**编译原理**

班级：21计算机科学与技术

姓名：陈昊天 学号：2021329600006

实验一 TINY语言的词法分析

1. 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。

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**

1. 实验设计

实验设计：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。

相关正则表达式

数字标识符：[0-9]+

字符串标识符：[a-zA-Z]+

关键字标识符：read|if|then|repeat|until|write|end

操作符标识符：":="|";"|"<"|">"|"="|"+"|"-"|"\*"|"/"|"%"

注释的处理方法：

%x COMMENT

%%

[\n]\*        {printf("\nline %d:\n",line); line++;}

"{"                     { BEGIN(COMMENT); }

<COMMENT>[^{}]\*         {

                            printf("COMMENT BEGIN\n");

                            for(int i = 0; i < strlen(yytext); i++){

                                if(yytext[i] == '\n'){

                                    printf("\nline %d:\n",line); line++;

                                }

                                else{

                                    printf("%c",yytext[i]);

                                }

                            }

                            //printf("COMMENT BEGIN{ %s } COMMENT END\n",yytext);

                            printf("COMMENT END\n");

                        }

<COMMENT>"}"            { BEGIN(INITIAL); }

%%

1. 内容和步骤

1代码：

%{

    //包含C头文件、定义全局变量、声明函数

    #include <stdio.h>

    #include <stdlib.h>

    #include <string.h>

    int line=2;

%}

%x COMMENT

%%

[\n]\*        {printf("\nline %d:\n",line); line++;}

"{"                     { BEGIN(COMMENT); }

<COMMENT>[^{}]\*         {

                            printf("COMMENT BEGIN\n");

                            for(int i = 0; i < strlen(yytext); i++){

                                if(yytext[i] == '\n'){

                                    printf("\nline %d:\n",line); line++;

                                }

                                else{

                                    printf("%c",yytext[i]);

                                }

                            }

                            //printf("COMMENT BEGIN{ %s } COMMENT END\n",yytext);

                            printf("COMMENT END\n");

                        }

<COMMENT>"}"            { BEGIN(INITIAL); }

"read"       printf("READ ");

"if"         printf("IF ");

"then"       printf("THEN ");

"else"       printf("ELSE ");

"repeat"     printf("REPEAT ");

"until"      printf("UNTIL ");

"write"      printf("WRITE ");

"end"        printf("END ");

":="         printf("ASSIGN ");

";"          printf("SEMICOLON\n");

"<"          printf("LESS\_THAN ");

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

"="          printf("EQUALS ");

"+"          printf("PLUS ");

"-"          printf("MINUS ");

"\*"          printf("MULTIPLY ");

"/"          printf("DIVIDE ");

"%"          printf("MODULO ");

[0-9]+       printf("[NUMBER, Value = %s] ",yytext);

[a-zA-Z]+    printf("[IDENTIFIER, Name = %s] ",yytext);

[\t]         /\* ignore\*/

[ ]          /\* ignore\*/

.            printf("UNKNOWN ");

%%

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;

    printf("line1:\n");

    yylex();

    fclose(file);

    return 0;

