**课程编号：C0801003030**

**编译方法实验报告**



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **岑哲栋** | **学号** | | **20165018** |
| **班级** | **软件1607** | **指导教师** | | **刘洪娟** |
| **实验名称** | **扫描器设计及中间代码生成器设计** | | | |
| **开设学期** | **2018-2019学年秋季学期** | | | |
| **开设时间** | **第15周和第17周** | | | |
| **报告日期** | **2018年12月30日** | | | |
| **评定成绩** |  | | **评定人** | **刘洪娟** |
| **评定日期** | **2018年12月31日** |

**东北大学软件学院**

1. 实验目的

**实验 1：**

熟悉并实现一个扫描器（词法分析程序）。

1. 实验内容

**实验 1：**

首先了解了扫描器的工作原理，之后仔细阅读题目中的要求，原本应当根据题目中的要求编写出相应的程序，但实验指导书中已经有现成的参考代码，因此在充分了解该代码运作机制的情况下直接使用该代码进行测试，发现了一些存在的问题，并针对这些问题修改了部分的参考代码直至测试结果与预期结果无误。最后撰写实验报告。

1. 实验原理及基本步骤

**实验 1：**

扫描器的工作方式如图所示

扫描器

单词 Token

扫描器具体有两个任务，一是识别单词，即从用户的源程序中把单词分离出来，二是翻译单词，即把单词转换成机内表示，便于后续处理。识别单词的任务本质上是由一个有限自动机来完成，这个有限自动机需要通过状态转换图或状态转换矩阵来确定。识别完成一个token之后，它的最终状态即代表了这个token的类型，在本实验中用一个(code,value)对的方式来表示token,特别要注意的是关键字和界符，它们需要专门存储起来用以比对token是否为关键字或界符。

**实验2：**

1. 数据结构设计

**实验 1：**

根据状态转换图设计如下的状态转换矩阵：

int aut[11][8]=

{

0, 0, 0, 0, 0, 0, 0, 0,

0, 2, 0, 0, 0, 8, 9,15,

0, 2, 3, 5,11, 0, 0,11,

0, 4, 0, 0, 0, 0, 0, 0,

0, 4, 0, 5,11, 0, 0,11,

0, 7, 0, 0, 6, 0, 0, 0,

0, 7, 0, 0, 0, 0, 0, 0,

0, 7, 0, 0,11, 0, 0,11,

0, 8, 0, 0, 0, 8, 0,12,

0, 0, 0, 0, 0, 0,10,14,

0, 0, 0, 0, 0, 0, 0,13

}

原本只有10种状态和7种类型的符号，为何这里要设计出11行来表示状态，8行来表示符号类型呢？因为C语言中数组下标从0开始，如果只设计10\*7的矩阵的话，每次根据行和列取得矩阵中的元素时，行和列都要-1，不符合一般人的理解方式，且增大了无谓的计算量。直接设计11\*8的矩阵，使得第0行和第0列都用0填充，这样根据行和列取得矩阵元素时就不需要-1，符合一般人的理解规律。

设计一个30\*12的字符二维数组来存储关键字和界符：

char keywords[30][12]={"program","begin","end","var","while","do","repeat",

"until","for","to","if","then","else",";", ":", "(", ")", ",",

":=", "+", "-", "\*", "/", ">", ">=", "==", "<", "<="} 尽管关键字和界符的总数只有28个，但为了方便以后扩展关键字或界符，将容量设为30。

char ID[50][12]; 用来存储源程序中读取出来的标识符。

double C[20]; 用来存储源程序中读取出来的常数。

结构体

struct token

{

int code;

int value;

}; 来表示token结构。

struct token tok[100]; 来存储识别出的token序列。

char strTOKEN[15]; 用来存储当前正在识别的token。

char w[50]; 来作为源程序的缓冲区，从该数组中取单词。

结构体：

struct map

{

char str[50];

int col;

