

编译原理词法分析实验报告

一、实验题目

词法分析程序的设计与实现

本程序严格按照教材题目要求实现功能，是独立完成。同时本程序完善课本给的样例（教材给的那个自动机太不全了，很多符号都无法识别，因此本程序大大扩展了可识别符号）

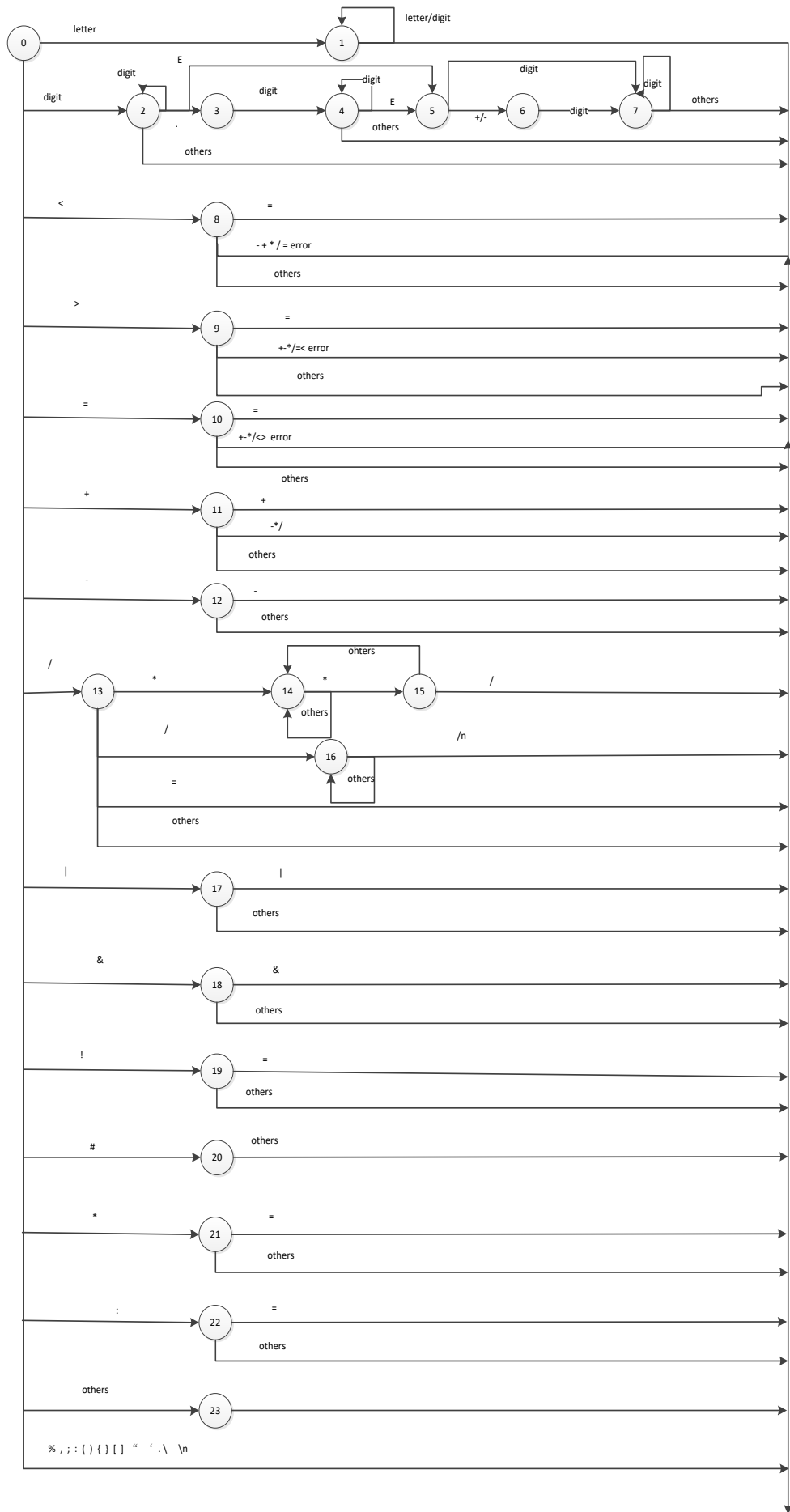
二、实验要求

- 1.可以识别出用 C 语言编写的源程序中的每个单词符号，并以记号的形式输出每个单词符号。
- 2.可以并识别并跳过硬程序中的注释。
- 3.可以统计源程序中的语句行数、各类单词个数，以及字符总数，并输出统计结果。
- 4.检查源程序中存在的词法错误，并报告错误所在位置。
- 5.发现对源程序中存在的错误，进行适当修复，使词法分析可以继续进行，通过一次词法分析处理，可以检查并报告源程序中存在的所有错误。

三、程序设计思路

- 1.根据课上所学知识，使用 c 语言实现确定自动机结构，来完成词法分析过程。
- 2.源程序的输入采用缓冲区的方式。设置一个分成左右两个半区的缓冲区，每个半区可保存 N 个字符（N=2048）。一次读入 N 个字符，已备送到左半区或右半区。两个半区交替使用。
- 3.建立 c 语言 32 个保留字字符数组以及用户自定义字符数组，记录不同类型单词。其中用户自定义字符保存可实现查重功能，用户定义的字符重复出现不会产生歧义。
- 4.设置多个变量来记录统计语句行数、各类单词个数，以及字符总数以及词法错误数。
- 5.分析标识符、常数和运算符的词法错误，并输出结果到 error_file.txt 文件中。
- 6.将源程序按记号形式（记号，属性二元组）输出到控制台以及 output.txt 文件中。

下图为使用 visio 画的状态转移图。其中画出部分了对词法异常的处理。



四、全局变量以及函数说明

- 1.**state**：整型变量，当前状态指示。
- 2.**C**：字符变量，存放当前读入的字符。
