编译原理课程实验报告

实验 1：词法分析

姓名	姚敏敏		院系	软件学院		学号	170720231
任课教师	韩希先			指导教师	韩希先		
实验地点	研究院中 517			实验时间	2019 年 10 月 19 日		
实验课表现	出勤、表现得分			实验报告得分		实验总分	
	操作结果得分						

一、实验目的

要求：需分析本次实验的基本目的，并综述你是如何实现这些目的的？

1. 巩固对词法分析的基本功能和原理的认识。

   词法分析器的基本功能就是将构成源程序的字符串转化成等价的单词序列。词法分析器的原理是在已知文法的情况下利用递归向下分析。

2. 能够应用自动机的知识进行词法分析。

   我们通过有穷自动状态机来描述语言的词法模型，通过该词法模型从而实现对程序的词法分析。

3. 理解并处理词法分析中的异常和错误。

   对于词法分析过程中出现的例如：非法字符、单词拼写错误、注解或字符常数不封闭、变量重复说明等错误，我们通常采用非法字符检查、关键字拼写错误检查、不封闭错误检查、重复说明检查、以及紧急恢复的方式来进行错误的发现与处理。

二、实验内容

1. 给出语言的词法规则描述

   对于要求的语法规则，建立了种别码表

单词符号	种别码	单词符号	种别码
!	-1	*	24
#	0	/	25
begin	1	(	26
if	2	)	27
then	3	[	28
while	4	]	29
do	5	{	30
end	6	}	31
int	7	,	32
main	8	:	33
l（l|d）*	10	;	34
return	12	>	35
cout	13	<	36

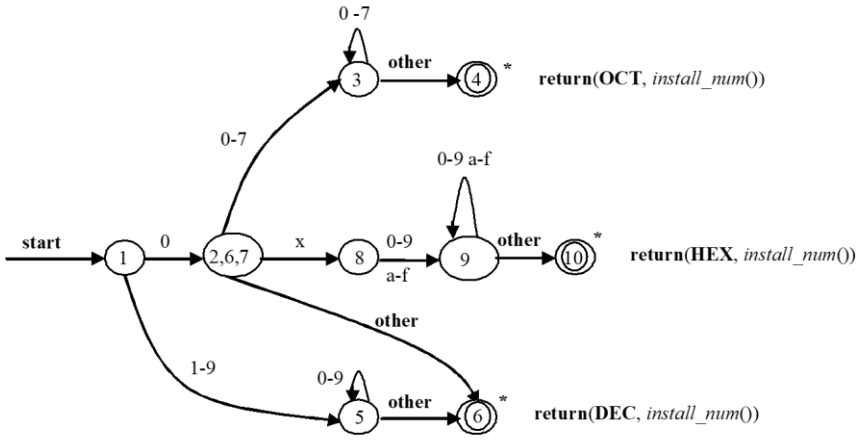
:=	18	>=	37
dd*	20	<=	38
==	21	!=	40
+	22	“	41
-	23	\0	1000