}

1. 结果：

编译命令：

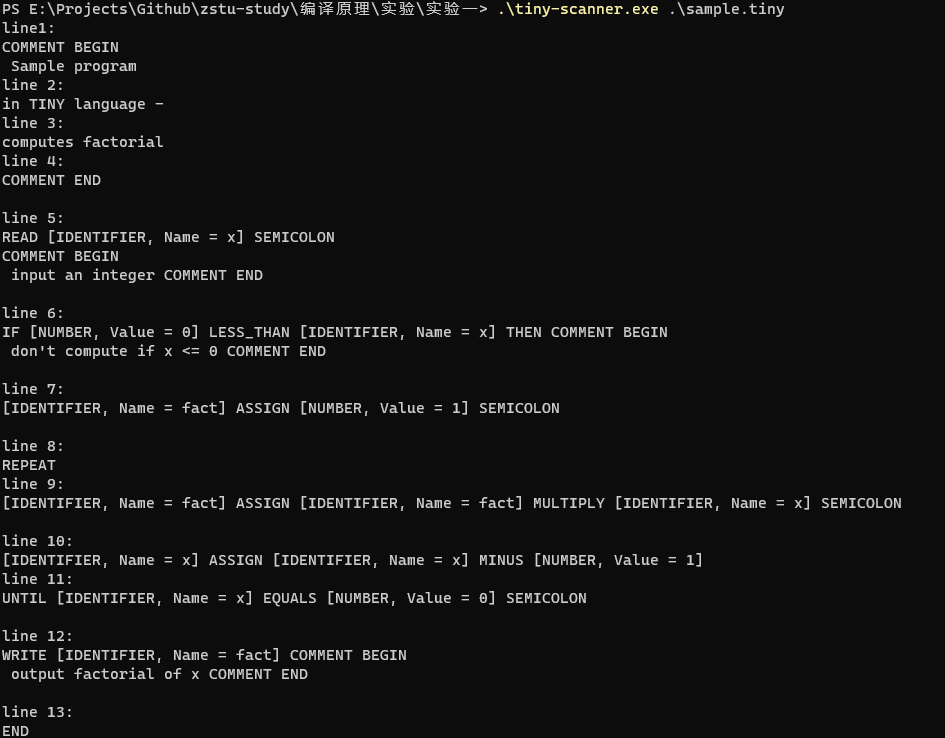
$ flex tiny.l

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

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



程序输出：



1. 实验结论:

1 理论基础（评价依据 理论知识非常清楚）

词法分析器（也称为扫描器）是编译器的一个组成部分，用于读取源代码并将其分解为单个单词或标记（token），以便进一步处理。在本实验中，使用了第三方的lex工具Flex来构造tiny语言的词法分析器，这个工具可以根据给定的正则表达式生成相应的有限状态自动机（DFA），并将输入的文本流分解为匹配的标记序列。通过实验的结果，可以验证该词法分析器能够正确地读入tiny语言的示例代码，并将其分解为对应的token序列进行输出。

2、分析和总结（评价依据：是否能够对实验结果作出完整和准确的描述，是否能够捕捉到实验中的各种现象，是否有强的信息综合能力，是否能得出正确的结论。）

在本实验中，我们成功地构造了一个tiny语言的词法分析器，使用第三方lex工具Flex进行构造，并成功读入教材样例中给出的tiny语言的示例代码，将其分解为token输出。在实验过程中，我们发现在编写正则表达式时需要考虑到各种情况的可能性，例如标识符、数字、关键字等等。同时，在测试扫描器的过程中也需要注意一些细节问题，如输入是否符合要求、输出结果是否正确等。

通过本次实验，我们深入理解了计算机编程语言中的词法分析原理和技术，并学会了如何使用第三方lex工具Flex构造词法分析器。此外，我们还学习了如何编写相关的测试用例，以检查词法分析器的正确性。这些知识和技能对于编写编译器或其他软件工具的人员来说是非常重要的。

总的来说，本次实验让我们通过实践掌握了词法分析相关的知识和技能，而实验结果也表明我们构造的扫描器能够准确地读入和分解tiny语言的示例代码，因此我们可以得出结论，该词法分析器的性能良好。

3、对工具的评价（优缺点及其局限性的总结）

优点：

- 利用lex工具可以方便快捷地生成词法分析器，无需手动编写有限状态自动机代码。

- lex工具支持正则表达式，可以轻松地实现复杂的文本匹配任务，适用于各种编程语言和领域。

缺点和局限性：

- 可能需要花费一定时间学习lex工具的使用方法和相关知识。

- 由于lex工具是基于有限状态自动机的，对于一些更复杂的词法分析问题（如嵌套语言或不规则模式），可能需要手动编写自定义的有限状态自动机，或者使用其他工具或技术来完成。

- 在某些情况下，可能需要使用其他工具或技术来实现更高效或更精确的匹配操作，例如使用正则表达式引擎或语法分析器等。

综上所述，lex工具是一个非常有用的词法分析器构建工具，它具有快速、简单、灵活等优点，但也存在一些局限性和缺陷。在实际应用中，需要根据具体情况选择适合的工具和技术来解决问题。