- 3.**iskey**：整型变量。值为-1时，表示识别的单词是用户自定义标识符，否则，表示识别的单词是关键词，其值为关键字的记号。
- 4.**token**：字符数组，存放当前正在识别的单词的字符串。
- 5.**forward**：字符指针，向前指针，这里用整型变量表示。
- 6.**buffer**：字符数组，输入缓冲区。
- 7.**get_nbc ()**：函数，每次调用时，检查 C 中的字符是否为空格，若是，则反复调用函数 **get_char ()** 直到 C 中进入非空字符为止。
- 8.**cat ()**：函数，把 C 中的字符连接到 token 中的字符后面。
- 9.**letter ()**：布尔函数，判断 C 中的字符数是否为字母，若是则返回 true，否则，返回 false。
- 10.**digit ()**：布尔函数，判断 C 中字符数是否为数字，若是则返回 true，否则返回 false。
- 11.**retract ()**：函数，向前函数 forward 后退一个字符。
- 12.**reserve ()**：函数，根据 token 找那个的单词查关键字表，若 token 中的单词是关键字，则返回该关键字的记号，否则，返回-1。
- 13.**table_insert ()**：函数，将识别出来的用户自定义标识符（即 token 中的单词）插入符号表，返回该单词在符号表中的指针。
- 14.**error ()**：对非法字符输入进行报错。
- 15.**my_return ()**：函数，将识别出来的单词记号输出到控制台和文件。
- 16.**get_char ()**：从输入缓冲区中读取 forward 指向的字符到 C。
- 17.**fillbuffer ()**：读文件到缓冲区函数。
- 18.**initial ()**：初始化函数，初始化各种统计量以及指针变量。
- 19.**struct idInfo** 定义了一个结构体来表示当前单词的记号以及属性

