编译原理 实验一：词法分析

09019106 牟倪

目录

一、实验目标

- 理解词法分析过程。
- 熟练掌握 FA、RE 和 Grammar 之间的转换。
- 自主实现词法分析程序。

二、实验内容

- 自主定义 token 和对应的 RE。
- 完成词法分析程序：
  - 输入字符串，输出解析结果 token 串。
  - 对不合法的输入字符串，报错。

三、基本思路

首先利用 Thompson's construction rules 将 RE 转换为 NFA，利用 subset method 将 NFA 转换为 DFA，利用 partition method 将 DFA 转换为 minimum DFA。

然后，对 DFA 的状态进行标号，构造相应的状态转换逻辑。逐个读入字符，模拟状态转换过程，从而判定字符串是否合法以及 token 类型。

四、假设与名词定义

基本 RE 定义：

- letter → A | B | … | Z | a | b | … | z
- nonzero_digit → 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- digit → 0 | nonzero_digit

token 的 RE 定义：

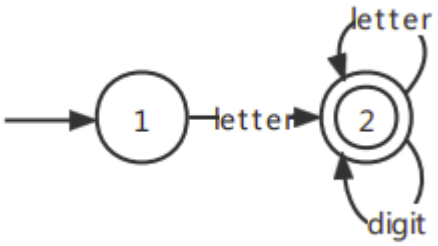
- identifier → letter (letter | digit)*
- integer → nonzero_digit (digit)*
- float → (integer | 0) . (digit)+
- operator → (+ | - | * | /)

