厦門大學

软件学院

《编译技术》实验报告

题　　目　标识符和实数的识别

姓　　名　陈澄

学　　号　32420212202930

班　　级　软工三班

实验时间　2024/03/19

2024　年　03　月　19　日

1 实验目的

基本掌握计算机语言的词法分析程序的开发方法。

2 实验内容

编制一个能够分析三种整数、标识符、主要运算符和主要关键字的词法分析程序。

3 实验要求

1. 根据以下的正规式，编制正规文法，画出状态图；

基本要求：

标识符 <字母>(<字母>|<数字字符>)

十进制整数 (1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)* |0

八进制整数 0o(0|1|2|3|4|5|6|7)(0|1|2|3|4|5|6|7)*

十六进制整数

0x(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)

运算符和分隔符 + - * / > < = ( ) ;

关键字 if then else while do

附加要求：

标识符 <字母>(<字母>|<数字字符>)*(ε |_|.)(<字母>|<数字字符>)

十进制整数 (1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)* |0 (ε |.(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)*)

八进制整数 0o(0|1|2|3|4|5|6|7)(0|1|2|3|4|5|6|7)* (ε |.(0|1|2|3|4|5|6|7)(0|1|2|3|4|5|6|7)*)

十六进制整数

0x(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)* (ε |.(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)(0|1|2|3|4|5|6|7|8|9|a|b|c|d|e|f)*)

2. 根据状态图，设计词法分析函数 int scan( )，完成以下功能：
   1） 从键盘读入数据，分析出一个单词。
   2） 返回单词种别（用整数表示），
   3） 返回单词属性（不同的属性可以放在不同的全局变量中）。
3. 编写测试程序，反复调用函数 scan( )，输出单词种别和属性。

4 实验环境

PC 微机

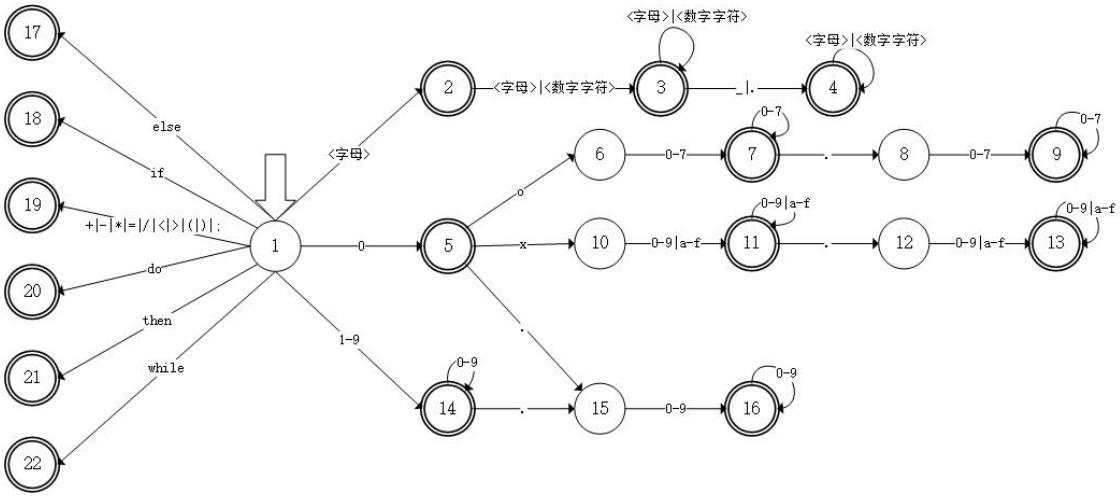
Windows 操作系统

Turbo C 程序集成环境或 Visual C++ 程序集成环境

5 实验步骤

1. 根据状态图，设计词法分析算法。

状态图如下：



设计算法：

采用状态转移算法，大致思路：检查当前状态->根据当前状态将第 i 个字符唯一转化为下一状态->字符串指针+1

伪代码：

```
While(字符串未结束){
        Switch(状态):
        Case 状态 1:{
                Switch(字符):
                Case a:状态 1 转化为状态 2;break;
                Case b:状态 2 转化为状态 3;break;
                ......
        }
        Case 状态 2:{
                ......
        }
        ......
        字符串指针后移;
}
```

