```c
// 文件: scanner.h
// 内容: 声明词法分析器的对外接口( C 语言版 )
// 作者: WXQ#2018

#ifndef _SCANNER_H
#define _SCANNER_H

/*                关于输入中的存在错误之设计
 * 1. 词法分析阶段只能识别到“无效字符”、“无效单词”这2种错误。
 * 2. 对于这两种错误，词法分析器向调用者返回的记号之种类统一填写为"ERRTOKEN".
 * 3. 对于“无效字符”，还会被词法分析内部直接丢弃，但将该字符的16进制值
 *    存放到记号的文本中，以便调用者得到。
 * 4. 所有错误，需要调用者（通常是语法分析器）来处理。
 */

//
// 词法分析器类中的类型定义与常量声明
//

enum Token_Type   // 记号种类
{
    COMMENT, // 用于表驱动型分析器，可识别行注释
    ID,      // 用于表驱动型分析器，可识别下述所有单词

    ORIGIN, SCALE,   ROT, IS,  TO,         // 保留字
    STEP,   DRAW,    FOR,    FROM,         // 保留字
    T,                                     // 参数

    SEMICO, L_BRACKET, R_BRACKET, COMMA,  // 分隔符号
    PLUS, MINUS, MUL, DIV, POWER,         // 运算符
    FUNC,                          // 函数
    CONST_ID,                      // 常数

    NONTOKEN,                      // 空记号
    ERRTOKEN                       // 出错记号
};

typedef enum Token_Type Token_Type;

// 有效记号的最大长度，超长部分将被丢弃.
#define  TOKEN_LEN_MAX  20 // 与被分析语言相关

struct position  // 记号所在位置的数据结构
{
    unsigned int line; // 行号
    unsigned int col;  // 列号
};

typedef double (*t_func)(double);

struct Token   // 记号的数据结构
{
    Token_Type  type;                       // 记号的类别
    char        lexeme[TOKEN_LEN_MAX+1];    // 构成记号的字符串
    double      value;                      // 若为常数，则是常数的值
    t_func      FuncPtr;                    // 若为函数，则是函数的指针
    // double    (*FuncPtr)(double);  该声明与上行等价


    struct position   where;                // 记号在输入中的位置
};

```

```
63
64 //
65 //  Part II  操作接口
66 //
67
68 // 初始化词法分析器，成功时返回非0，失败返回0
69 int InitScanner(const char* fileName);
70
71 // 识别并返回一个记号。
72 // 遇到非法输入时 .type=ERRTOKEN、文件结束时 .type=NONTOKEN
73 struct Token GetToken();
74
75 // 关闭词法分析器
76 void CloseScanner();
77
78 #endif
79 // end of file
80
```