（为了简化情况，减少二义性（如-0.2 可以被解析为负 0.2 或减 0.2），假设 integer 和 float 均为正数）

五、相关 FA 描述

5.1 identifier 的 FA 描述

可以直接得到 minimum DFA：

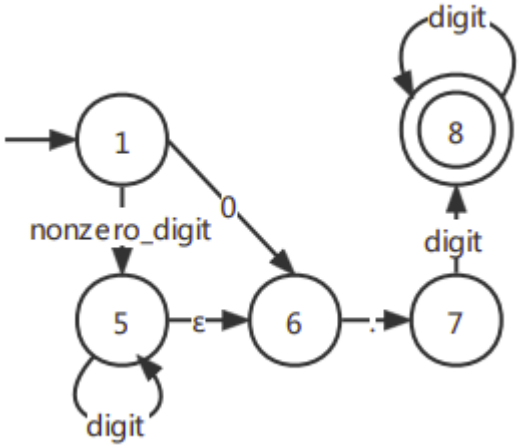


5.2 integer 的 FA 描述

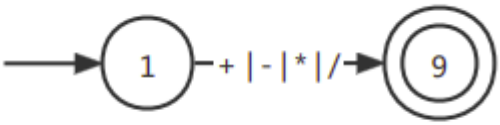
可以直接得到 minimum DFA：

5.3 float 的 FA 描述

根据 integer 的 minimum DFA，得到 float 的 NFA：



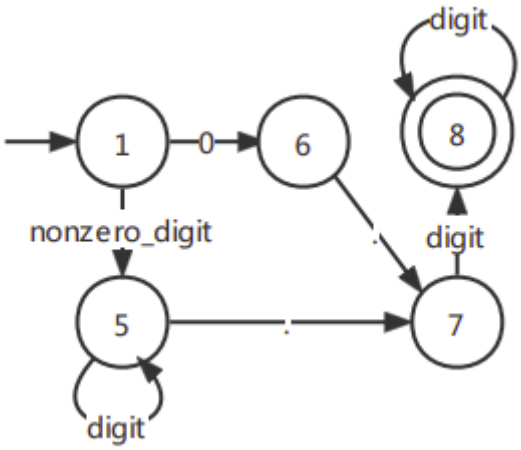
从 NFA 转化为 DFA：



已经是 minimum DFA。

5.4 operator 的 FA 描述

可以直接得到 minimum DFA：



六、核心算法

6.1 程序框架

构建 `LexicalAnalysis` 类。

成员变量：

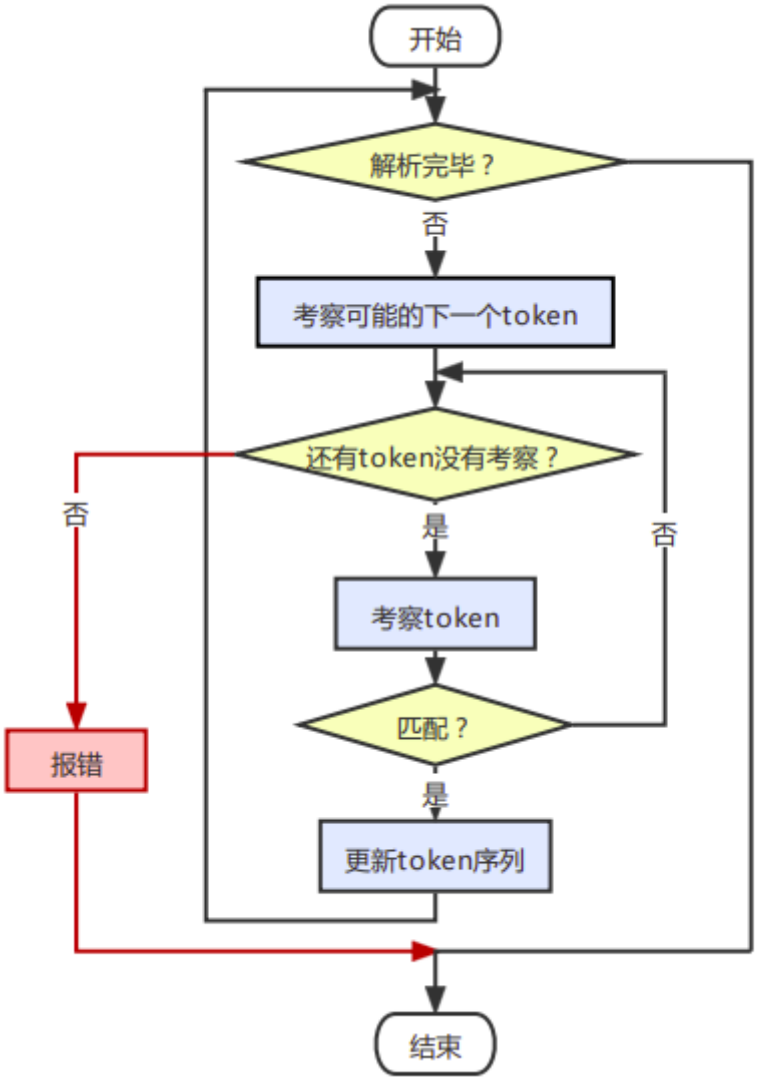
- `state`，`int` 型变量，表示目前所处的状态。
- `input`，`string` 型变量，用来记录待解析的字符串。
- `pos`，`int` 型变量，起指针的作用，指向字符串中已经成功解析的位置。
- `cur`，`int` 型变量，起指针的作用，指向字符串中目前解析到的位置。
- `output`，`queue<string>`型变量，记录解析结果。

成员函数：

- `void init()`，读入字符串 `input`，为 `input` 添加后缀`'#'`。

- `void analysis()`，词法分析。
- `char curChar()`，返回当前字符
- `char nextChar()`，返回当前字符的下一个字符。
- `bool isLetter(char c)`、`bool isNonzeroDigit(char c)`、`bool isDigit(char c)`、`bool isPoint(char c)`、`bool isOperator(char c)`、`bool isZero(char c)`，用来判断字符类型。
- `bool anaIdentifier()`、`bool anaInteger()`、`bool anaFloat()`、`bool anaOperator()`，解析出下一个 token，返回值为是否成功。
- `void fail()`，用来报错。
- `void printOutput()`，打印解析结果。

利用 `LexicalAnalysis` 类，按照如下流程图，可以实现词法分析程序：



在 6.2-6.6 节，我会详细介绍 `LexicalAnalysis` 类的成员函数，将功能与上述流程图对应。

6.2 analysis()函数

analysis()函数通过当前字符的类型判断下一个可能的 token，并对所有可能的 token 逐个尝试。如果所有尝试都失败，调用 fail()函数。analysis()函数对应前两个带判断条件的 while 循环。代码如下：

```
void analysis(){
        while(curChar() != '#'){
            if(isLetter(curChar())){
                if(anaIdentifier()) continue;
            };
            if(isNonzeroDigit(curChar())){
                if(anaFloat()) continue;
                if(anaInteger()) continue;
            }
            if(isZero(curChar())){
                if(anaFloat()) continue;
            }
            if(isOperator(curChar())){
                if(anaOperator()) continue;
            }
            fail();
        }
    }
```