表格 1 种别码表

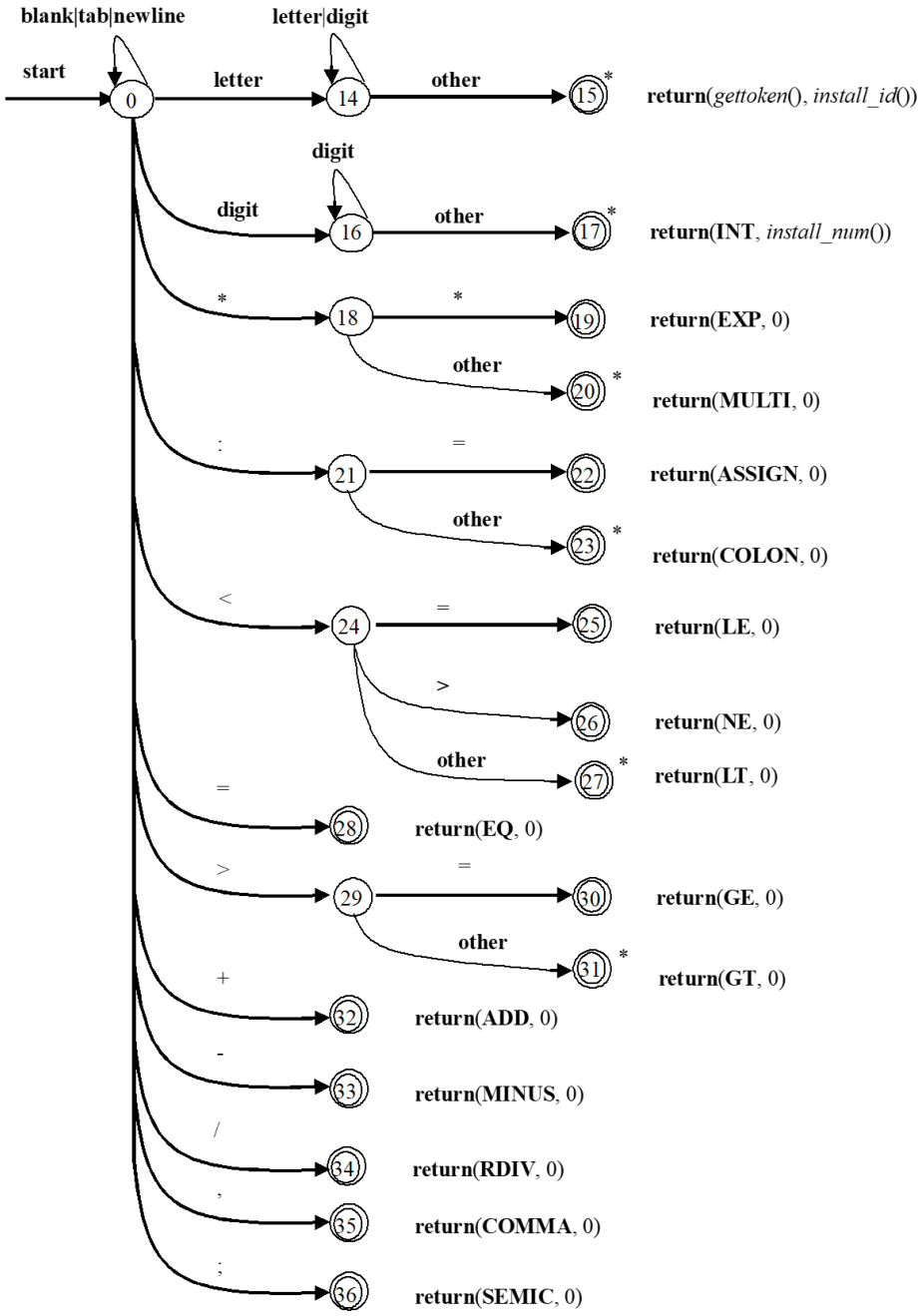
而且，我们可以得出该语言的正则文法如下

- <id> → letter <id_left>
- <id_left> →ε| letter <id_left> | digit <id_left>
- <int> → digit <int_left>
- <int_left> →ε| digit <int_left>
- <assignment> → :=
- <relop> → < | <= | = | < > | > | >=
- <op> → + | - | * | / | **
- <delimiter> → : | , | ;

