**编译原理实验四实验报告**

计科 MisakaCJQ

1. **实验目的**

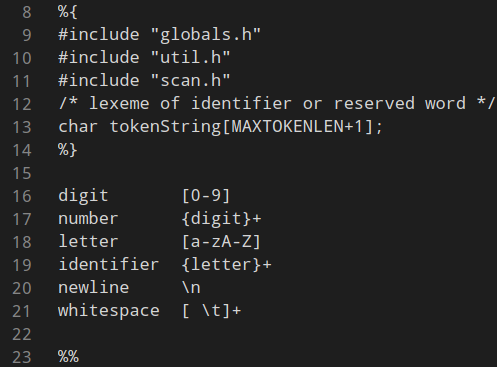
学习使用词法分析自动工具LEX。

学习使用语法分析程序自动生成工具YACC

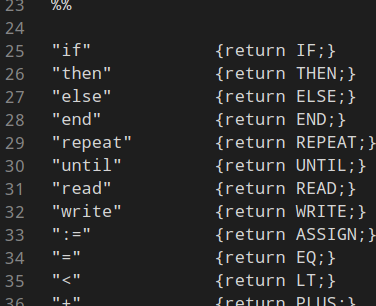
1. **实验任务**
2. 使用LEX工具实现编译器的词法分析程序
3. 使用YACC工具实现编译器的词法分析程序
4. **实验内容**
5. 生成LEX版本的TINY语言词法分析器并测试

tiny.l文件如下分为三部分，使用flex tiny.l即可生成lex.yy.c文件

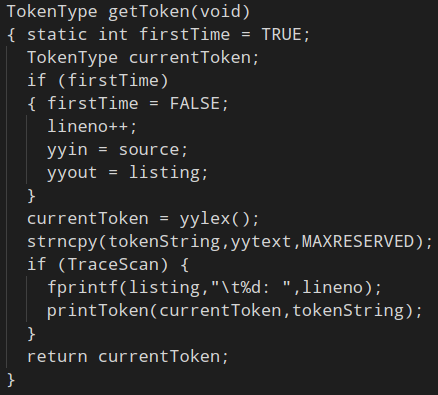
声明部分:



词法规则部分：



辅助函数部分：



编写makefile如下：

scanner: main.o lex.yy.o util.o

    gcc main.o lex.yy.o util.o -o scanner -lfl

main.o: main.c globals.h util.h

    gcc main.c -c

util.o: util.c util.h globals.h

    gcc util.c -c

lex.yy.o: tiny.l

    flex tiny.l

    gcc lex.yy.c -c

clean:

