

北京交通大学
BEIJING JIAOTONG UNIVERSITY

《编译原理》 实验报告

实验名称:	专题 1_词法分析程序设计原理与实现
学 号:	
姓 名:	
学 院:	计算机与信息技术学院
日 期:	2022 年 10 月 17 日

目录

1.	实验目的	3
2.	实验内容	3
3.	实验要求	3
4.	程序实现	3
4.1.	相关环境介绍	3
4.2.	主要数据结构	3
4.3.	程序结构描述	4
4.3.1.	设计方法	4
4.3.2.	函数定义	4
5.	程序测试	5
6.	实验汇总	6
6.1.	技术难点及解决方案	6
6.2.	实验感想和经验总结	7

1. 实验目的

通过实验，掌握词法分析的算法，如 NFA 到 DFA、状态的最小化和程序编写。

2. 实验内容

通过实验完成给定的 C 语言子集的单词符号划分和二元组输出。完成相应的报错功能。

3. 实验要求

- (1) 给出各单词符号的类别编码；
- (2) 词法分析程序应能发现输入串中的错误；
- (3) 词法分析作为单独一遍编写，词法分析结果为二元式序列组成的中间文件；
- (4) 设计两个测试用例（尽可能完备），并给出测试结果。

4. 程序实现

4.1. 相关环境介绍

操作系统：window 10 21H2

开发环境：Clion-2022.2.1-Windows

编译器：mwing-10.0

4.2. 主要数据结构

主要是识别出的单词，建立了一个 struct

```
01: struct Keyword{
02:     string notation;
03:     int class_num;
04:     int line;
05:     Keyword(string str, int num, int line_){
06:         notation = str;
07:         class_num = num;
08:         line = line_;
09:     }
```

```

10:   Keyword(char* str, int num, int line_){
11:       notation = string(str);
12:       class_num = num;
13:       line = line_;
14:   }
15:   Keyword(char str, int num, int line_){
16:       notation = str;
17:       class_num = num;
18:       line = line_;
19:   }
20:};

```

其中 notation 为单词的值，class_num 为单词所属的类别，line 是单词在源程序中的行号。

4.3. 程序结构描述

4.3.1. 设计方法

参照课程所讲述的过程，首先由每一类的单词构建 NFA，然后是确定化，得到 DFA，最后可以适当地进行状态的最小化，这样我们就可以得到，所有情况的状态转化图。由此可以进行代码的编写。

我们把所有的保留字，存在 reserved_word.txt 文件中，在主程序中进行文件读写，创建保留字的相关类型编号矩阵，这样做，程序不失一般性。Txt 文件中只需要空格分割保留字即可，简单易懂。之后我们可以配合其他，给出相应的函数文件，方便单词的划分，给与了扩展空间。

实验的输入是一个文本文件，按行读入，这样做有利于保留行号信息，方便报错。遇到空格和制表符、换行符就跳过。

词法分析函数，主要是通过状态转化图，使用 if-else，Switch-case 进行控制转移。

4.3.2. 函数定义

int init()

实验初始化函数，主要是读取 reserved_word.txt 文件创建保留字表。

int lexical_analysis(string file_path)

词法分析函数，也是本实验的核心函数，传入参数，file_path 为待分析的源程序路径。

函数中主要使用全局变量 vector<Keyword> output，每次完成单词的匹配，则 output.emplace_back(Keyword(word, class_num, line)，即写入单词的值、类别、行号到输出 vector。

Int main()

主函数主要完成，基本的参数设定，包括输入输出文件路径、保留字路径和文件的读入写

出等等。

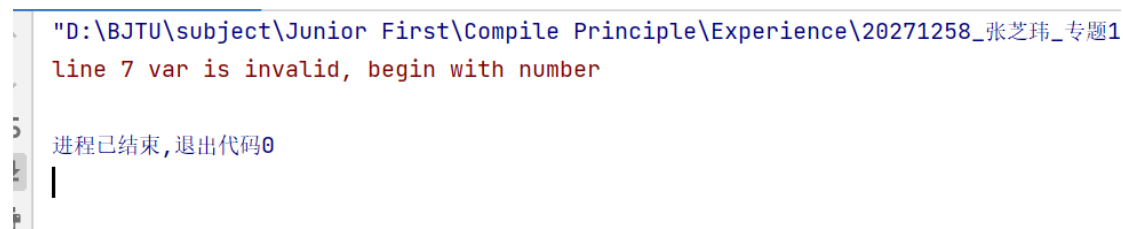
5. 程序测试

Test1.txt:

```
21:#include <iostream>
22:
23:int main(){
24:    int a = 1;
25:    int b = 3;
26:    int c = a+b;
27:    int 2434acb343;
28:    printf("%d", c);
29:}
```