五、测试样例

1.test1

测试源码

```
1  /*这段注释会被跳过*/
2  #include<stdio.h>
3  #include<stdlib.h>
4  int main(){
5      printf("Hello World!\n"); //也可以跳过此段注释
6
7      int n=6,m;
8      m=(6*6)/n;
9      m++;
10     double x=6.6E+6;
11     if(x>m||x>m||x==m)
12     { printf("This is 666 program.");
13     }
14 }
```

分析结果：

```
C:\Users\Rocair\Desktop\词法分析win.exe
请输入词法分析文件名(e.g. test1.cpp):
test1.cpp

----词法分析记号结果----:
< #, header >
< #, header >
< int, - >
< ID, main >
< (, - >
< {, - >
< ID, printf >
< (, - >
< ID, Hello >
< ID, World >
< !, - >
< \, - >
< ID, n >
< ", - >
< ), - >
< ;, - >
< int, - >
< ID, n >
< relop, EQ >
< NUM, 6 >
< ,, - >
< ID, m >
< ;, - >
< ID, m >

选择C:\Users\Rocair\Desktop\词法分析win.exe
< relop, EQ >
< (, - >
< NUM, 6 >
< *, - >
< NUM, 6 >
< ), - >
< /, - >
< ID, n >
< ;, - >
< ID, m >
< ++, - >
< ;, - >
< double, - >
< ID, x >
< relop, EQ >
< NUM, 6.6E+6 >
< ;, - >
< if, - >
< (, - >
< ID, x >
< relop, - >
< ID, m >
< ||, - >
< ID, x >
< relop, - >
< ID, m >
< ||, - >
< ID, x >
< ==, - >
< ID, m >

选择C:\Users\Rocair\Desktop\词法分析win.exe
< ID, m >
< ), - >
< {, - >
< ID, printf >
< (, - >
< ", - >
< ID, This >
< ID, is >
< NUM, 666 >
< ID, program >
< ;, - >
< ", - >
< ), - >
< ;, - >
< }, - >
< ), - >
-程序记号表达已保存到output.txt文件中
-程序错误分析已保存到error_file.txt文件中

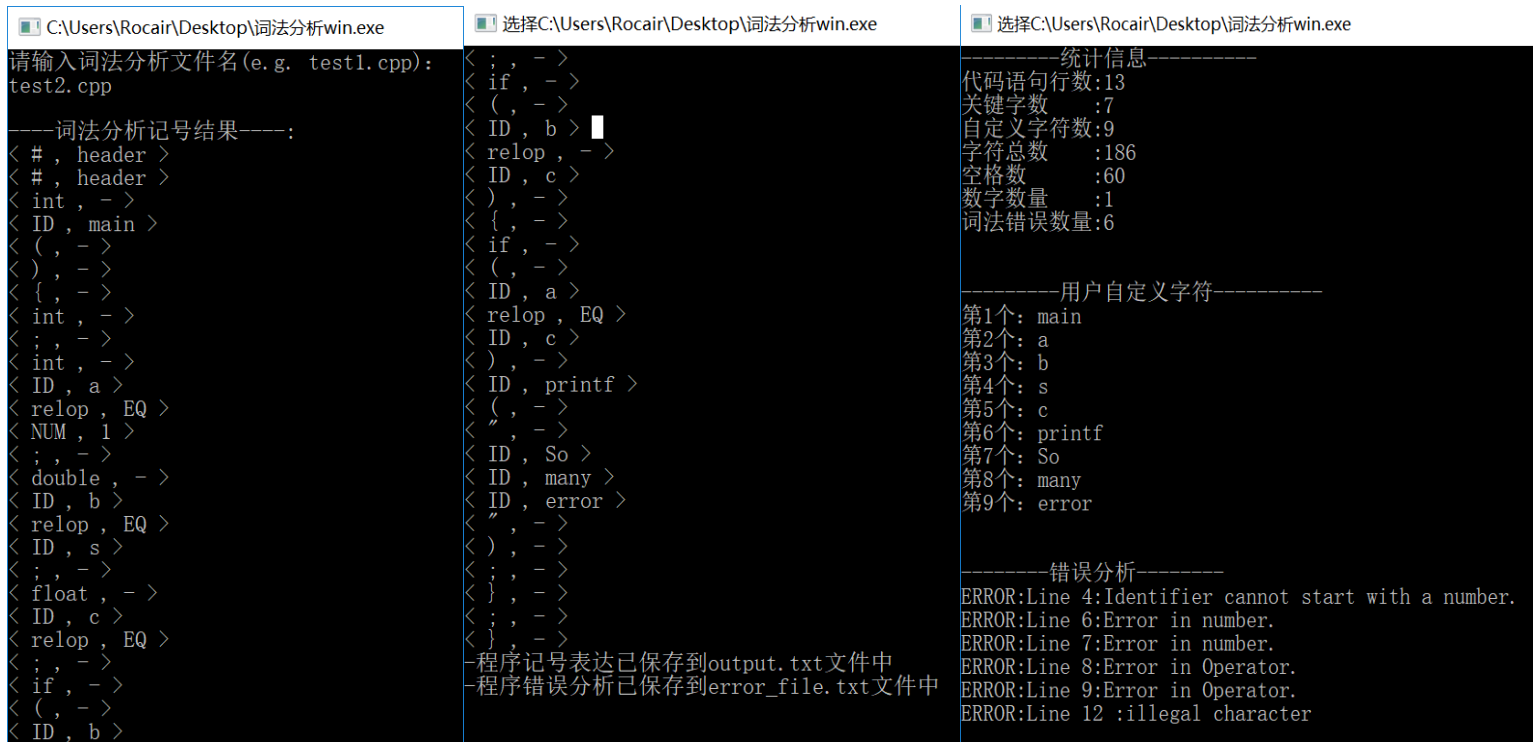
-----统计信息-----代码语句行数:13
关键字数:4
自定义字符数:10
字符总数:232
空格数:50
数字数量:5
词法错误数量:0
```

