**深 圳 大 学 实 验 报 告**

**课程名称：­ 编译原理**

**实验项目名称： 词法分析程序设计**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 张昊迪**

**报告人：缪克达 学号：2020274045 班级： 二班**

**实验报告提交时间： 2023年4月6日星期四**

**教务处制**

1. 实验内容

* TINY 语言的词法由 TINY Syntax.ppt 描述；
* TINY 语言的词法分析器由 TINY Scanner.rar 的 C 语言代码实现；
* TINY+语言的词法由 TINY+ Syntax.doc 描述。

任务：理解 TINY 语言的词法及词法分析器的实现，并基于该词法分析器， 实现拓展语言 TINY+的词法分析器。

要求 :

1. TINY+词法分析器以 TINY+源代码为输入，输出为识别出的 token 序 列；
2. 词法分析器以最长匹配为原则，例如‘:=’应识别为赋值符号而非单 独的‘：’及‘=’；
3. Token 以（种别码，属性值）表示，包含以下类型的种别码：
4. KEY 为关键字；
5. SYM 为系统特殊字符；
6. ID 为变量；
7. NUM 为数值常量；
8. STR 为字符串常量。
9. 识别词法错误。词法分析器可以给出词法错误的行号并打印出对应的 出错消息，主要包含以下类型的词法错误：
10. 非法字符。即不属于 TINY+字母表的字符，比如$就是一个非法字符；
11. 字符串匹配错误，比如右部引号丢失，如‘scanner
12. 注释的右部括号丢失或匹配错误，如 {this is an example
13. 实验要求

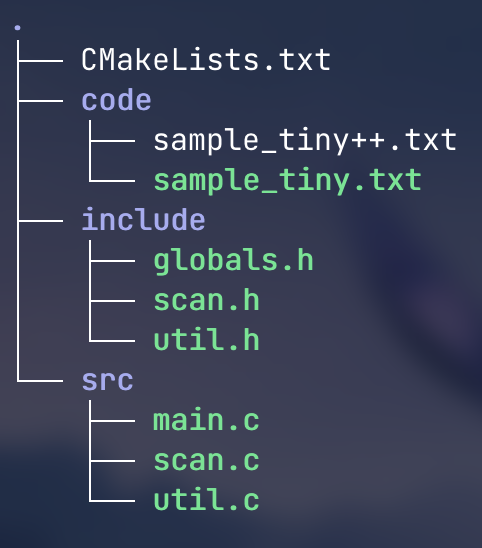
* 完成 TINY+词法分析程序的编写及测试（使用提供的测试代码或自己编写测 试文件）；
* 使用实验所提供的模板撰写实验报告，要求内容详实，有具体的设计描述、 关键的代码片段、及实验结果屏幕截图；
* 在截止日期前将代码、实验报告、测试文件（如有）等所有实验相关文件压 缩到一个压缩包姓名\_学号\_实验二.zip 上传至 Blackboard

