**中山大学数据科学与计算机学院本科生实验报告**

课程名称：编译器构造实验 任课教师：陈炬桦 教学助理（TA）：

|  |  |  |  |
| --- | --- | --- | --- |
| 学年度 | 2018-2019 | 学期 | 第二学期 |
| 年级 | 2016级 | 专业（方向） | 计算机科学与技术 |
| 学号 | 16337327 | 姓名 | 郑映雪 |
| 电话 | 18956073450 | Email | 646352420@qq.com |
| 开始日期 | 2019.3.30 | 完成日期 | 2019.3.30 |

**实验题目**

## 1. Description

设一语言的关键词、运算符、分界符的个数与单词如下：

struct { int number; string str[10]; } keywords={3,"int","main","return"} ; //关键词

struct { int number; string str[10]; } operators ={5,"+","\*","=","+=","\*="}; //运算符

struct { int number; string str[10]; } boundaries ={6,"(",")","{","}",",",";"} ; //分界符

struct { int number; string str[100];} identifieres={0}; //标识符

struct { int number; string str[100];} Unsigned\_integer={0}; //无符号整数

以上类号分别为1~5，序号从0开始；

标识符是字母开头的字母数字串；常量为无符号整数；

用C++设计一程序实现词法分析。

## 2. Input

输入一程序，结束符用”#”；

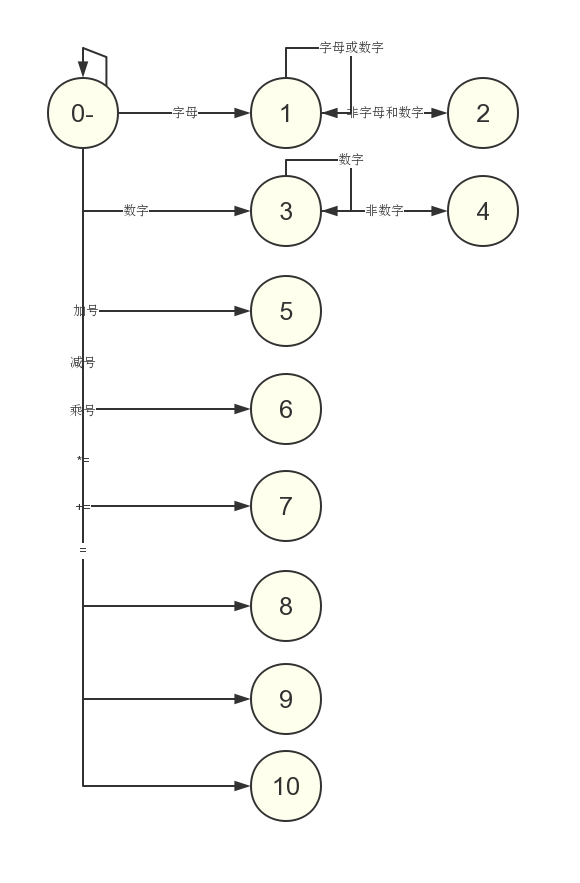
## 3. Output

输出单词数对：<类号，序号>。 输出标识符表，用空格分隔； 输出无符号整数表，用空格分隔；

**算法描述**

## 1. 流程总览

根据课本所说绘制如下状态图，并根据状态图设计算法（下面为部分状态）：



## 2.各模块功能

**a.初始模块**

本次的实验处理是比较简单的。一整个程序段分别处理的方法是有多种选择的。我选择getline函数，逐行读取和分析。getline读取之后会把整行的程序段放进一个字符串中。每次读取之后，进行结尾的井号判断。如果有井号则在该行处理之后退出循环。

**b.分析模块**

本次实验实际上是对书本上扫描程序的设计进行应用。

单词符号一般分为五类：关键字、标识符、常数、运算符和界限符。题目里设计好的struct也已经说明。那么可以根据上面的状态设计分析模块。

前面提及，使用getline函数得到的str为每一行程序代码。判断str的每个字符的类型，做下面的分析：

①字母，则继续叠加该字符接下来的若干字符，直到遇到非数字且非字母的字符。这时进入关键词判断。如果这串字符串是在关键词表中的，则输出该关键词在表中的位置。否则，则说明这是一个标识符，需要判断该串是在标识符表中，如果在，则输出在表中的位置，如果不在，需要添加到标识符表中。

②数字，则继续叠加该字符接下来的若干字符，直到遇到非数字的字符。这时进入数表判断。如果该值已经在数表中，则输出位置；否则，将此数字添加到数表之中。

③运算符。此时需要判断紧接的字符是否为运算符，如是，则二者叠加并输出该字符串在运算符表中的位置；否则，输出字符本身在运算符表中的位置。

④界限符。此时可直接输出该符号在界限符表中的位置。

**c.算法总结**

本次实验算法难度不大，总结来说就是两点——程序的分割和大量的查表操作。程序分割这方面，由于是直接对内容进行判断，而不是对不合法内容进行筛除，所以分割十分容易。查表判断方面，由于本次实验给出的表比较简单，只需要根据状态机设计逻辑，实现也十分的轻松。词法分析关键是划分各个单词符号的类型，提前设计好状态机，打代码自然胸有成竹。