}; 用来表示当前字符到状态转换矩阵标记的映射。

struct map col1[4]={{"0123456789",1},{".",2},{"Ee",3},{"+-",4}}; struct map col2[2]={{"abcdefghijklmnopqrstuvwxyz",5},{"0123456789",1}}; struct map col3[1]={{";:(),+-\*/=><",6}}; 分别表示数字、关键字或标识符、界符符号的映射。

1. 关键代码分析（带注释）及运行结果

**关键代码：**

**实验 1：**

*判断单词类别：*

while (w[i]==' ') //滤空格

i++;

//判定单词类别

if (w[i]>='a' && w[i]<='z')

{

ptr=col2; num\_map=2; //关键字或标识符

}

else if (w[i]>='0' && w[i]<='9')

{

ptr=col1; num\_map=4; //数字

}

else if (strchr(col3[0].str,w[i])==NULL)

{

printf("非法字符%c\n",w[i]); //非法字符

i++; //跳过

continue;

}

else

{

ptr=col3; num\_map=1; //界符

}

*进行状态变换：*

int find(int s,char ch)

{

int i,col=7;

struct map \*p;

p=ptr; //ptr代表映射到当前字符对应的map数组指针，在判断单词类别时确定

for (i=0;i<num\_map;i++) //num\_map代表对应的map数组中包含的元素个数，和ptr同时确定

if (strchr((p+i)->str,ch)!=NULL&&ch!='\0')

{

col=(p+i)->col; //确定新状态在状态转换矩阵中的列号

break;

}

return aut[s][col];

}

**运行结果：**

**实验 1：**

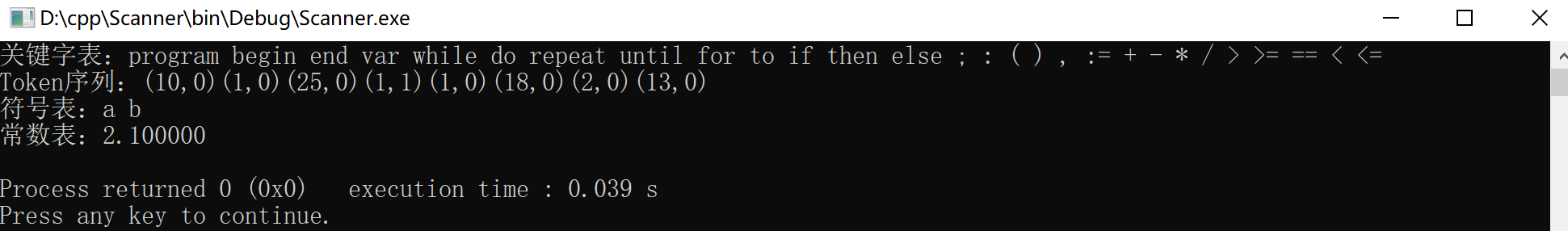
实验指导书中的示例代码在源程序代码只有一行时会出错，而且token数量会莫名其妙多出来一个，同时由于它用int型数组来保存常数值，因此无法正确显示小数。经修改后的程序解决了这些问题。

设计源程序如下：

if a==b

a:=2.1;

运行结果如图：



可以看到token序列完美地满足了题目中的要求：标识符的编码（code）为1，值（value）为其在符号表中的位置；常数的编码（code）为2，值（value）为其在常数表中的位置；关键字和界符的编码（code）为其在关键字表中的位置，值（value）为0。

**实验 2：**

1. 总结与分析

**成绩评价表格**：

|  |  |
| --- | --- |
| 考核标准 | 得分 |
| （1）正确理解和掌握实验所涉及的基本概念和原理；（10%）； |  |
| （2）按实验要求合理设计数据结构和程序结构（20%）； |  |
| （3）实验中要设计各种测试用例，考虑词法、语法和语义正确和不正确等各种情况（20%）； |  |
| （4）认真记录实验结果，对实验结果的分析准确（20%）； |  |
| （5）实验过程中，具有严谨、认真的学习态度，不做与实验内容无关的其他学习工作（10%）； |  |
| （6）实验2应具有一定的创新性（10%）； |  |
| （7）实验报告内容充实、格式符合规范（10%）。 |  |