1. 实验步骤与结果
   * + 1. 配置环境

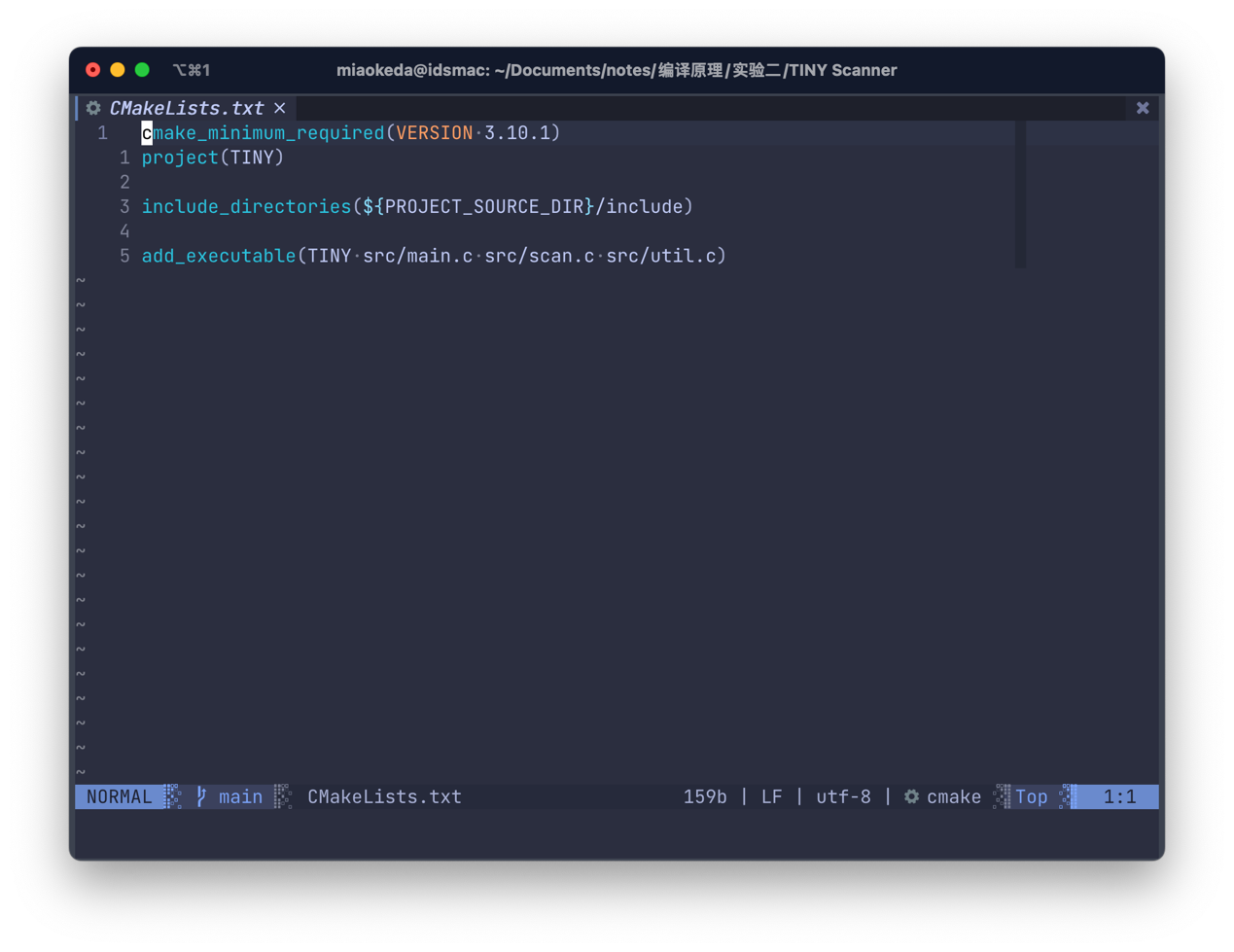
将给定的TINY程序：



按照C++的工程规范，建立项目文件夹，文件树如下：



其中CMakeLists.txt配置了C++的跨平台环境，因为只使用系统头文件和include文件夹下的头文件，所以CMakeLists.txt的内容如下：



其中解释如下：

cmake\_minimum\_required(VERSION 3.10.1)：指定 CMake 最低版本要求，这里要求版本为 3.10.1 或更高版本。

project(TINY)：定义 CMake 项目名称为 TINY。

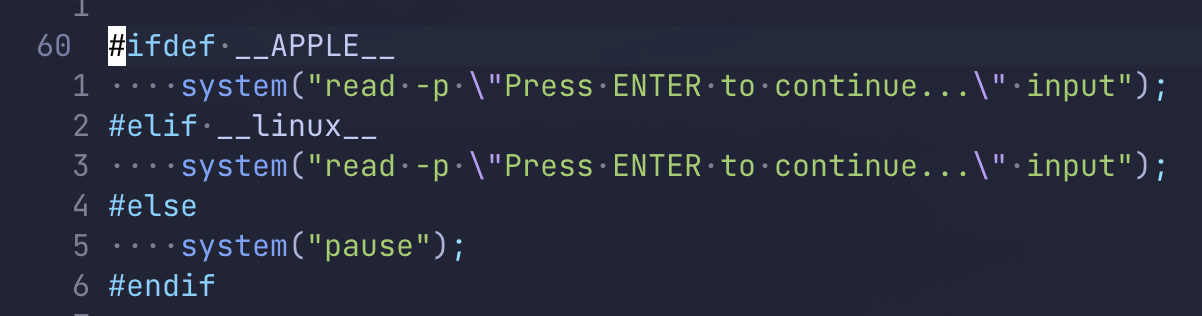
include\_directories(${PROJECT\_SOURCE\_DIR}/include)：将 ${PROJECT\_SOURCE\_DIR}/include 目录添加到编译器的头文件搜索路径中，以便在源代码中使用 #include <header.h> 这样的语句引入头文件。

add\_executable(TINY src/main.c src/scan.c src/util.c)：指定生成可执行文件 TINY，由 src/main.c、src/scan.c、src/util.c 三个源文件编译而成。

总之，这段代码的作用是为 TINY 项目生成一个 Makefile 文件，以便使用 make 命令编译、链接和生成可执行文件。

再构建之前，我们需要修改main.c，这是因为这个程序是针对Windows平台编写的，里面用到了cmd.exe的指令pause，这在Unix和类Unix（如Linux）上是没有的，需要用指令read来代替。

