**编译原理课程设计**

**代码说明文档**

罗文水

计算机科学与工程学院

专业：计算机科学与技术

2021.05

目录

[一 文档结构说明 3](#_Toc71385761)

[二 问题一：词法分析文档说明 3](#_Toc71385762)

[三 问题二：语法分析文档说明 4](#_Toc71385763)

[四 问题三：语义分析文档说明 5](#_Toc71385764)

[五 程序运行方法 5](#_Toc71385765)

[5.1词法分析 5](#_Toc71385766)

[5.2 语法分析 6](#_Toc71385767)

[5.3 语义分析 7](#_Toc71385768)

[六 程序使用与扩展方法 8](#_Toc71385769)

# 一 文档结构说明

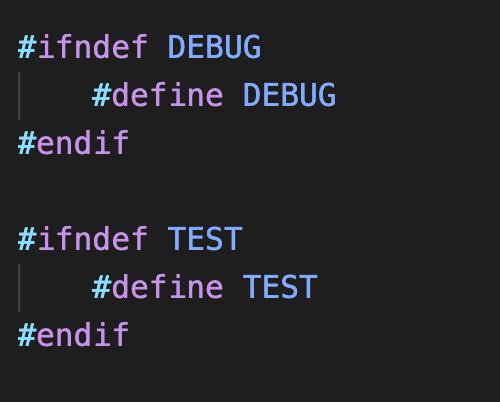
本次课程设计中涉及的文档以及结构如下所示：

* 1. **文件夹 lexical：**
     1. 文件 code.txt
     2. 文件 code2.txt
     3. 文件 code3.txt
     4. 文件 lexical-rule.txt
     5. 文件 lexical-rule2.txt
     6. 文件 lexical-rule3.txt
     7. 代码文档 lexicalAnalysis.cpp
     8. 头文件 lexicalAnalysis.h
     9. 文件 problem3\_code.txt
     10. 文件 token.txt
  2. 文件夹 grammer
     1. 文件 error\_file.txt
     2. 文件 grammer\_rule.txt
     3. 文件 grammer\_rule2.txt
     4. 代码文档 grammerAnalysis.txt
     5. 头文件 grammerAnalysis.h
     6. 文件 test\_rule.txt
     7. 文件 test\_rule2.txt
     8. 文件 test\_rule3.txt
     9. 文件 test\_rule4.txt
     10. 文件 test\_rule5.txt
     11. 文件 test\_rule6.txt
  3. 文件夹 semantic
     1. 文件 error\_file.txt
     2. 文件 rule.txt
     3. 代码文档 semanticAnalysis.cpp
     4. 头文件 semanticAnalysis.h
     5. 测试文件 test1.cpp

# 二 问题一：词法分析文档说明

词法分析程序文件主要为lexicalAnalysis.cpp，其中大部分数据结构抽离并放置于lexicalAnalysis.h中。

在lexicalAnalysis.cpp中，有两个宏分别用于测试和调试输出，如下所示：



将其中DEBUG宏取消注释之后将会得到NFA转DFA过程描述，即子集法算法每一步的执行过程（包括标记DFA节点，弧转换等）。

将其中TEST宏取消注释之后将会得到四个单项DFA的测试输出结果，即cpp文档中构造的测试序列（标识符、运算符、限定符、常量）分别在DFA上的测试结果，返回结果是false或者true表示符号串是否符合词法分析文件lexical-rule2.txt中指定的规则。程序的一部分测试输出如下所示（测试科学计数法）：

4234 : true

44323e10 : true

.1E10 : true

.1e-10 : true

103213.3213e+10 : true

.3123 : true

4234i : true

44323e10i : true

.231E10i : true

.312e-10i : true

.3213e+10i : true

e10 : false

e++10 : false

. : false

e : false

.43f3e10 : false

.e10 : false

将两个宏同时注释掉之后将会得到程序需要的输出，包含读取的产生式，字母表，非终结符对整数的索引表，NFA转移表，DFA状态转移表以及token序列。

词法分析的产生式规则输入文档为lexical-rule2.txt。其中包括四个规则，分别对应于常量、标识符、限定符和运算符规则，对于每一类，第一个符号对应于开始符号，而由于三型文法，终止状态只有一个，故无需指定，由程序制定（以非终结符数量为标准进行自动指定）。

