TINY+ 词法分析程序实验

实验目的

通过扩充已有的样例语言TINY语言的词法分析程序，为扩展TINY语言TINY＋构造词法分析程序，从而掌握词法分析程序的构造方法。

实验内容

了解样例语言TINY及TINY编译器的实现，了解扩展TINY语言TINY＋，用C/C++语言在已有的TINY词法分析器基础上扩展，构造TINY＋的词法分析程序。

实验要求

将TINY＋源程序翻译成对应的TOKEN序列，并能检查一定的词法错误。

实验环境

- **Linux(Ubuntu)**
- g++
- make

TINY+语言

1. TOKEN序列表现形式：(Kind,Value)
   其中，Kind为词的种类，Value表示词的实际值。下面是Kind的种类：

KEY	SYM	ID	NUM	STR
关键字	特殊符号	标识符	数字常量	字符串常量

2. 关键字：

Tiny关键字	if then else end repeat until read write
Tiny+新增关键字	**Or and int bool string while do true false not**

   其中所有的关键字是程序设计语言保留使用的，并且用小写字母表示，用户自己定义的标识符不能和关键字重复。

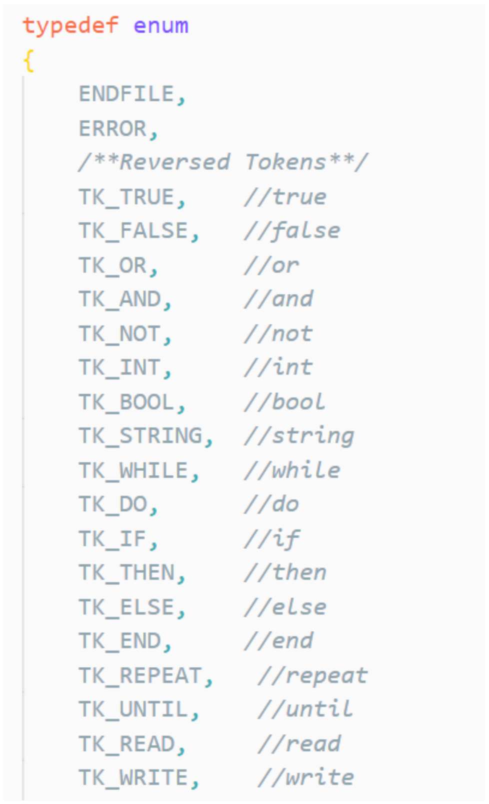
3. 特殊符号：

Tiny符号	{ } ; := + - * / ( ) < =
Tiny+新增符号	**> <= >= , '**

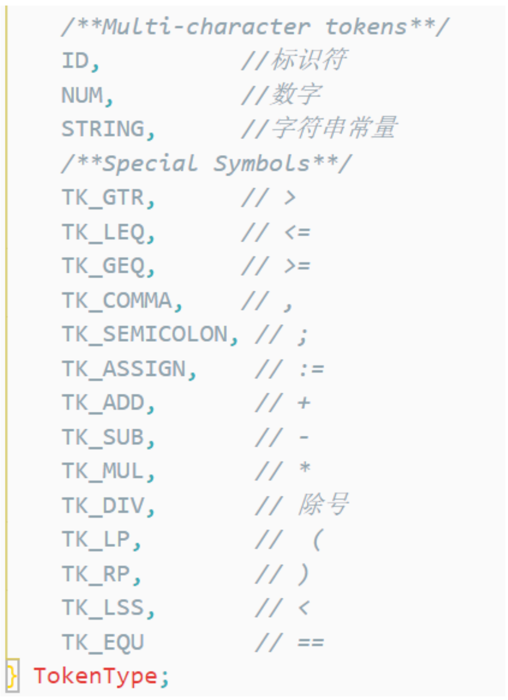
4. 词法分析
   - 词法要求
     - 关键词必须以字母开头
     - 字符不能包含在数字中
     - 注释用方括号括起来，不能嵌套，但可以包含不止一行
     - 字符串用单引号括起来
     - .....
   - 错误检查
     - ALPHA_AFTER_NUMBER_ERROR 字母紧接数字错误
     - ASSIGN_LEXICAL_ERROR 赋值符号没有打全错误
     - SINGLE_QUOTES_MISSING_FOR_STRING_ERROR 字符串缺失单引号错误
     - LEFT_BRACE_MISSING_FOR_COMMENTS_ERROR 注释左大括号缺失错误
     - RIGHT_BRACE_MISSING_FOR_COMMENTS_ERROR 注释右大括号缺失错误
     - ILLEGAL_CHARACTER非法字符
     - .....

实验过程

1. 在声明全局变量的global.h文件中枚举TOKEN类型，分别包括上面所述的五大种类，细分下来36种之多，如下图：

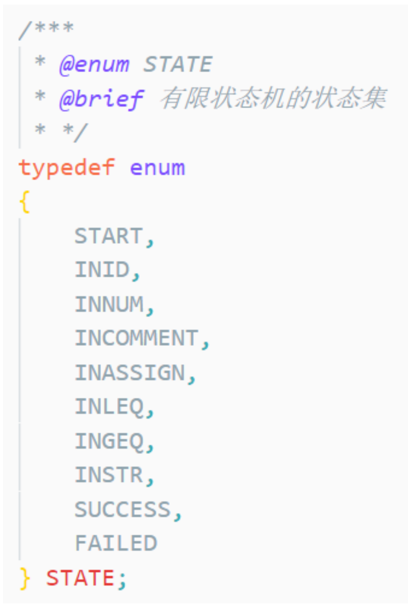


```
typedef enum
{
    ENDFILE,
    ERROR,
    /**Reversed Tokens**/
    TK_TRUE,     //true
    TK_FALSE,    //false
    TK_OR,       //or
    TK_AND,      //and
    TK_NOT,      //not
    TK_INT,      //int
    TK_BOOL,     //bool
    TK_STRING,   //string
    TK_WHILE,    //while
    TK_DO,       //do
    TK_IF,       //if
    TK_THEN,     //then
    TK_ELSE,     //else
    TK_END,      //end
    TK_REPEAT,    //repeat
    TK_UNTIL,     //until
    TK_READ,      //read
    TK_WRITE,     //write
```

