一、 实验目的与方法

1．实验目的

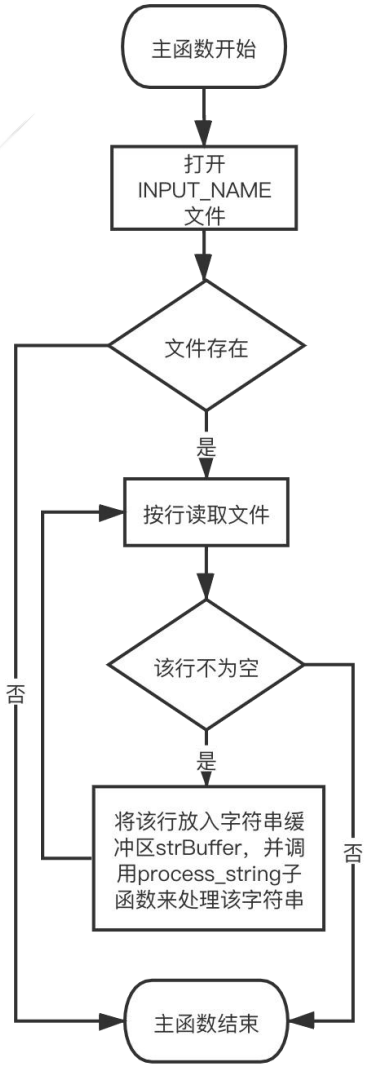
通过设计调试词法分析程序，实现从源程序中分出各种单词的方法，加深对课堂教学的理解，提高词法分析方法的实践能力。

2．实验方法

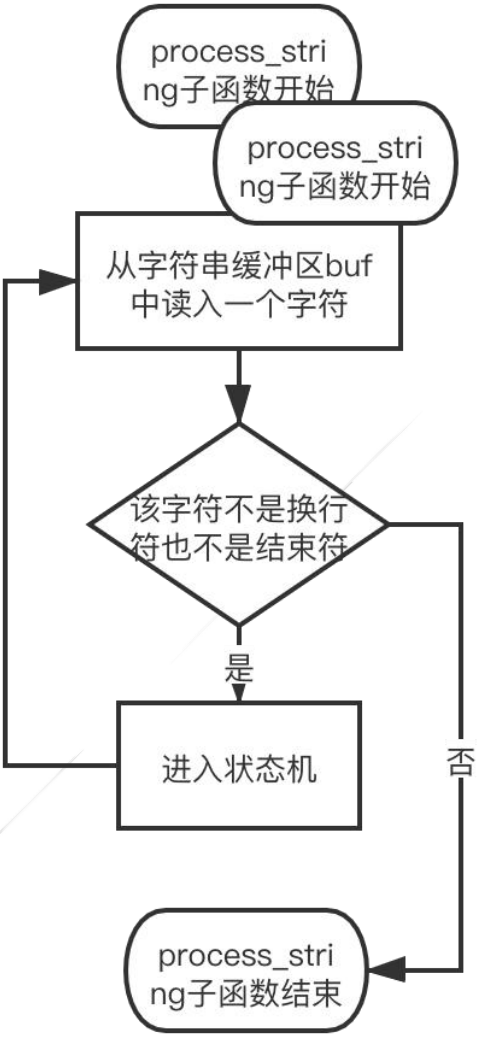
所使用的语言为 C 语言；操作系统环境为 macOS Mojave；软件环境为 CLion。

