上海大学

SHANGHAI UNIVERSITY

<编译原理>实验报告

学　　院　计算机工程与科学学院

组　　号　8

实 验 题 号　二

日　　期　2022 年 5 月 6 日

学号	姓名	主要工作	贡献因子
16121803	许睿	代码调试	20%
19120188	孙瑶	代码优化	20%
19120191	汪雨卿	代码测试	20%
19121442	曹卓文	代码编写	20%
19122408	严邹莹	报告撰写	20%

实验二 词法分析

一、实验目的与要求

1、根据 PL/0 语言的文法规范，编写 PL/0 语言的词法分析程序。

2、通过设计调试词法分析程序，实现从源程序中分出各种单词的方法；加深对课堂教学的理解；提高词法分析方法的实践能力。

3、掌握从源程序文件中读取有效字符的方法和产生源程序的内部表示文件的方法。

4、掌握词法分析的实现方法。

5、上机调试编写出词法分析程序。

二、实验环境

c/c++、visual studio

三、实验内容

输入 PL/0 语言程序，输出程序中各个单词符号（关键字、专用符号以及其它标记）。

四、实验内容的设计与实现

4.1 实验设计

利用了状态机的思想，将状态分为 START（初始）, WORD（字母）, NUMBER（数字）, OPERATOR（操作符）, BOUND（边界）五个状态，通过状态转移对输入的文本进行分词操作，状态转移表如下表所示：

其中，S0->START　　S1->WORD　　S2->NUMBER

　　S3->OPERATOR　　S4->BOUND

表 1 状态转移表

输入 / 状态	字母/下划线	数字	运算符	标点	空格/换行符	非法符号
S0	S1	S2	S3	S4	S0	ERROR
S1	S1	S1	S3	S4	S0	ERROR
S2	ERROR	S2	S3	S4	S0	ERROR
S3	S1	S2	S3	S4	S0	ERROR
S4	S1	S2	S3	S4	S0	ERROR

每当接收一个字符时，都会根据状态转移表进行状态转移。

当状态改变时判断为接收到一个完整的分词，然后根据其上一个状态（prestate）以及保留字、操作符和分隔符列表中的元素进行比对，判断该分词属于单词，数字，运算符，标点和非法符号中的哪一项，并输出结果。

其中，当上一个状态(prestate)为数字，当前状态(state)为字母时，判断为非法标识符转换为 ERROR 状态。在 ERROR 状态下会输出错误信息，然后返回到 S0 状态继续接收输入。

4.2 主要代码实现

1、保留字、操作符、分隔符列表

```
const unordered_map<string, string> alphabet::basic_word = {
        {"begin",       "beginsym"},
        {"call",        "callsym"},
        {"const",       "constsym"},
        {"do",          "dosym"},
        {"end",         "endsym"},
        {"if",          "ifsym"},
```

```
        {"odd",          "oddsym"},
        {"procedure", "proceduresym"},
        {"read",         "readsym"},
        {"then",         "thensym"},
        {"var",          "varsym"},
        {"while",        "whilesym"},
        {"write",       "writesym"}
};

const unordered_map<string, string> alphabet::operators = {
        {"+",            "plus"},
        {"-",            "minus"},
        {"*",            "times"},
        {"/",            "slash"},
        {"=",            "eql"},
        {"#",            "neq"},
        {"<",            "lss"},
        {"<=",           "leq"},
        {">",            "gtr"},
        {">=",           "geq"},
        {":=",           "becomes"}
};

const unordered_map<string,string> alphabet::boundary = {
```

```
        {"(",           "lparen"},
        {")",           "rparen"},
        {",",           "comma"},
        {";",           "semicolon"},
        {".",           "period"}
};

const vector<string> alphabet::bound = {"(",")",",",";",".","};
const vector<string>
alphabet::operators_list = {"+","-","*","/",":=","#","<","<=",">",">=","="};
```

2、状态转移代码

```
if (isalpha(ch) || ch == '_') {
    prestate = state;
    state = WORD;
    // 如果是非字母状态下连接数字，认为是数字开始
} else if (isdigit(ch)) {
    prestate = state;
    if (state != WORD)
        state = NUMBER;
    // 切换操作符状态
} else if (ch == '=' || ch == ':' || ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '#' ||
            ch == '<' || ch == '>') {
    prestate = state;
```

```
        state = OPERATOR;
        // 边界状态
    } else if (ch == '.' || ch == ';' || ch == '(' || ch == ')' || ch == ',') {
        prestate = state;
        state = BOUND;
        // 换行
    } else if (ch == ' ' || ch == '\n' || ch == '\t') {
        if (ch == '\n')count_line++;
        prestate = state;
        state = START;
        // 异常状态，重置到 start
    } else {
        cout << "error \"illegal character\" occur at line " << count_line << ": " << ch << endl;
        outfile << "error \"illegal character\" occur at line " << count_line << ": " << ch << endl;
        prestate = state;
        state = START;
    }
```

4.3 实验结果

左图为测试数据，右图为输出结果。

数据一：

infile1.txt - 记事本

```
const a=10;
var b,c;
begin
    read(b);
    c:=a+b;
    write(c)
end.
```

output1.txt - 记事本

```
(constsym, const)
(ident, a)
(eql, =)
(number, 10)
(semicolon, ;)
(varsym, var)
(ident, b)
(comma, ,)
(ident, c)
(semicolon, ;)
(beginsym, begin)
(readsym, read)
(lparen, ()
(ident, b)
(rparen, ))
(semicolon, ;)
(ident, c)
(becomes, :=)
(ident, a)
(plus, +)
(ident, b)
(semicolon, ;)
(writesym, write)
(lparen, ()
(ident, c)
(rparen, ))
(endsym, end)
(period, .)
```