2.test2

测试源码：

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 int main(){ //报错功能测试样例
4     int 2m;
5     int a=1;
6     double b=6.6Es;
7     float c=6.0;
8     if(b >< c)
9     { if(a =< c)
10         printf("So many error");
11     }
12     @;
13 }
```

分析结果：



六、实验总结

通过此次实验，加大对词法分析的理解。

同时我也理解了最初学习 C 语言的困惑，就是为啥每行结束一定要有“；”。在最初的编写中，我总是忘记在最后加上分号。当时心中就暗想，这语言设计真麻烦，非得加一个看似没用的符号。可是通过此次实验，终于了解到，加分号，是让编译器有一个对不同语句有一个很好的分隔符，这样可以更加方便的进行编译。

七、源程序

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <iostream>
#define bufferlength 4095
#define halflength 2047
#define tokenlength 100
#define tablelength 150

using namespace std;

FILE *file=NULL,*output=NULL,*error_file=NULL;

//统计量
int NumSpace,NumKeyword,NumRow,NumError,NumDigit,NumChar;
```

```

int state;
char C;
int iskey;
char token[tokenlength];
int tokenptr;
int tableptr;
int forward;
char buffer[bufferlength];

char table[tablelength][30];
char keywords[32][15]={
    "auto","break","case","char","const","continue","default","do"
    ,"double","else","enum","extern","float","for","goto","if","int","long"
    ,
    "register","return","short","signed","sizeof","static","struct","switch",
    "typedef","union","unsigned","void","volatile","while"};

void initial()
{
    state=0;
    NumSpace=0;NumKeyword=0;NumRow=0;NumError=0;NumDigit=0;NumChar=0;
    forward=0;tokenptr=0;tableptr=0;
}
//填充缓冲区
void fillbuffer(int p)
{
    int i;
    for(i=0;i<=halflength&&!feof(file);i++)
        buffer[p+i]=fgetc(file);
    if(feof(file)) buffer[p+i]='\0';
    NumChar=i-2;
}

void get_char()
{
    C=buffer[forward];
    if(C=='\n')
        NumRow++;

    if(forward==halflength){
        fillbuffer(halflength+1);
        forward++;
    }
    else if(forward==bufferlength){
        fillbuffer(0);
    }
}