首先进入main.c，然后增加宏定义的判断：

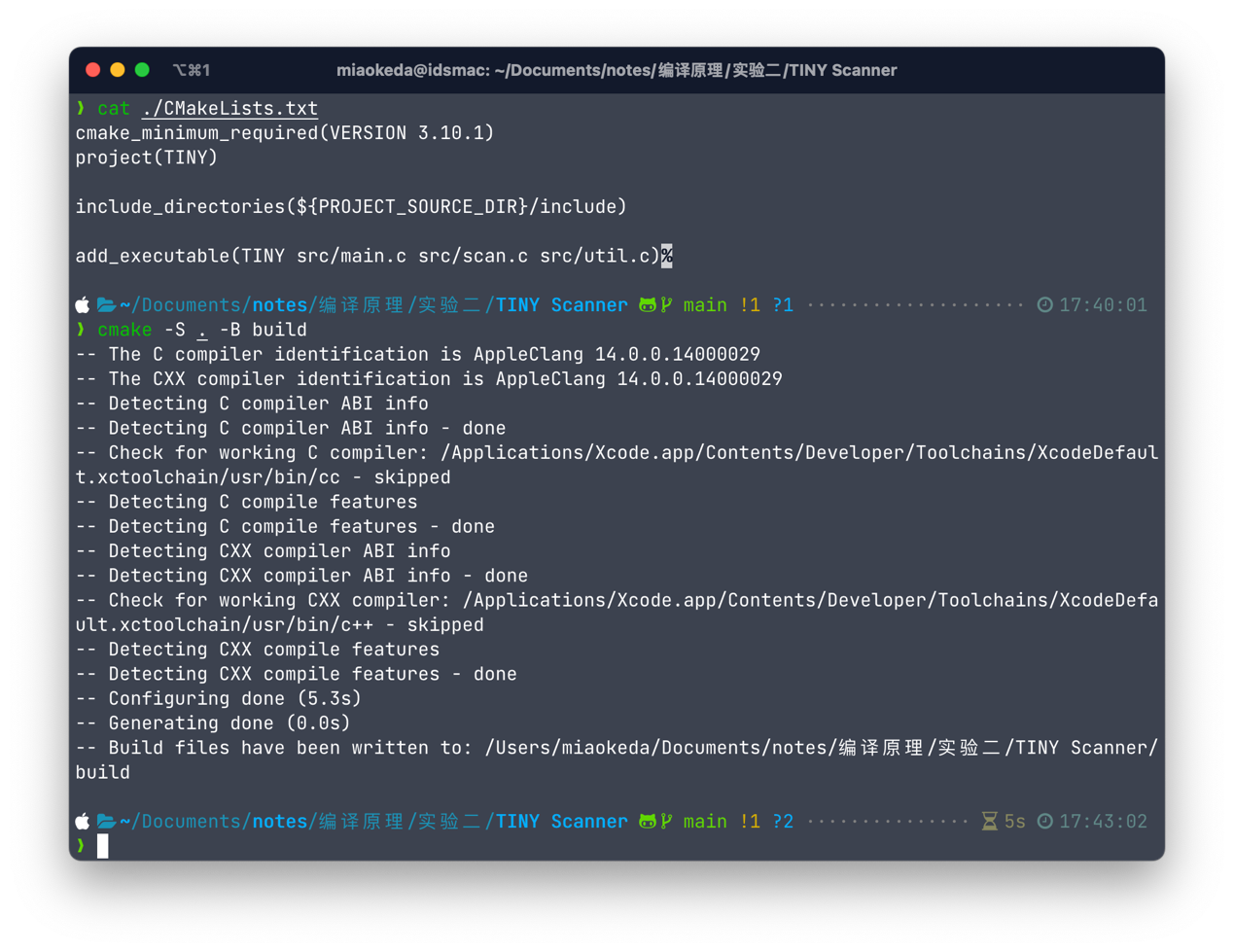


这段代码，通过判断宏定义来判断使用的系统，如果是MacOS或者Linux的任意发行版，那么就会执行read -p “Press ENTER to continue..” input来等待用户回车结束程序，这个指令等效于windows中cmd的pause指令。

