****

**院 系：计算机学院**

**实验课程：编译原理实验**

**实验项目：拓展TINY语言语法**

**指导老师：黄煜廉**

**开课时间： 2017-2018第一学年**

**专 业： 软件工程（软件开发）**

**班 级： 2015级2班**

**学 生：**

**学 号：**

**实验完成时间： 2017年12月21日**

## (一)实验要求

### 实验二：TINY扩充语言的语法分析

扩充的语法规则有：实现 while、do while、for语句和求余计算式子，具体文法规则自行构造。

可参考：P97及P136的文法规则。

(1) While-stmt --> while exp do stmt-sequence endwhile

(2) Dowhile-stmt-->do stmt-sequence while exp

(3) for-stmt-->for identifier:=simple-exp to simple-exp do stmt-sequence enddo 步长递增1

(4) for-stmt-->for identifier:=simple-exp downto simple-exp do stmt-sequence enddo 步长递减1

### 要求：

（1）要提供一个源程序编辑界面，以让用户输入源程序（可保存、打开源程序）

（2）可由用户选择是否生成语法树，并可查看所生成的语法树。

（3）应该书写完善的软件文档

**实验环境：Qt 5.6.3**

## (二)需求分析

(1)扩展文法while、do while、for语句和求余计算式子

(2)使用Qt开发

**(三)概要设计**

**(1)具体文法规则设计：**

(1)While-stmt --> while exp do stmt-sequence endwhile

(2) Dowhile-stmt-->do stmt-sequence while exp

(3) For-stmt-->for identifier:=simple-exp to|downto simple-exp do stmt-sequence enddo

to步长递增1,downto步长递减1

1. **在课本附录B代码的基础上进行以上文法的扩充。**

**(四)详细设计**

**(1)扩充文法while、do while、for 语句的语法图：**





（1）在SCAN.cpp中修改typedef enum{}TokenType

增加保留字： WHILE、DO、TO、DOWNTO、FOR、ENDDO、ENDWHILE

增加求余运算符：MOD

（2）修改typedef enum{}StmtKind

增加语句类型：WhileK、DoWhileK、ForK

（3）修改static struct

{ char\* str;

Typetoken tok;

} reservedWords[MAXRESERVED]= {}

增加保留字关联字符{{"while",WHILE},{"endwhile",ENDWHILE},

{"do",DO},{"to",TO},{"downto",DOWNTO},{"for",FOR},{"enddo",ENDDO}};

（4）PARSE.cpp添加以下代码：

static TreeNode \* while\_stmt(void);

static TreeNode \* dowhile\_stmt(void);

static TreeNode \* for\_stmt(void);

TreeNode \* while\_stmt(void) {

TreeNode \* t=newStmtNode(WhileK);

match(WHILE);

if(t!=NULL)t->child [0]=exp();

match(DO);

if(t!=NULL)t->child [1]=stmt\_sequence();

match(ENDWHILE);

return t; }

TreeNode \* dowhile\_stmt(void) {

TreeNode \* t=newStmtNode(DowhileK);

match(DO);

if(t!=NULL)t->child [0]=stmt\_sequence();

match(WHILE);

if(t!=NULL)t->child [1]=exp();

return t; }

TreeNode \* for\_stmt(void) {

TreeNode \* t=newStmtNode(ForIncK);

match(FOR);

if(t!=NULL){

t->child [0]=assign\_stmt();

if(tokn==TO)

match(TO);

else if(tokn==DOWNTO) {

match(DOWNTO);

t->kind.stmt =ForDecK;}

t->child [1]=simple\_exp();

match(DO);

t->child [2]=stmt\_sequence();

match(ENDDO); }

return t; }

TreeNode \* statement(void)的switch中添加以下代码：

case WHILE: t=while\_stmt();break;

case DO: t=dowhile\_stmt();break;

case FOR: t=for\_stmt();break;

(5)UTIL.cpp中修改以下代码：

switch (token)中添加代码：

case WHILE:

case DO:

case FOR:

case ENDWHILE:

case ENDDO:

case TO:

case DOWNTO:

case MOD: fprintf(listing,"%c\n",ch);break;

switch (tree->kind.stmt)中添加代码：

case WhileK:

fprintf(listing,"While\n");

break;

case DowhileK:

fprintf(listing,"Do\n");

break;

case ForIncK:

fprintf(listing,"For(步长加1)\n");

break;

case ForDecK:

fprintf(listing,"For(步长减1)\n");

break;

1. main.c这个文件去掉，重新写一个mymain.h的头文件。

因为时Qt开发的，所以不能直接使用main.c这个文件，得重新写。

Mymain.h主要就是导入一些tiny的头文件，还有一个生成语法树的函数，供qt里的mainwindow.cpp这个文件里的生成语法树这个按钮调用。具体函数内容：

void syntree()