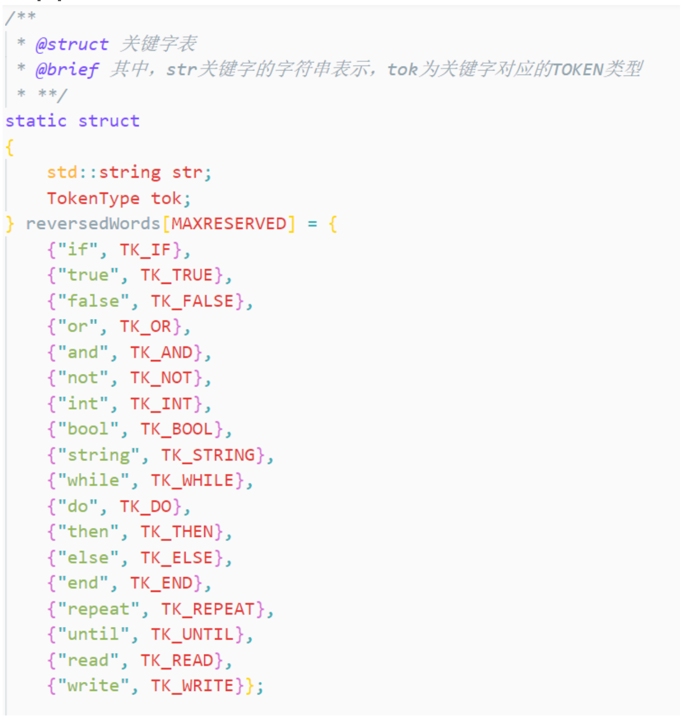


```
    /**Multi-character tokens**/
    ID,            //标识符
    NUM,           //数字
    STRING,        //字符串常量
    /**Special Symbols**/
    TK_GTR,        // >
    TK_LEQ,        // <=
    TK_GEQ,        // >=
    TK_COMMA,      // ,
    TK_SEMICOLON, // ;
    TK_ASSIGN,     // :=
    TK_ADD,        // +
    TK_SUB,        // -
    TK_MUL,        // *
    TK_DIV,        // 除号
    TK_LP,         //  (
    TK_RP,         // )
    TK_LSS,        // <
    TK_EQU         // ==
} TokenType;
```

2. 在词法分析程序实现文件scan.cpp中实现词法分析状态机以及关键字枚举等。如下图：



```
/***
 * @enum STATE
 * @brief 有限状态机的状态集
 * */
typedef enum
{
    START,
    INID,
    INNUM,
    INCOMMENT,
    INASSIGN,
    INLEQ,
    INGEQ,
    INSTR,
    SUCCESS,
    FAILED
} STATE;
```



```
/**
 * @struct 关键字表
 * @brief 其中，str关键字的字符串表示，tok为关键字对应的TOKEN类型
 * **/
static struct
{
    std::string str;
    TokenType tok;
} reversedWords[MAXRESERVED] = {
    {"if", TK_IF},
    {"true", TK_TRUE},
    {"false", TK_FALSE},
    {"or", TK_OR},
    {"and", TK_AND},
    {"not", TK_NOT},
    {"int", TK_INT},
    {"bool", TK_BOOL},
    {"string", TK_STRING},
    {"while", TK_WHILE},
    {"do", TK_DO},
    {"then", TK_THEN},
    {"else", TK_ELSE},
    {"end", TK_END},
    {"repeat", TK_REPEAT},
    {"until", TK_UNTIL},
    {"read", TK_READ},
    {"write", TK_WRITE}};
```