```

```

        forward=0;
    }
    else forward++;

}

void get_nbc()
{ while(1)
    {if(C==' '||C=='\n' ||C==' '){
        if(C==' ')
            NumSpace++;
        else if(C==' ')
            NumSpace=NumSpace+4;

        get_char();
    }
    else
        break;
    }
}

void cat()
{ token[tokenptr++]=C;
}

bool letter() //判断字符 C 是否为字母
{ return ((C>='a'&&C<='z')||(C>='A'&&C<='Z'));
}

bool digit() //判断字符 C 是否为数字
{ return (C>='0'&&C<='9');
}

void retract() //缓冲区指针回退
{ if(forward==0)
    forward=bufferlength;
    else
        forward--;
}

int reserve()
{ int i;
    for(i=0;i<32;i++)
        if(strcmp(token,keywords[i])==0)

```

```

        { NumKeyword++;
          return i;
        }
    if(i==32)
        return -1;
}

int table_insert()
{ int i;
  for(i=0;i<tableptr;i++)
      if(strcmp(token,table[i])==0)
          return i;

  if(i==tableptr)
  { strcpy(table[tableptr],token);
    return tableptr++;
  }
}

void error()
{ fprintf(error_file,"ERROR:Line %d :illegal character\n",NumRow+1);
  NumError++;
}

typedef struct{
    string id;
    string type;
}CInfo;

void my_return(CInfo temp)
{
    //输出控制台 和文件
    printf("< %s , %s >\n",temp.id.c_str(),temp.type.c_str());
    fprintf(output,"< %s , %s >\n",temp.id.c_str(),temp.type.c_str());
}

int main()
{ char filename[30];
  printf("请输入词法分析文件名(e.g. test1.cpp): \n");
  scanf("%s",filename);
  file=fopen(filename,"r");

  while(output==NULL)

```



```

        output=fopen("output.txt","w");
while(error_file==NULL)
    error_file=fopen("error_file.txt","w");
while(file==NULL){
    printf("The file don't exist,please input again: \n");
    scanf("%s",filename);
    file=fopen(filename,"r");
}
printf("\n----词法分析记号结果----:\n");
initial();
fillbuffer(0);
get_nbc();

CInfo temp;

do{
    switch (state)
    {
        case 0:
            memset(token,0,100);//token='';
            tokenptr=0;
            get_char();
            get_nbc();

            switch (C)
            {
                case 'a': case 'b': case 'c': case 'd': case 'e':
                case 'f': case 'g': case 'h': case 'i': case 'j':
                case 'k': case 'l': case 'm': case 'n': case 'o':
                case 'p': case 'q': case 'r': case 's': case 't':
                case 'u': case 'v': case 'w': case 'x': case 'y':
                case 'z':
                    case 'A': case 'B': case 'C': case 'D': case 'E':
                case 'F': case 'G': case 'H': case 'I': case 'J':
                case 'K': case 'L': case 'M': case 'N': case 'O':
                case 'P': case 'Q': case 'R': case 'S': case 'T':
                case 'U': case 'V': case 'W': case 'X': case 'Y':
                case 'Z':
                        state=1;break;//

                case '0': case '1': case '2': case '3': case '4':
                case '5': case '6': case '7': case '8': case '9':
                        state=2;break;//
            }
        }
    }