{

TreeNode \* syntaxTree;

char pgm[]="SAMPLE.txt";

source = fopen(pgm,"rt");

syntaxTree = parse();

char ch[]="syntax tree.txt"; //语法树输出文件

listing = fopen(ch,"wt+");

if (TraceParse) {

fprintf(listing,"Syntax tree:\n");

printTree(syntaxTree);

fclose(listing);

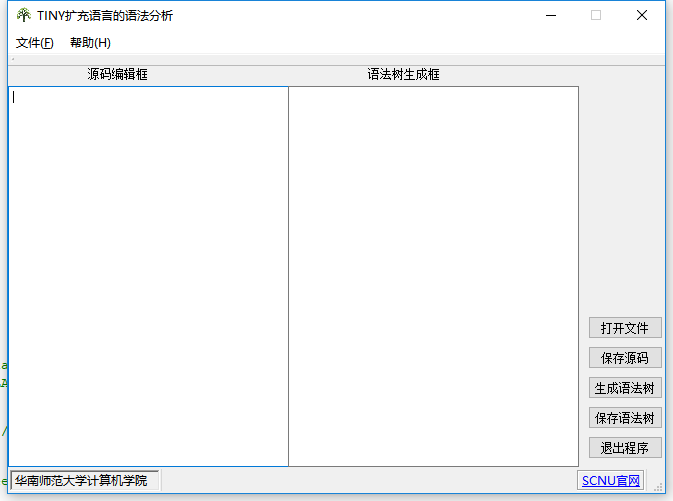
}

fclose(source);

}

**(五)软件测试与运行**

开始界面



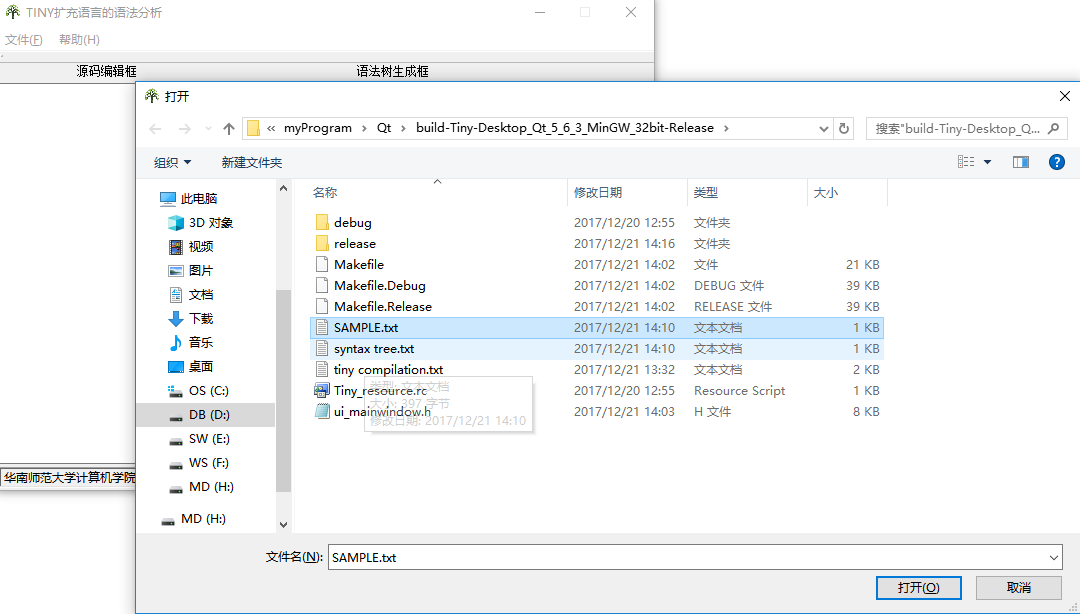
说明：**源码编辑框**是可以**手动输入**tiny源程序，也可以采用**打开文件**按钮打开一个已经写好了的**text文本来读取源码**。

**保存源码**按钮可以把源码编辑框里的源码**保存到指定的路径**。

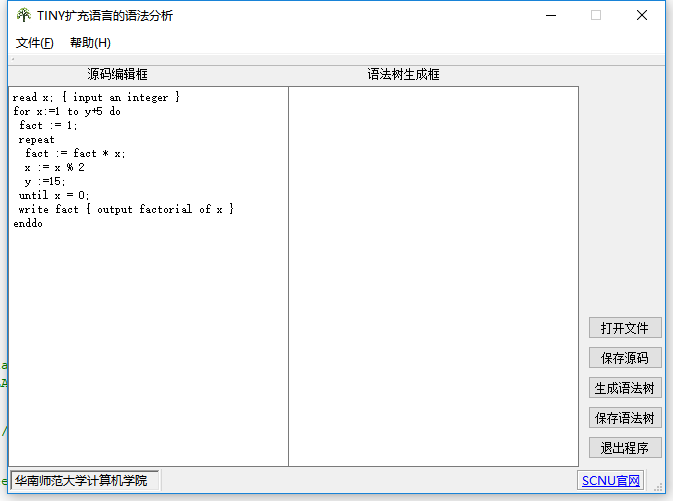
**生成语法树**按钮可以把左边源码编辑框里的源程序生成语法树，并**把生成的语法树显示到右边的语法树生成框中**。

