**实验报告一**

一、 实验目的

1. 学会针对DFA转换图实现相应的高级语言源程序。

2. 深刻领会状态转换图的含义，逐步理解有限自动机。

3. 掌握手工生成词法分析器的方法，了解词法分析器的内部工作原理。

二、 实验内容

实现计算机语言的编译程序的词法分析功能。具体包括：从左至右逐行扫描源程序的字符，把源程序分解为一个个单词，这些单词有且只有五类，分别是标识符、保留字、常数、运算符和界符，用词法单元（token）表示，便于下一步进行语法分析。为了简化程序的编写，对识别的源文件要求如下：

1. 包含标识符（关键字）、整数和浮点数。

2. 多余空白符仅仅是空格、回车符、制表符（跳格键）。

3. 代码是自由格式。

4. 注释应放在/\* \*/或者 // 之内，并且不允许嵌套。

三、 实验要求（尽可能地实现所有功能，可采用c/java/python任一语言实现，可任意删除和修改代码）

理解已有的lex\_main.cpp源程序，要求补充部分代码以实现编译器的如下功能：

1. 读入源程序，逐行打印源程序，注意在每行的前面加上行号和空格。

2. 对源程序进行过滤，删除注释内容（/\* \*/或者//），删除除空格以外的空白符 (如制表符\t、回车符\r等，换行符\n不要删除)，注意用一个数组记录下删除了的行列位置，方便后期对错误的行和列进行定位。

3. 识别五类词法单元，并转换成二元式token形式逐行输出，实现GetToken函数，指出所有词法分析错误的行的位置：

1） 识别所有进制的整数，如int cc=0x4567。可用不同种别码表示不同的进制数。

2） 识别所有十进制的浮点数，如45.58e-10。

3） 识别复合赋值运算符，如+=，-=，\*=,|=,&=等。

4） 定位所有词法分析错误的列的位置（附加题）。

​

词法分析的具体要求：

1. 所有单词（token）二元式形式的种别码已提前分配好，自己可以根据需要进行添加；其中保留字和特殊字符是每个都一个种类，标识符自己是一类，数字是一类；单词的属性值就是表示的字符串值。

2. 词法分析的具体功能实现是一个函数GetToken()，每次调用都对剩余的字符串分析得到一个单词或记号识别其种类，收集该记号的符号串属性，当识别一个单词完毕，采用返回值的形式返回符号的种类，同时采用程序变量的形式提供当前识别出记号的属性值。这样配合语法分析程序的分析需要的记号及其属性，生成一个语法树。

3. 标识符和保留字的词法构成相同，为了更好的实现，把语言的保留字建立一个表格存储，这样可以把保留字的识别放在标识符之后，用识别出的标识符对比该表格，如果存在该表格中则是保留字，否则是一般标识符。

四、 实现方法

为了便于扫描器在构造单词的二元式时能方便的找到该单词的种别码，先建立一个足够容量的数组按字典顺序存放实现语言的单词编码表，每从源程序读入一个单词，先决定是否非法字符，然后再查编码表，根据返回的种别码进行处理：关键字和操作符直接生成token字；标识符、整常数、实常数和字符常数则先查填符号表（只填入标识符名），再生成相应的token字。

五、 错误处理

在词法分析阶段，可以发现以下错误：

1. 非法字符，即程序语言的字符集以外的字符

2. 注释或字符常数不闭封（即注释未配对或常数的格式不对）

程序遇到错误，输出包含错误的行号和列号的错误信息到外部文件error.txt，然后跳过错误继续分析。

六、 运行结果

测试源文件（根据testHard.c或另外增加内容）：

// Lexical\_Analysis.cpp : 定义控制台应用程序的入口点。

//

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

#include "iostream"

using namespace std;

//词法分析程序

//首先定义种别码

/\*

第一类：标识符  letter(letter | digit)\* 无穷集

第二类：常数  (digit)+ 无穷集

第三类：保留字(32)

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

第四类：界符 ‘/\*’、‘//’、 () { } [ ] " " ?'

第五类：运算符 <、<=、>、>=、=、+、-、\*、/、^、

对所有可数符号进行编码：

<$,0>

<auto,1>

...