**测试数据(2组)**

|  |  |  |
| --- | --- | --- |
| 测试数据 | 测试结果 | 结果判定 |
| main()  { int a=2,b=3;  return 2\*b+a;  }# | （由于是逐行读取所以结果也是逐行显示） | 正确 |
| int a=1+2\*3# |  | 正确 |

**程序清单**

1. #include<iostream>
2. #include<string>
3. #include<cstring>
4. **using** **namespace** std;
5. **struct** { **int** number; string str[10]; } keywords = { 3,"int","main","return" }; //关键词
6. **struct** { **int** number; string str[10]; } operators = { 5,"+","\*","=","+=","\*=" }; //运算符
7. **struct** { **int** number; string str[10]; } boundaries = { 6,"(",")","{","}",",",";" }; //分界符
8. **struct** { **int** number; string str[100]; } identifieres = { 0 }; //标识符
9. **struct** { **int** number; string str[100]; } Unsigned\_integer = { 0 }; //无符号整数
11. **int** find(string str,**int** length, string str2[]) {
12. **int** flag = 0;
13. **for** (**int** i = 0; i < length; i++)
14. **if** (str == str2[i]) **return** (i+1);
16. **return** 0;
17. }
19. **int** main() {
20. string str;
21. **int** flag = 0;
22. **while** (**true**) {
23. getline(cin, str);
24. **if** (str[str.size() - 1] == '#') {
25. str.pop\_back();
26. flag = 1;
27. }
28. string tmpstr = "";
29. **int** i = 0;
30. **while** (i < str.length()) {
31. **if** (str[i]!=' ') tmpstr += str[i];
32. **if** (find(tmpstr, boundaries.number, boundaries.str)) {//分界符
33. cout << "<" << 3 << "," << find(tmpstr, boundaries.number, boundaries.str)-1 << ">";
34. }
35. **else** {
36. **if** (str[i] >= 'a'&& str[i] <= 'z' || str[i] >= 'A'&& str[i] <= 'Z' &&str[i]!=' ') {//字母
37. **while** (**true**) {
38. **if** (str[i + 1] >= 'a'&& str[i + 1] <= 'z' || str[i + 1] >= 'A'&& str[i + 1] <= 'Z' || str[i + 1] >= '0'&&str[i + 1] <= '9') {
39. tmpstr += str[i+1];
40. i++;
41. }
42. **else** {
43. **break**;
44. }
45. }//在关键词则输出关键词
46. **if** (find(tmpstr, keywords.number, keywords.str)) cout << "<" << 1 << "," << find(tmpstr, keywords.number, keywords.str)-1 << ">";
47. //不在则输入标识符中
48. **else** {
49. **int** flag2 = find(tmpstr, identifieres.number, identifieres.str);
50. **if** (flag2==0) {
51. identifieres.number++;
52. **int** tmp = identifieres.number - 1;
53. identifieres.str[tmp] = tmpstr;
54. }
55. cout << "<" << 4 << "," << find(tmpstr, identifieres.number, identifieres.str)-1 << ">";
56. }
57. }
58. **else** **if** (str[i] >= '0' && str[i] <= '9') {//数字
59. **while** (**true**) {
60. **if** (str[i + 1] >= '0'&&str[i + 1] <= '9') {
61. tmpstr += str[i + 1];
62. i++;
63. }
64. **else** {
65. **break**;
66. }
67. }
68. **int** newflag = 1;
69. **for** (**int** j = 0; j < Unsigned\_integer.number; j++) {
70. **if** (tmpstr == Unsigned\_integer.str[j]) newflag = 0;
71. }
72. **if** (newflag==1) {
73. Unsigned\_integer.number++;
74. **int** tmp = Unsigned\_integer.number - 1;
75. Unsigned\_integer.str[tmp] = tmpstr;
76. }
77. cout << "<" << 5 << "," << find(tmpstr, Unsigned\_integer.number, Unsigned\_integer.str)-1 << ">";
78. }
79. **else** **if**(find(tmpstr, operators.number, operators.str)){//符号
80. **while** (**true**) {
81. string stri1 = "" + str[i + 1];
82. **if** (find(stri1, operators.number, operators.str)) {
83. tmpstr += str[i + 1];
84. i++;
85. }
86. **else** **break**;
88. }
89. cout << "<" << 2 << "," << find(tmpstr, operators.number, operators.str)-1 << ">";
90. }
91. }
92. tmpstr = "";
93. i++;
94. }
95. **if** (flag) **break**;
96. }
98. cout << endl << "identifieres:";
99. **for** (**int** i = 0; i < identifieres.number; ++i) {
100. cout << identifieres.str[i];
101. **if** (i != identifieres.number - 1) cout << " "; **else** cout << endl;
102. }
104. cout << endl << "Unsigned\_integer:";
105. **for** (**int** i = 0; i < Unsigned\_integer.number; ++i) {
106. cout << Unsigned\_integer.str[i];
107. **if** (i != Unsigned\_integer.number - 1) cout << " "; **else** cout << endl;
108. }
109. }