这样我们的程序就可以在MacOS和Linux下正确运行了。

* + - 1. 构建

接下来我们构建一下这个项目，首先需要在根目录下输入cmake -S . -B build来用cmake生成Makefile：



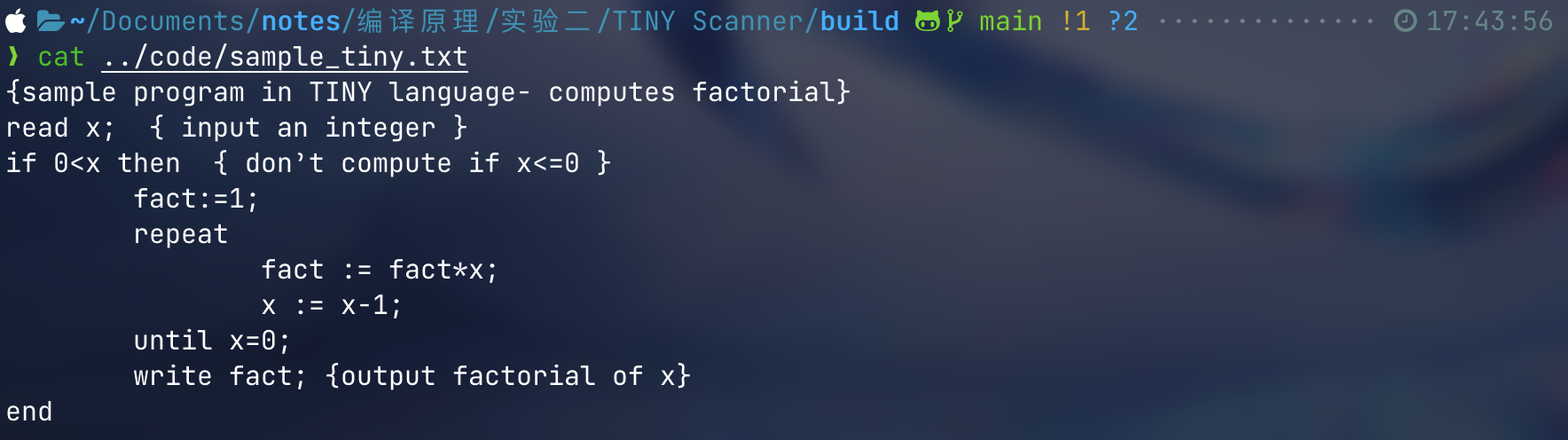
接下来我们进入build文件夹，然后输入make即可开始构建：

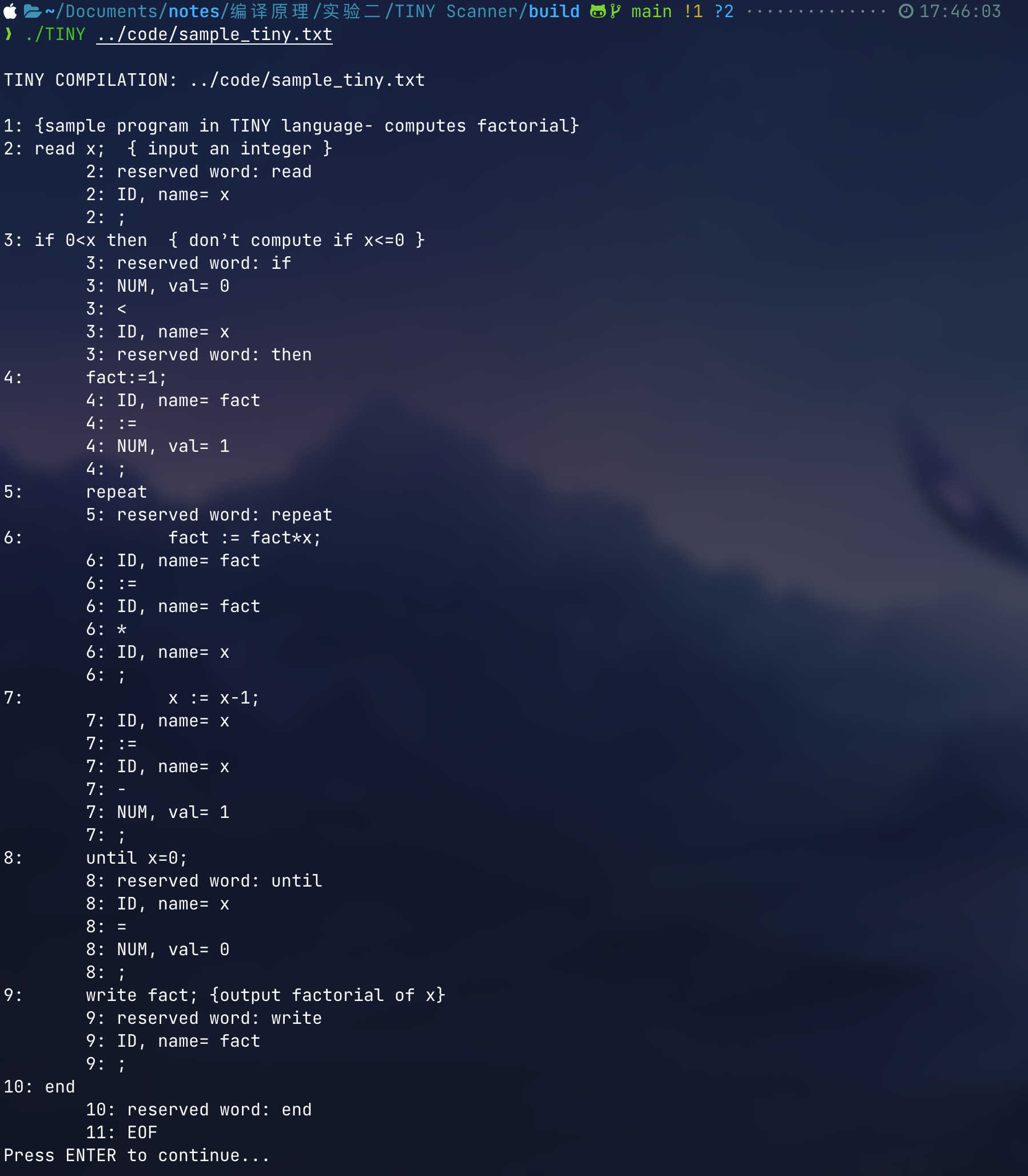


忽略warning，可以发现构建成功。

接下来，我们使用构建好的TINY程序，来进行基本的TINY程序语法分析：

首先准备TINY程序样例：