2. 针对这种单词的状态转换图和程序框图
   - 状态转换图

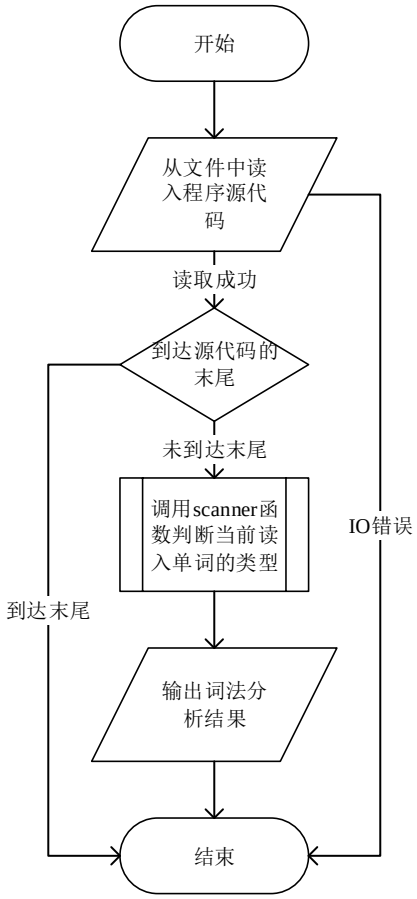


图表 1 识别不同进制无符号整数的状态转换图

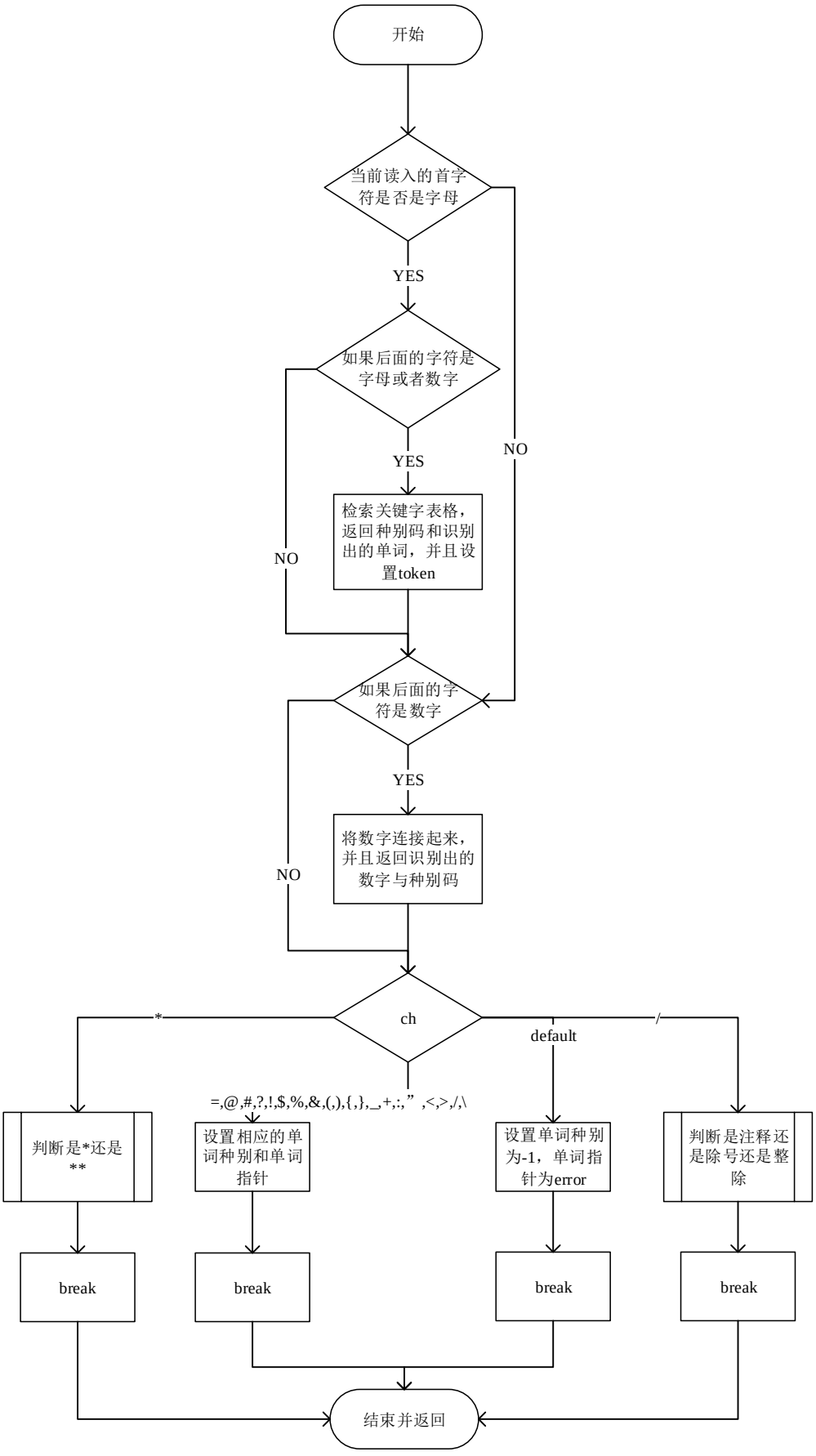


图表 2 识别基本单词的状态转图

- 程序框图



图表 3 main 函数的流程图



图表 4 Scanner 函数的流程图

3. 核心数据结构的设计

- 通过一个数组 keywordTable 来存储关键字

```
char *keywordTable[] = {"begin","if","then","while","do","end","int","main",
                        "else","float","double","return","cout",KEY_WORD_END
                        };
```

- 符号表的设计如下所示

  采用数组的形式来储存符号表，将标识符长度放在符号表中。

```
//符号表结点
struct snode
{
 string text;    //符号名称
 string type;    //符号类型
 union {int ival;double rval;}value;//值
 int offset;     //偏移量
 snode *nextn;   //指向下一个节点
 stable *header;  //指向下属符号表的表头
};
```

- 其他数据结构

```
typedef struct //词的结构，二元组形式（单词种别，单词自身的值）
{
    int typeNum; //单词种别
    char * word; //单词指针
} WORD;
char input[255];//储存输入
char token[255] = "";//储存token