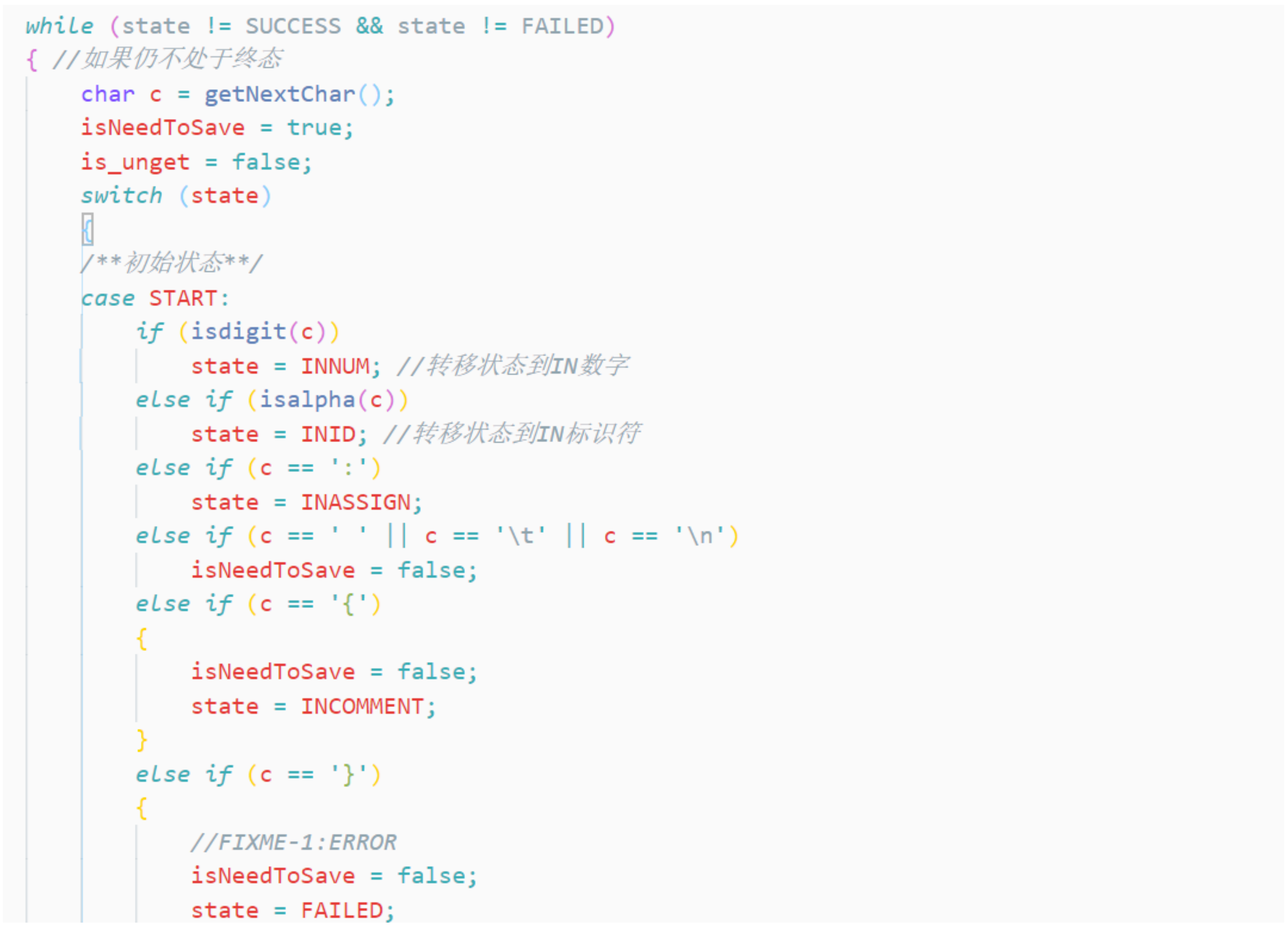
实现关键字表的查询函数，方便使用，其实就是遍历即可

```
/**
 * @brief 查询关键字表
 * @details 当获得了一个ID类型的token之后，还需要调用此函数进行判断其是不是关键字，如果是，就返回其对应的关键字TOKEN类型
 * @param 该ID类型token的字符串表示
 * @return 如果是关键字，返回对应的TOKEN类型；如果不是，直接返回ID即可。
 * **/
static TokenType reversedLookUp(const std::string &s)
{
    for (auto &i : reversedWords)
    {
        if (i.str == s)
            return i.tok;
    }
    return ID;
}
```

接下来就是实现词法分析最关键的函数getToken，读取行然后输出以pair作为表示的TOKEN序列，下面给出部分函数片段(详见scan.cpp文件)：
声明函数所需的中间变量：

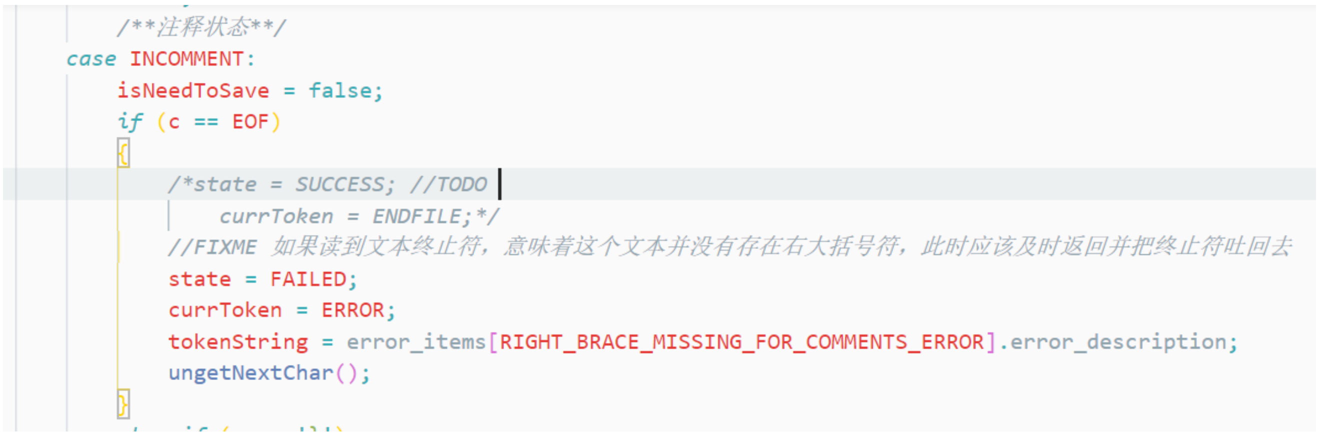
```
/***
 * @brief 获取TOKEN函数，词法分析的核心算法
 * @param ret_lineno out 用于返回值，token对应的行号
 * @return std::pair类型，其中第一个元素为读取到的TOKEN类型，第二个元素为token对应的字符串表示
 * */
std::pair<TokenType, std::string> getToken(int &ret_lineno)
{
    bool isNeedToSave;
    TokenType currToken;
    std::string tokenString;
    bool is_unget; //是否已经出现了回滚操作

    /**
     * 一开始处于START状态
     * @update 如果大括号栈中还有东西，应让它处于注释状态（错误处理）
     * **/
    STATE state = (brace_num_nested == 0 ? START : INCOMMENT);
```

循环只当词法分析状态为成功或者失败时跳出，且用一个switch将每种状态的情况罗列出来，分情况来实现：



```
while (state != SUCCESS && state != FAILED)
{ //如果仍不处于终态
    char c = getNextChar();
    isNeedToSave = true;
    is_unget = false;
    switch (state)
    {
    /**初始状态**/
    case START:
        if (isdigit(c))
            state = INNUM; //转移状态到IN数字
        else if (isalpha(c))
            state = INID; //转移状态到IN标识符
        else if (c == ':')
            state = INASSIGN;
        else if (c == ' ' || c == '\t' || c == '\n')
            isNeedToSave = false;
        else if (c == '{')
        {
            isNeedToSave = false;
            state = INCOMMENT;
        }
        else if (c == '}')
        {
            //FIXME-1:ERROR
            isNeedToSave = false;
            state = FAILED;
```

