编译原理实验-语法分析

17341146 王程钥

1 简介

1.1 实验目的

通过扩展已有的样例语言TINY的语法分析程序，为扩展TINY语言TINY + 构造语法分析程序，从而掌握语法分析程序的构造方法。

1.2 实验内容

用EBNF描述TINY + 的语法，用C语言扩展TINY的语法分析程序，构造TINY + 的递归下降语法分析器。

1.3 实验要求

将TOKEN序列转换成语法分析树，并能检查一定的语法错误。

2 TINY+词法的EBNF

首先将上次实验得到的Token分为以下几个类别

类别	解释
Key	标识符
Number	数字
String	字符串
Char	字符
Type	数据类型
Reserved	保留字

将这些类别的Token（除保留字和特殊字符外，保留字需要特殊判断）作为终端符号，得到TINY+的EBNF。

```
Program -> ( Func | Vars )*
Func -> Type Key '(' ForMalParams ')' Block
Vars -> Type Key ( ',', key )* ';'

ForMalParams -> ϵ | ForMalParam ( ',' ForMalParam )*
ForMalParam -> Type Key

Block -> Statement*
Statement -> Vars | Assignment | IfStmt | ReturnStmt | RepeatStmt |
ReadStmt | WriteStmt

Assignment -> Key ':=' ( Expression | String ) ';'
```

```
IfStmt -> 'if' Expression 'then' Block [ 'else' Block ] 'endif'
ReturnStmt -> 'return' Expression ';'
RepeatStmt -> 'repeat' Block 'until' Expression
ReadStmt -> 'read' Key ';'
WriteStmt -> 'write' Key ';'

Expression -> Exp0 [ ( '==' | '!=' | '>=' | '<=' | '>' | '<' ) Exp0 ]
Exp0 -> [ Exp1 ] ( ('+' | '-') Exp1 )*
Exp1 -> Exp2 ( ('*' | '/') Exp2 )*
Exp2 -> ( '(' Expression ')' ) | Number | Key | Char

```

根据TINY+的EBNF表构造First集合，字符太多的直接用终端符号表示（Type, Digit, Alpha, AnyChar）。

Key	**First**
Program	Type
Func	Type
Vars	Type
ForMalParams	Type
ForMalParam	Type
Block	'begin'
Statement	Type, 'if', 'return', 'repeat', 'write'
Assignment	Key
IfStmt	'if'
ReturnStmt	'return'
RepeatStmt	'repeat'
WriteStmt	'write'
Expression	'(', Number, Key, Char
Exp0	'(', Number, Key, Char
Exp1	'(', Number, Key, Char
Exp2	'(', Number, Key, Char

3 类定义与相关说明

词法分析器Scanner，记录代码串，代码串长和当前位置

```
typedef struct Scanner
{
    int pos;
    int length;
    char* buf;
}Scanner;
```

语法树节点TrieNode，每个节点存节点名字和指向孩子的指针，这里默认MAXCHILD=1000。

```
typedef struct trie TrieNode;
struct trie
{
    TrieNode* Child[MAXCHILD];
    int ChildSize;
    char* NodeName;
    Token* NodeToken;
    char* Place;
};
```

错误记录ErrCode，记录错误类型，错误代码而错误位置。

```
typedef struct ErrCode
{
    char* ErrInfo;
    ErrType ErrTp;
    int Posx, Posy;
}ErrCode;
```

其中错误类型包括以下几种

编号	错误名称	解释
1	$KeyError$	标识符错误
2	$AnnError$	注释格式错误
3	$SynError$	语法错误
4	$UknError$	出现未知符号
5	$MemError$	代码过长
6	$ExpError$	表达式错误

Token类记录每个token的信息，包括名称，类型和位置。

```
typedef struct Token{
    char* string;
    TokenType type;
    int Posx, Posy;
}Token;
```

TokenList类记录所有token的信息。

```
typedef struct TokenList{
    int size;
    int CurPos;
    Token* list[MAXN];
}TokenList;
```

4 语法分析

本次实验采用递归下降语法进行语法分析。将EBNF翻译程代码，对于每个节点根据first集合判断下一步的走向，构建语法树即可。
定义文法的部分代码比较长，不过多赘述。这里只举几个简单的例子。