数据二：

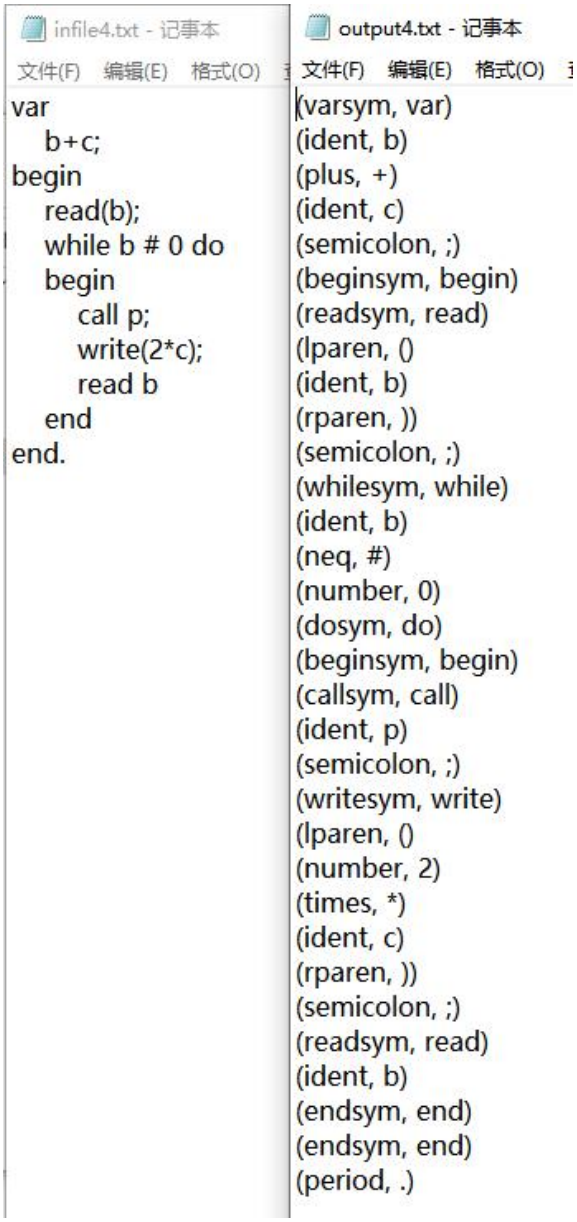
infile2.txt - 记事本

```
var n,i,apple;
procedure fordiy;
        const a=7;
        var b,c;
        begin
                while i>0 do
                begin
                        apple:=(apple+1)*2;
                        i:=i-1;
                end;
        end;
begin
   read(n);
   i:=n;
   apple:=1;
   call fordiy;
   write(apple);
end.
```

output2.txt - 记事本

```
(varsym, var)
(ident, n)
(comma, ,)
(ident, i)
(comma, ,)
(ident, apple)
(semicolon, ;)
(proceduresym, procedure)
(ident, fordiy)
(semicolon, ;)
(constsym, const)
(ident, a)
(eql, =)
(number, 7)
(semicolon, ;)
(varsym, var)
(ident, b)
(comma, ,)
(ident, c)
(semicolon, ;)
(beginsym, begin)
(whilesym, while)
(ident, i)
(gtr, >)
(number, 0)
(dosym, do)
(beginsym, begin)
(ident, apple)
(becomes, :=)
(lparen, ()
(ident, apple)
(plus, +)
(number, 1)
(rparen, ))
(times, *)
(number, 2)
```

数据三：



五、收获与体会

许睿：

本次词法分析实验主要还是沿用了状态机思想，最深的感受还是在测试用例的编写上，界符和非法字符是之前自己没有考虑到的一部分。验收时通过老师的指点找到了疏忽的漏洞，希望在接下来的几个实验里能做的更好。

孙瑶：

本次词法分析的实验中，我刚开始的想法是将不同种类包括标识符、单词、标点等放入字符常量数组，再不断扫描的过程中，循环判断去区分字符，但是在代码编写中发现存在一定的冗余，因此之后使用状态机的思想，利用状态转移的结果进行词法分析，使得整个分析过程更加便捷和易于理解。在这次实验中，通过具体代码的编写让我更深刻的体会对于每个词的分析过程以及有效字符的提取，对词法分析有了更进一步的认识。

汪雨卿：

通过本次词法分析的实验，帮助我更好的理解了编译器对于识别不同类型标识符的实现过程。在编译程序的过程中，我们小组采用状态机的机制去进行词法的分割，通过状态的变化进行词法的分割。在整个实验的过程中，我遇到的最大的困难在于设计样例，发现 bug，并且修改 bug 的过程。尤其是对于双字符和单字符的判定，是我们小组最开始没有处理好的。之后通过交流和讨论，我们找到了解决方法，优化了之前的程序。这个实验让我收获颇丰。

曹卓文：

这次实验是进行词法分析，通过这次实验，我学会了如何对输入的文本进行分词。练习了使用 C++实现状态机，对 PL0 语言实现分词操作，并且在测试过程中我对异常输入的处理有了更深刻的认识，通过实践练习如何识别异常的数字输入。加深了我对于编译程序是如何提取有效字符的认识。

严邹莹：

这次实验是进行词法分析，通过这次实验，我学会了如何进行语言处理，会对输入的文本进行简单的单词分割和词性判断。在这次实验中，我们组用到了状态机的思想，通过对每次输入的状态转移结果进行分词操作，加深了我对于编译程序是如何提取有效字符以及如何对这些字符进行内部表示的理解。