处理注释：



```
        /**注释状态**/
    case INCOMMENT:
        isNeedToSave = false;
        if (c == EOF)
        {
            /*state = SUCCESS; //TODO
              currToken = ENDFILE;*/
            //FIXME 如果读到文本终止符，意味着这个文本并没有存在右大括号符，此时应该及时返回并把终止符吐回去
            state = FAILED;
            currToken = ERROR;
            tokenString = error_items[RIGHT_BRACE_MISSING_FOR_COMMENTS_ERROR].error_description;
            ungetNextChar();
        }
```

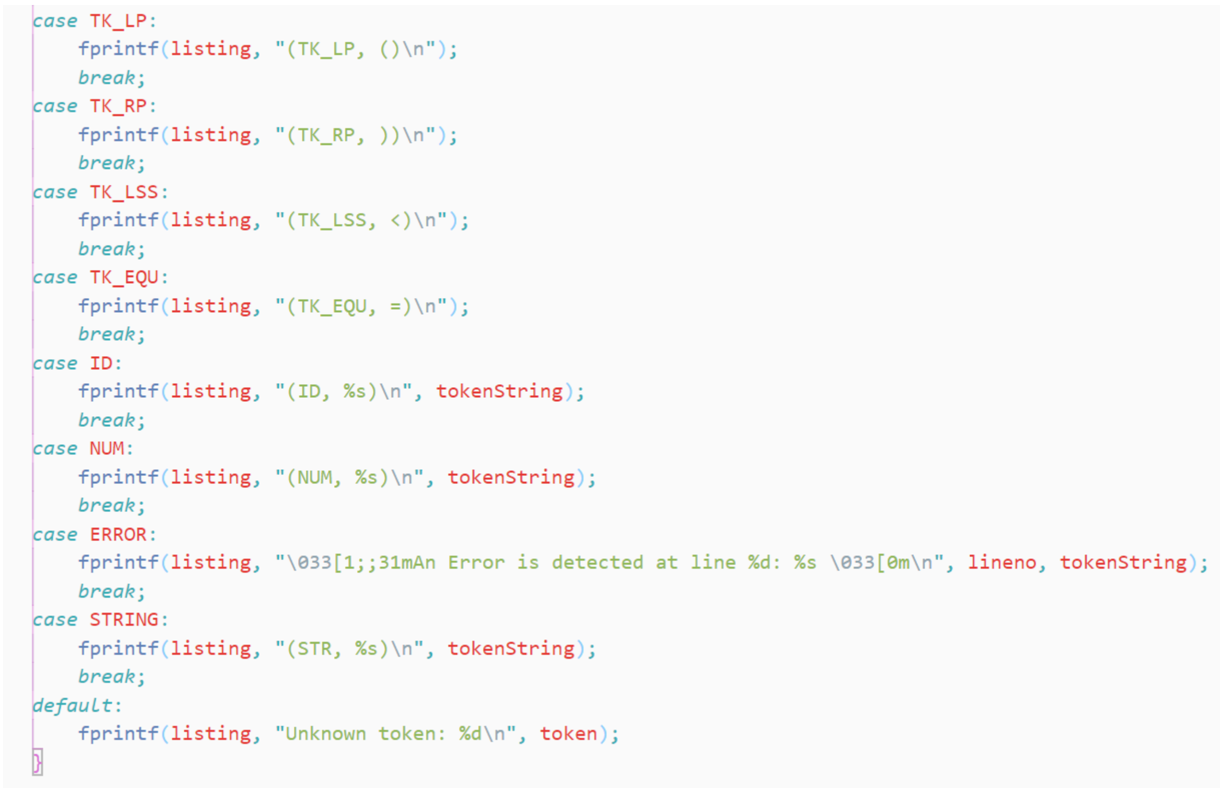
处理数字：

```
    /**数字状态**/
case INNUM:
    if (isalpha(c))
    { //当数字紧接字母的时候，是错误的
        ungetNextChar();
        is_unget = true;
        isNeedToSave = false;

        state = FAILED;
        currToken = ERROR;
        tokenString = error_items[0].error_description;
    }
    else if (!isdigit(c))
    {
        state = SUCCESS;
        currToken = NUM;
        ungetNextChar();
        is_unget = true;
    }
    break;
```

还有很多代码，此处不一一展示，详见scan.cpp，重要部分均给出注释。

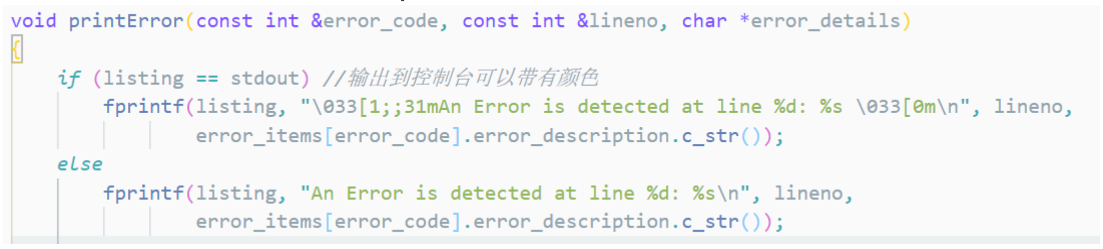
3. 在输出TOKEN序列实现函数的print.cpp文件中，实现printToken函数来输出Token序列，实现也不难，由于函数较长，仅展示部分如下(详见print.cpp)：

```
void printToken(TokenType token, const char *tokenString, const int &lineno)
{
    switch (token)
    {
    case TK_TRUE:
    case TK_FALSE:
    case TK_OR:
    case TK_AND:
    case TK_NOT:
    case TK_INT:
    case TK_BOOL:
    case TK_STRING:
    case TK_WHILE:
    case TK_DO:
    case TK_IF:
    case TK_THEN:
    case TK_ELSE:
    case TK_END:
    case TK_REPEAT:
    case TK_UNTIL:
    case TK_READ:
    case TK_WRITE:
        fprintf(listing, "(KEY, %s)\n", tokenString);
        break;
    case TK_GTR:
        fprintf(listing, "(TK_GTR, >)\n");
        break;
```