int p_input; //指向输入的指针
int p_token; //指向token的指针
char ch;
```

4. 错误处理

   对于不合法的标识符，采取了紧急方式恢复。反复删除剩余输入最前面的字符，直到词法分析器能够发现一个正确的单词为止。

三、实验结果

要求：将实验获得的结果进行描述，基本内容包括：

1. 针对某测试程序输出其词法分析结果；

   测试用例：

```
int main(){
int a = 1, b = 2;
    b / a;//test
    b > a;
    c = a + b;
    cout << c;
    return 0;
}
```

   输出的结果：

```
Lexical analysis result:
[ 7     int ]
[ 8     main ]
[ 26    ( ]
[ 27    ) ]
[ 30    { ]
[ 7     int ]
[ 10    a ]
[ 18    = ]
[ 20    1 ]
[ 32    , ]
[ 10    b ]
[ 18    = ]
[ 20    2 ]
[ 34    ; ]
[ 10    b ]
[ 25    / ]
[ 10    a ]
[ 34    ; ]
[ 25    / ]
[ 25    / ]
[ 10    test ]
[ 10    b ]
[ 35    > ]
[ 10    a ]
[ 34    ; ]
[ 10    c ]
[ 18    = ]
[ 10    a ]
[ 22    + ]
[ 10    b ]
[ 34    ; ]
[ 13    cout ]
[ 42    << ]
[ 10    c ]
[ 34    ; ]
[ 12    return ]
[ 20    0 ]
[ 34    ; ]
[ 31    } ]
```