<while,32>

<+，33>

<-,34>

<\*,35>

</,36>

<<,37>

<<=,38>

<>,39>

<>=,40>

<=,41>

<==,42>

<!=,43>

<;,44>

<(,45>

<),46>

<^,47>

<,,48>

<",49>

<',50>

<#,51>

<&,52>

<&&,53>

<|,54>

<||,55>

<%,56>

<~,57>

<<<,58>左移

<>>,59>右移

<[,60>

<],61>

<{,62>

<},63>

<\,64>

<.,65>

<?,66>

<:,67>

<!,68>

"[","]","{","}"

<常数99,数值>

<标识符100 ，标识符指针>

\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//全局变量，保留字表

static char reserveWord[32][20] = {

    "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"

};

//界符运算符表,根据需要可以自行增加

static char operatorOrDelimiter[43][10] = {

    "+", "-", "\*", "/", "<", "<=", ">", ">=", "=", "==",

    "!=", ";", "(", ")", "^", ",", "\"", "\'", "#", "&",

    " &&", "|", "||", "%", "~", "<<", ">>", "[", "]", "{",

    "}", "\\", ".", "\?", ":", "!","+=","-=","\*=","/=","&=","++","--"

};

static char IDentifierTbl[1000][50] = { "" };//标识符表

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*查找是否是保留字\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int searchReserve(char reserveWord[][20], char s[]) {

    //自己实现

    for (int i = 0; i < 32; i++)

    {

        if(strcmp(s,reserveWord[i])==0) {

            return i+1;

        }

    }

    return 0;

}

/\*\*\*\*\*\*\*\*查找保留字\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*判断是否为字母\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool IsLetter(char letter)

{//注意C语言允许下划线也为标识符的一部分可以放在首部或其他地方

    if (letter >= 'a'&&letter <= 'z' || letter >= 'A'&&letter <= 'Z'|| letter=='\_')

        return true;

    else

        return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*判断是否为字母\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*判断是否为数字\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool IsDigit(char digit)

{

    if (digit >= '0'&&digit <= '9')

        return true;

    else

        return false;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*判断是否为数字\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*编译预处理，取出无用的字符和注释\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void filterResource(char r[], int pProject)

{

    char tempString[10000];

    int count = 0;

    for (int i = 0; i <= pProject; i++)

    {//自己实现，从源程序中删除换行符\n、制表符\t、回车符\r，删除/\* ?\*/或者 //包含的注释。

        if(r[i]=='\t' || r[i]=='\r') {

            continue;

        }

        if(r[i]=='/' && r[i+1]=='/') {

            i += 2;

            while(r[i] != '\n') {

                i++;

            }

            continue;

        }

        int temp = i;

        bool flag = true;

        if(r[i]=='/' && r[i+1]=='\*') {

            i += 2;

            while(r[i] != '\*' || r[i+1] != '/') {

                i++;

            }

            if(i > pProject)

            {

                flag = false;

            }

            if(flag)

            {

                 i++;

                 continue;

            }

            else

            {

                i = temp;

            }

        }

        tempString[count++] = r[i];

     }

     tempString[count] = '\0';

     strcpy(r, tempString);//产生净化之后的源程序

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*编译预处理，取出无用的字符和注释\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*分析子程序，算法核心\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int row = 1, col = 1;

void GetToken(int &syn, char resourceProject[], char token[], int &pProject)

{//根据DFA的状态转换图设计

     int i, count = 0;//count用来做token[]的指示器，收集有用字符

     char ch;//作为判断使用

     ch = resourceProject[pProject];

     if(ch == '\0')

     {

         exit(0);

     }

     while (ch == ' ')

     {//过滤空格，防止程序因识别不了空格而结束

         pProject++;

         col++;

         ch = resourceProject[pProject];

     }

     if(ch == '\n')

     {

         pProject++;

         row++;

         col = 1;

         ch = resourceProject[pProject];

     }

     for (i = 0; i<20; i++)

     {//每次收集前先清零

         token[i] = '\0';

     }

     if (IsLetter(resourceProject[pProject]))

     {

         bool flag = false;

         if(resourceProject[pProject-1]=='\"')

         {

             flag = true;