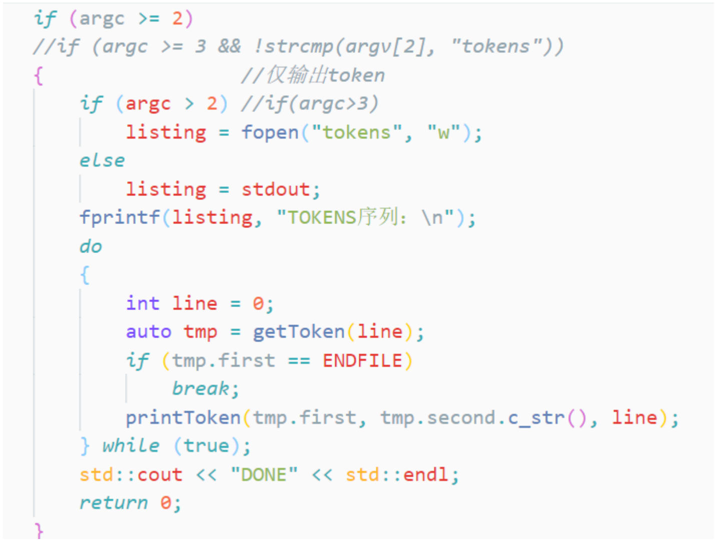
```
case TK_LP:
    fprintf(listing, "(TK_LP, ()\n");
    break;
case TK_RP:
    fprintf(listing, "(TK_RP, ))\n");
    break;
case TK_LSS:
    fprintf(listing, "(TK_LSS, <)\n");
    break;
case TK_EQU:
    fprintf(listing, "(TK_EQU, =)\n");
    break;
case ID:
    fprintf(listing, "(ID, %s)\n", tokenString);
    break;
case NUM:
    fprintf(listing, "(NUM, %s)\n", tokenString);
    break;
case ERROR:
    fprintf(listing, "\033[1;;31mAn Error is detected at line %d: %s \033[0m\n", lineno, tokenString);
    break;
case STRING:
    fprintf(listing, "(STR, %s)\n", tokenString);
    break;
default:
    fprintf(listing, "Unknown token: %d\n", token);
}
```

同时，定义输出error的函数printError，可以将error的输出弄成红色。



```
void printError(const int &error_code, const int &lineno, char *error_details)
{
    if (listing == stdout) //输出到控制台可以带有颜色
        fprintf(listing, "\033[1;;31mAn Error is detected at line %d: %s \033[0m\n", lineno,
                error_items[error_code].error_description.c_str());
    else
        fprintf(listing, "An Error is detected at line %d: %s\n", lineno,
                error_items[error_code].error_description.c_str());
```

