



编译技术课程实验报告

实验名称:	实验一 词法分析程序编制
实验日期:	2024-2-27
实验地点:	西部片区 4 号楼 208

学号:	33920212204567
姓名:	任宇
专业年级:	软工 2021 级
学年学期:	2023-2024 学年第二学期

目录

一、 实验目的	3
二、 实验内容	3
三、 实验环境	3
四、 实验过程	3
1) 根据以下的正规式，编制正规文法，画出状态图：	3
2) 程序的数据结构：	4
3) 程序的算法：	4
4) 测试程序：	8
五、 思考题	10
六、 实验心得	11

一、 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。

二、 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。

三、 实验环境

- PC 微机 Windows10 操作系统
- 开发环境为 VS2022

四、 实验过程

1) 根据以下的正规式，编制正规文法，画出状态图：

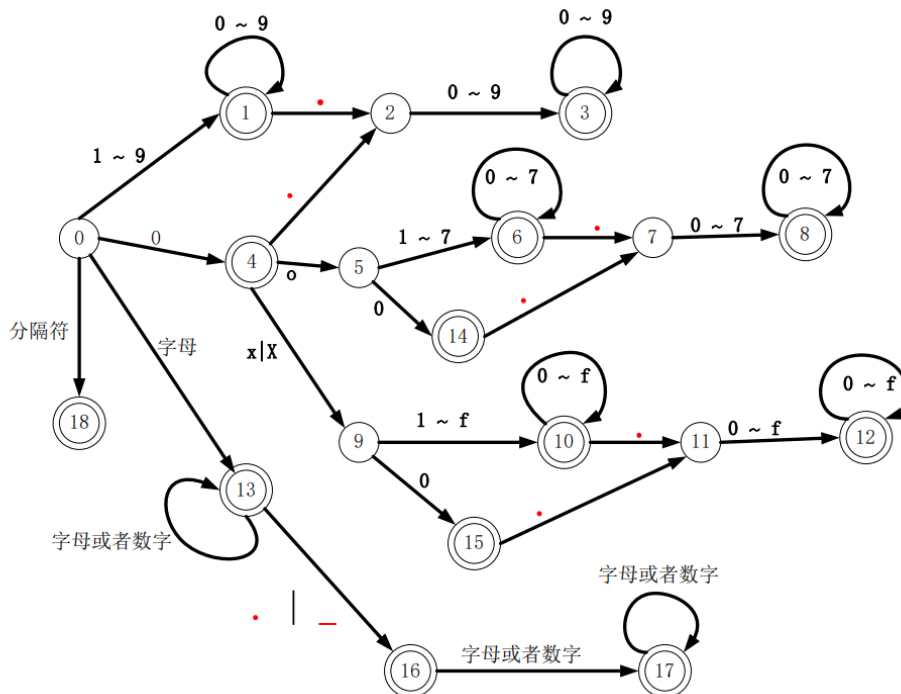
标识符	$\langle \text{字母} \rangle (\langle \text{字母} \rangle \langle \text{数字字符} \rangle)^* (\epsilon _) (\langle \text{字母} \rangle \langle \text{数字字符} \rangle)^*$
十进制数	$(1 2 3 4 5 6 7 8 9) (0 1 2 3 4 5 6 7 8 9)^* 0 (\epsilon . (0 1 2 3 4 5 6 7 8 9) (0 1 2 3 4 5 6 7 8 9)^*)$
八进制数	$0o (0 1 2 3 4 5 6 7) (0 1 2 3 4 5 6 7)^* (\epsilon . (0 1 2 3 4 5 6 7) (0 1 2 3 4 5 6 7)^*)$
十六进制数	$0x (0 1 2 3 4 5 6 7 8 9 a b c d e f) (0 1 2 3 4 5 6 7 8 9 a b c d e f)^* (\epsilon . (0 1 2 3 4 5 6 7 8 9 a b c d e f) (0 1 2 3 4 5 6 7 8 9 a b c d e f)^*)$
运算符和分隔符	$+ \quad - \quad * \quad / \quad > \quad < \quad = \quad (\quad) \quad ;$
关键字	$\text{if} \quad \text{then} \quad \text{else} \quad \text{while} \quad \text{do}$