2. 采用 C 语言，设计函数 scan( )，实现该算法。

由于 switch...case 内部嵌套的字符选择逻辑较简单，改用 if...else 结构进行选择

```
int scan(char s[]) {
    int status = 1, i = 0;
    //初始状态为1, 不合法状态为0, i为指针
    while (s[i] && status) {
        switch (status)
        {
        case 1: {
            /*char s_copy[10];
            strncpy(s_copy, s, n);
            s_copy[n] = '\0';*/
            if (strcmp(s, "else") == 0) {
                status = 17; i += 3; break;
            }
            if (strcmp(s, "if") == 0) {
                status = 18; i += 1; break;
            }
            if (strcmp(s, "do") == 0) {
                status = 20; i += 1; break;
            }
            if (strcmp(s, "then") == 0) {
                status = 21; i += 3; break;
            }
            if (strcmp(s, "while") == 0) {
                status = 22; i += 4; break;
            }
            if (s[i] == '0') status = 5;
            else if (isAlpha(s[i]))status = 2;
            else if (isPunct(s[i]))status = 19;
            else if (s[i] >= '1' && s[i] <= '9')status = 14;
            else status = 0;
            break;
        }
        case 2: {
            if (isAlpha(s[i]) || isNum(s[i]))status = 3;
            else status = 0;
            break;
        }
        case 3: {
            if (isAlpha(s[i]) || isNum(s[i]))status = 3;
            else if (s[i] == '.' || s[i] == '_')status = 4;
            else status = 0;
            break;
        }
        case 4: {
            if (isAlpha(s[i]) || isNum(s[i]))status = 3;
            else status = 0;
            break;
        }
        case 5: {
            if (s[i] == 'o' || s[i] == 'O')status = 6;
            else if (s[i] == 'x' || s[i] == 'X')status = 10;
            else if (s[i] == '.')status = 16;
            else status = 0;
            break;
        }
        case 6: {
            if (isInt8(s[i]))status = 7;
            else status = 0;
            break;
        }
        case 7: {
            if (isInt8(s[i]))status = 7;
            else if (s[i] == '.')status = 8;
            else status = 0;
            break;
        }
        case 8: {
            if (isInt8(s[i]))status = 9;
            else status = 0;
            break;
        }
        case 9: {
            if (isInt8(s[i]))status = 9;
            else status = 0;
            break;
        }
        case 10: {
            if (isInt16(s[i]))status = 11;
            else status = 0;
            break;
        }
        case 11: {
            if (isInt16(s[i]))status = 11;
            else if (s[i] == '.')status = 12;
            else status = 0;
            break;
        }
        case 12: {
            if (isInt16(s[i]))status = 13;
            else status = 0;
            break;
        }
        case 13: {
            if (isInt16(s[i]))status = 13;
            else status = 0;
            break;
        }
        case 14: {
            if (isNum(s[i]))status = 14;
            else if (s[i] == '.')status = 15;
            else status = 0;
            break;
        }
        case 15: {
            if (isNum(s[i]))status = 16;
            else status = 0;
            break;
        }
        default:status = 0;
        }
        i++;
    }
    if (status==1 || status == 6 || status == 8
        || status == 10 || status == 12 || status == 15)status = 0;
    //除去非终结态
    return status;
    //将状态返回
}//状态转移算法
string processStatus(int status) {
```

3. 编制测试程序（主函数 main）。

主要思路：

Step1:识别前 k 个字符，若满足前 k 个字符可以被识别，而前 k+1 个字符无法被识别，则说明前 k 个字符为一个完整的终结符。

Step2:通过 scan 方法识别该终结符，并输出。

Step3:指针后移，将已经识别过的终结符去除，回到 Step1 重复。

```
//循环读取并识别
int i = 0, j = 0;
char temp[100];
while (s[i + j]) {
    if (!(s[i + j] == ' ' || isPunct(s[i + j]))) {
        temp[j] = s[i + j];
        j++;
        continue;
    }
    temp[j] = '\0';
    cout << temp << "\t" << processStatus(scan(temp)) << endl;
    char punct = 0;
    while (s[i + j] == ' ' || isPunct(s[i + j])) {
        if (isPunct(s[i + j]))punct = s[i + j];
        j++;
    }
    if (punct)cout << punct << "\t_" << endl;
    i += j;
    j = 0;
}
```