**保存语法树**按钮可以把右边语法树生成框的**语法树保存到指定路径的路径**。

点击打开按钮，读取一个源码：

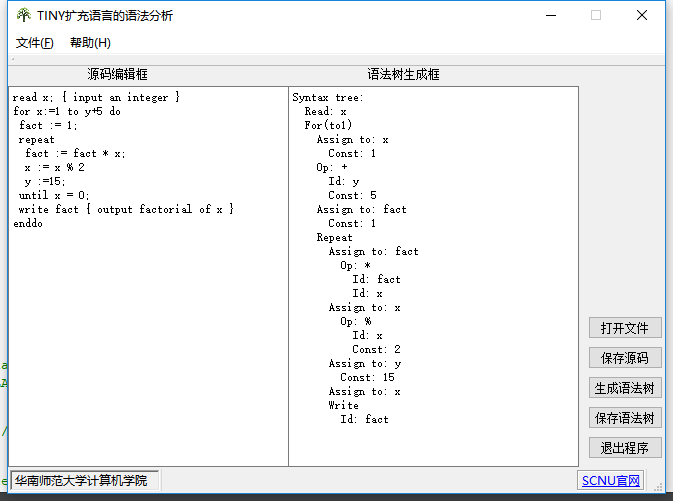


读取结果如下：

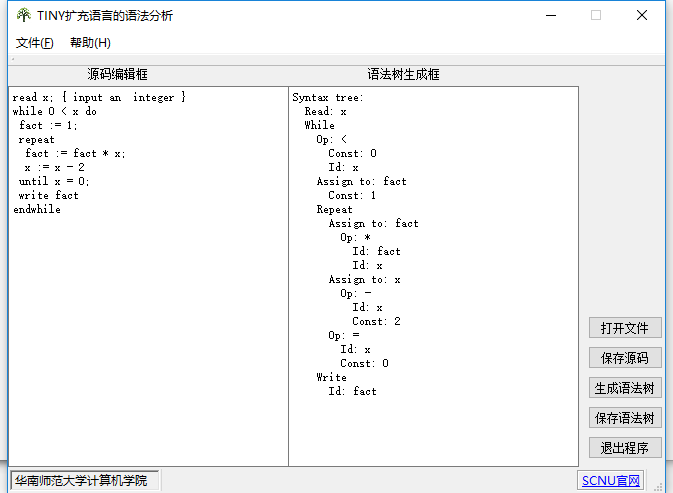


点击生成语法树：

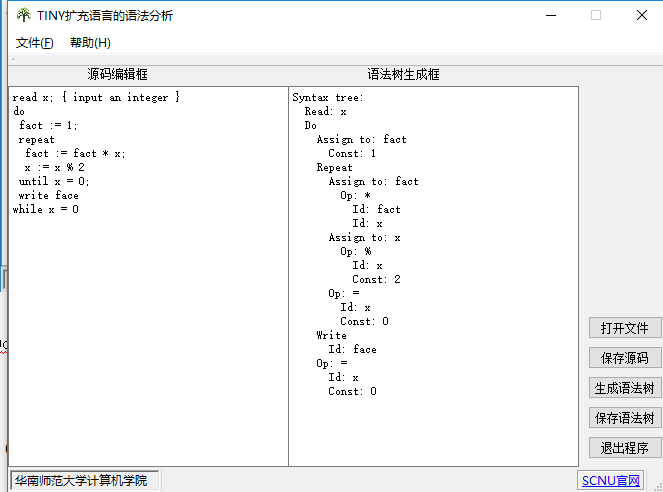
for-stmt-->for identifier:=simple-exp to simple-exp do stmt-sequence enddo 文法规则测试



While-stmt --> while exp do stmt-sequence endwhile 文法规则测试



Dowhile-stmt-->do stmt-sequence while exp 文法规则测试



**（六）实验小结:**

这次实验基于课本后面的小型编译器代码，成功实现了Tiny语言while和dowhile和for以及余式计算的扩充，这次的实验主要是在原有代码基础上进行修改，核心代码的实现还是离不开语法分析的基本思想，沿着这个思路就可以很好地理解代码。

本程序在运行时会在程序的根目录下生成两个txt文档，分别时SAMPLE.txt和syntax tree.txt。这两个文件的作用分别是SAMPLE.txt是保存源码编辑框里的源程序，syntax tree.txt里的内容显示到语法树编辑框中。这两个文件只有在点击生成语法树按钮是才会生成。至于为什么要采弄多两个这个文件，主要是因为还没找到更好的方法，把源程序的内容作为生成语法树的输入，把生成的语法树直接输出到语法树生成框。所以就采用这样的办法。