本文中的产生式规则应该满足：对任意产生式，左侧只有一个非终结符，右侧非终结符在左，终结符号在右，且都是一个字符。

# 三 问题二：语法分析文档说明

词法分析主要程序代码在grammerAnalysis.cpp中，其中大部分数据结构抽离在grammerAnalysis.h中。在cpp程序文件中，同样有一个DEBUG宏用于控制程序的过程性输出，本程序中的DEBUG宏用于控制如下三种过程性输出：

* 求FIRST集合算法过程中，对于每个非终结符是否能推出epsilon的判断，该算法分为三个阶段进行迭代，直到所有的非终结符都确定是否能推出epsilon则终止迭代。
* FIRST集合求解的迭代过程，该过程中算法一共有五种情况，一旦first集合发生变化就控制输出，直到FIRST集合全部收敛。
* 可归前缀DFA状态转移图，使用map进行存储，debug过程输出map中每个节点可经过弧转换到达状态的信息（以该信息构建ACTION和GOTO表）。

本问题中，error\_file.txt用于输出错误信息。test-rule.txt、test-rule2.txt、test-rule3.txt用于测试符号的FIRST集合是否求解正确（经过本算法的测试，能正确求解所有合法二型文法的FIRST集合）。其次，test-rule4.txt、test-rule5.txt、test-rule6.txt用于测试项目集规范族求解算法以及LR(1)分析是否正确，经过本程序的测试，所有合法LR(1)文法均能通过测试并自动构建可归前缀自动机。

注意：使用测试FIRST集合的文件进行测试时会在随后求LR过程中报assertion错误，因为只求FIRST集合而不指定LR分析串会导致本人定义的assert错误，并导致操作系统自陷。但过程中对于FIRST集合等的求解结果都将在assertion错误之前显示。测试项目集规范族算法时，ACTION和GOTO表也将在assertion错误之前显示。

注：所有的测试样例都为《编译原理》教材中的典型例子。

在正确的按照流程执行问题一、二的代码后，将不会遇到assertion错误，并显示正确结果。

本程序的**产生式规则文件**为**grammer-rule2.txt**，该文档中规定产生式规则，非终结符号使用英文双引号进行包围，终结符号使用单引号进行标注。文档结尾有一个$符号表示产生式录入结束。Grammer-rule.txt为产生式测试文件，其中的产生式为部分C语言的产生式，测试正确后方修改为自定义语言产生式。

# 四 问题三：语义分析文档说明

语义分析文档根据问题二LR(1)分析法程序进行二次开发。产生式规则在semantic文件夹下的rule.txt中指定（属性文法）。错误信息提示文件在semantic文件夹下的error\_file.txt文件中指定。

# 五 程序运行方法

## 5.1词法分析

进入目录lexical，进入lexicalAnalysis.cpp文件，该程序使用C++编写，编辑环境为vscode，使用**C++11标准**，使用任何可以运行**C++11标准**代码的环境并且**点击运行**即可得到结果（保证在环境中能够链接到目录下的txt文件以及头文件）。对于本词法分析，使用的分析程序在code.txt中。正确运行后会得到如下tokenList（部分）并且该结果将会保存到token.txt文件中，若出现错误则错误信息将打印在控制台:

token表：

( 1 , 关键字 , 'func' )

( 1 , 标识符 , 'main' )

( 1 , 限定符 , '(' )

( 1 , 关键字 , 'Void' )

( 1 , 标识符 , 'args' )

( 1 , 限定符 , ',' )

( 1 , 关键字 , 'Double' )

( 1 , 标识符 , 'args2' )

( 1 , 限定符 , ',' )

( 1 , 关键字 , 'String' )

( 1 , 标识符 , 'args3' )

( 1 , 限定符 , ')' )

( 1 , 运算符 , '-' )

( 1 , 运算符 , '>' )

( 1 , 关键字 , 'Void' )

( 1 , 限定符 , '{' )