         }

         //开头为字母，读入后面的字符，判断是否为标识符，并将字符串存储在token中，自己实现。

         while(IsLetter(resourceProject[pProject]) || IsDigit(resourceProject[pProject])) {

             token[count++] = resourceProject[pProject];

             pProject++;

             col++;

         }

         token[count] = '\0';

         int t = searchReserve(reserveWord, token);

         //查表找到关键字的种别码，自己实现

         if(t) {

             syn = t;

         }

         if (syn == -1)

         {//若不是关键字则是标识符

             syn = 100;//标识符种别码

         }

         if(flag)

         {

             syn = 102;

         }

         return;

     }

     else if (IsDigit(resourceProject[pProject]))

     {//首字符为数字

         bool flag = false, flag0 = false;

         if (resourceProject[pProject] == '0' && resourceProject[pProject + 1] == 'x')

         {

             token[count++] = '0';

             token[count++] = 'x';

             pProject += 2;

             col += 2;

             while(IsDigit(resourceProject[pProject]) || (resourceProject[pProject]>='a'&&resourceProject[pProject]<='f') || (resourceProject[pProject]>='A'&&resourceProject[pProject]<='F'))

             {

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

             }

             if((resourceProject[pProject]>'f'&&resourceProject[pProject]<='z')||(resourceProject[pProject]>'F'&&resourceProject[pProject]<='Z'))

             {

                 flag0 = true;

                 FILE \*fp;

                 if ((fp = fopen("E:\\私人\\实验报告\\编译原理\\实验一\\error.txt", "a+")) == NULL)

                 {//打开源程序

                     cout << "can't open this file";

                     exit(0);

                 }

                 printf("error：源文件 (%d,%d) 位置发生错误，16进制格式不对，there is no exist %c \n", row, col, resourceProject[pProject]);

                 fprintf(fp, "error：源文件 (%d,%d) 位置发生错误，16进制格式不对，there is no exist %c \n", row, col, resourceProject[pProject]);

                 fclose(fp);

             }

             while(IsDigit(resourceProject[pProject]) || IsLetter(resourceProject[pProject]))

             {

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

             }

         }

         else

         {

             while (IsDigit(resourceProject[pProject]) || resourceProject[pProject]=='.')

             {//后跟数字

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

             }//多读了一个字符既是下次将要开始的指针位置

             if((resourceProject[pProject]=='e'||resourceProject[pProject]=='E') && (resourceProject[pProject+1]=='+'||resourceProject[pProject+1]=='-'))

             {

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

                 flag = true;

             }

             while (IsDigit(resourceProject[pProject]))

             {//后跟数字

                 token[count++] = resourceProject[pProject];//收集

                 pProject++;

                 col++;

             }

         }

         if(flag0)

             return;

         token[count] = '\0';

         syn = 99;//常数种别码

         if (flag)

             syn = 101;

     }

     else if (ch == ';' || ch == '(' || ch == ')' || ch == '^'

         || ch == ',' || ch == '\"' || ch == '\'' || ch == '~' || ch == '#' || ch == '%' || ch == '['

         || ch == ']' || ch == '{' || ch == '}' || ch == '\\' || ch == '.' || ch == '\?' || ch == ':')

     {//若为运算符或者界符，查表得到结果

         token[0] = resourceProject[pProject];

         token[1] = '\0';//形成单字符串

         for (i = 0; i<36; i++)

         {//查运算符界符表

             if (strcmp(token, operatorOrDelimiter[i]) == 0)

             {

                 syn = 33 + i;//获得种别码，使用了一点技巧，使之呈线性映射

                 break;//查到即推出

             }

         }

         pProject++;//指针下移，为下一扫描做准备

         col++;

         return;

     }

     else if (resourceProject[pProject] == '+')

     {//+,+=,++

         pProject++;//后移，超前搜索

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 69;

         }

         else if (resourceProject[pProject] == '+')

         {

             syn = 74;

         }

         else

         {

             pProject--;

             col--;

             syn = 33;

         }

         pProject++;//指针下移

         col++;

         return;

     }

     else if (resourceProject[pProject] == '-')

     {//-,-=,--

         pProject++;//后移，超前搜索

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 70;

         }

         else if (resourceProject[pProject] == '-')