正规文法为：

标识符	$S \rightarrow L \mid LT$ $T \rightarrow L \mid D \mid _ \mid TL \mid TD$ $L \rightarrow a \mid b \mid \dots \mid z \mid A \mid B \mid \dots \mid Z$ $D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$
十进制数	$S \rightarrow D \mid D.F$ $D \rightarrow 0 \mid N$ $F \rightarrow D \mid DF$ $N \rightarrow 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \mid 1N \mid 2N \mid 3N \mid 4N \mid 5N \mid 6N \mid 7N \mid 8N \mid 9N$
八进制数	$S \rightarrow 0o N \mid 0o N.F$ $N \rightarrow D \mid DN$ $F \rightarrow .D \mid FD$ $D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7$
十六进制数	$S \rightarrow 0x HA$ $A \rightarrow HA \mid \epsilon \mid .B$ $B \rightarrow HB \mid \epsilon$ $H \rightarrow 0 \mid 1 \mid 2 \mid \dots \mid 9 \mid a \mid b \mid \dots \mid f \mid A \mid B \mid \dots \mid F$

运算符和分隔符	$S \rightarrow + \mid - \mid * \mid / \mid > \mid < \mid = \mid (\mid) \mid ;$
关键字	$S \rightarrow \text{if} \mid \text{then} \mid \text{else} \mid \text{while} \mid \text{do}$

状态图为：



接着就可以根据状态图，设计词法分析算法。采用 C 语言，设计函数 `scan()` 实现词法分析算法并编制测试程序（主函数 `main`）。

2) 程序的数据结构：

- 枚举体 `TokenType`：定义了一系列可能的词汇单元类型，包括标识符、整数（十进制、八进制、十六进制）、操作符、关键字（`if`、`then`、`else`、`while`、`do`）和未识别的类型。

```

7  typedef enum {
8      IDENTIFIER, DECIMAL_INTEGER, OCTAL_INTEGER, HEXADECIMAL_INTEGER, FLOAT,
9      OPERATOR, KEYWORD_IF, KEYWORD_THEN, KEYWORD_ELSE, KEYWORD_WHILE, KEYWORD_DO,
10     UNRECOGNIZED
11 } TokenType;

```

- 结构体 `Token`：用于存储单个词汇单元的类型（使用 `TokenType` 枚举）和对应的文本（字符数组）。

```

41  typedef struct {
42      TokenType type;
43      char text[100];
44  } Token;
45

```

3) 程序的算法：

- 本程序的核心算法流程集中在词法分析部分，即 `scan` 函数，其主要目的是从给定的输入字符串中识别并分类词汇单元，其流程为：

a) 跳过空白字符:

当输入的当前字符是空白 (例如空格、制表符、换行符等) 时, 指针前移直到指向非空白字符。

```
46 // 扫描输入并生成一个词汇单元
47 Token scan(const char** input) {
48     Token token = { UNRECOGNIZED, {0} };
49
50     while (isspace(**input)) (*input)++; // 跳过后导空白字符
51 }
```

b) 开始标记词汇单元:

记录当前字符的位置, 作为词汇单元开始的位置。

```
51
52     const char* start = *input; // 记录开始位置
53 }
```

c) 分类处理:

■ 标识符或关键字: 如果当前字符是字母或下划线, 继续读取后续字符, 直到遇到非字母、非数字、非下划线字符为止。接着使用从开始位置到当前位置的字符串去判断是否是关键字, 如果不是关键字则分类为标识符。

```
54     if (isalpha(**input) || **input == '_') { // 标识符或关键字
55         while (isalnum(**input) || **input == '_' || **input == '.') (*input)++;
56         strncpy(token.text, start, *input - start);
57         token.text[*input - start] = '\0';
58         token.type = checkKeyword(token.text);
59     }
```

