编译原理作业3

林伟业 20152100121

1. 实验内容

扩充的语法规则有：实现 while、do while、for语句、大于>比较运算符号以及求余计算式子，具体文法规则自行构造。

(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

(5) 大于>比较运算符号以及求余计算式子的文法规则请自行组织。

(6) 把TINY语言原有的if语句书写格式

```
if_stmt-->if exp then stmt-sequence end  |  | if exp then stmt-sequence else stmt-sequence end
```

改写为：

```
 if_stmt-->if(exp) stmt-sequence else stmt-sequence | if(exp) stmt-sequence
```

2. 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

3. EBNF中的TINY语言的文法

```
program → stmt-sequence
stmt-sequence → statement{;statement}
statement → if-stmt | repeat-stmt | assign-stmt | read-stmt | write-stmt

改造过的if文法
if-stmt → if(exp) stmt-sequence [else stmt-sequence]

新添加的文法
while-stmt → while exp do stmt-sequence endwhile
Dowhile-stmt → do stmt-sequence while(exp)
for-stmt → for identifier:=simple-exp to simple-exp do stmt-sequence enddo
for-stmt → for identifier:=simple-exp downto simple-exp do stmt-sequence enddo

repeat-stmt → repeat stmt-sequence until exp
assign-stmt → identifier := exp
read-stmt → read identifier
write-stmt → write exp
exp → simple-exp [comparison-op simple-exp]

添加大于号文法
```

```
comparison-op → < | = | >

simple-exp → term [addop term]
addop → + | -
term → factor {mulop factor}
mulop → * | /
factor → (exp) | number | identifier
```

修改 globals.h

修改最大保留字

```
#define MAXRESERVED 15
```

添加类型

```
typedef enum
    /* book-keeping tokens */
   {ENDFILE,ERROR,
    /* reserved words */
    IF,THEN,ELSE,END,REPEAT,UNTIL,READ,WRITE,DO,ENDDO,ENDWHILE,TO,DOWNTO,FOR,WHILE,
    /* multicharacter tokens */
    ID,NUM,
    /* special symbols */
    ASSIGN,EQ,LT,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,SEMI,NEQ,MT
   } TokenType;
```

添加节点

```
typedef enum {IfK,RepeatK,AssignK,ReadK,WriteK,DoWhileK,ForK,WhileK} StmtKind;
```

修改SCAN.C 文件

添加保留字

```
static struct
    { char* str;
      TokenType tok;
    } reservedWords[MAXRESERVED]
   = {{"if",IF},{"then",THEN},{"else",ELSE},{"end",END},
      {"repeat",REPEAT},{"until",UNTIL},{"read",READ},
      {"write",WRITE},{"while",WHILE},{"endwhile",ENDWHILE},
      {"do",DO},{"to",TO},{"downto",DOWNTO},{"for",FOR},{"enddo",ENDDO}};
```

在getToken(void)函数的switch(c)添加大与号语法

```
 case '>':
   currentToken = MT;
   break;
```

修改UTIL.C

在void printToken(TokenType token, const char* tokenString)添加

```
case WHILE:
case DO:
case TO:
case DOWNTO:
case FOR:
case ENDDO:
case ENDWHILE:
case MT: fprintf(listing,">\n"); break;
```

在void printTree(TreeNode * tree)添加

```
        case DoWhileK:
          fprintf(listing, "DO while\n");
          break;
        case ForK:
          fprintf(listing, "FOR\n");
          break;
        case WhileK:
          fprintf(listing, "while\n");
          break;
```

修改PARSE.C代码

修改已经有的函数，添加没有的函数 文件前面添加

```
static TreeNode * while_stmt(void);
static TreeNode * Dowhile_stmt(void);
static TreeNode * for_stmt(void);
```

if代码

```
TreeNode * if_stmt(void)
{
  TreeNode * t = newStmtNode(IfK);
  match(IF);
  match(LPAREN);
  if (t!=NULL)
    t->child[0] = exp();
  match(RPAREN);
  if (t!=NULL)
    t->child[1] = stmt_sequence();
  if (token==ELSE)
  {
    match(ELSE);
    if (t!=NULL)
      t->child[2] = stmt_sequence();
  }
  return t;
}
```

添加 > 号

parse.c 文件

```
TreeNode * exp(void)
{
  TreeNode * t = simple_exp();
  if ((token==LT)||(token==EQ)||(token==MT))
  {
    TreeNode * p = newExpNode(OpK);
    if (p!=NULL)
    {
      p->child[0] = t;
      p->attr.op = token;
      t = p;
    }
    match(token);
    if (t!=NULL)
      t->child[1] = simple_exp();
  }
  return t;
}
```

实现 while

parse.c 文件 添加

```
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;
}
```

在TreeNode * statement(void)函数添加

```
case WHILE : t = while_stmt(); break;
```

在TreeNode * stmt_sequence(void)函数的while条件里添加

```
&& (token != ENDWHILE)
```

实现 do-while

parse.c 文件添加

```
TreeNode * Dowhile_stmt(void)
{
  TreeNode * t = newStmtNode(DoWhileK);
  match(DO);
  if (t!=NULL)
    t->child[0] = stmt_sequence();
  match(WHILE);
  match(LPAREN);
  if (t!=NULL)
    t->child[1] = exp();
  match(RPAREN);
  return t;
}
```