4.1 ForMalParams(形参)

文法

> ForMalParams -> ε | ForMalParam (',' ForMalParam)*

对应代码

```
TrieNode* ForMalParams(TokenList* ToList){
    Token* CurToken;
    TrieNode* ChildNode;
    TrieNode* CurNode = (TrieNode*)malloc(1);
    InitTrieNode(CurNode, "ForMalParams");

    ChildNode = ForMalParam(ToList);
    if(ChildNode==NULL) return NULL;
    AddChild(CurNode, ChildNode);

    while(1){
        CurToken = GetCurToken(ToList);
        if(CurToken==NULL || CurToken->string!=",") break;
        AddTerminal(CurNode, CurToken->string);
        MoveToken(ToList, 1);

        ChildNode = ForMalParam(ToList);
        if(ChildNode==NULL){
            BackToken(ToList, 1); break;
        }
        AddChild(CurNode, ChildNode);
    }

    return CurNode;
}
```

4.2 Statement

文法

> Statement -> Vars | Assignment | IfStmt | ReturnStmt | RepeatStmt | ReadStmt | WriteStmt

对应代码

```
TrieNode* Statement(TokenList* ToList){
```

```
    Token* CurToken;
    TrieNode* ChildNode;
    TrieNode* CurNode = (TrieNode*)malloc(1);
    InitTrieNode(CurNode, "Statement");

    CurToken = GetCurToken(ToList);
    if(CurToken==NULL) return NULL;
    char* CurStr = CurToken->string;

    if(CurToken->type==_Type){
        ChildNode = Vars(ToList);
    } else if(CurToken->type==_Key){
        ChildNode = Assignment(ToList);
    } else if(!strcmp(CurStr, "if")){
        ChildNode = IfStmt(ToList);
    } else if(!strcmp(CurStr, "return")){
        ChildNode = ReturnStmt(ToList);
    } else if(!strcmp(CurStr, "repeat")){
        ChildNode = RepeatStmt(ToList);
    } else if(!strcmp(CurStr, "read")){
        ChildNode = ReadStmt(ToList);
    } else if(!strcmp(CurStr, "write")){
        ChildNode = WriteStmt(ToList);
    } else {
        SetErrorToken(UknError, "unknown statement", CurToken);
        return NULL;
    }

    if(ChildNode==NULL) return NULL;
    AddChild(CurNode, ChildNode);

    return CurNode;
}
```

因为代码太多了，篇幅所限，其它文法略，具体见**parser.h**。

5 实验结果

5.1 语法树

样例代码

```
int main ()
    int x , y , fact;
    string str ;
    char ch ;
    x := 5 ;
    y := ( 2 + 4 ) * 5 ;
    str := "STR1!!!!" ;
    ch := 'a';

    if 0 < x then // don't compute if x <= 0
        fact := 1;
        repeat
            fact := fact * x;
            x := x - 1 ;
```

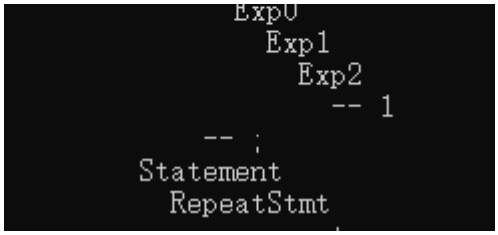
```
15          until x == 0
16          write fact ; // output factorial of
17      endif
18
19      write str ;
```