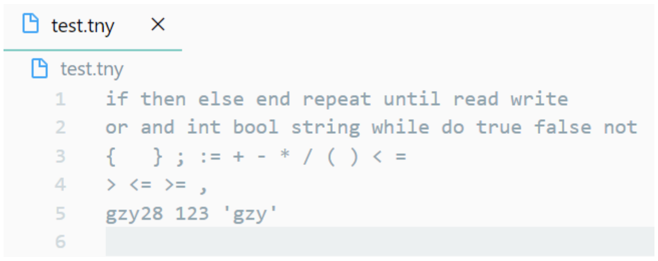
4. main中处理输出方式，支持输出到控制台(屏幕)或文件中。



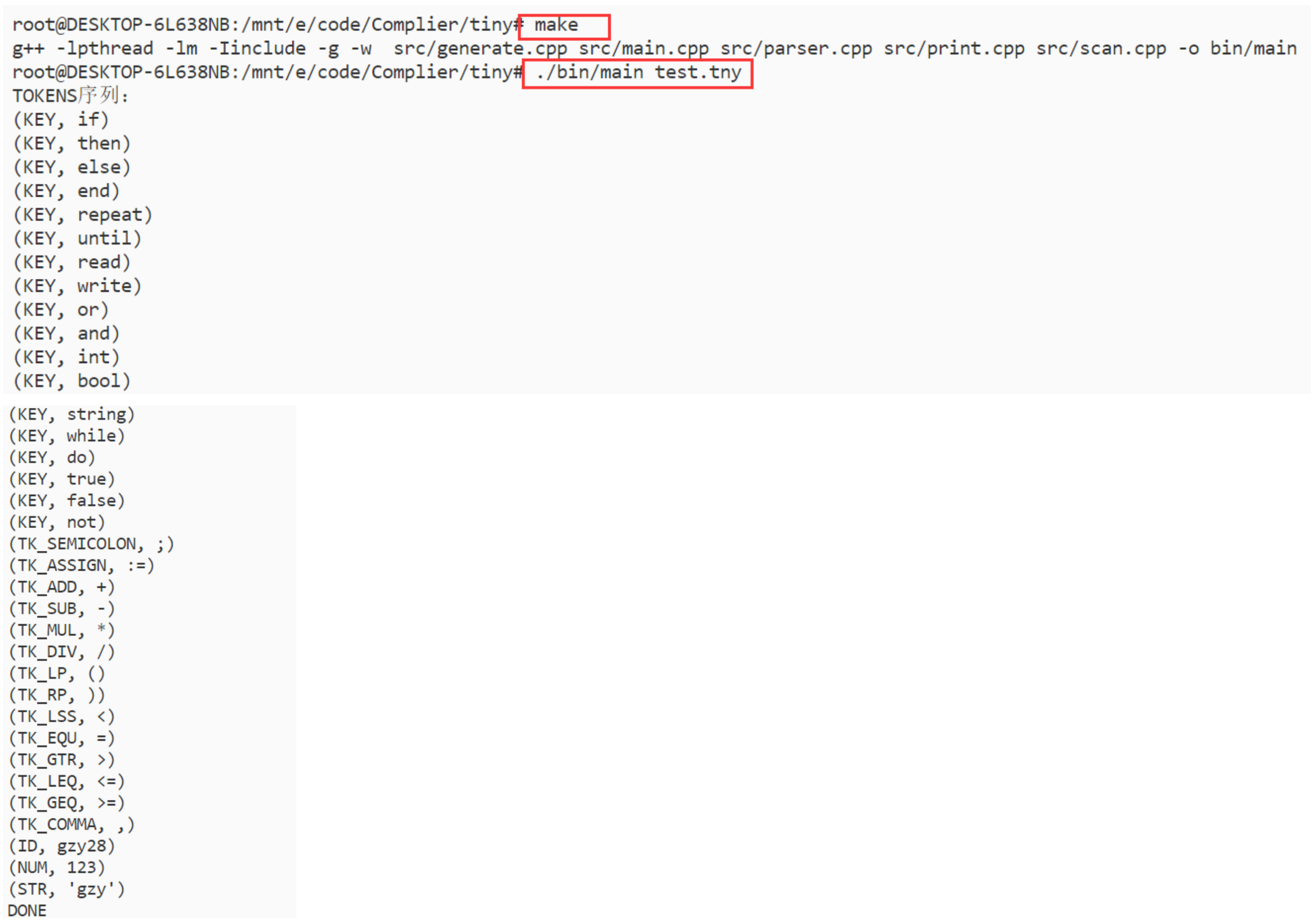
```
if (argc >= 2)
//if (argc >= 3 && !strcmp(argv[2], "tokens"))
{                   //仅输出token
    if (argc > 2) //if(argc>3)
        listing = fopen("tokens", "w");
    else
        listing = stdout;
    fprintf(listing, "TOKENS序列: \n");
    do
    {
        int line = 0;
        auto tmp = getToken(line);
        if (tmp.first == ENDFILE)
            break;
        printToken(tmp.first, tmp.second.c_str(), line);
    } while (true);
    std::cout << "DONE" << std::endl;
    return 0;
}
```

实验测试结果展示

- 输入make得到可执行程序main，并在命令行输入参数运行，测试Tiny源程序以及输出TOKEN序列如下图（测试test.tny）：



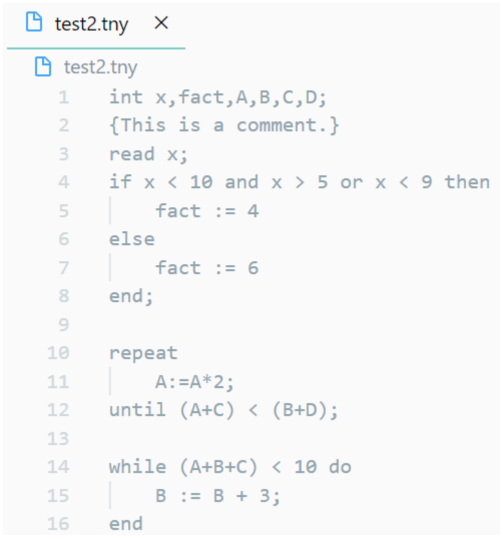
```
test.tny
if then else end repeat until read write
or and int bool string while do true false not
{   } ; := + - * / ( ) < =
> <= >= ,
gzy28 123 'gzy'
```



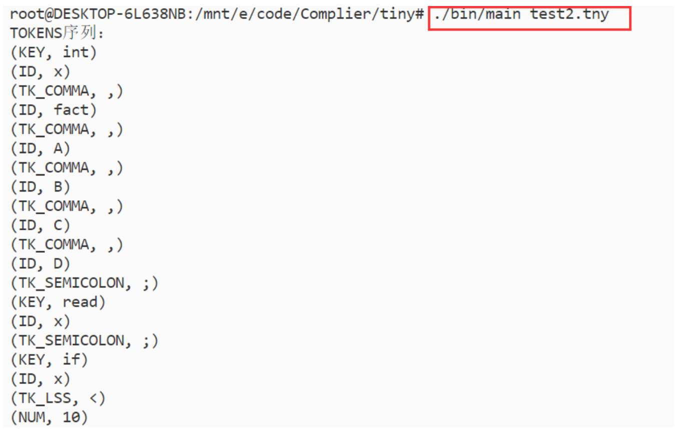
```
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# make
g++ -lpthread -lm -Iinclude -g -w  src/generate.cpp src/main.cpp src/parser.cpp src/print.cpp src/scan.cpp -o bin/main
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# ./bin/main test.tny
TOKENS序列:
(KEY, if)
(KEY, then)
(KEY, else)
(KEY, end)
(KEY, repeat)
(KEY, until)
(KEY, read)
(KEY, write)
(KEY, or)
(KEY, and)
(KEY, int)
(KEY, bool)
(KEY, string)
(KEY, while)
(KEY, do)
(KEY, true)
(KEY, false)
(KEY, not)
(TK_SEMICOLON, ;)
(TK_ASSIGN, :=)
(TK_ADD, +)
(TK_SUB, -)
(TK_MUL, *)
(TK_DIV, /)
(TK_LP, ()
(TK_RP, ))
(TK_LSS, <)
(TK_EQU, =)
(TK_GTR, >)
(TK_LEQ, <=)
(TK_GEQ, >=)
(TK_COMMA, ,)
(ID, gzy28)
(NUM, 123)
(STR, 'gzy')
DONE
```