         {

             syn = 75;

         }

         else

         {

             pProject--;

             col--;

             syn = 34;

         }

         pProject++;//指针下移

         col++;

         return;

     }

     else if (resourceProject[pProject] == '\*')

     {//\*,\*=

         pProject++;//后移，超前搜索

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 71;

         }

         else

         {

             pProject--;

             col--;

             syn = 35;

         }

         pProject++;//指针下移

         col++;

         return;

     }

     else if (resourceProject[pProject] == '/')

     {///,/=,/\*

         pProject++;//后移，超前搜索

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 72;

         }

         else if(resourceProject[pProject] == '\*')

         {

             FILE \*fp;

             if ((fp = fopen("E:\\私人\\实验报告\\编译原理\\实验一\\error.txt", "a+")) == NULL)

             {//打开源程序

                 cout << "can't open this file";

                 exit(0);

             }

             printf("error：源文件 (%d,%d) 位置发生错误，注释未配对，there is no exist %c%c \n", row, col, resourceProject[pProject-1], resourceProject[pProject]);

             fprintf(fp, "error：源文件 (%d,%d) 位置发生错误，注释未配对，there is no exist %c%c \n", row, col, resourceProject[pProject-1], resourceProject[pProject]);

             fclose(fp);

             syn = 0;

         }

         else

         {

             pProject--;

             col--;

             syn = 36;

         }

         pProject++;//指针下移

         col++;

         return;

     }

     else if (resourceProject[pProject] == '<')

     {//<,<=,<<

         pProject++;//后移，超前搜索

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 38;

         }

         else if (resourceProject[pProject] == '<')

         {//左移

             syn = 58;

         }

         else

         {

             pProject--;

             col--;

             syn = 37;

         }

         pProject++;//指针下移

         col++;

         return;

     }

     else if (resourceProject[pProject] == '>')

     {//>,>=,>>

         pProject++;

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 40;

         }

         else if (resourceProject[pProject] == '>')

         {

             syn = 59;

         }

         else

         {

             pProject--;

             col--;

             syn = 39;

         }

         pProject++;

         col++;

         return;

     }

     else if (resourceProject[pProject] == '=')

     {//=.==

         pProject++;

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 42;

         }

         else

         {

             pProject--;

             col--;

             syn = 41;

         }

         pProject++;

         col++;

         return;

     }

     else if (resourceProject[pProject] == '!')

     {//!,!=

         pProject++;

         col++;

         if (resourceProject[pProject] == '=')

         {

             syn = 43;

         }

         else

         {

             syn = 68;

             pProject--;

             col--;

         }

         pProject++;

         col++;

         return;

     }

     else if (resourceProject[pProject] == '&')

     {//&,&&

         pProject++;

         col++;

         if (resourceProject[pProject] == '&')

         {

             syn = 53;

         }

         else if (resourceProject[pProject] == '=')

         {

             syn = 73;

         }

         else

         {

             pProject--;

             col--;

             syn = 52;

         }

         pProject++;

         col++;

         return;

     }

     else if (resourceProject[pProject] == '|')

     {//|,||

         pProject++;

         col++;

         if (resourceProject[pProject] == '|')

         {

             syn = 55;

         }

         else

         {

             pProject--;

             col--;

             syn = 54;

         }

         pProject++;

         col++;

         return;

     }

     else if (resourceProject[pProject] == '$')

     {//结束符

         syn = 0;//种别码为0

     }

     else

     {//不能被以上词法分析识别，则出错。

         FILE \*fp;

         if ((fp = fopen("E:\\私人\\实验报告\\编译原理\\实验一\\error.txt", "a+")) == NULL)

         {//打开源程序

             cout << "can't open this file";

             exit(0);

         }

         printf("error：源文件 (%d,%d) 位置发生错误，有非法字符，there is no exist %c \n", row, col, resourceProject[pProject]);

         fprintf(fp, "error：源文件 (%d,%d) 位置发生错误，有非法字符，there is no exist %c \n", row, col, resourceProject[pProject]);

         fclose(fp);

         pProject++;

         col++;

     }

}

int main()