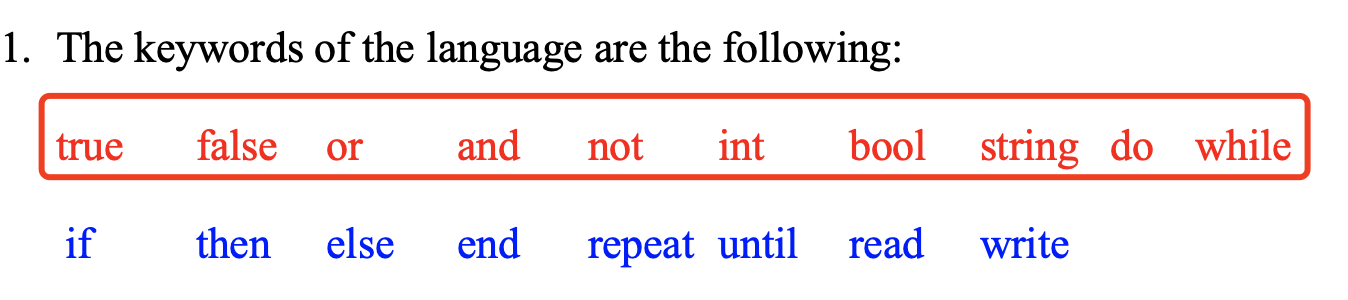
接下来输入./TINY ../code/sample\_tiny.txt：  


可以发现我们的程序可以正常运行。

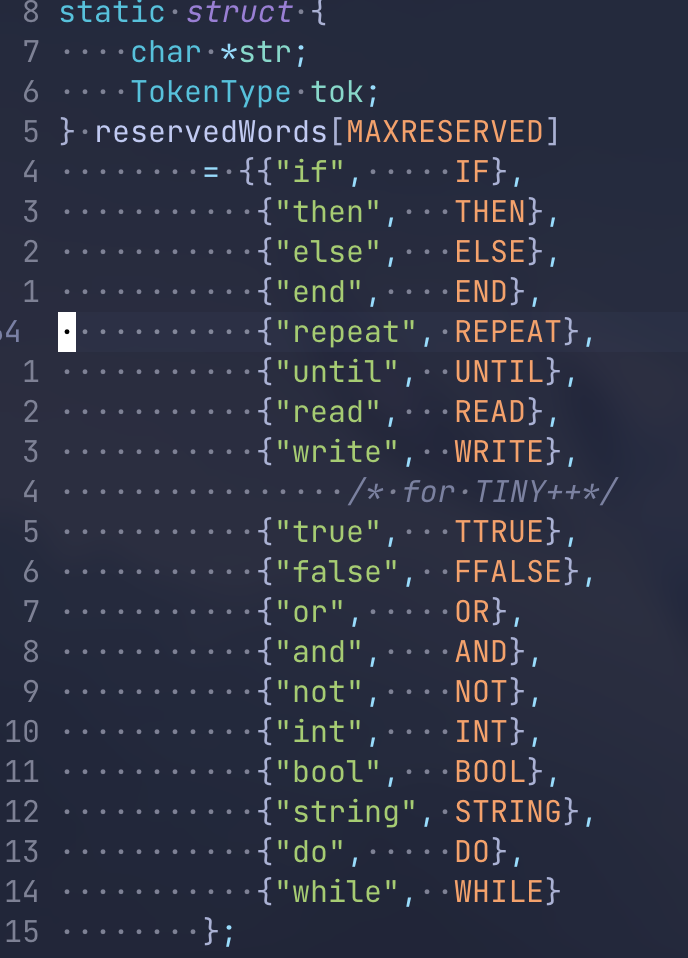
* + - 1. 修改程序，使得可以分析tiny++的语法

1. 添加新的关键字

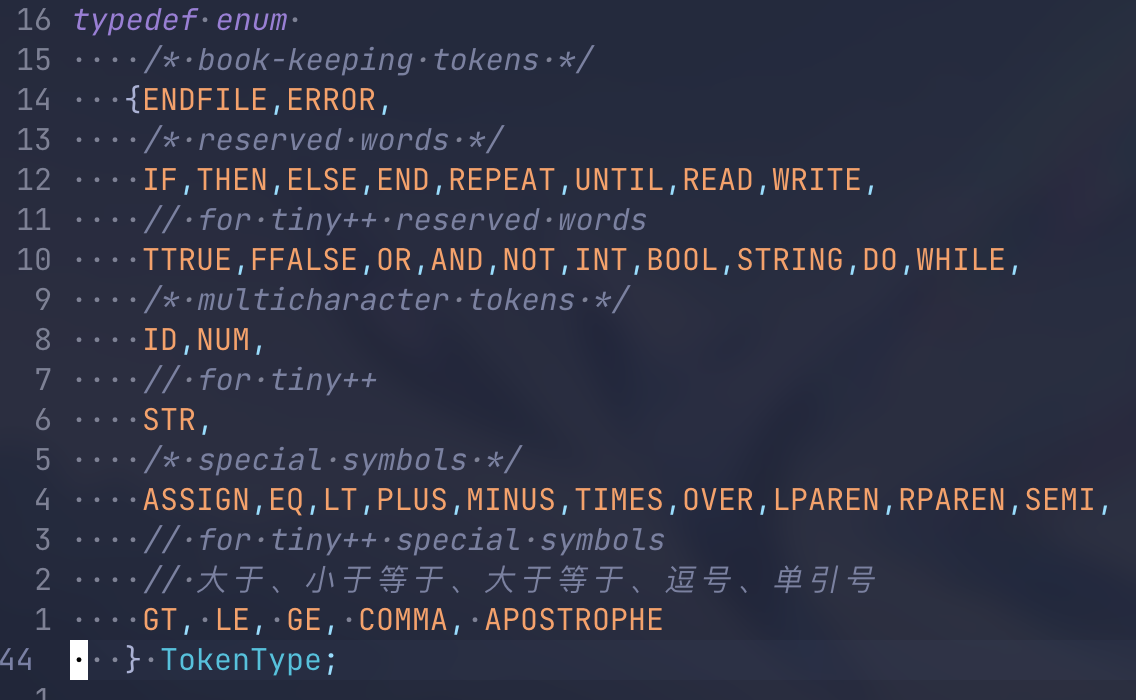
根据文档，我们需要添加如下的新关键字：



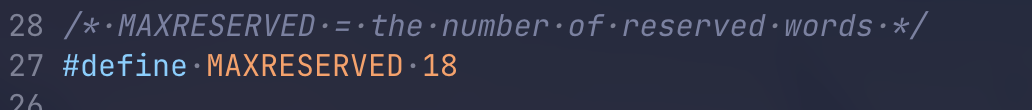
所以，我们首先在scan.c的reversedWords结构体添加如下的键值对：