```

```
case '<':
    state=8;
    break;
case '>':
    state=9;
    break;
case '=':
    state=10;
    break;
case '+':
    state=11;
    break;
case '-':
    state=12;
    break;
case '/':
    state=13;
    break;
case '|':
    state=17;
    break;
case '&':
    state=18;
    break;
case '!':
    state=19;
    break;
case '#':
    state=20;
    break;
case '*':
    state=21;
    break;
case ':':
    state=22;
    break;
case '%':
    state=0;
    temp.id = "%";
    temp.type = "-";
    my_return(temp);
    break;
case ',':
```

```

        state=0;
        temp.id=',';
        temp.type='-';
        my_return(temp);
        break;
    case ';':
        state=0;
        temp.id=';';
        temp.type='-';
        my_return(temp);
        break;

    case '(':
        state=0;temp.id='(';temp.type='-';
';my_return(temp);break;
    case ')':
        state=0;temp.id=')';temp.type='-';
';my_return(temp);break;
    case '{':
        state=0;temp.id='{';temp.type='-';
';my_return(temp);break;
    case '}':
        state=0;temp.id='}';temp.type='-';
';my_return(temp);break;
    case '[':
        state=0;temp.id='[';temp.type='-';
';my_return(temp);break;
    case ']':
        state=0;temp.id=']';temp.type='-';
';my_return(temp);break;
    case '"':
        state=0;temp.id='"';temp.type='-';
';my_return(temp);break;
    case '\\':
        state=0;temp.id='\\';temp.type='-';
';my_return(temp);break;
    case '.':

state=0;temp.id='.';temp.type='_';my_return(temp);break;
    case '\n':
        state=0;temp.id='\n';temp.type='-';
';my_return(temp);break;
    case '\\':

```

```

        state=0;temp.id='\\';temp.type='-
';my_return(temp);break;

        default:
            state=23;break;
    }

    break;

case 1:
    cat();
    get_char();
    if( letter() || digit() )
        state=1;
    else
    { retract();
      state=0;
      iskey=reserve();
      if(iskey!=-1)
      { string str(keywords[iskey]);
        //temp.id=str;
        temp.id=str;//strcpy(temp.id,keywords[iskey].

    );

        temp.type="-";
        my_return(temp);
      }

      else
      { int identry=table_insert();
        temp.id="ID";

        string st(table[identry]);
        temp.type=st;
        // printf("***%s***\n",st.c_str());
        my_return(temp);
      }
    }
    break;

case 2:
    cat();
    get_char();

```

```

switch (C)
{
    case '0': case '1': case '2': case '3': case '4':
    case '5': case '6': case '7': case '8': case '9':
        state=2;break;

    case '.':
        state=3;break;

    case 'E': case 'e':
        state=5;break;

    case 'a': case 'b': case 'c': case 'd':
    case 'f': case 'g': case 'h': case 'i': case 'j':
    case 'k': case 'l': case 'm': case 'n': case 'o':
    case 'p': case 'q': case 'r': case 's': case 't':
    case 'u': case 'v': case 'w': case 'x': case 'y':
    case 'z':
        case 'A': case 'B': case 'C': case 'D':
    case 'F': case 'G': case 'H': case 'I': case 'J':
    case 'K': case 'L': case 'M': case 'N': case 'O':
    case 'P': case 'Q': case 'R': case 'S': case 'T':
    case 'U': case 'V': case 'W': case 'X': case 'Y':
    case 'Z':
        state=0;

    fprintf(error_file,"ERROR:Line %d:Identifier cannot start with a
    number.\n",NumRow+1);

        NumError++;
        break;
    default:
        retract();
        state=0;
        temp.id="NUM";
        temp.type=token;//字符串转数字  SToI
        my_return(temp);
        NumDigit++;
        break;
}
break;

case 3:
    cat();