在TreeNode * statement(void)函数添加

```
case DO : t = Dowhile_stmt(); break;
```

在TreeNode * stmt_sequence(void)函数的while条件里添加

```
&& (token != WHILE)
```

实现 for-stmt

```
TreeNode * for_stmt(void)
{
  TreeNode * t = newStmtNode(ForK);
  match(FOR);
  assign_stmt();
  if(token==TO)
  {
      match(TO);
      if (t!=NULL)
      t->child[0] = simple_exp();
      match(DO)
      if (t!=NULL)
      t->child[1] = stmt_sequence();
      match(ENDDO);
  }
  else(token==DOWNTO)
  {
      match(DOWNTO);
      if (t!=NULL)
      t->child[0] = simple_exp();
      match(DO);
      if (t!=NULL)
      t->child[1] = stmt_sequence();
      match(ENDDO);
  }
  return t;
}
```

在TreeNode * statement(void)函数添加

```
case FOR : t = for_stmt(); break;
```

在TreeNode * stmt_sequence(void)函数的while条件里添加

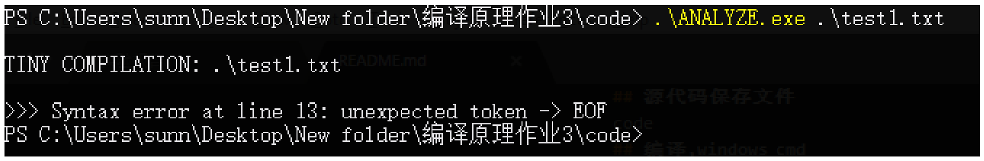
```
&& (token != DOWNTO) && (token != DO && (token != ENDDO))
```

运行结果

测试代码1

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if ( x>0 ) { don't compute if x <= 0 }
  fact := 1;
  do
    fact := fact * x;
    x := x - 1
  while ( x>0 );
  write fact;  { output factorial of x }
```

结果



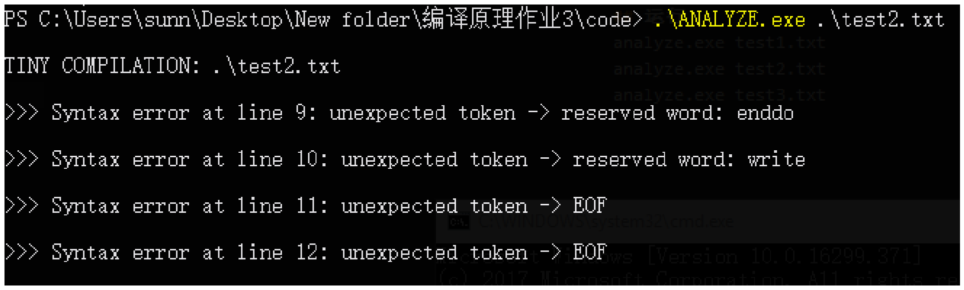
```
PS C:\Users\sunn\Desktop\New folder\编译原理作业3\code> .\ANALYZE.exe .\test1.txt

TINY COMPILATION: .\test1.txt

>>> Syntax error at line 13: unexpected token -> EOF
PS C:\Users\sunn\Desktop\New folder\编译原理作业3\code>
```

测试代码1结果

测试代码2

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if ( x>0 ) { don't compute if x <= 0 }
  for  fact := x downto 1 do
    fact := fact * x;
  enddo
  write fact;  { output factorial of x }
```

结果



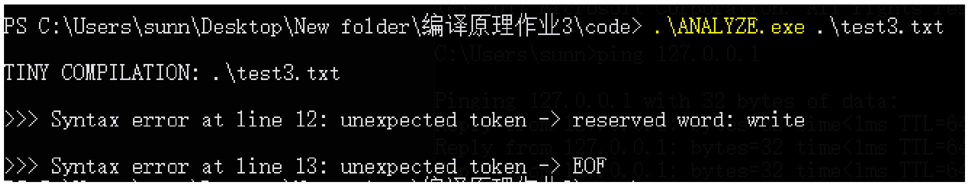
```
PS C:\Users\sunn\Desktop\New folder\编译原理作业3\code> .\ANALYZE.exe .\test2.txt

TINY COMPILATION: .\test2.txt

>>> Syntax error at line 9: unexpected token -> reserved word: enddo

>>> Syntax error at line 10: unexpected token -> reserved word: write

>>> Syntax error at line 11: unexpected token -> EOF

>>> Syntax error at line 12: unexpected token -> EOF
```

测试代码2结果

测试代码3

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if ( x>0 ) { don't compute if x <= 0 }
  fact := 1;
  while x>0 do
    fact := fact * x;
    x := x - 1
  endwhile
  write fact;  { output factorial of x }
```

结果



```
PS C:\Users\sunn\Desktop\New folder\编译原理作业3\code> .\ANALYZE.exe .\test3.txt

TINY COMPILATION: .\test3.txt

>>> Syntax error at line 12: unexpected token -> reserved word: write

>>> Syntax error at line 13: unexpected token -> EOF
```

测试代码3结果