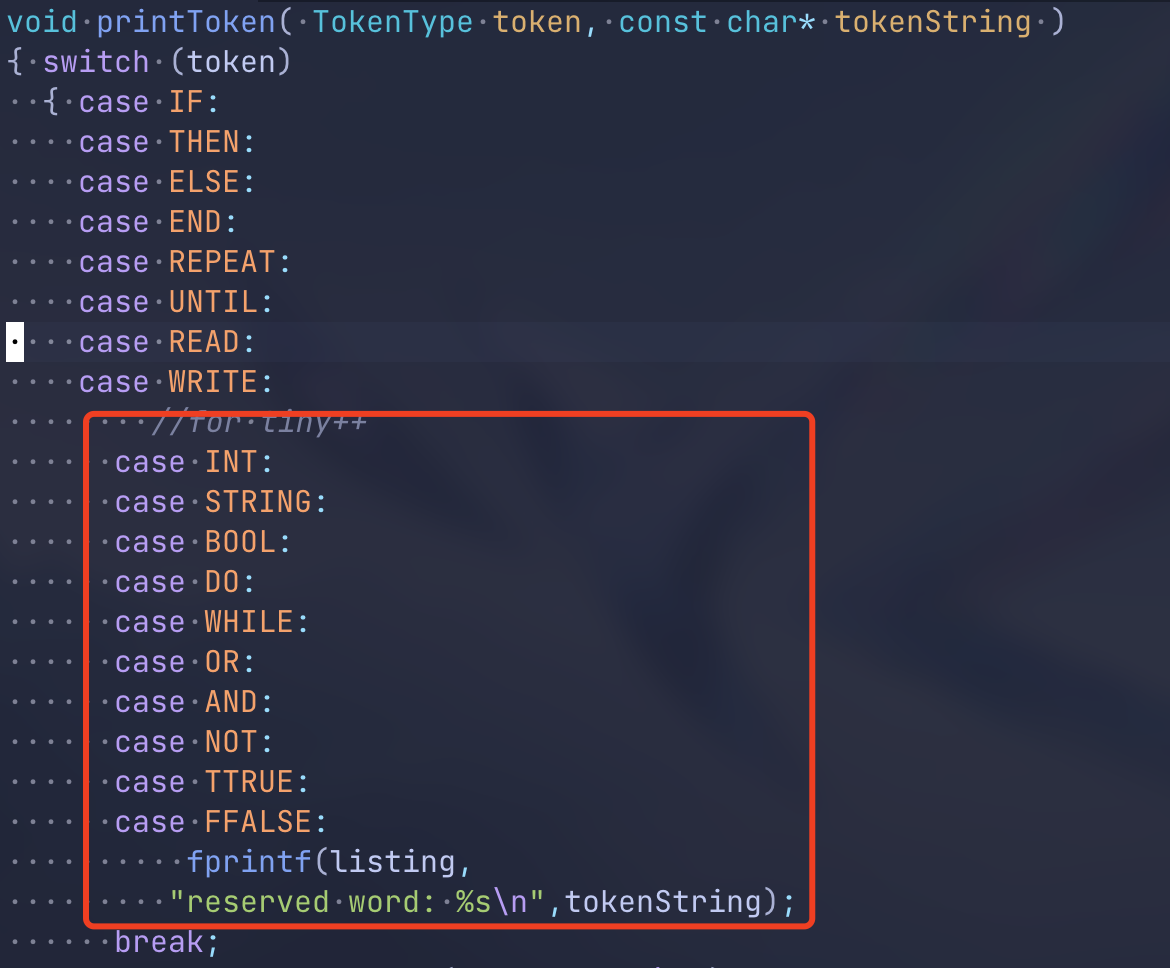
其中值在global.h中定义：



需要注意的是，由于MAXRESERVED的长度为8，而我们对该键值对进行了扩充，所以需要修改这个长度为18，在global.h修改即可：



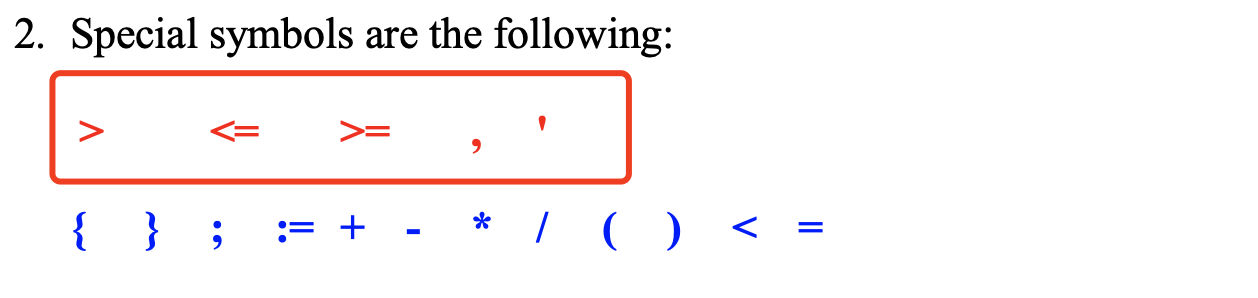
最后在util.c的printToken中添加新的关键字的token输出：



至此我们完成了对新关键字的语法识别。

1. 特殊字符的识别

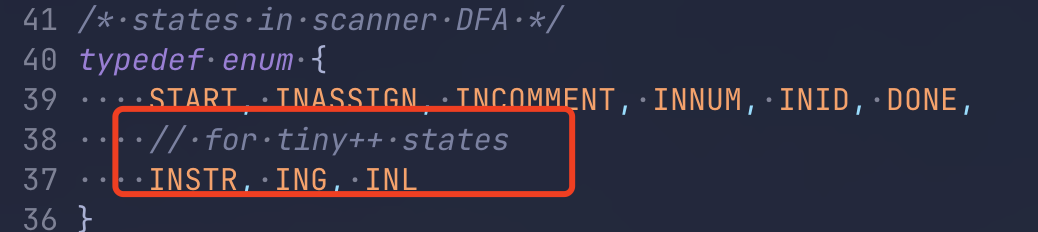
根据文档，我们需要添加如下的字符识别：



首先在global.h定义这些新特殊字符的TokenTytpe：



由于单引号表示字符串，所以我们还得定义一个新的状态INSTR，表示当前的DFA进入了INSTR状态，即字符串的状态；同时，因为存在大于等于和小于等于，所以当DFA识别到了大于号或者小于号，我们还需要在检测后一位字符是不是等于符号，所以我们还需要额外定义两个状态INL和ING，分别是小于状态和大于状态，在scan.c的枚举类型中添加：



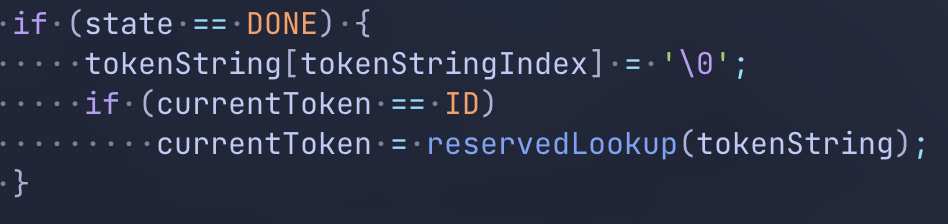