二、 实验总体流程与函数功能描述

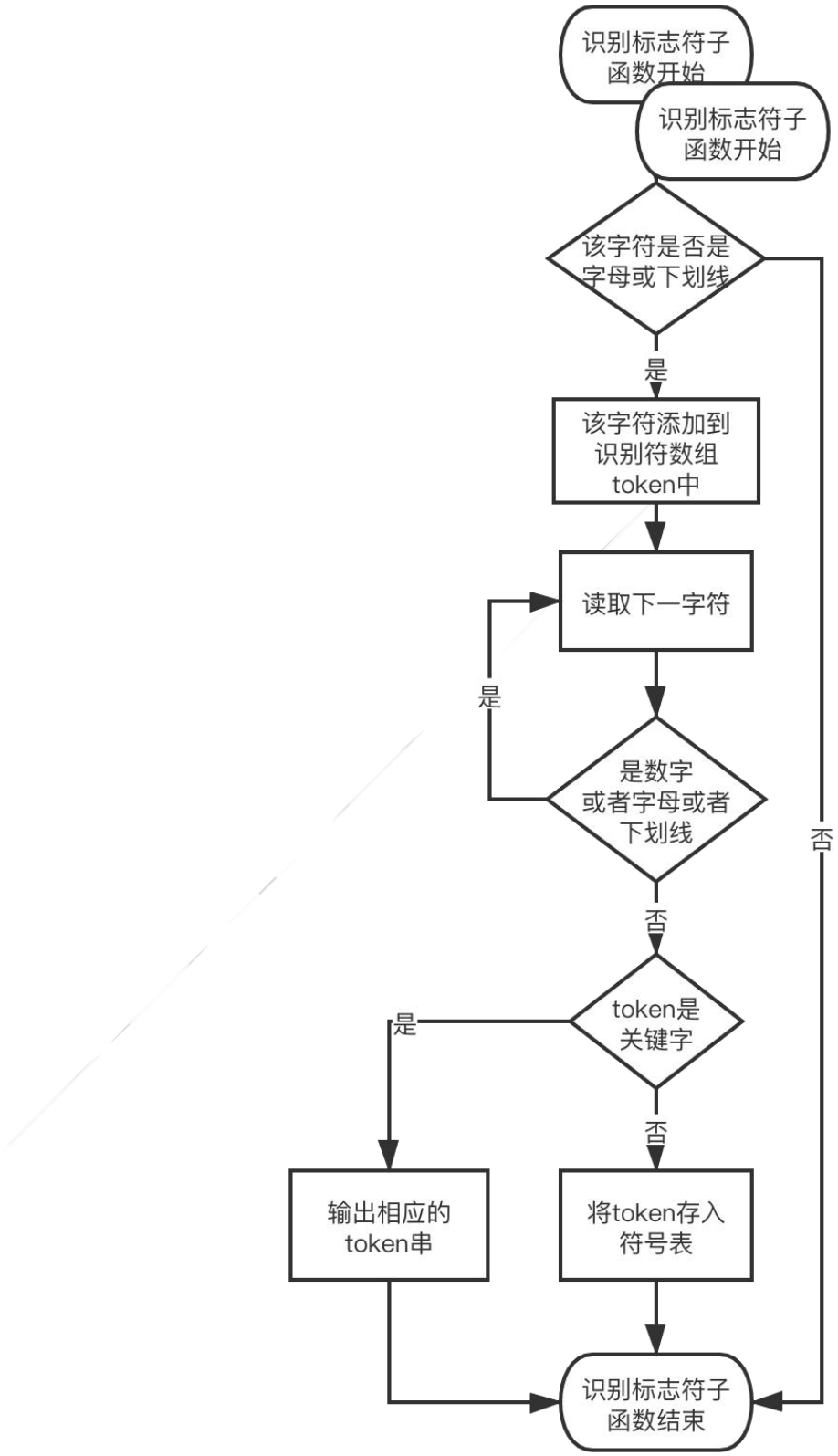
1．主函数



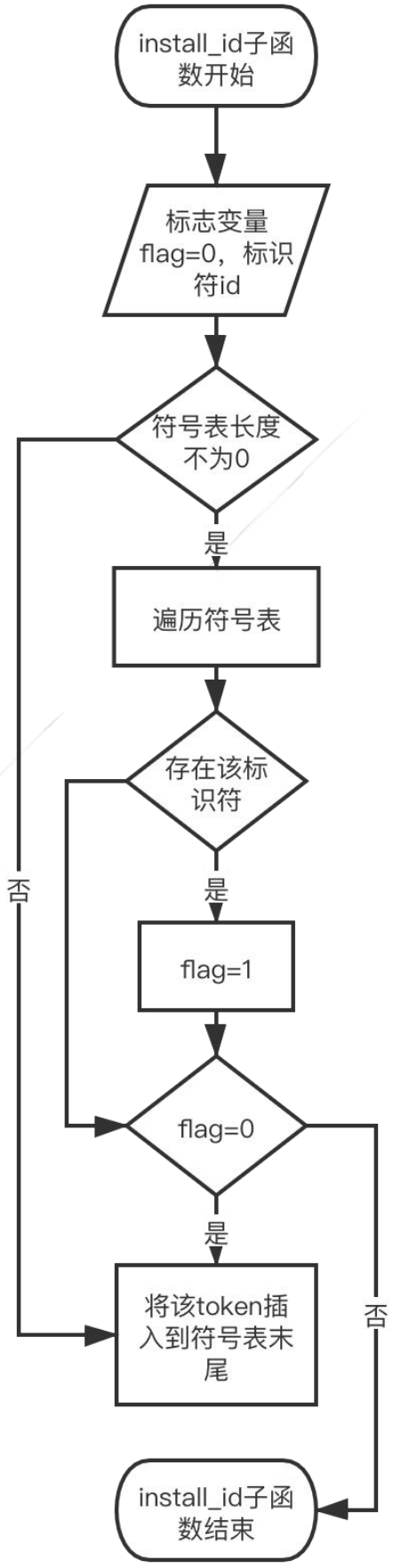
2．void process_string(char* buf) 对一行字符串的内容进行词法分析