■ 数字 (整数或浮点数): 根据第一个字符是否为 0 来确定是否可能是八进制或十六进制。如果是 0x 或 0X 开头, 处理为十六进制数字; 如果是 0o 或 0O 开头, 处理为八进制数字。对于十六进制和八进制, 继续读取直到字符不再符合该进制的数字规则。对于普通的十进制数字或浮点数 (以 0 开头或其他数字开头), 读取直到遇到非数字且非点号的字符, 根据是否包含点号判断是整数还是浮点数。

```
60     else if (isdigit(**input) || (**input == '.' && isdigit(*(input + 1)))) { // 数字或浮点数
61         bool isFloat = false;
62         if (**input == '0') {
63             start = *input;
64             (*input)++;
65             if (**input == 'x' || **input == 'X') { // 十六进制
66                 (*input)++;
67                 while (isxdigit(**input) || **input == '.') {
68                     if (**input == '.') isFloat = true;
69                     (*input)++;
70                 }
71                 token.type = isFloat ? FLOAT : HEXADEcimal_INTEGER;
72             }
73             else if (**input == 'o' || **input == 'O') { // 八进制
74                 (*input)++;
75                 while (**input >= '0' && **input <= '7') (*input)++;
76                 token.type = OCTAL_INTEGER;
77             }
78             else { // 十进制或浮点数以0开始
79                 while (isdigit(**input) || **input == '.') {
80                     if (**input == '.') isFloat = true;
81                     (*input)++;
82                 }
83                 token.type = isFloat ? FLOAT : DECIMAL_INTEGER;
84             }
85         }
86         else { // 十进制或浮点数不以0开始
```

```

86     else { // 十进制或浮点数不以0开始
87         while (isdigit(**input) || **input == '.') {
88             if (**input == '.') isFloat = true;
89             (*input)++;
90         }
91         token.type = isFloat ? FLOAT : DECIMAL_INTEGER;
92     }
93 }

```

- 操作符：如果当前字符属于操作符集合，则直接分类为操作符。

```

94     else if (strchr("+*-/>=<()", **input)) { // 运算符或界定符
95         token.type = OPERATOR;
96         (*input)++;
97     }

```

- 未识别的字符

```

98     else { // 未识别的字符
99         token.type = UNRECOGNIZED;
100         (*input)++;
101     }

```

- 设置词汇单元文本：根据已识别的词汇单元类型，将从开始位置到当前位置的文本复制到词汇单元结构的文本字段。

```

102 // 为非标识符和非关键字设置文本
103 if (token.type != IDENTIFIER && token.type != KEYWORD_IF && token.type != KEYWORD_THEN && token.type != KEYWORD_ELSE && token.type != KEYWORD_WHILE && token.t
104     size_t length = *input - start;
105     strncpy(token.text, start, length);
106     token.text[length] = '\0';
107 }
108

```

- 返回词汇单元：

```

109     return token;
110 }
111

```

- 除此之外还有一些必要的辅助函数：

- checkKeyword：接受一个字符串，检查是否匹配某个关键字，如果是则返回相应的 TokenType，否则返回 IDENTIFIER。

```

13 // 检查是否为关键字
14 TokenType checkKeyword(const char* str) {
15     if (strcmp(str, "if") == 0) return KEYWORD_IF;
16     if (strcmp(str, "then") == 0) return KEYWORD_THEN;
17     if (strcmp(str, "else") == 0) return KEYWORD_ELSE;
18     if (strcmp(str, "while") == 0) return KEYWORD_WHILE;
19     if (strcmp(str, "do") == 0) return KEYWORD_DO;
20     return IDENTIFIER;
21 }

```

- TokenTypeToString：将 TokenType 转换为对应的的字符串形式。

```

23 // 将词汇单元类型转换为字符串
24 const char* TokenTypeToString(TokenType type) {
25     switch (type) {
26         case IDENTIFIER: return "IDN";
27         case DECIMAL_INTEGER: return "INT10";
28         case OCTAL_INTEGER: return "INT8";
29         case HEXADECIMAL_INTEGER: return "INT16";
30         case FLOAT: return "FLOAT";
31         case OPERATOR: return "OPR";
32         case KEYWORD_IF:
33         case KEYWORD_THEN:
34         case KEYWORD_ELSE:
35         case KEYWORD_WHILE:
36         case KEYWORD_DO: return "KEYWORD";
37         default: return "UNRECOGNIZED";
38     }
39 }