然后在scan.c中添加对这些字符识别，当识别到了’、>和<，进入对应的状态：



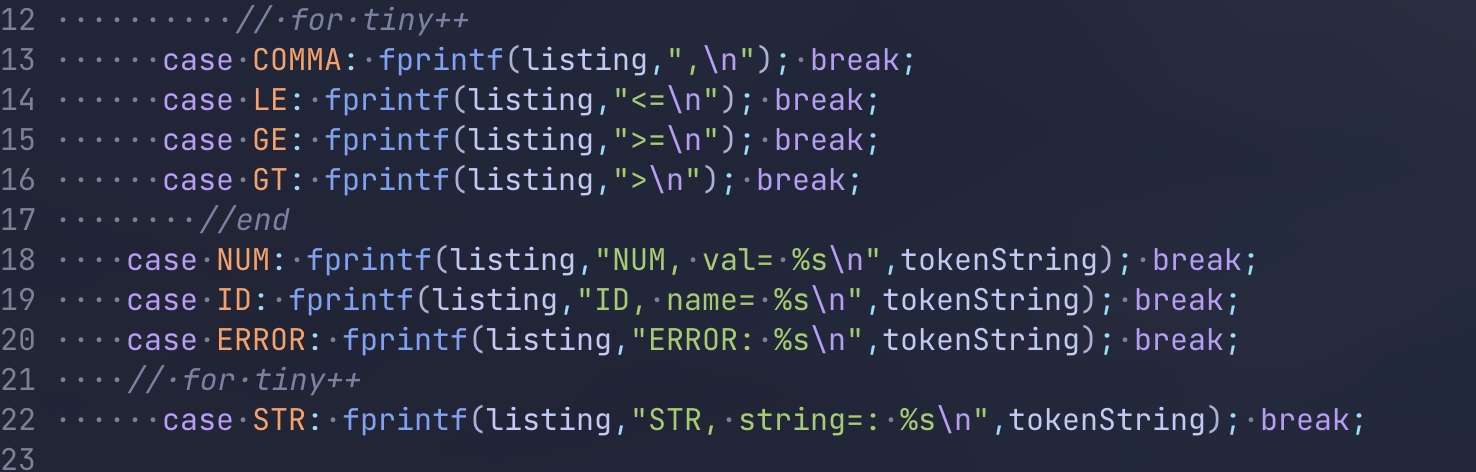
然后补充这三个状态的具体执行步骤：



其中，ING和INL都增加了对等于号的识别，并修改对应的状态为LE和GE。INSTR直接暴力读取接下来的字符串内容，当再次读到’的时候，表明字符串读取结束，于是进入DONE状态，同时记录这个token：

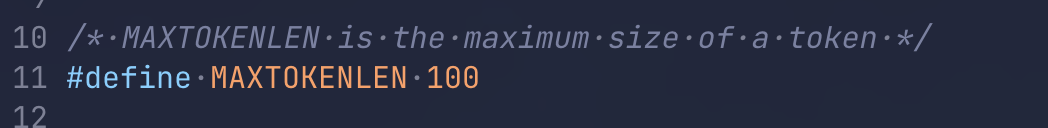
 

最后补充printToken这个函数对这些特殊字符的print：



在print的时候，字符串token输出不全的问题。

需要更改最大字符串token的长度，位于scan.h里：



* + - 1. Tiny++程序验证

为了验证上面的修改是否是正确的，我们首先定义这样的tiny++程序：

string str;

int x, fact;

str:= 'sample program in TINY+ language- computes factorial' ;

read x;

if x>0 and x<100 then {don’t compute if x<=0}

fact:=1;

do

fact:=fact\*x;