3．识别标识符的算法



4．void install_id(char id[]) 将 token 存入符号表


install_id子函数开始
标志变量flag=0，标识符id
符号表长度不为0
是
遍历符号表
存在该标识符
否
是
flag=1
flag=0
是
否
将该token插入到符号表末尾
install_id子函数结束

5．char getChar(char* str) 从字符串中获取指针当前所指位置的字符
6．int isDigit09(char c) 判断是否是数字 0-9
7．int isDigit19(char c) 判断是否是数字 1-9
8．int isDigit07(char c) 判断是否是数字 0-7
9．int isDigit09Letteraf(char c) 判断是否是 0-9 或 a-f
10．int isLetter(char c) 判断是否是字母
11．int isPunctuation(char c) 判断是否是标点符号
12．int isKey(char* str) 判断是否是关键字

三、 实验内容

1．文法描述

① 标识符

<id> → letter<rid> | _<rid>
<rid> → ε | digit<rid> | letter<rid> | _<rid>

② 十进制无符号整数

<dec_num> → 1~9<r_dec_num>
<r_dec_num> → ε | 0~9<r_dec_num>

③ 八进制无符号整数

<oct_num> → 0<r_oct_num>
<r_oct_num> → 0~7<rr_oct_num>
<rr_oct_num> → ε | 0~7<rr_oct_num>

④ 十六进制无符号整数

<hex_num> → 0<r_hex_num>
<r_hex_num> → x<rr_hex_num>
<rr_hex_num> → 0~9<rrr_hex_num> | a~f<rrr_hex_num>
<rrr_oct_num> → ε | 0~9<rrr_oct_num> | a~f<rrr_hex_num>