{

     //打开一个文件，读取其中的源程序

     char resourceProject[10000];

     char token[20] = { 0 };

     int syn = -1, i;//初始化

     int pProject = 0;//源程序指针

     FILE \*fp, \*fp1;

     if ((fp = fopen("E:\\私人\\实验报告\\编译原理\\实验一\\testHard.c", "r")) == NULL)

     {//打开源程序

         cout << "can't open this file";

         exit(0);

     }

     resourceProject[pProject] = fgetc(fp);//读取一个字符

     while (resourceProject[pProject] != EOF)

     {//将源程序读入resourceProject[]数组

         pProject++;

         resourceProject[pProject] = fgetc(fp);

     }

     resourceProject[++pProject] = '\0';

     fclose(fp);

     cout << endl << "源程序为:" << endl;

     for (int i = 0, k = 1; i <= pProject;i++)

     {

         if(i==0 || resourceProject[i-1]=='\n')

         {

             cout << k++ << " ";

         }

         cout << resourceProject[i];

     }

     cout << endl;

     //对源程序进行过滤

     filterResource(resourceProject, pProject);

     cout << endl << "过滤之后的程序:" << endl;

     cout << resourceProject << endl;

     pProject = 0;//从头开始读

     if ((fp1 = fopen("E:\\私人\\实验报告\\编译原理\\实验一\\test\_compile.txt", "w+")) == NULL)

     {//打开源程序

         cout << "can't open this file";

         exit(0);

     }

     cout << endl;

     int num = 0;

     while (syn != 0)

     {

         //启动扫描

         syn = -1;

         GetToken(syn, resourceProject, token, pProject);

         if (syn == 100)

         {//判断是否在已有标识符表中，自己实现

             bool flag = false;

             for (int i = 0; i < num;i++)

             {

                 if(strcmp(token,IDentifierTbl[i]) == 0)

                 {

                     flag = true;

                     break;

                 }

             }

             if(!flag)

             {

                 for (int i = 0; i < strlen(token);i++)

                 {

                     IDentifierTbl[num][i] = token[i];

                 }

                 num++;

             }

             printf("(标识符 ,%s)\n", token);

             fprintf(fp1, "(标识符 , %s)\n", token);

         }

         else if (syn >= 1 && syn <= 32)

         {//保留字

             printf("(%s , 保留字)\n", reserveWord[syn - 1]);

             fprintf(fp1, "(%s , 保留字)\n", reserveWord[syn - 1]);

         }

         else if (syn == 99)

         {//const 常数

             printf("(常数 , %s)\n", token);

             fprintf(fp1, "(常数 , %s)\n", token);

         }

         else if (syn >= 33 && syn <= 74)

         {

             printf("(%s , 符号)\n", operatorOrDelimiter[syn - 33]);

             fprintf(fp1, "(%s , 符号)\n", operatorOrDelimiter[syn - 33]);

         }

         else if (syn == 101)

         {

             printf("(浮点数 , %s)\n", token);

             fprintf(fp1, "(浮点数 , %s)\n", token);

         }

         else if (syn == 102)

         {

             printf("(字符串常量 , %s)\n", token);

             fprintf(fp1, "(字符串常量 , %s)\n", token);

         }

     }

     cout << endl;

     for (i = 0; i<100; i++)

     {//插入标识符表中

         if(IDentifierTbl[i][0]=='\0')

         {

             break;

         }

         printf("第%d个标识符： %s\n", i + 1, IDentifierTbl[i]);

         fprintf(fp1, "第%d个标识符： %s\n", i + 1, IDentifierTbl[i]);

     }

     fclose(fp1);