4. 调试程序：输入一组单词，检查输出结果。

输入数据 1：0 92+data> 0x3f 0o0 while a+acc>xx do x=x-1;

```
0  92+data>   0x3f  0o0  while a+acc>xx do x=x-1;
0       INT10
92      INT10
+       _
data    IDN
>       _
0x3f    INT16
0o0     INT8
while   WHILE
a       IDN
+       _
acc     IDN
>       _
xx      IDN
do      DO
x       IDN
=       _
x       IDN
-       _
1       INT10
;       _
```

输入数据 2：a=6.2+a*0X88.80;

```
a=6.2+a*0X88.80;
a        IDN
=        _
6.2      INT10
+        _
a        IDN
*        _
0X88.80  INT16
;        _
```

输入数据 3：if a>b then a=b else a=b-1+c;

```
if a>b then a=b else a=b-1+c;
if      IF
a       IDN
>       _
b       IDN
then    THEN
a       IDN
=       _
b       IDN
else    ELSE
a       IDN
=       _
b       IDN
-       _
1       INT10
+       _
c       IDN
;       _
```

6 实验报告要求

1. 词法的正规式描述

   实验要求中已经给出

2. 变换后的正规文法

标识符：

文法 G=({S,A,B,C},{<字母>,<数字>,.,_},P,S)其中 P 由下列产生式生成：

S-><字母>A

A->ε

A-><字母>B|<数字>B

B-><字母>B|<数字>B

B->ε

B->.C|_C

C-><字母>C|<数字>C

C->ε

十进制整数：

文法 G=({S,A,B,C,D},{<数字>,.,},P,S)其中 P 由下列产生式生成：

S->0A

A->ε

A->.C

S->1B|2B|3B|4B|5B|6B|7B|8B|9B

B-><数字>B

B->ε

B->.C

C-><数字>D

D-><数字>D

D->ε

八进制整数：

文法 G=({S,A,B,C,D},{0,1,2,3,4,5,6,7,.,},P,S)其中 P 由下列产生式生成：

S->0oA

A->0B|1B|2B|3B|4B|5B|6B|7B

B->0B|1B|2B|3B|4B|5B|6B|7B

B->ε

B->.C

C->0D|1D|2D|3D|4D|5D|6D|7D

D->0D|1D|2D|3D|4D|5D|6D|7D

D->ε

十六进制整数：

文法 G=({S,A,B,C,D},{0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f,.,},P,S)其中 P 由下列产生式生成：

S->0xA

A->0B|1B|2B|3B|4B|5B|6B|7B|8B|9B|aB|bB|cB|dB|eB|fB

B->0B|1B|2B|3B|4B|5B|6B|7B|8B|9B|aB|bB|cB|dB|eB|fB

B->ε

B->.C

C->0D|1D|2D|3D|4D|5D|6D|7D|8B|9B|aB|bB|cB|dB|eB|fB

D->0D|1D|2D|3D|4D|5D|6D|7D|8B|9B|aB|bB|cB|dB|eB|fB

D->ε

运算符和分隔符：

文法 G=({S},{+,-,*,/,<,>,(,),=,;},P,S)其中 P 由下列产生式生成：

S->+|-|*|/|<|>|(|)|=|;

关键字：

文法 G=({S},{while,do,if,else,then},P,S)其中 P 由下列产生式生成：

S->while|do|if|else|then

3. 状态图

实验步骤中已经给出

4. 词法分析程序的数据结构与算法

实验步骤中已经给出

7 思考题

1. 词法分析能否采用空格来区分单词？

答：不能，单词与单词间不仅可以通过空格也可以通过运算符，换行符，制表符等等连接，因此不能仅仅通过空格来区分单词。

2. 程序设计中哪些环节影响词法分析的效率？如何提高效率？

答：

(1)数据结构的选择：选择合适的数据结构来存储和组织文本数据对词法分析效率至关重要。例如，使用合适的数据结构来存储单词、标记或者字符流，能够提高词

法分析的效率。常见的数据结构如哈希表、树结构等可以被用来加快查找和匹配过程。

(2)算法的设计：采用高效的算法能够显著提高词法分析的效率。例如，使用适当的字符串匹配算法来识别和提取单词，或者利用自动机等数据结构来进行状态转换和匹配。

(3)输入/输出：在词法分析中，输入输出操作可能成为性能瓶颈。因此，优化输入输出操作，减少文件读写次数，或者采用缓冲技术来减少 IO 开销，都能提高词法分析的效率。

(4)并行处理：采用并行处理技术，将词法分析任务拆分成多个子任务并行处理，可以加速整个词法分析过程。