```
if( FOLLOWPositions(ch).length == 2 ) {
            if( FOLLOWPositions(ch)(1).toString == "T" ) {
                result += FIRST_Group( FOLLOWPositions(ch)(0).toString )
                FOLLOW_Group(ch) = result.distinct
            }
            else if( FOLLOWPositions(ch)(1).toString == "W" ) {
                result += dfsFOLLOW( FOLLOWPositions(ch)(0).toString )
                FOLLOW_Group(ch) = result.distinct
            }
        }
        FOLLOW_Group(ch).replace("ε", "")
    }
```

```
def analyse( expression: String ): Boolean = {
        val stack = new mutable.Stack[String]()
        var localExpression = expression
        val table = createMatrix()
        val localVT = VT
        val localVN = VN
        val localRelations = relations
        stack.push("#")
        stack.push( localRelations(0)._1 )
        var cnt = 0
        staticAnalyseList.append(new Analyse("步骤","分析栈","剩余字符串","所用产生式","动作"));
        staticAnalyseList.append(new                                    Analyse(cnt.toString,
displayStack(stack).reverse.toString,localExpression.toString,"","initiate"));
        while( stack.isEmpty == false ) {
            val stackTop = stack.top
            stack.pop()
            // 栈顶符号属于   非终结符
            if( localVN.contains(stackTop) == true ) {
                // 栈顶符号与表达式左端首字符   存在   关系
                if( table( getRow(stackTop) )( getColumn( localExpression(0).toString ) ) != null ) {
                    val                             lastHalf                                =
table( getRow(stackTop) )( getColumn( localExpression(0).toString ) ).split( "->", 2 ).last
                    val length = lastHalf.length
                    for( i <- 0 to (length - 1) ) {
                        if( lastHalf != "ε" ) {
                            stack.push(lastHalf(length - 1 - i).toString)
                        }
                    }
                    cnt += 1
```