以下为生成的语法树。因为语法树过长，只截取其中的某些部分。完整的语法树见**trie.txt**。

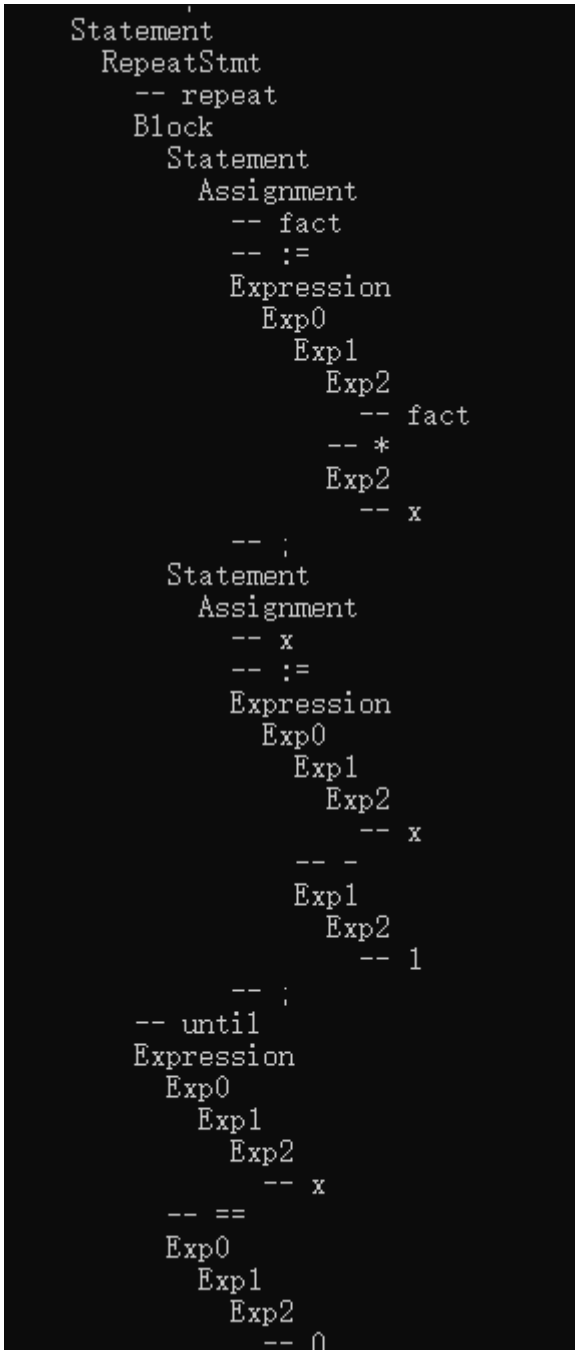
```
Parsing Tree :
Root
  Func
    -- int
    -- main
    -- (
    -- )
    Block
      Statement
        Vars
          -- int
          -- x
          -- ,
          -- y
          -- ,
          -- fact
          -- ;
      Statement
        Vars
          -- string
          -- str
          -- ;
      Statement
        Vars
          -- char
          -- ch
          -- ;
      Statement
        Assignment
          -- x
          -- :=
          Expression
            Exp0
              Exp1
                Exp2
                  -- 5
          -- ;
      Statement
        Assignment
          -- y
          -- :=
          Expression
```

以上为程序的开始部分，主要是一些定义和赋值。

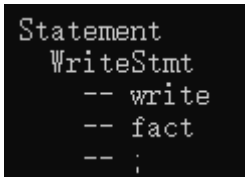
```
  Statement
    IfStmt
      -- if
      Expression
        Exp0
          Exp1
            Exp2
              -- 0
        -- <
        Exp0
          Exp1
            Exp2
              -- x
      -- then
      Block
        Statement
          Assignment
            -- fact
            -- :=
            Expression
```



以上为if语句接上循环语句的语法树。



以上为循环语句的语法树。



这是write语句的语法树。

5.2 编译错误展示

样例代码

```
int main ()
    int x , fact ;
    x := 5 ;

    if 0 < x  // don't compute if x <= 0
      fact := 1;
      repeat
        fact := fact * x;
        x := x - 1 ;
      until x == 0
      write fact ; // output factorial of x
```

编译器输出

```
wcy[1220LAPTOP-943J2FC]:/mnt/c/WCY/sysu_study/20_spring/complier/homework/lab1/mywork$ ./test.exe code1.tny
Lexical Analysis Success
Tokens:
int main ( ) int x , fact ; x := 5 ; if 0 < x fact := 1 ; repeat fact := fact * x ; x := x - 1 ; until x == 0 write fact ;
Syntax Analysis Fail
code1.tny:5:4 Syntex Error: expect ';' after vars
```

词法分析正确，语法分析错误，if语句错误。