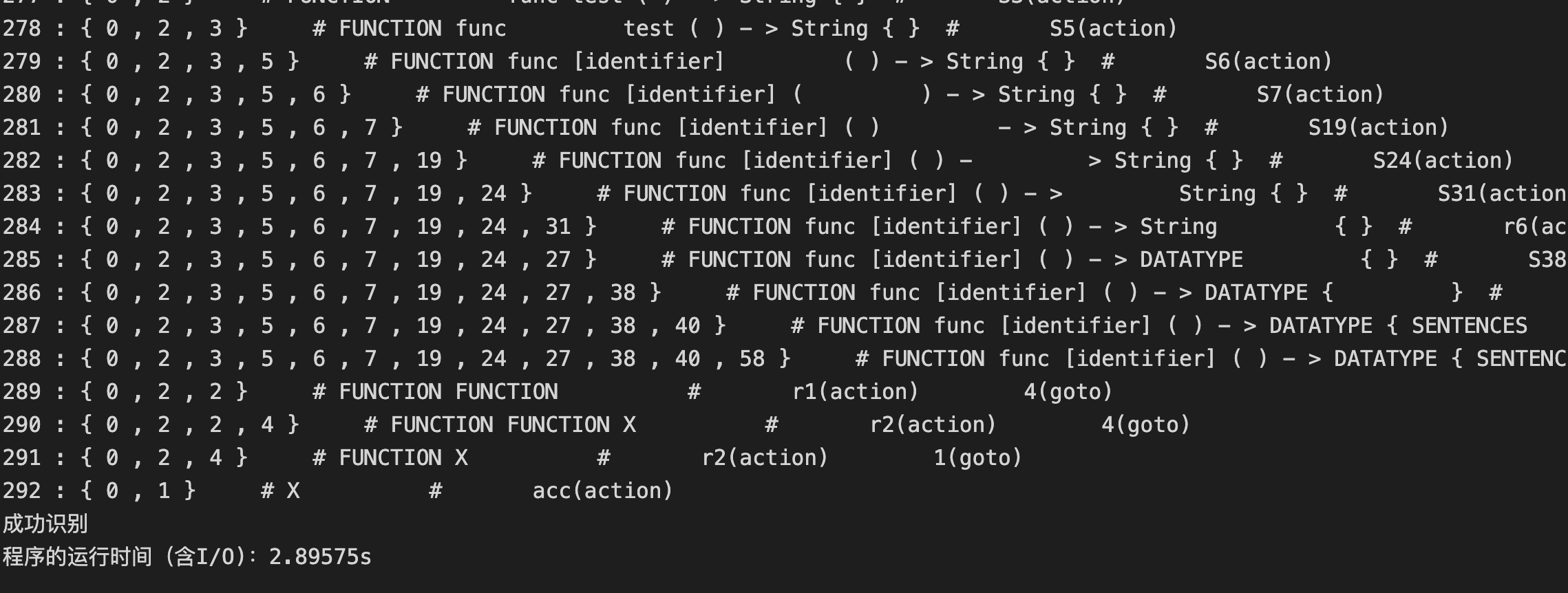
( 2 , 标识符 , 'function' )

**说明**：其余文档lexical\_rule 、lexical\_rule3文档都是产生式的测试文档，在运行时不必考虑。而code2.txt与code3.txt也是程序编写过程中的测试文档，若不修改代码，则无需考虑，直接运行默认程序。

## 5.2 语法分析

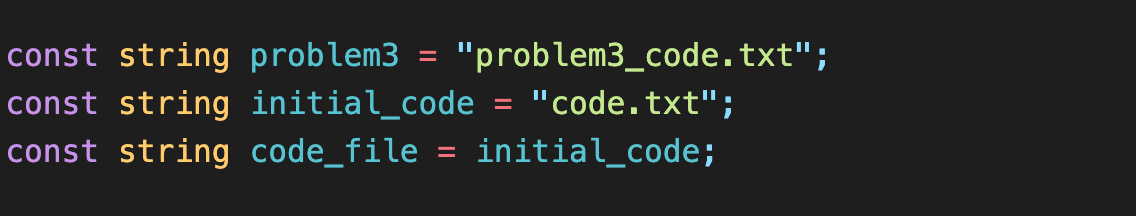
前提条件：执行完词法分析程序，并且不改变默认的文件目录。

执行方法：进入grammer文件夹中，进入grammerAnalysis.cpp文件，该程序使用C++编写，编辑环境为vscode，使用**C++11标准**，使用任何可以运行**C++11标准**代码的环境并且**点击运行**即可得到结果（保证在环境中能够链接到目录下的txt文件以及头文件）。使用默认的程序以及语法分析的产生式，直接运行得到程序的分析结果。包括ACTION和GOTO分析表以及程序的LR(1)识别过程，若出现错误将会将错误同时显示在控制台以及error\_file.txt文件中。结果部分如下图所示：



