**实验一 TINY语言的词法分析**

实验目的

构造tiny语言的词法分析器（扫描器），要求利用第三方的lex工具进行构造。实验结果：构造出的扫描器，能够读入教材样例中给出的tiny语言的示例代码，分解成token输出。

Experimental Requirement

Build the lexical parser (a scanner) of TINY language, using the Lex tool of third party. The experimental result Scanner can receive the sample program of Tiny language, and output is a sequence of tokes that are defined by regular expression.

输入：

**{ Sample program**

**in TINY language -**

**computes factorial**

**}**

**read x; { input an integer }**

**if 0 < x then { don't compute if x <= 0 }**

**fact := 1;**

**repeat**

**fact := fact \* x;**

**x := x - 1**

**until x = 0;**

**write fact { output factorial of x }**

**end**

预习要求：

Preview requirements:

1.查阅各种专业文献，针对任意二种现代程序设计语言编译器的词法分析技术，找出各自的标志性的特征点，并加以比较分析和比较，指出其优缺点，并试图给出自已的结论。

Referring to professional literatures, aiming at the lexical analysis technology of any two modern programming language compilers, finds out their respective features, compares them, points out their advantages and disadvantages, and tries to give your Evaluation and conclusions.

现代程序设计语言的编译器可以采用多种不同的词法分析技术。以下是两种常见的技术及其特征点、优缺点和比较分析：

技术1：基于表驱动的词法分析

特征点：

- 通过预处理阶段生成一个有限状态自动机（DFA）。

- DFA中的每个状态对应一个词法单元，转移条件为输入字符。

- 识别输入中的所有词法单元时，只需要遍历DFA一次。

优点：

- 比较高效，因为它们使用DFA进行匹配，可以快速跳过非法字符序列。

- 可以很好地适应大多数编程语言，因为这些语言通常具有固定的词法规则，而且这些规则可以轻松地表示为DFA。

缺点：

- 首次生成DFA需要花费一些时间。

- 如果输入程序包含大量注释或空格等无用字符，则会影响解析速度。

比较分析：

基于表驱动的词法分析技术广泛应用于许多编程语言的编译器中，如C、C++等。由于这些语言的词法规则相对简单，易于建立DFA进行匹配，因此这种技术可以很好地适应这些语言，具有高效、稳定的特点。

技术2：基于正则表达式的词法分析

特征点：

- 通过使用正则表达式定义词法规则，然后将其转换为确定性有限自动机（DFA）。

- DFA中每个状态对应一个正则表达式，表示匹配该状态的所有输入字符。

- 在识别输入中的所有词法单元时，需要遍历DFA，并根据当前状态匹配相应的正则表达式。

优点：

- 可以轻松地处理复杂的词法规则，如注释、字符串等。

- 可以采用多种不同的匹配算法进行实现，包括NFA、DFA和回溯匹配算法等。

缺点：

- 基于正则表达式的词法分析器通常比基于表驱动的词法分析器慢，因为每次都需要遍历DFA。

- 在处理大型输入文件时，可能会出现内存问题，因为它们需要存储整个DFA，而不仅仅是输入字符流。

比较分析：

基于正则表达式的词法分析技术广泛应用于许多编程语言的编译器中，如Python、Perl等。由于这些语言的词法规则比较复杂，需要处理注释、字符串等，因此这种技术可以很好地适应这些语言。然而，与基于表驱动的词法分析相比，它的解析速度较慢，并且可能会出现内存问题。

综上所述，选择何种词法分析技术取决于具体的编程语言和应用场景。如果需要快速解析简单的词法规则，可以使用基于表驱动的词法分析技术；如果需要处理复杂的词法规则，则可以使用基于正则表达式的词法分析技术。

2.请描述自已拟定的实验计划和步骤，以及对输入和输出的设计。

Please design your experimental plan and procedures, as well as the design of input and output of the SCANNER

实验计划：安装flex，编写lex程序，使用flex将其编译为c程序，使用gcc将c程序编译为可执行文件（扫描器）。

输入：tiny语言的示例代码的文件名。

输出：分解的token。

扫描器代码如下：

%{

    //包含C头文件、定义全局变量、声明函数

    #include <stdio.h>

    #include <stdlib.h>

%}

%%

"{"          printf("LEFT\_BRACE\n");

"}"          printf("RIGHT\_BRACE\n");

"read"       printf("READ\n");

"if"         printf("IF\n");

"then"       printf("THEN\n");

"else"       printf("ELSE\n");

"repeat"     printf("REPEAT\n");

"until"      printf("UNTIL\n");

"write"      printf("WRITE\n");

"end"        printf("END\n");

":="         printf("ASSIGN\n");

";"          printf("SEMICOLON\n");

"<"          printf("LESS\_THAN\n");

">"          printf("GREATER\_THAN\n");

"="          printf("EQUALS\n");

"+"          printf("PLUS\n");

"-"          printf("MINUS\n");

"\*"          printf("MULTIPLY\n");

"/"          printf("DIVIDE\n");

"%"          printf("MODULO\n");

[0-9]+       printf("NUMBER\n");

[a-zA-Z]+    printf("IDENTIFIER\n");

[\n\t ]+     /\* 忽略空格和换行符 \*/;

.            printf("UNKNOWN\n");

%%

int yywrap() {

    return 1;

}

int main(int argc, char \*argv[]) {

    if (argc != 2) {

        fprintf(stderr, "usage: %s filename\n", argv[0]);

        exit(1);

    }

    FILE \*file = fopen(argv[1], "r");

    if (!file) {

        fprintf(stderr, "could not open file: %s\n", argv[1]);

        exit(1);