```

```

        get_char();
        if( digit())
            state=4;
        else{
            fprintf(error_file,"ERROR:Line %d:Error in
number.\n",NumRow+1);
            state=0;
            NumError++;
        }
        break;

    case 4:
        cat();
        get_char();

        switch (C)
        {
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
                state=4;break;

            case 'E':
                state=5;break;
            default:
                retract();
                state=0;
                temp.id="NUM";
                temp.type=token;//字符串转浮点数 STOf
                my_return(temp);
                NumDigit++;
                break;
        }
        break;

    case 5:
        cat();
        get_char();

        switch (C)
        {
            case '0': case '1': case '2': case '3': case '4':
            case '5': case '6': case '7': case '8': case '9':
                state=7;break;

```

```

        case '+':case '-':
            state=6;break;
        default:
            retract();
            fprintf(error_file,"ERROR:Line %d:Error
in number.\n",NumRow+1);

            state=0;
            NumError++;
            break;
    }
    break;

case 6:
    cat();
    get_char();
    if( digit() ) state=7;
    else{
        retract();
        fprintf(error_file,"ERROR:Line %d:Error in
number.\n",NumRow+1);
        state=0;
        NumError++;
    }
    break;

case 7:
    cat();
    get_char();
    if( digit() ) state=7;
    else{
        retract();
        state=0;
        temp.id="NUM";
        //string s(token)
        temp.type=token; //字符串转浮点数 SToF
        my_return(temp);
        NumDigit++;
    }
    break;

case 8:// '<'
    cat();
    get_char();

```

```

switch (C)
{
    case '=':
        cat();
        state=0;
        temp.id="relop";
        temp.type="LE";
        my_return(temp);
        break;
    case '>':
        cat();
        state=0;
        temp.id="relop";
        temp.type="NE";
        my_return(temp);
        break;
    case '+':case '-':case '*':case '/':
        state=0;
        fprintf(error_file,"ERROR:Line %d:Error in
Operator.\n",NumRow+1);
        NumError++;
        break;
    default:
        retract();
        state=0;
        temp.id="relop";
        temp.type="LT";
        my_return(temp);
        break;
}
break;

case 9: //'>'
    cat();
    get_char();
    if(C=='='){
        cat();
        state=0;
        temp.id="relop";
        temp.type="GE";
        my_return(temp);
    }
    else if(C=='+' || C=='-' || C=='*' || C=='/' || C=='<'){

```



```

        fprintf(error_file, "ERROR:Line %d:Error in
Operator.\n", NumRow+1);
        NumError++;
    }

    else{
        retract();
        state=0;
        temp.id="relop"; temp.type="-";
        my_return(temp);
    }
    break;

case 10:// '='
    // cat();
    get_char();
    if(C=='='){
        cat();
        state=0;
        temp.id=="=";temp.type="-";
        my_return(temp);
    }
    else if(C=='+' || C=='-'
|| C=='*' || C=='/' || C=='<' || C=='>'){
        fprintf(error_file, "ERROR:Line %d:Error in
Operator.\n", NumRow+1);
        NumError++;
    }
    else{
        retract();
        state=0;
        temp.id="relop";temp.type="EQ";
        my_return(temp);
        //NumAriOprt++;
    }
    break;

case 11:// '+'
    cat();
    get_char();
    if(C=='+'){
        cat();
        state=0;
        temp.id="++";temp.type="-";

```

```

        my_return(temp);
    }
    else if(C=='='){
        cat();
        state=0;
        temp.id="+=";temp.type="-";
        my_return(temp);
    }
    else if(C=='-'||C=='*'||C=='/'||C=='<'||C=='>'){
        fprintf(error_file,"ERROR:Line %d:Error in
Operator.\n",NumRow+1);
        NumError++;
    }
    else{
        retract();
        state=0;
        temp.id="+=";temp.type="-";
        my_return(temp);
    }
    break;

case 12:/'-'
    cat();
    get_char();
    if(C=='-'){
        cat();
        state=0;
        temp.id="--";temp.type="-";
        my_return(temp);
    }
    else if(C=='='){
        cat();
        state=0;
        temp.id="-=";temp.type="-";
        my_return(temp);
    }
    else if(C=='+'||C=='*'||C=='/'){
        fprintf(error_file,"ERROR:Line %d:Error in
Operator.\n",NumRow+1);
        NumError++;
    }
    else{
        retract();