可以看到每种Tiny原来的关键字或特殊符号或字符串等以及新增的均能被识别出来。

- 接下来测试正常的Tiny程序，如下图（test2.tny）：



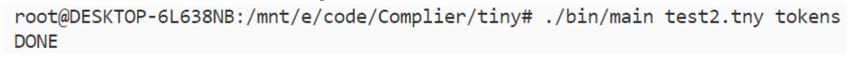
```
test2.tny
int x,fact,A,B,C,D;
{This is a comment.}
read x;
if x < 10 and x > 5 or x < 9 then
    fact := 4
else
    fact := 6
end;

repeat
    A:=A*2;
until (A+C) < (B+D);

while (A+B+C) < 10 do
    B := B + 3;
end
```



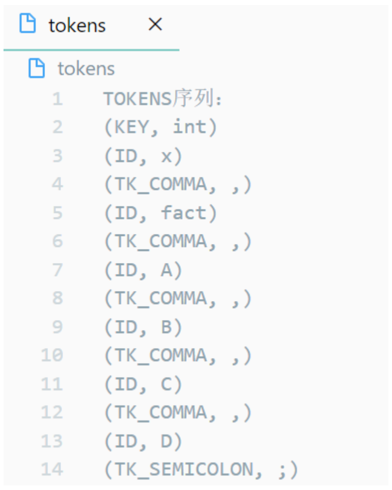
```
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# ./bin/main test2.tny
TOKENS序列:
(KEY, int)
(ID, x)
(TK_COMMA, ,)
(ID, fact)
(TK_COMMA, ,)
(ID, A)
(TK_COMMA, ,)
(ID, B)
(TK_COMMA, ,)
(ID, C)
(TK_COMMA, ,)
(ID, D)
(TK_SEMICOLON, ;)
(KEY, read)
(ID, x)
(TK_SEMICOLON, ;)
(KEY, if)
(ID, x)
(TK_LSS, <)
(NUM, 10)
```

此处由于输出序列较长仅展示一部分，详见tokens文件。

- 在./bin/main test2.tny 加上 tokens 可以将结果输出到一个名为tokens的文件。



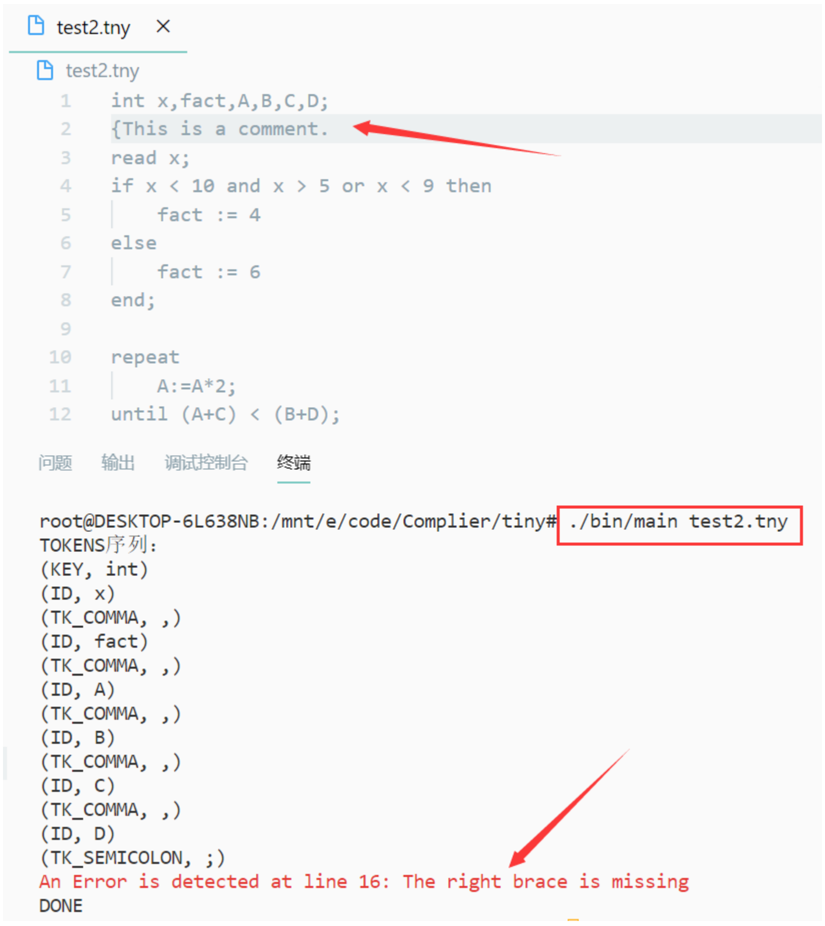
```
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# ./bin/main test2.tny tokens
DONE
```



```
TOKENS序列:
(KEY, int)
(ID, x)
(TK_COMMA, ,)
(ID, fact)
(TK_COMMA, ,)
(ID, A)
(TK_COMMA, ,)
(ID, B)
(TK_COMMA, ,)
(ID, C)
(TK_COMMA, ,)
(ID, D)
(TK_SEMICOLON, ;)
```

(仅展示部分，详见具体文件)