6.3 anaIdentifier()函数

anaIdentifier()函数用来解析 identifier。代码如下：

```
bool anaIdentifier(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                ++cur; state = 2;
                break;
            case 2:
                curc = curChar();
                if(!(isLetter(curc) || isDigit(curc))){
                    output.push("<identifier>");
                    pos = cur;
```

```
                    state = 1;
                    return true;
                }
                else{
                    ++cur; break;
                }
            default:
                state = 1; cur = pos; return false;
            }
        }
    }
```

6.4 anaInteger()函数

anaInteger()函数用来解析 integer。因为 integer 和 float 具有左公共因子，因此额外判断下一个字符是否为'.'，若是则返回 false，而非取 float 的整数部分作为 integer。代码如下：

```
bool anaInteger(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                ++cur; state = 3;
                break;
            case 3:
                if(!isDigit(curChar())){
                    if(isPoint(curChar())){
                        state = 1;
                        cur = pos;
                        return false;
                    }
                    output.push("<integer>");
                    pos = cur;
                    state = 1;
                    return true;
                }
                else{
                    ++cur; break;
                }
            default:
                state = 1; cur = pos; return false;
```

```
            }
        }
    }
```

6.5 anaFloat()函数

anaFloat()函数用来解析 float。代码如下：

```
bool anaFloat(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                if(isZero(curChar())) state = 6;
                else state = 5;
                ++cur;
                break;
            case 5:
                if(isDigit(curChar())){
                    ++cur; break;
                }
                else if(isPoint(curChar())){
                    state = 7; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
            case 6:
                if(isPoint(curChar())){
                    state = 7; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
            case 7:
                if(isDigit(curChar())){
                    state = 8; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
            case 8:
```

```
                if(!isDigit(curChar())){
                    output.push("<float>");
                    pos = cur;
                    state = 1;
                    return true;
                }
                else{
                    ++cur; break;
                }
            default:
                state = 1; cur = pos; return false;
            }
        }
    }
```

6.6 anaOperator()函数

anaOperator()函数用来解析 operator。代码如下：

```
bool anaOperator(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                ++cur; state = 9; break;
            case 9:
                output.push("<operator>");
                pos = cur;
                state = 1;
                return true;
            default:
                state = 1; cur = pos; return false;
            }
        }
    }
```

七、实验

- 目的：验证 identifier 是否成功解析。
  输入：muni
  输出：<identifier>

- 目的：验证 integer 是否成功解析。

  输入：`123893827483328234`

  输出：`<integer>`
- 目的：验证 float 是否成功解析。

  输入：`0.32482734823773732`

  输出：`<float>`
- 目的：验证多个 token 的情形。

  输入：`123+456+abc`

  输出：`<integer> <operator> <integer> <operator> <identifier>`
- 目的：验证多个 token 的情形。

  输入：`muni123+haha+yes+123`

  输出：`<identifier> <operator> <identifier> <operator> <identifier> <operator> <integer>`
- 目的：检查格式错误的 float 是否报错。

  输入：`0123.23232`

  输出：`lexical analysis fail!`
- 目的：检查格式错误的 float 是否报错。

  输入：`0.123.456.789`

  输出：`lexical analysis fail!`
- 目的：检查格式错误 identifier 的是否报错。

  输入：`@a123bdbe`

  输出：`lexical analysis fail!`

运行截图（部分）：

```
please input your string:
muni
the result of lexical analysis is:
<identifier>

please input your string:
123893827483328234
the result of lexical analysis is:
<integer>

please input your string:
0.32482734823773732
the result of lexical analysis is:
<float>

please input your string:
123+456+abc
the result of lexical analysis is:
<integer> <operator> <integer> <operator> <identifier>

please input your string:
muni123+haha+yes+123
the result of lexical analysis is:
<identifier> <operator> <identifier> <operator> <identifier> <operator> <integer>

please input your string:
0123.23232
lexical analysis fail!
```

可以看出，实验结果正确，程序编写合理。

八、出现问题与解决方案

没有出现问题。

九、实验体会

经过本次实验，我复习了词法分析的内容，RE、Grammar、FA 之间的转换也更熟练了。将抽象的、看起来庞大难以解决的词法分析问题，转化成 RE 的表达形式，再转化成 FA 这样的数学模型，最后基于 FA 编写程序，让我因前人的智慧而震撼赞叹，深刻感受到了数学的魅力与重要性。