## 5.3 语义分析

前提条件：修改lexical/lexicalAnalysis.cpp中的字符号串常量定义（文件名称定义），如下图所示，将其中的code\_file=initial\_code，修改为code\_file=problem3以链接问题三的计算式文件（从而生成问题三的输入token词例表）。在**C++11环境**下运行代码。



计算式书写文件：lexical/lexicalAnalysis.cpp文件（需符合指定文法，否则会提示错误）。

执行方法：进入semantic/semanticAnalysis.cpp文件中，在**C++11环境**下运行代码即可得到程序输出，程序输出包括LR（1）分析过程以及语义动作、最终的四元式序列。四元式的输出结果应当如下所示（默认表达式的分析，有两种表述形式的四元式，情况分别列出）：

t1:=10

t2:=t1

t3:=t2

t4:=10

t5:=t4

t6:=321

t7:=t6

t8:=t7

t9:=213

t10:=t9

t12:=321

t13:=t12

t15:=t14

t17:=321

t19:=t18

t20:=t19

t21:=t20

t22:=32

t23:=t22

t24:=t23

t25:=321

t26:=t25

t28:=t27

t31:=t30

t32:=t31

t33:=t32

( := ,10 , \_ , t1 )

( := ,t1 , \_ , t2 )

( := ,t2 , \_ , t3 )

( := ,10 , \_ , t4 )

( := ,t4 , \_ , t5 )

( := ,321 , \_ , t6 )

( := ,t6 , \_ , t7 )

( := ,t7 , \_ , t8 )

( := ,213 , \_ , t9 )

( := ,t9 , \_ , t10 )

( - , t8 , t10 , t11 )

( := ,321 , \_ , t12 )

( := ,t12 , \_ , t13 )

( + , t11 , t13 , t14 )

( := ,t14 , \_ , t15 )

( \* , t5 , t15 , t16 )

( := ,321 , \_ , t17 )

( \* , t16 , t17 , t18 )

( := ,t18 , \_ , t19 )

( := ,t19 , \_ , t20 )

( := ,t20 , \_ , t21 )

( := ,32 , \_ , t22 )

( := ,t22 , \_ , t23 )

( := ,t23 , \_ , t24 )

( := ,321 , \_ , t25 )

( := ,t25 , \_ , t26 )

( + , t24 , t26 , t27 )

( := ,t27 , \_ , t28 )

( \* , t21 , t28 , t29 )

( + , t3 , t29 , t30 )

( := ,t30 , \_ , t31 )

( := ,t31 , \_ , t32 )

( := ,t32 , \_ , t33 )

# 六 程序使用与扩展方法

**程序对于任意词法规则的构造与扩展需要满足如下条件：**

* 词法分析中产生式规则仅修改lexical-rule2.txt文件，其中每一类规则的修改都在指定区域内，比如对于常量的产生式修改必须在[scientific-format]以及下一个类型的表示符号之间修改。
* epsilon使用@符号表示。
* 产生式右侧，第一个符号是非终结符，第二个符号是终结符，产生式左边第一个符号是非终结符，且每一个符号只占一个字符（本文档书写规则，同时也是三型文法的要求，非终结符与终结符号的大小写无关，只关注位置）。

**程序对于任意语法分析规则的构造与扩展需要满足如下条件：**

* 语法分析的规则在grammer-rule2.txt文件中修改。
* 产生式满足非终结符用英文双引号进行包括，终结符使用英文单引号进行包括。
* 标识符务必使用[identifier]并使用英文单引号进行包括。
* epsilon使用@符号表示。并且使用英文单引号进行包括。
* 若修改成自身指定规则的语言，需关注词法和语法分析程序头文件中的keywordList表，修改其中的关键字，否则可能导致关键字识别成标识符的问题。