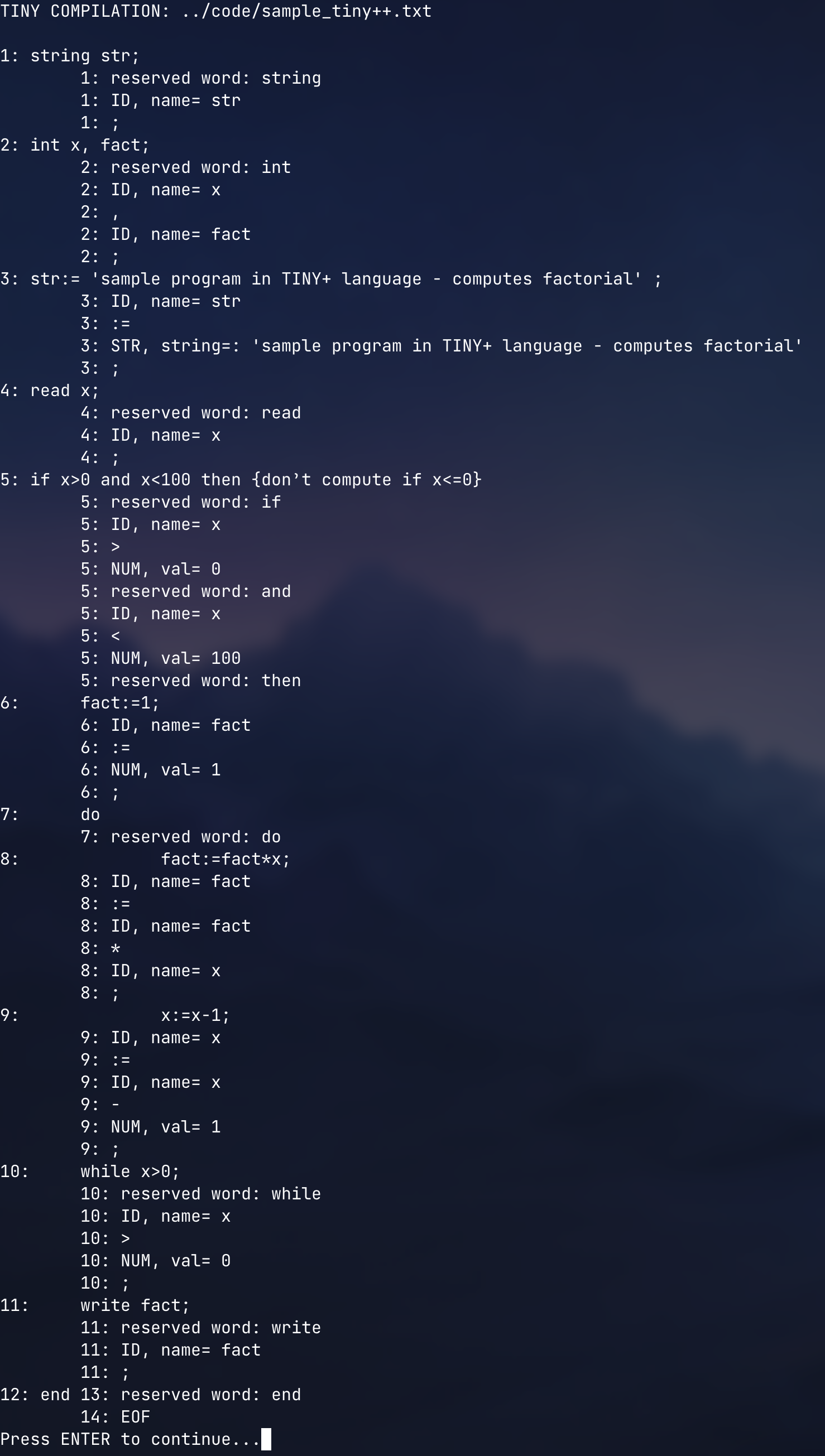
x:=x-1；

while x>0;

write fact；

end

接下来进行验证：



可以发现，成功识别了所有关键字，特殊字符和类型。

1. 实验总结

本次实验主要是对 TINY 语言的词法及词法分析器进行理解和实现，同时也拓展了 TINY+ 语言的词法分析器。

首先，我们通过 TINY Syntax.ppt 和 TINY+ Syntax.doc 对 TINY 和 TINY+ 的词法进行了了解和描述。在此基础上，我们使用 TINY Scanner.rar 中的 C 语言代码实现了 TINY 语言的词法分析器，并基于该分析器实现了拓展语言 TINY+ 的词法分析器。

在实现过程中，我们遵循了最长匹配原则，将符号识别为赋值符号而不是单独的冒号和等号。同时，我们使用了（种别码，属性值）表示 token，包含了关键字、系统特殊字符、变量、数值常量和字符串常量等类型的种别码。

此外，我们也实现了识别词法错误的功能，包括非法字符、字符串匹配错误和注释的右部括号丢失或匹配错误等类型的词法错误。词法分析器可以给出词法错误的行号并打印出对应的出错消息。

通过本次实验，我们深入了解了词法及词法分析器的实现原理，同时也拓展了对 TINY+ 语言的理解和应用能力。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  年 月 日 |
| 备注： |