    -rm main.o

    -rm lex.yy.o

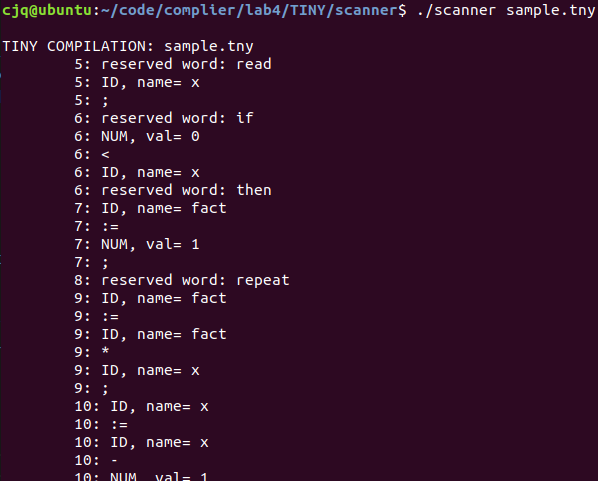
    -rm lex.yy.c

    -rm util.o

    -rm scanner

链接和生成最终的词法分析器scanner。

运行./scanner sample.tny进行测试

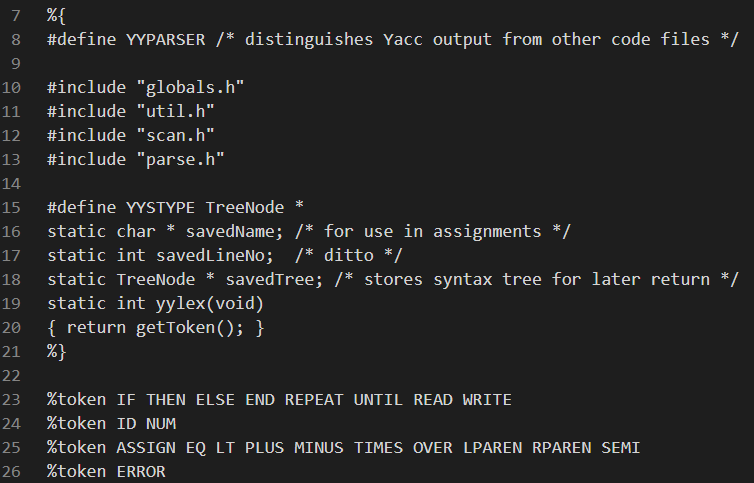


正确的输出了词法单元序列

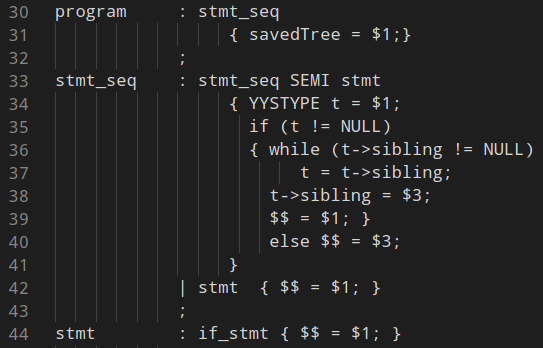
1. 生成YACC版本的TINY语言语法分析器

tiny.y分为三部分。使用bison -d tiny.y指令可生成tiny.tab.c和tiny.tab.h

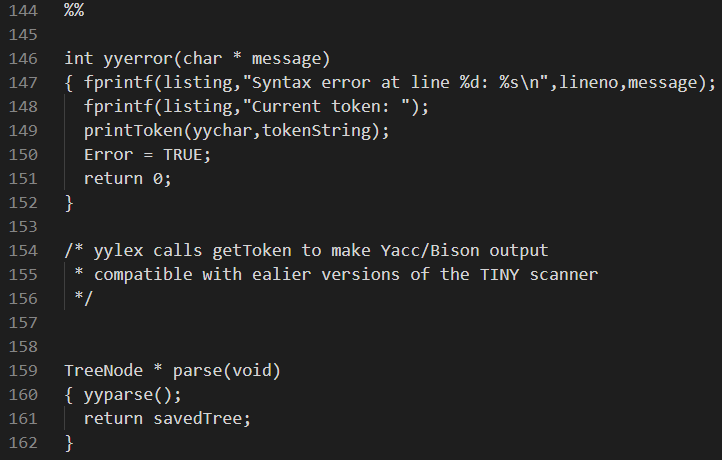
声明部分：



语义规则部分：



辅助函数部分：



编写makefile如下：

parser: main.o util.o scan.o parse.o

    gcc main.o util.o scan.o parse.o -o parser -lfl

main.o: main.c globals.h util.h scan.h

    gcc -c main.c

util.o: util.c util.h globals.h

    gcc -c util.c

scan.o: lex.yy.c scan.h util.h globals.h

    gcc -c lex.yy.c -o scan.o

lex.yy.c: tiny.l

    flex tiny.l

tiny.tab.c: tiny.y

    bison -d tiny.y

parse.o: tiny.tab.c parse.h scan.h globals.h util.h

gcc -c tiny.tab.c -o parse.o

clean:

    -rm lex.yy.c

    -rm tiny.tab.c

    -rm parse.o

    -rm scan.o

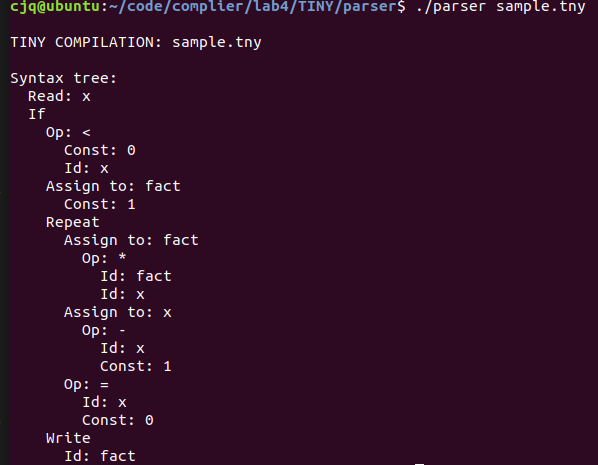
    -rm util.o

    -rm main.o

    -rm parser

链接和生成最终的语法分析器parser

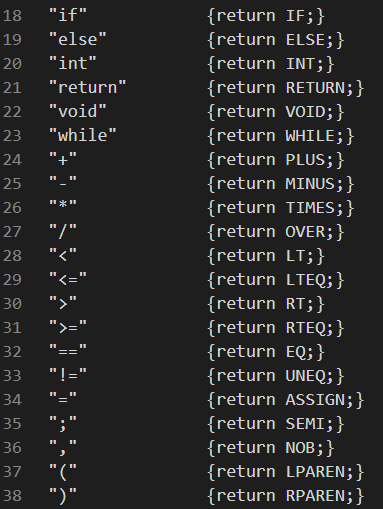
运行./parser sample.tny进行测试。正确生成了语法树。



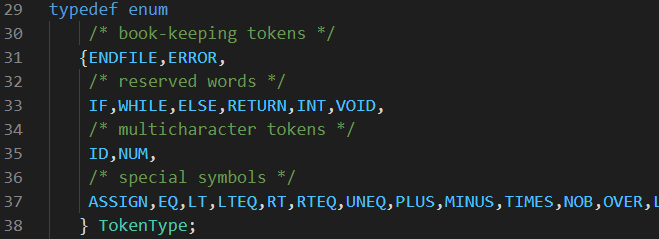
1. 生成LEX版本的C-语言词法分析器

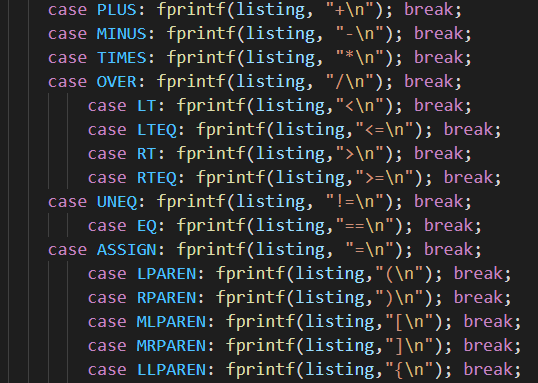
首先在之前tiny.l的基础上修改。

将中间的词法规则修改为C-语言的词法规则。



修改global.h和util.c中的词法单元定义和printToken函数，改为C-语言使用。

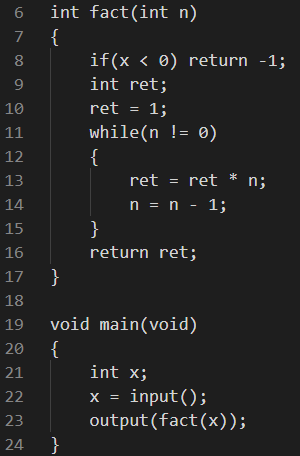




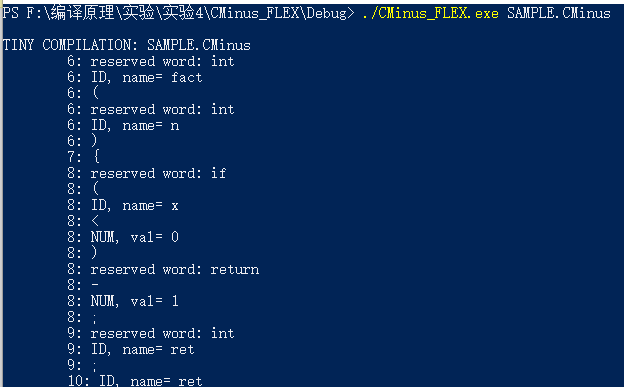
使用flex.exe生成lex.yy.c文件。

将lex.yy.c和修改过后的global.h util.h util.c main.c等共同放入一个工程中，即可编译生成对应的C-语言词法分析器程序CMinus\_FLEX.exe。