```

- hexCharToInt: 将单个十六进制/八进制字符转换为其对应的整数值。

```

112 // 十六进制/八进制字符转十进制整数
113 int hexCharToInt(char c) {
114     if (c >= '0' && c <= '9') return c - '0';
115     if (c >= 'a' && c <= 'f') return 10 + (c - 'a');
116     if (c >= 'A' && c <= 'F') return 10 + (c - 'A');
117     return -1; // 非法字符
118 }
119

```

- hexStringToDouble: 将十六进制或八进制字符串(可能包含小数点)转换为对应的十进制浮点数。

```

120 // 十六进制/八进制字符串转双精度浮点数
121 double hexStringToDouble(const char* hexStr, int base) {
122     if (hexStr[0] != '0' || (hexStr[1] != 'x' && hexStr[1] != 'X' && hexStr[1] != 'o')) {
123         printf("输入无效\n");
124         return 0;
125     }
126     double result = 0.0;
127     int isDecimal = 0;
128     double decimalFactor = 1.0;
129     for (const char* p = hexStr + 2; *p; ++p) {
130         if (*p == '.') {
131             isDecimal = 1;
132             continue;
133         }
134         int val = hexCharToInt(*p);
135         if (val < 0) {
136             printf("十六进制字符串中含有无效字符: %c\n", *p);
137             return 0;
138         }
139         if (isDecimal) {
140             decimalFactor /= (float)base;
141             result += val * decimalFactor;
142         }
143         else {
144             result = result * base + val;
145         }
146     }
147     return result;
148 }
149
150 int main() {
151

```

- 测评程序是否可行的主函数：

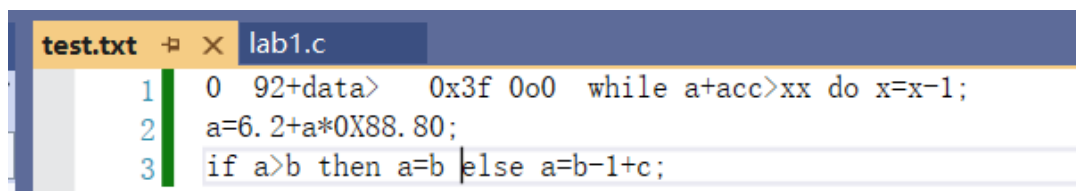
主函数（main）中，程序打开指定的文本文件，逐行读取内容。对每行使用 scan 函数分析，识别其中的词汇单元。对于每个词汇单元，根据其类型进行处理，如果是数字类型还可能进行数值转换，然后打印出来。文件读取完毕后关闭文件。

```

151 int main() {
152     FILE* file;
153     char buffer[1024]; // 用于存储文件中的每一行
154     const char* filename = "test.txt";
155
156     file = fopen(filename, "r");
157     if (file == NULL) {
158         perror("无法打开文件");
159         return 1;
160     }
161
162     while (fgets(buffer, sizeof(buffer), file)) {
163         const char* input = buffer; // 指向当前行的起始位置
164         while (*input != '\0') {
165             Token token = scan(&input); // 调用词法分析器处理每个词汇单元
166             if (*input == '\n') {
167                 break; // 遇到换行符时，跳出内部循环，继续读取下一行
168             }
169             char* type = TokenTypeToString(token.type);
170             double result;
171             if (!strcmp(type, "FLOAT") || !strcmp(type, "INT8") || !strcmp(type, "INT16")) {
172                 if (token.text[1] == 'x' || token.text[1] == 'X' || token.text[1] == 'o') {
173                     int base = (token.text[1] == 'o') ? 8 : 16;
174                     result = hexStringToDouble(token.text, base);
175                     if (token.type == FLOAT) {
176                         if (base == 8)
177                             printf("INT8\t %f\n", result);
178                         else
179                             printf("INT16\t %f\n", result);
180                     }
181                     else {
182                         printf("%s\t %d\n", type, (int)result);
183                     }
184                 }
185                 else {
186                     printf("INT10\t %s\n", token.text);
187                 }
188             }
189             else {
190                 printf("%s\t %s\n", type, token.text);
191             }
192         }
193     }
194     fclose(file); // 关闭文件
195     return 0;
196 }

```

4) 测试程序：
输入为：



```

test.txt  X lab1.c
1  0 92+data> 0x3f 0o0 while a+acc>xx do x=x-1;
2  a=6.2+a*0X88.80;
3  if a>b then a=b else a=b-1+c;