    }

    yyin = file;

    yylex();

    fclose(file);

    return 0;

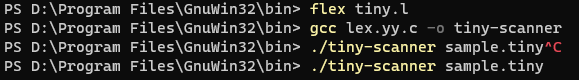
}

编译命令：

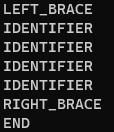
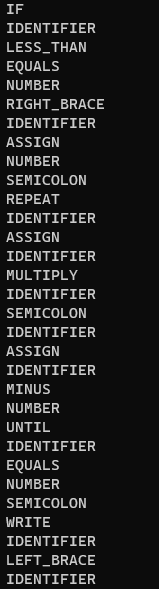
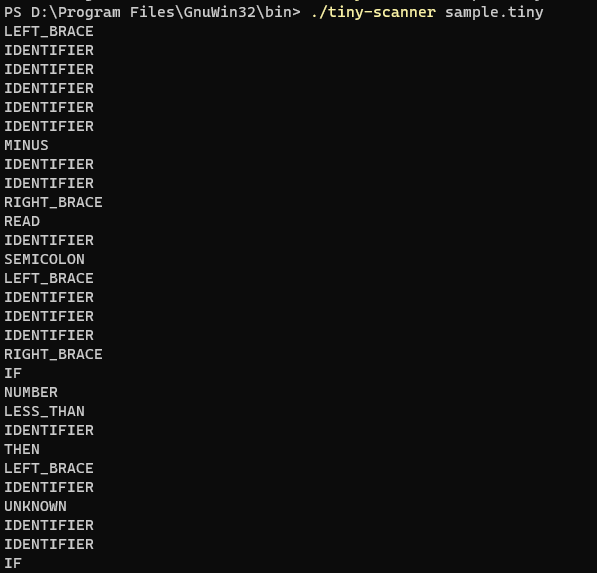
$ flex tiny.l

$ gcc lex.yy.c -o tiny-scanner

$ ./tiny-scanner sample.tiny #tiny语言示例代码文件



程序输出：

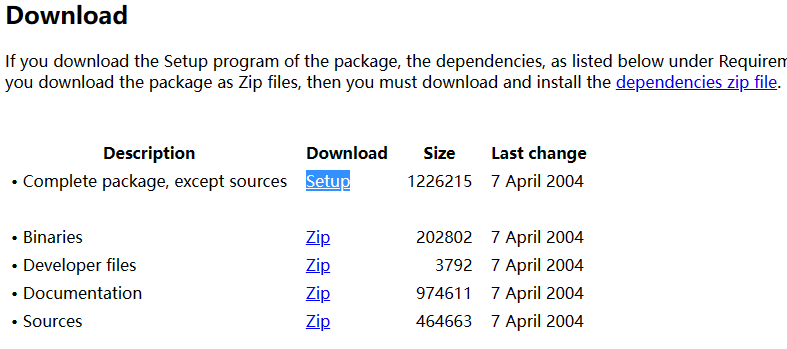


3.选择三方工具，阅读其使用说明，并尝试建立起工作环境。

Choose the LEX tools, read their instructions, and try to establish a working environment.

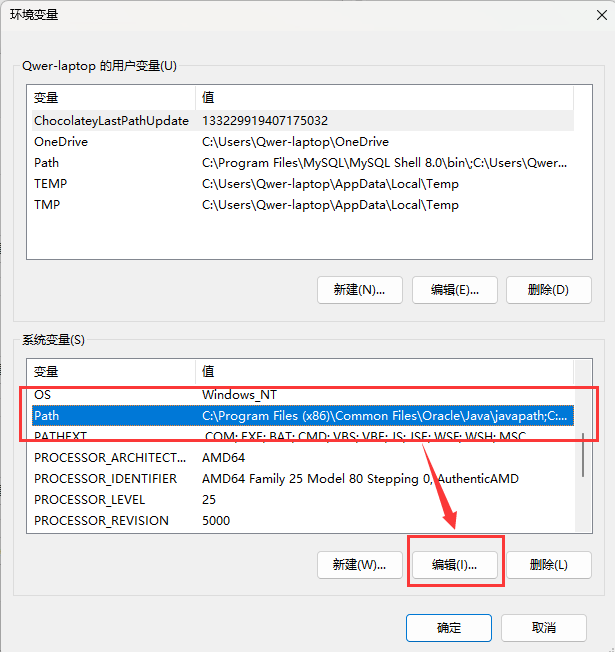
### **下载安装包**

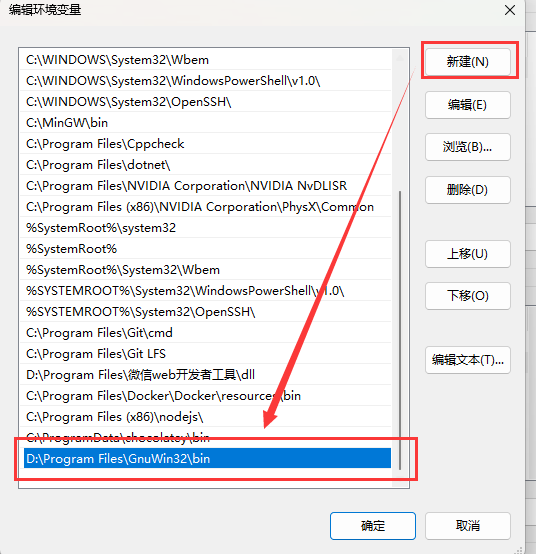
Flex 安装包下载地址: <https://gnuwin32.sourceforge.net/packages/flex.htm>



单击 Setup，会跳转到 SourceForge 下载。

### **配置环境变量**



然后根据自己的安装位置配置环境变量。