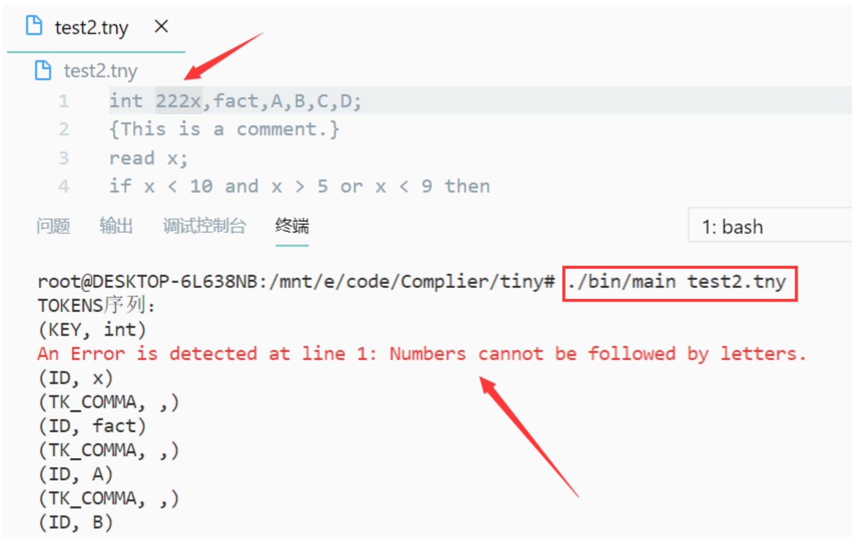
- 测试词法错误
  注释括号未打全:



```
int x,fact,A,B,C,D;
{This is a comment.
read x;
if x < 10 and x > 5 or x < 9 then
    fact := 4
else
    fact := 6
end;

repeat
    A:=A*2;
until (A+C) < (B+D);
```

```
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# ./bin/main test2.tny
TOKENS序列:
(KEY, int)
(ID, x)
(TK_COMMA, ,)
(ID, fact)
(TK_COMMA, ,)
(ID, A)
(TK_COMMA, ,)
(ID, B)
(TK_COMMA, ,)
(ID, C)
(TK_COMMA, ,)
(ID, D)
(TK_SEMICOLON, ;)
An Error is detected at line 16: The right brace is missing
DONE
```

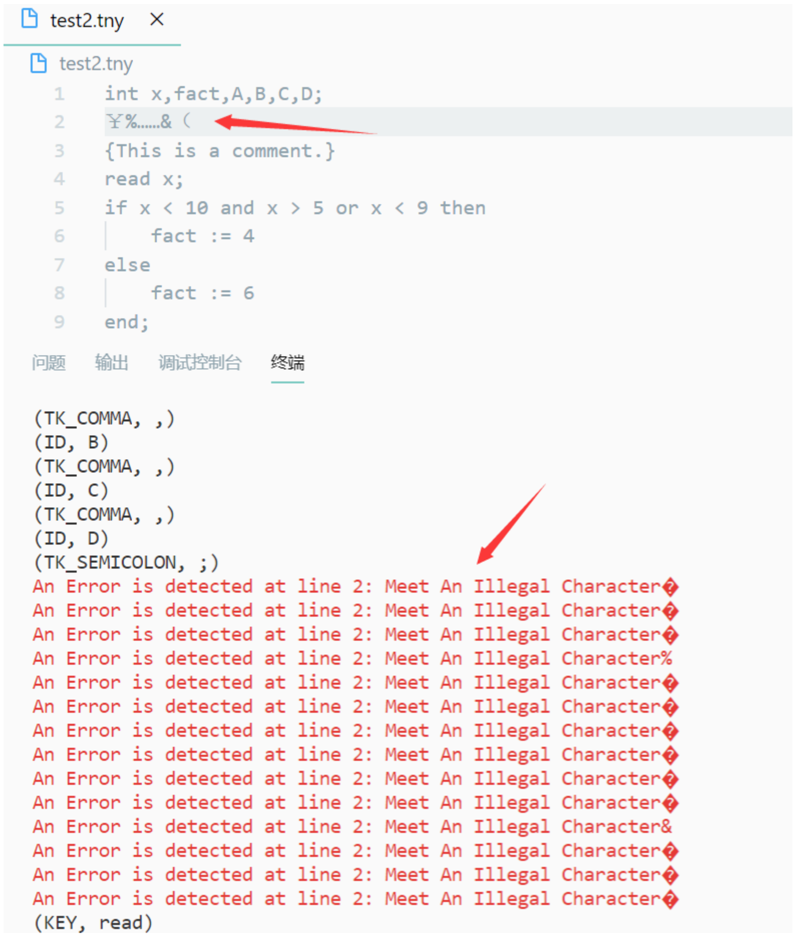
变量命名数字在字母前面:



```
int 222x,fact,A,B,C,D;
{This is a comment.}
read x;
if x < 10 and x > 5 or x < 9 then
```

```
root@DESKTOP-6L638NB:/mnt/e/code/Complier/tiny# ./bin/main test2.tny
TOKENS序列:
(KEY, int)
An Error is detected at line 1: Numbers cannot be followed by letters.
(ID, x)
(TK_COMMA, ,)
(ID, fact)
(TK_COMMA, ,)
(ID, A)
(TK_COMMA, ,)
(ID, B)
```

出现非法字符：



```
int x,fact,A,B,C,D;
￥%……&（
{This is a comment.}
read x;
if x < 10 and x > 5 or x < 9 then
    fact := 4
else
    fact := 6
end;
```

```
(TK_COMMA, ,)
(ID, B)
(TK_COMMA, ,)
(ID, C)
(TK_COMMA, ,)
(ID, D)
(TK_SEMICOLON, ;)
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character%
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character&
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
An Error is detected at line 2: Meet An Illegal Character�
(KEY, read)
```

本词法分析程序还能检查字符串 ' 缺失以及赋值符号缺失等等问题，此处就不一一展示，略占篇幅，欢迎尝试。

心得体会

词法分析程序实现的难点还是在于getToken函数的实现，根据读取到的token，查询关键字表判断是否为关键字然后确定其类型，进行输出。其次就是词法分析错误的判断，在errors中声明了多种错误，然后在getToken中进行相应的判断，当注释的左 { 大括号时，应该判断接下来的token中有没有右大括号，若直至终止符都没有右大括号，那么应该输出错误。
经过这次实验，我对词法分析的实现有了更深的理解，并实现了从课堂理论知识到实际应用实现的映射，获益匪浅。