notation	class_num	line
#	24	1
include	19	1
<	30	1
iostream	20	1
>	28	1
int	11	3
main	24	3
(45	3
)	46	3
{	47	3
int	11	4
a	0	4
=	26	4
1	1	4
;	44	4
int	11	5

图表 5-1 test1 中 out.txt 结果



```

"D:\BJTU\subject\Junior First\Compile Principle\Experience\20271258_张芝玮_专题1
line 7 var is invalid, begin with number

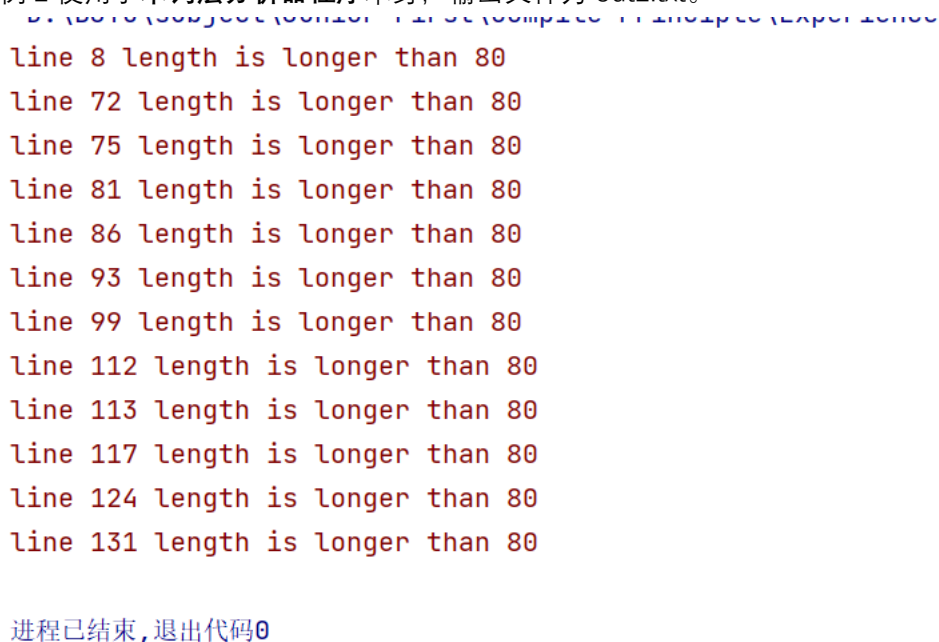
进程已结束,退出代码0

```

图表 5-2 test1 程序报错提示

这里可以看出，程序圆满的完成了要求，并且，给出了相应的报错，为变量定义错误。

测试样例 2 使用了**本词法分析器程序**本身，输出文件为 out2.txt。



```

D:\BJTU\subject\Junior First\Compile Principle\Experience
line 8 length is longer than 80
line 72 length is longer than 80
line 75 length is longer than 80
line 81 length is longer than 80
line 86 length is longer than 80
line 93 length is longer than 80
line 99 length is longer than 80
line 112 length is longer than 80
line 113 length is longer than 80
line 117 length is longer than 80
line 124 length is longer than 80
line 131 length is longer than 80

进程已结束,退出代码0

```

图表 5-3 test2 报错信息

样例 2 本身代码语法是完全正确的，但是在词法分析程序中，我们加入了每行最多字符限制 80，这也是比较老的 C++ 标准，但是词法分析程序是按照 C++14 编写的，行字符上限是 120，所以会有这样的报错信息。

这样看来实验结果完成正确。

6. 实验汇总

6.1. 技术难点及解决方案

实验本身编码没有太多的难度，主要是前期的 DFA 和状态转化图的绘制比较繁琐。状态转换图也就是一个程序流程图，可以帮助我们不漏掉部分单词。

6.2. 实验感想和经验总结

实验主要是按照课堂上所讲述的算法和流程进行编写，锻炼了编码能力，以及了解编译的全过程，对本人的学习十分有帮助。