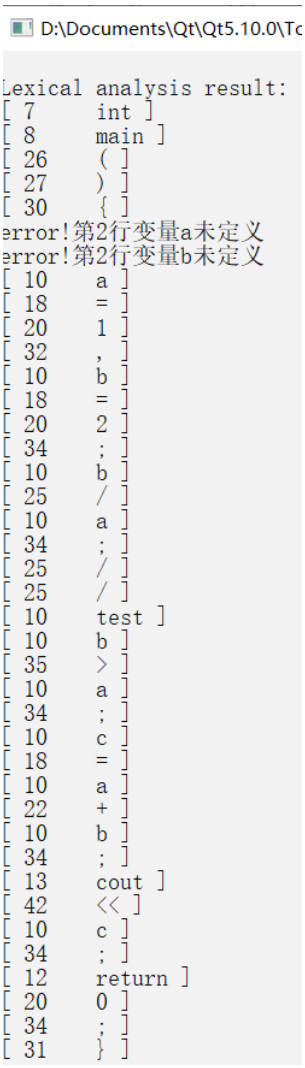
2. 输出针对此测试程序对应的词法错误报告；

   对于以下存在错误：“a，b 未定义”的代码，

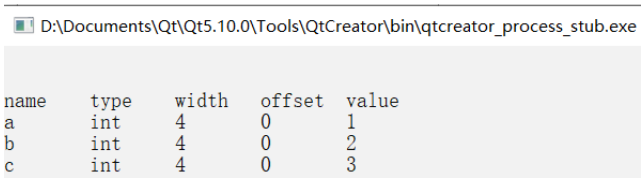
```
int main(){
    a = 1, b = 2;
    b / a;//test
    b > a;
    c = a + b;
    cout << c;
    return 0;
}
```

   程序提示以下错误：

   程序提示 a，b 未定义。



3. 输出针对此测试程序对应的符号表。



注：其中的测试样例自行产生。

四、实验中遇到的问题总结

要求：主要阐述两方面的问题

一、 实验过程中遇到的问题如何解决的？

- 在一开始编写程序的时候，并没有采取任何方式来存储符号表，这样使得各种名字管理十分混乱，后来，在看了书上的符号表一章后，采取了线性表的方式来存储符号表，从而方便了管理，也提升了程序的可拓展性。
- 在对注释进行识别的时候，容易将单行注释符号和整除搞混，同样是“//”，可以通过判断“//”之后的字符串的内容来确定“//”的实际含义。
- 在编写程序的时候，应该先确定正则文法、状态转换图、种别码表这些基

础的内容，这样在编写代码的时候可以顺水推舟。

二、 思考题的思考与分析

思考题 1：你编写的程序是如何体现自动机的，这样做有什么好处？

自动机的初始态为 0，每次根据自动机的当前状态，词法分析器分析当前得到的单词，再依据根据状态转换图，进入下一个状态。就这样不断读入，不断处理，不断进行状态转移，直到到达状态转换图的终止态。

这样做的好处是程序简洁明了，逻辑清晰，方便进行调试。

思考题 2：符号表是怎样处理的，可不可以提出改进方法？

一开始是以数组的形式直接存储所有符号，这样的时间复杂度很高，线性表中插入 n 个符号，执行 e 次查找的时间复杂度为$O\big(n(n+e)\big)$。

改进方法：建立一张 *hash table*，通过哈希表存储符号表，这样可以极大地减少符号表的插入和查找的时间复杂性，提升符号表插入和查找操作的性能。根据分析，长度为 n 的线性表被分割成 m 个长度为 n/m 的线性表，此时的时间复杂度是$\mathrm{T}(n,e,m)=O(n(n+e)/m)$。如果 m 取 n 时，$\mathrm{T}(n,e,m)=O(n+e)$。相比于线性表存储符号表，时间复杂度从平方级别降到一次方级别。

五、实验体会

通过本次实验，我对编译原理的词法分析器有了更加深刻的了解。

词法分析是编译的第一个阶段，将构成源程序的字符串转换成等价的单词序列的程序。词法分析器从文件中读入表示源程序的字符流，按照程序功能等价的要求，将其转换成对应的单词序列，并提出其中的空格、注解等不影响程序语义的字符。

同时我发现了自身在理论知识上的不足，没有做到及时预习、及时复习，导致在绘制状态转换图时需要重新学习一遍，影响了效率。有限自动状态机那一部分想要学好，还是要去学习一些《形式语言与自动机》课程的相关知识，否则，理解起来还是有些困难。

指导教师评语：

日期：