

实验一

张配天-2018202180

2021 年 4 月 8 日

目录

1 实验内容	1
2 程序设计原理与方法	1
2.1 原理	1
2.2 方法	1
3 程序设计流程	2
4 程序设计清单	3
5 运行结果	3
6 程序使用说明	3
7 总结与完善	4

1 实验内容

用 C++ 实现对 C-语言的词法分析器。

2 程序设计原理与方法

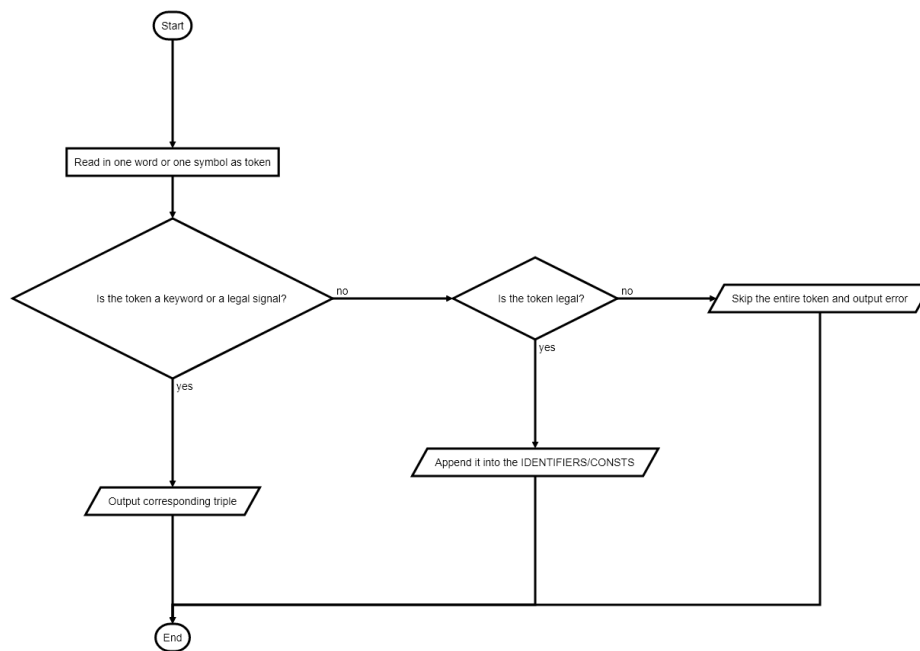
2.1 原理

在教材上给出了 C-语言的部分单词类型的有限状态自动机, 老师给出了更健全的正则表达式, 对两者进行分析, 结合有限状态自动机, 以老师给出的伪代码为框架, 自己实现词法分析器。

2.2 方法

在明确了各个组成成为的正则表达式后, 利用 C++ 的 if, else 语句, switch, case 语句实现状态之间的切换, 同时处理各种错误。

需要注意的, 词法分析器不涉及语法语义的处理, 所有要做的事情可以抽象为下面的流程图:



因此程序需要做的变得很简单, 即通过一个 while 循环连续地读入 token 之后处理 token 即可。

3 程序设计流程

根据上一个 section, 我们可以直观地讲词法分析器抽象成如下格式:

Algorithm 1: Lexer

```

while getchar() do
    if word then
        | parse_keywords_and_identifiers();
    else if digit then
        | parse_digit();
    else if operator then
        | parse_operator();
        if // then
            | parse_comment_line();
        else if /* then
            | parse_comment_block();
    else if separator then
        | parse_separator();
    else
        | parse_error();

```

在每个 `parse_xxx` 的函数中进行判断是否读到了当前 token 的结束, 以及输出和最后对已读取 token 的清除。

4 程序设计清单

如上, 我们要设计实现七种 **parse function**; 除此之外, 由于我预先将 **keywords, operator 和 separator** 都存放在分别的文件里, 需要设计相应的 **utility function** 来进行加载, 并且需要封装读取字符功能以及输出功能。

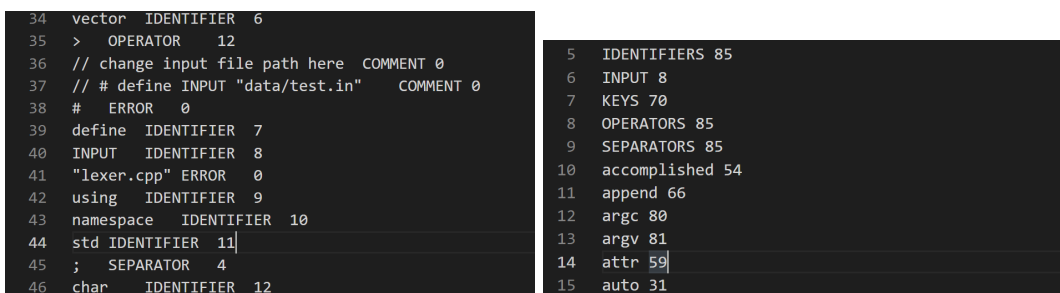
我使用 `std::map` 来保存一系列预先定义/代码生成的表, 包括预先定义的 **Keywords, Operators, Separators**, 和代码生成的 **Identifiers, Consts**; 对于所有字典, 我将 `token` 作为字典的键, 将其在该字典中的序号作为对应的值; 同时分别写入文件, 使用 `t` 作为分隔符, 每一个 token 占一行。

5 运行结果

对 `lexer.cpp` 进行词法分析, 即将源代码作为输入文件, 得到结果部分展示如图 1:

6 程序使用说明

1. 在 `data/KeyWords.txt` 中修改预设关键词;
2. 在 `data/Operators.txt` 中修改预设操作符;
3. 在 `data/Separators.txt` 中修改预设分隔符;



```
34 vector IDENTIFIER 6
35 > OPERATOR 12
36 // change input file path here COMMENT 0
37 // # define INPUT "data/test.in" COMMENT 0
38 # ERROR 0
39 define IDENTIFIER 7
40 INPUT IDENTIFIER 8
41 "lexer.cpp" ERROR 0
42 using IDENTIFIER 9
43 namespace IDENTIFIER 10
44 std IDENTIFIER 11
45 ; SEPARATOR 4
46 char IDENTIFIER 12
```

```
5 IDENTIFIERS 85
6 INPUT 8
7 KEYS 70
8 OPERATORS 85
9 SEPARATORS 85
10 accomplished 54
11 append 66
12 argc 80
13 argv 81
14 attr 59
15 auto 31
```

图 1: result(左), identifier(右)

4. 在 `lexer.cpp` 的宏定义中修改 `INPUT` 为测试文件路径, 即可编译运行;
5. 最后默认会将每个词的分析结果写入 `result.txt`, 并将 `identifier` 写入 `identifiers.txt`, 将 `consts` 写入 `consts.txt`。

7 总结与完善

- 之前由于没有分清楚词法分析和语法分析, 纠结了很长时间, 后来才理清逻辑。
- 在解析 (尤其是小数) 的时候, 很难按照有限状态自动机的逻辑去处理, 只能判断一些条件看是否报错, 和师兄交流过是否有更好的逻辑, 发现难以做到。