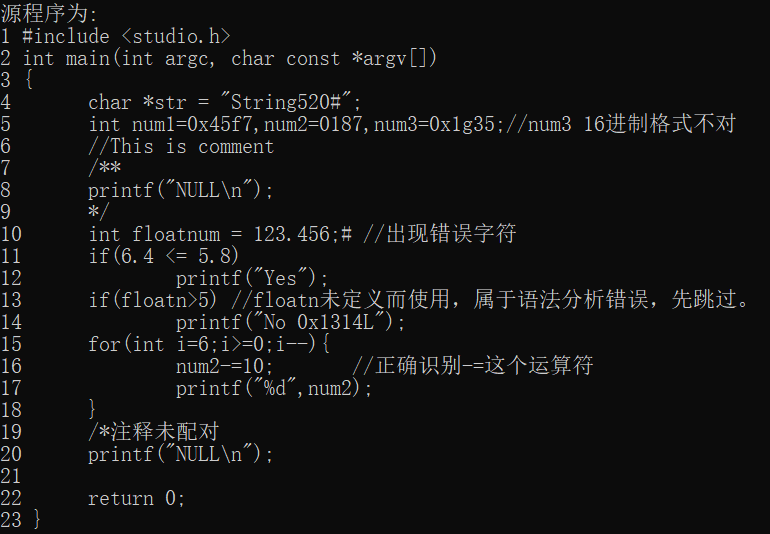
     return 0;

}

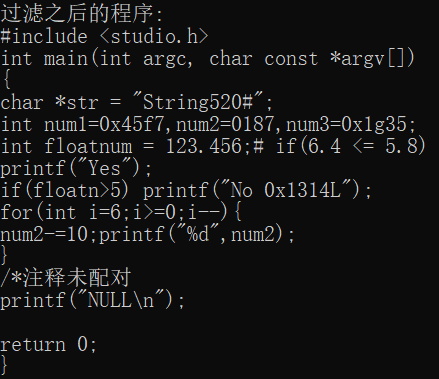
分析结果（截图）：

如：实现了1-3的所有功能，除附加题外。

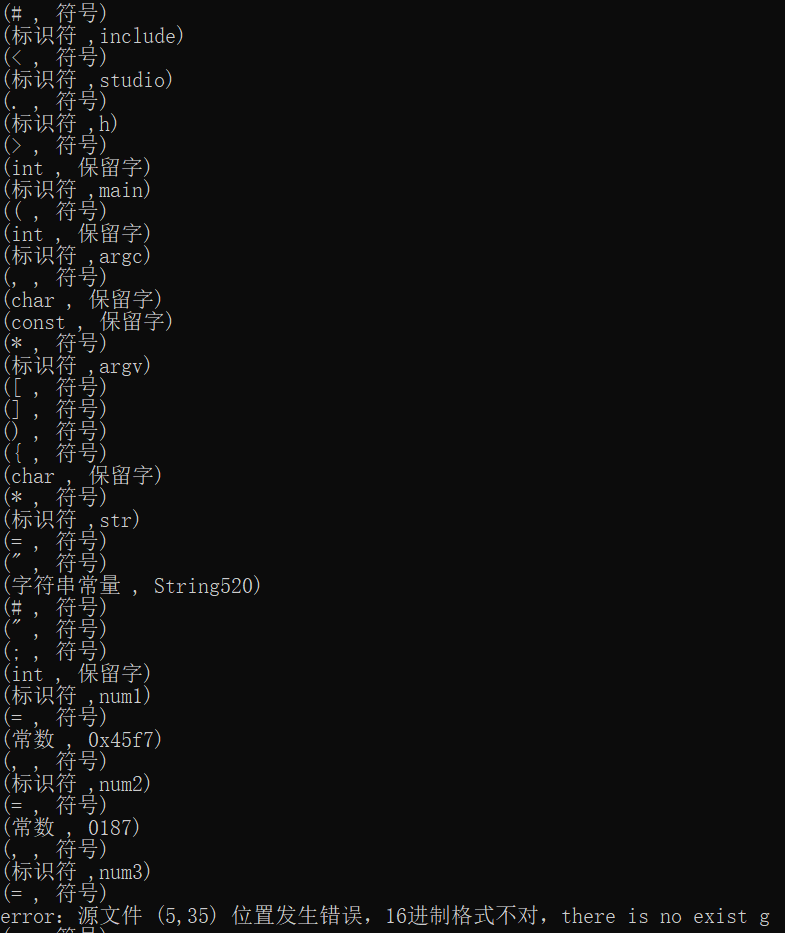
1. 读入乐园程序testHard.cpp，打印了源程序，并且在每行前面加上了行号和空格

