函数绘图语言的解释器-参考实现（C 语言）

0. 目录组织直观地反映了模块划分：

　　scanner　 　词法分析器的全部代码

　　parser　　　语法分析器的全部代码，仅完成了语法分析

　　semantics　语义分析（计算），应用于在 parser 代码中，以构成“解释器”

　　errlog　　　错误处理，信息输出

　　ui\_main　　人机界面+主程序，包括两种实现：

　　　　　　　　(a) 文本交互界面(console), 主函数是 main()

　　　　　　　　(b) 图形界面(WinGUI)，主函数是 WinMain()

1. 词法分析器（目录 scanner）

~~~~~~~~

1.1 源文件 & 编译

scanner.h 词法分析器对外提供的接口声明

scanner.c 词法分析器的实现

dfa.c 定义了 表驱动型 DFA 及其操作，共 scanner.c 使用

scannermain.c 词法分析测试主程序，仅测试词法分析时使用.

\*\* 编译生成可执行的词法分析测试程序时，在你的开发工具中，应建立一个 \

\*\* 控制台项目(Win32 Console Application)。

1.2 采用“表驱动”模式实现，定义在文件 dfa.c 中。

（1）将 DFA 的使用抽象为三个接口操作，从而将实现细节封装起来。

　　　当你要构造“直接编码型”词法分析器时，仅需要修改这三个函数的实现即可。

（2）将表驱动型 DFA 的状态转换矩阵当作“稀疏矩阵”，并采用了基于三元组的“压缩”存储策略。

1.3 部分程序基本沿用了《习题解答》中的代码，但 struct Token 的结构中，

记号文本的存储采用了“直接存储”，而不是《解答》中的“间接”存储。

\*\* 这样就可以将 记号 对象存储到 语法树上。

1.4 错误处理策略

(1) 当打开文件失败时，仅通过 InitScanner() 的返回值向调用者报告。

(2) 对于词法分析发现的错误，统一由 GetToken() 返回一个种类为 ERRTOKEN 的记号，

而该错误的显示、处理均交由调用者负责。

词法分析测试主程序中，凡是输出记号类别为 ERRTOKEN 的都是这些错误。

2. 语法分析器

~~~~~~~~

2.1 源文件 & 编译

parsermain.c 语法分析测试主程序，仅测试语法分析时使用.

|\_\_\_ 该文件已失效，因为其代码以移动到 ui\_main/console.c 中。

parser.h 语法分析器对外提供的接口声明

parser.c 语法分析器的实现

\*\* 编译生成可执行的语法分析测试程序时，在你的开发工具中，应建立一个

控制台项目(Win32 Console Application)。

编译时，需加上

词法分析的 scanner/scanner.c , scanner/dfa.c ，

错误处理 errlog/errlog.c

基于控制台界面的例程和主程序 ui\_main/console.c

2.2 实现模式: 这里实现的是‘递归下降的语法分析器’。

2.3 关于语法树

按题目要求，仅为各类表达式构造语法树，且一个语句分析结束后，

该语句对应的全部语法树会被销毁（参见 statement 的递归下降函数）。

2.4 程序输出

错误信息输出：写入文件 error.log，也会显示到屏幕上。

其他信息输出：写入文件 parser.log ，并不输出到屏幕上。

2.5 错误处理策略

但凡发现语法错误、词法错误，则显示错误信息之后，立即结束程序（很简单，也很粗暴）。

3. 语义分析与解释器

~~~~~~~~~~~~

这个实现将 语法分析、语义分析与绘图操作交织在一起，所以该模块

定义了必要的语义子程序 semantics.c，并对 parser/parser.c 中，

四个语句（\*Statement）对应的递归下降子程序体中，增加了语义计算操作。

3.1 源文件

-- 下面两个文件是语义计算的辅助程序，为 parser.c 服务。

semantics.h 语义计算的辅助函数声明

semantics.c 语义计算函数的实现

-- 下面两个文件是语法分析器的，需要配合语义分析进行修改：

parser.h 语法分析器对外、对语义分析提供的接口声明

parser.c {添加了语义动作}的语法分析器，它 #include "semantics.h"

\*\* 编译生成可执行的能绘图的解释器程序时，在你的开发工具中，应建立一个

(Win32 Application)项目，不是控制台应用项目。

编译时，需加上

词法分析的 scanner/scanner.c , scanner/dfa.c ，

错误处理 errlog/errlog.c

基于控制台的界面例程和主程序 ui\_main/console.c （这个不能绘图）

OR 基于窗口的界面例程和主程序 ui\_main/WinGUI.c （这个 能绘图）

【 ！！！ 提 示 ！！！ 】

编译时需在编译器的配置中增加宏定义 "\_PARSER\_WITH\_SEMANTICS"，

否则将没有语义计算，图形界面无法执行。

3.2 程序输出

错误信息输出：写入文件 error.log。

其他信息输出：写入文件 parser.log ，并不输出到屏幕上。

3.3 错误处理策略

但凡发现语法错误、词法错误，则显示错误信息之后，立即结束程序（很简单，也很粗暴）。