⑤ 关系运算符

<relop> → < | < <equal> | > | ><equal> | =<equal> | !<equal>
<equal> → =

⑥ 赋值符号

<assign> → =

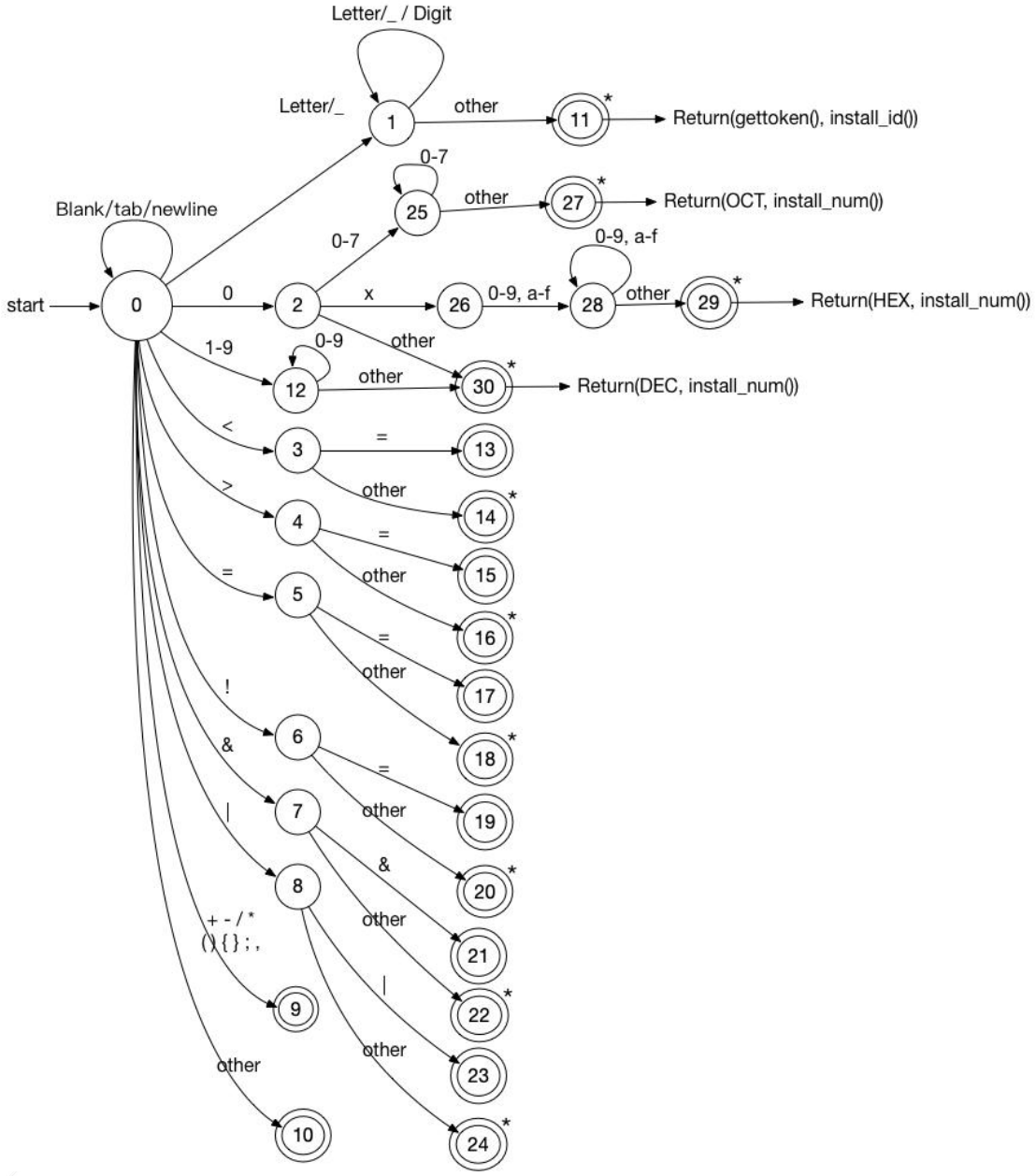
⑦ 逻辑运算符

<op> → + | - | * | / | &<rop> | |<rop>
<rop> → ε | &<rop> | |<rop>

⑧ 分隔符

<delimiter> → : | , | ; | (|) | { | }

2．有穷自动机



3．词类编码表

单词	种别码	单词	种别码	单词	种别码	单词	种别码
for	1	int	11	{	21	LS	31
while	2	float	12	}	22	LE	32
do	3	return	13	;	23	EQ	33
switch	4	main	14	,	24	NE	34
case	5	+	15	ID	25	ASSIGN	35
break	6	-	16	DECNUM	26	NOT	36
continue	7	/	17	HEXNUM	27	AND	37
if	8	*	18	OCTNUM	28	DAND	38
else	9	(	19	GT	29	OR	39
char	10	)	20	GE	30	DOR	40

4．符号表的逻辑结构及存贮结构

```
// 符号表的数据结构
struct ID_NAME{
    char name[KEY_LEN];
    char type[KEY_LEN];
    char address[KEY_LEN];
};
struct ID_LIST{
    struct ID_NAME names[ID_MAX];
    int length;
};
```

① 符号表的存储结构

ID_NAME 是存储标识符的结构体，包括其名字、类型、入口指针。

ID_LIST 是符号表的结构体，包括标识符结构体数组（最大只能存储 1000 个）和符号表的长度。

② 符号表的逻辑结构

当开始读入文件时，将符号表的长度设置为 0。

读入文件后，若单词识别为标识符，则调用 install_id 函数，函数逻辑如二的流程图 4。

5．实验思考题

① 词法分析程序能否能够使用空格来区分单词?

不能。因为程序中有;，{ }等符号来构成分界符。例如关键字与分隔符之间可能没有空格来间隔开，无法区分。

② 词法分析程序作为一个独立子程序具有哪些优点?

避免中间文件的生成，可以提高效率。将程序模块化，一个模块完成一个任务，使整个工程较为清晰。

四、 实验结果与分析

输入：

```
S = 123;
mul_x = s * 2;
while(mul_x > 100)
{
    mul_x = mul_x - 1;
}
test = 012 / 0x1f;
```

输出：

```
*****输出单词*****
(25, S)
(35, =)
(26, 123)
(23, ;)
(25, mul_x)
(35, =)
(25, s)
(18, *)
(26, 2)
(23, ;)
(2, while)
(19, ()
(25, mul_x)
(29, >)
(26, 100)
(20, ))
(21, {)
(25, mul_x)
(35, =)
(25, mul_x)
(16, -)
(26, 1)
(23, ;)
(22, })
(25, test)
(35, =)
(28, 012)
(17, /)
(27, 0x1f)
(23, ;)
*****符号表*****
(S, -1, ?)
(mul_x, -1, ?)
(s, -1, ?)
(test, -1, ?)
```