使用示例C-代码代码进行测试



执行./CMinus\_FLEX.exe SAMPLE.CMinus

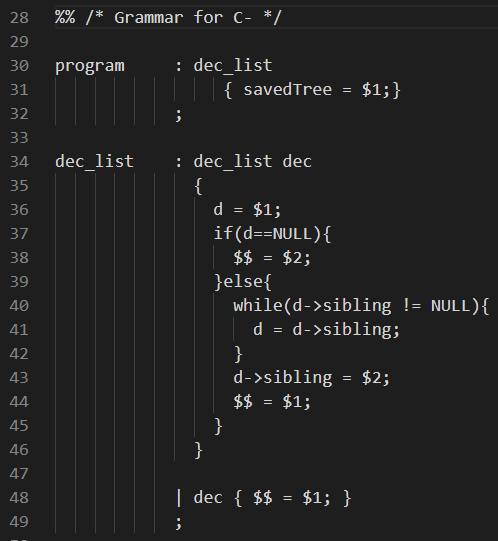


正确输出了词法单元序列。

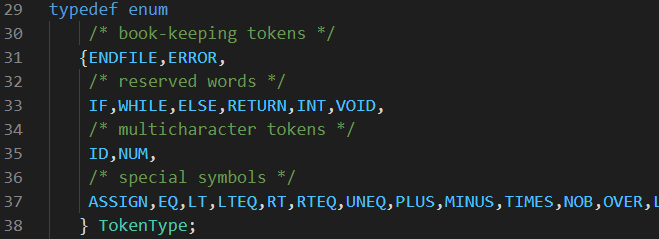
1. 生成YACC版本的C-语言语法分析器

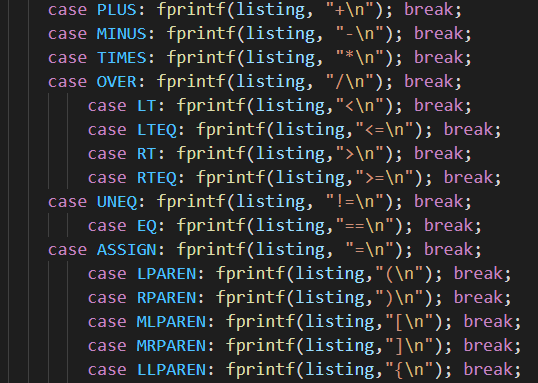
首先在之前tiny.l的基础上修改。

将中间的语法规则修改为C-语言的语法规则。

。

修改global.h和util.c中的词法单元定义和printToken函数，改为C-语言使用。

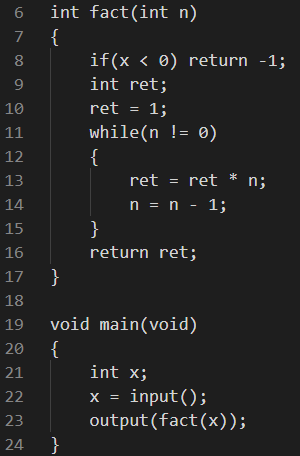




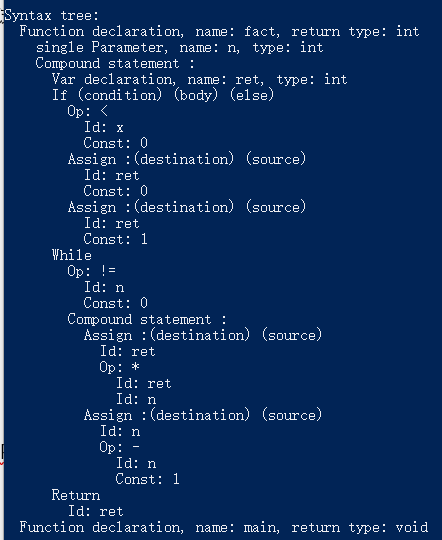
使用biason.exe -d指令生成cminus.tab.c和cminus.tab.h文件。

将cminus.tab.c和cminus.tab.h和修改过后的global.h util.h util.c main.c等共同放入一个工程中，即可编译生成对应的C-语言词法分析器程序CMinus\_YACC.exe

使用示例C-代码代码进行测试



执行./CMinus\_YACC.exe SAMPLE.CMinus



生成了正确的语法分析树。

1. **实验总结**

这一次实验的难点主要在于根据TINY语言的lex和yacc用法来改写C-语言的.l文件和.y文件，在修改这两个文件的基础上还要修改global.h和UTIL.c文件中的一些函数，从而使其能够配合C-语言的词法分析器和语法分析器使用。总之这次实验深入的让我了解了lex和yacc这两个工具的使用方法，这是我在学习编译原理课程的一大收获。