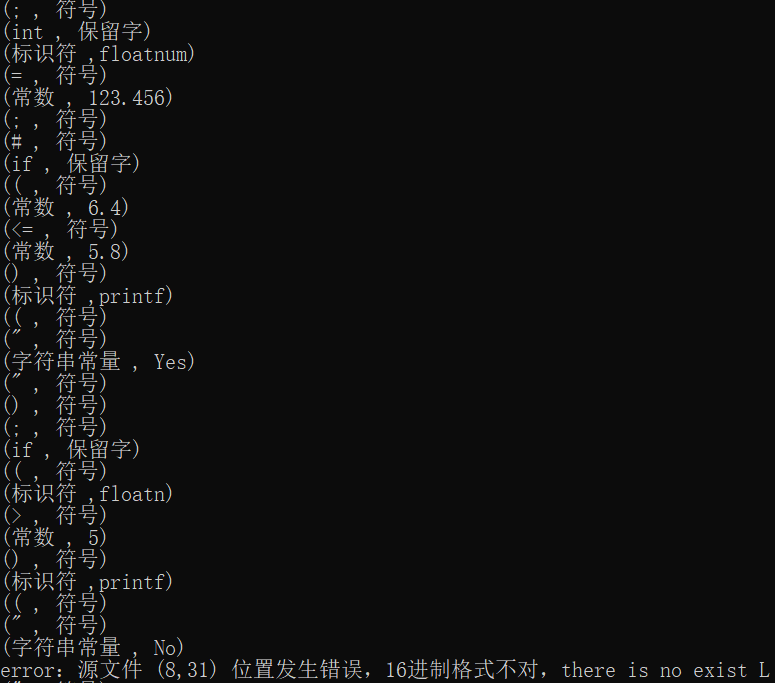


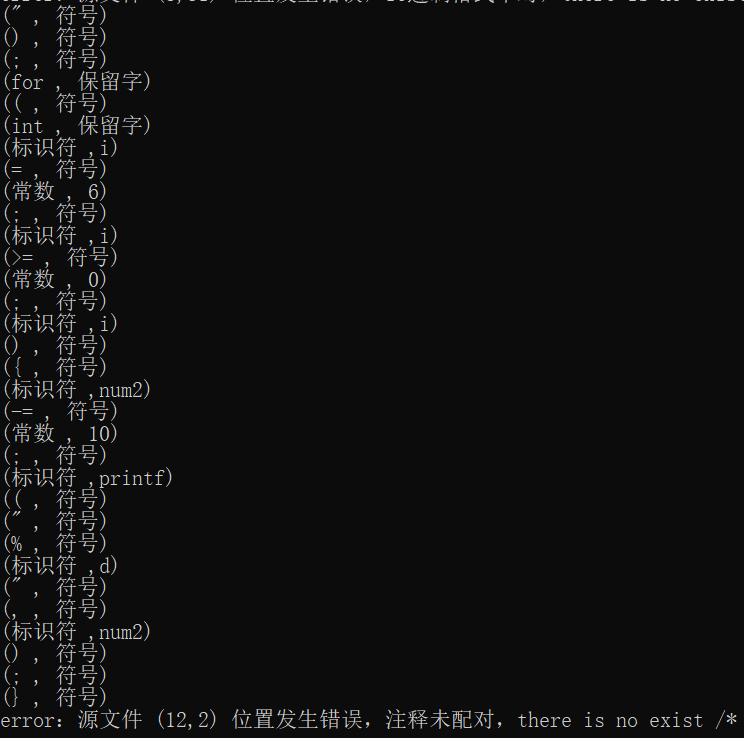
1. 对源程序进行过滤



1. 识别五类词法单元，并转换成二元式token形式逐行输出，实现GetToken函数



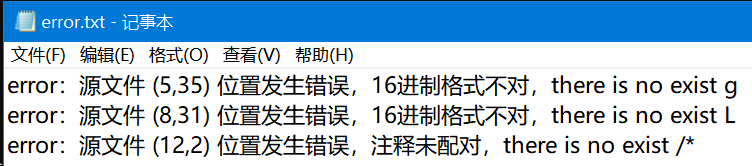




1. 识别源程序中的字符和字符串常量



1. 发现并定位错误



1. 识别所有进制整数



1. 识别所有十进制浮点数



1. 识别复合赋值运算符



1. 识别++、--等运算符
2. 输出所有用到的标识符

