几个例子，*test101*，test_test，*test101_test101*。如果出现其它字符，则报错`KeyError`。相关代码如下。

```
int GetFullKey(Scanner* Scan){
    int pos = Scan->pos - 1;
    int st = pos;
    int len = Scan->length;
    char* buf = Scan->buf;

    for(;pos<len;pos++)
    {
        if(isspace(buf[pos])) break;
        if(buf[pos] == ';') break;
        if(isletter(buf[pos])) continue;
        if(isdigit(buf[pos])) continue;
        return -1;
    }
    Scan->pos = pos;

    int NumLen = pos - st;
```

```
    return NumLen;
}
```

3.3 如果第一个字符是反斜杠 ('/')

根据TINY+语法的定义，第一个字符是反斜杠则这里出现一段注释。观察下一个符号，如果也是反斜杠则是同行注释，跳过该行继续进行词法分析。如果是 "*" *则两个字符构成* "/" 符号，则向后寻找 "*/" 符号，构成整段注释。若未找到则报错 `AnnError`。

3.4 如果第一个字符是引号 (' ', " ")

第一个字符是引号，则显然这是一个字符串。从当前字符开始向后找到第一个引号，这两个引号之间的字符就构成一个字符串。该字符串加上前后两个引号自动归为一个token。该规则对单引号和双引号都适用。若没找到则报错 `SynError`。

3.5 如果第一个字符是符号 (e.g. '+', '-')

如果第一个字符是符号，那么先看这个token的长度是否为1，即这个字符后接字符是否为空格或换行。token长度为1则该符号自动成为一个符号token。若不是1，则有可能是长度为2的符号token（e.g. '+=', '!=', ':='）。判断以下是否为这些token，若满足的产生一个符号token，否则报错 `UknError`。

3.6 如果第一个字符是左括号 ('(')

如果第一个字符是左括号，则该字符自动成为一个token。向后寻找，如果能找到右括号则合法，否则报错 `SynError`。

3.7 主要代码

以下为词法分析的主代码。

```
int NextToken(Scanner* Scan, TokenList* ToList)
{
    char ch = NextChar(Scan);
    int ret = -1;
    char* now = NULL;
    if(!ch)return 1;

    if(isdigit(ch)){
        ret = GetFullNumber(Scan);
        if(ret<0)return KeyError;
    }
    else if((ch=='+' || ch=='-') && isdigit(GetNow(Scan))){
        ret = GetFullNumber(Scan);
        if(ret<0) return KeyError;
        ret++;
    }
    else if(isletter(ch)){
        ret = GetFullKey(Scan);
        if(ret<0) return KeyError;
    }
    else if(ch=='/' && GetNow(Scan)=='*'){
        int res = GetSpecify2(Scan, "*/");
        if(!res)return AnnError;
        return 0;
    }
    else if(ch=='/' && GetNow(Scan)=='/'){
```

```
        int res = GetSpecify1(Scan, '\n');
        if(!res)return 1;
        return 0;
    }
    else if(ch=='('){
        int pos = Scan->pos;
        int res = GetSpecify1(Scan, ')');
        if(!res)return SynError;
        Scan->pos = pos;
        ret = 1;
    }
    else if(ch=='\"' || ch=='\''){
        int st = Scan->pos-1;
        int res = GetSpecify1(Scan, ch);
        if(res) ret = Scan->pos - st;
        if(ch=='\'' && ret>3) return SynError;
    }
    else if(isToken(ch) && isspace(GetNow(Scan))){
        ret = 1;
    }
    else if(isEqu(ch) && GetNow(Scan)=='='){
        Move(Scan, 1);
        ret = 2;
    }
    else return UknError;

    now = Copy(Scan, ret);
    bool res = Insert(ToList, now);
    if(!res) return 0;
    return 0;
}
```

4 测试样例与运行结果

4.1 成功编译

以下为TINY+的样例代码。

```
/* Sample program
  in TINY language -
  computes factorial
}*/
read x; /* input an integer */
read x1; /* input an integer */
x := 1e+3 ;
x1 := -3.444E-33 ;
str1 := "STR1!!!!" ;
ch1 := 'a';
x2 := ( 2 + 3 * 5 ) ;

if 0 < x then // don't compute if x <= 0
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
```

```
  write fact // output factorial of x
end
```

以下为测试命令。

```
./test.exe FileName
```

以下为编译后产生的token序列结果。

```
read x ; read x1 ; x := 1e+3 ; x1 := -3.444E-33 ; str1 := "STR1!!!!" ; ch1 := 'a' ; x2 := ( 2 + 3 * 5 ) ; if 0 < x then fact := 1 ; repeat fact := fact * x ; x := x - 1 until x = 0 ; write fact end
```

4.2 编译错误

样例代码1

```
int main ()
    int 1ab;
```

编译结果

```
Lexical Analysis Fail
code1.tny:1:6 Key Error: invalid number
```

报错KeyError，1ab不是一个正确的标识符/数字。

样例代码2

```
int main ()
    int x;
    x := ( 2 + 3 ;
```

编译结果

```
Lexical Analysis Fail
code1.tny:2:16 Syntex Error: missing terminating ')'character
```

报错SyntexError，缺右括号