```

输出为：

INT10	0	INT10	0
INT10	92	INT10	92
+	-	OPR	+
IDN	data	IDN	data
>	-	OPR	>
INT16	63	INT16	63
INT8	0	INT8	0
WHILE	-	KEYWORD	while
IDN	a	IDN	a
+	-	OPR	+
IDN	acc	IDN	acc
>	-	OPR	>
IDN	xx	IDN	xx
DO	-	KEYWORD	do
IDN	x	IDN	x
=	-	OPR	=
		IDN	y

二元组为类型和值，在我的程序中，将操作符和关键字分别合成一个类型

IDN	x	IDN	x
-	-	OPR	-
INT10	1	INT10	1
;	-	IDN	a
IDN	a	OPR	=
=	-	INT10	6.2
INT10	6.2	OPR	+
+	-	IDN	a
IDN	a	OPR	*
*	-	INT16	136.500000
INT16	136.5	KEYWORD	if
		IDN	a
		OPR	>
		IDN	b
		KEYWORD	then
;	-	IDN	a
IF	-	OPR	=
IDN	a	IDN	b
>	-	KEYWORD	else
		IDN	a

浮点数转为对应十进制表示

```

IDN      b
THEN     -
IDN      a
=        -
IDN      b
ELSE     -
IDN      a
=        -
IDN      b
-        -
INT10    1
+        -
IDN      c
;        -

```

```

OPR      /
IDN      b
KEYWORD  then
IDN      a
OPR      =
IDN      b
KEYWORD  else
IDN      a
OPR      =
IDN      b
OPR      -
INT10    1
OPR      +
IDN      c
OPR      ;

E:\visual_studio\work\编译lab
要在调试停止时自动关闭控制台，

```

五、 思考题

1. 词法分析能否采用空格来区分单词？

答：能否采用空格区分单词需要视情况而定，比如说在字符串或者注释中的空格不应该用作分隔符。例如，在字符串“Hello World”中，空格是字符串的一部分，不应该将其拆分为两个单独的单词，在这种情况下，词法分析器需要正确识别并处理字符串和注释。

在比如说某些不使用空格的语法结构，在某些编程语言中，如 python 中，空格和缩进被用来表达程序结构。同时，某些操作符或界定符可能紧挨着变量或者关键字，不由空格分隔，如 `a++` 或者 `!=`，在这些情况下，只使用空格来进行分割是不足的。

总的来说，虽然空格是一种有效的分隔符，可以用来区分大部分单词，但词法分析器的实现还需要考虑不同的上下文和特殊情况，确保所有的词法单元都能被正确解析。

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：在 `scan` 函数中，大量使用字符分类函数如 `isalpha`, `isdigit` 等，以及字符串操作函数如 `strcmp` 和 `strncpy`。这些函数的频繁调用可能会导致性能下降。同时关键字的识别也很耗费时间，因为词法分析器在识别出一个标识符后，就要调用函数比较该标识符是否是关键字，若关键字表中关键字较多，则查询比较耗时较长。

如果想提高效率，可以减少使用 `strncpy` 和 `strcmp`，尤其是在循环中。可以考虑使用指针操作来直接比较和复制字符串，或使用现代 C++ 库如

`std::string_view` 来避免不必要的字符串复制。同时，可以优化检查关键字的流程，比如说有些标识符不可能是关键字，就可以不再比较。假如长度大于关键字的最大长度，就可以不进行关键字的检查。

六、 实验心得

通过这次实验，我对词法分析程序的编制有了更深入的理解。实验不仅让我实际操作了从设计到实现的整个过程，还加深了我对词法分析在编译过程中的重要性的认识。在编写程序的过程中，我遇到了许多挑战，特别是在将理论转化为实际代码时。例如，处理不同类型的数值表示（十进制、八进制、十六进制）和区分关键字与普通标识符时，需要非常注意状态的转换和条件的判断。

总之，这次实验是非常宝贵的学习经验。它不仅提升了我的编程技能，也加深了我对编译原理中词法分析阶段的理解，这为完成期末大作业奠定坚实的基础。