```

```

        state=0;
        temp.id="-";temp.type="-";
        my_return(temp);

    }
    break;
case 13: '/'
    cat();
    get_char();
    if(C=='*') state=14;
    else if(C=='/') state=16;
    else if(C=='='){
        state=0;
        temp.id="/=";temp.type="-";
        my_return(temp);
    }
    else{
        retract();
        state=0;
        temp.id="/";temp.type="-";
        my_return(temp);
    }
    break;

case 14: '/' '*'
    cat();
    get_char();
    if(C=='*') state=15;
    else{
        state=14;
    }
    break;

case 15: '/' '*' '/'
    cat();
    get_char();
    if(C=='/') state=0;
    else state=14;
    break;

case 16: '/' '/' '/'
    cat();
    get_char();

```

```

        if(C == '\n')
            {state=0;
              //NumRow++;
            }
        else
            { //retract();
              state=16;
            }
        break;

case 17:// '|'
    cat();
    get_char();
    if(C=='|'){
        cat();
        state=0;
        temp.id="||";temp.type="-";
        my_return(temp);
    }
    else{
        retract();
        state=0;
    }
    break;

case 18: // '&'
    cat();
    get_char();
    if(C=='&'){
        cat();
        state=0;
        temp.id="&&";temp.type="-";
        my_return(temp);
    }
    else{
        retract();
        state=0;
    }
    break;

case 19: // '!'
    cat();
    get_char();
    if(C=='='){

```

```

        cat();
        state=0;
        temp.id="!=";temp.type="NE";
        my_return(temp);
    }
    else{
        retract();
        state=0;
        temp.id="!";temp.type="-";
        my_return(temp);
    }
    break;

case 20:// '#'
    //cat();

    get_char();
    while(C!='\n')
        get_char();
    //NumRow++;
    temp.id="#";
    temp.type="header";
    my_return(temp);
    // retract();
    state=0;
    break;
case 21:// '*'
    cat();
    get_char();
    if(C=='='){
        cat();
        state=0;
        temp.id="*=";temp.type="-";
        my_return(temp);
    }
    else{
        retract();
        state=0;
        temp.id="*";temp.type="-";
        my_return(temp);
    }
    break;

case 22:// ':'

```

```

        cat();
        get_char();
        if(C=='='){
            cat();
            state=0;
            temp.id="assign-op";temp.type="-";
            my_return(temp);
        }
        else{
            retract();
            state=0;
        }
        break;

    case 23:// 'other'
        error();
        state=0;
        break;
    }
    /* string ss(token);
    printf("^^^%s^^^\n",ss.c_str());*/

}while(C!=EOF);
printf("-程序记号表达已保存到 output.txt 文件中\n");
printf("-程序错误分析已保存到 error_file.txt 文件中\n\n");
printf("\n\n-----统计信息-----\n");
printf("代码语句行数:%d\n",NumRow);
printf("关键字数      :%d\n",NumKeyword);
printf("自定义字符数:%d\n",tableptr);
printf("字符总数      :%d\n",NumChar);
printf("空格数        :%d\n",NumSpace);
printf("数字数量      :%d\n",NumDigit);
printf("词法错误数量:%d\n",NumError);

printf("\n\n-----用户自定义字符-----\n");
for(int i=0;i<tableptr;i++)
{ printf("第%d 个: %s\n",i+1,table[i]);
}
fclose(error_file);
printf("\n\n-----错误分析-----\n");
char ch;          //读取的字符,判断准则为 ch 不等于结束符 EOF (end of
file)
FILE * fid = fopen("error_file.txt","r");

```

```
while(EOF!=(ch= fgetc(fid)))  
    printf("%c", ch);
```

```
}
```