附录：源代码

```
#include<iostream>
#include<queue>
#include<string>
#include<cstring>

using namespace std;

class LexicalAnalyzer{
private:
 // 成员变量
 string input;
 queue<string> output;
 int state;
 int pos; int cur;
public:
    // init
    void init(string i){
        pos = cur = 0; state = 1;
        input = i + '#';
    }

// utility function
char curChar() {return input[cur];}
char nextChar() {return input[cur+1];}
void fail(){
  cout<<"lexical analysis fail!\n";
  exit(1);
}
bool isLetter(char c){
  return (c >= 'a' && c <= 'z')||(c >= 'A' && c <= 'Z');
}
bool isZero(char c) {return c == '0';}
bool isNonzeroDigit(char c){return c >= '1' && c <= '9';}
bool isDigit(char c) {return isZero(c) || isNonzeroDigit(c);}
bool isPoint(char c) {return c == '.';}
bool isOperator(char c){
  return (c == '+') || (c == '-') || (c == '*') || (c == '/');
}

// get next token
    bool anaIdentifier(){
```

```
42.         cur = pos;
43.         char curc = curChar();
44.         while(1){
45.             switch(state){
46.             case 1:
47.                 ++cur; state = 2;
48.                 break;
49.             case 2:
50.                 curc = curChar();
51.                 if(!(isLetter(curc) || isDigit(curc))){
52.                     output.push("<identifier>");
53.                     pos = cur;
54.                     state = 1;
55.                     return true;
56.                 }
57.                 else{
58.                     ++cur; break;
59.                 }
60.             default:
61.                 state = 1; cur = pos; return false;
62.             }
63.         }
64.     }
65.     bool anaInteger(){
66.         cur = pos;
67.         char curc = curChar();
68.         while(1){
69.             switch(state){
70.             case 1:
71.                 ++cur; state = 3;
72.                 break;
73.             case 3:
74.                 if(!isDigit(curChar())){
75.                     if(isPoint(curChar())){
76.                         state = 1;
77.                         cur = pos;
78.                         return false;
79.                     }
80.                     output.push("<integer>");
81.                     pos = cur;
82.                     state = 1;
83.                     return true;
84.                 }
85.                 else{
```

```
                    ++cur; break;
                }
            default:
                state = 1; cur = pos; return false;
            }
        }
    }
    bool anaFloat(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                if(isZero(curChar())) state = 6;
                else state = 5;
                ++cur;
                break;
             case 5:
                if(isDigit(curChar())){
                    ++cur; break;
                }
                else if(isPoint(curChar())){
                    state = 7; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
             case 6:
                if(isPoint(curChar())){
                    state = 7; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
             case 7:
                if(isDigit(curChar())){
                    state = 8; ++cur; break;
                }
                else {
                    state = 1; cur = pos; return false;
                }
             case 8:
                if(!isDigit(curChar())){
                    output.push("<float>");
```

```
                    pos = cur;
                    state = 1;
                    return true;
                }
                else{
                    ++cur; break;
                }
            default:
                state = 1; cur = pos; return false;
            }
        }
    }
    bool anaOperator(){
        cur = pos;
        char curc = curChar();
        while(1){
            switch(state){
            case 1:
                ++cur; state = 9; break;
            case 9:
                output.push("<operator>");
                pos = cur;
                state = 1;
                return true;
            default:
                state = 1; cur = pos; return false;
            }
        }
    }

    // analysis
    void analysis(){
        while(curChar() != '#'){
            if(isLetter(curChar())){
                if(anaIdentifier()) continue;
            };
            if(isNonzeroDigit(curChar())){
                if(anaFloat()) continue;
                if(anaInteger()) continue;
            }
            if(isZero(curChar())){
                if(anaFloat()) continue;
            }
            if(isOperator(curChar())){
```

```
174.                 if(anaOperator()) continue;
175.             }
176.             fail();
177.         }
178.     }
179. 
180.     // print result
181.     void printOutput(){
182.         cout<<"the result of lexical analysis is:\n";
183.         while(!output.empty()){
184.             cout<<output.front()<<" "; output.pop();
185.         }
186.         cout<<"\n\n";
187.     }
188. };
189. 
190. int main(){
191.     LexicalAnalyzer la;
192.     string input;
193.     while(true){
194.         cout<<"please input your string:\n";
195.         cin>>input;
196.         la.init(input);
197.         la.analysis();
198.         la.printOutput();
199.     }
200. }
```