



南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

# 软件设计 (II)

作者: 刘明承

学号: 919106840423

班级: 9191062301

学院: 计算机科学与工程学院

专业: 计算机科学与技术

## 一、 课程设计总览:

### 1. 课程设计要求

- (1) 创建一个词法分析程序，该程序支持分析常规单词。
  - a) 输入：一个文本文档，包括一组 3° 型文法（正规文法）的产生式。
  - b) 输出：一个源代码文本文档，包含一组需要识别的字符串（程序代码）。
  - c) 要求：词法分析程序可以准确识别：科学计数法形式的常量（如 0.314E+1），复数常量（如10+12i），可检查整数常量的合法性，标识符的合法性（首字符不能为数字等），尽量符合真实常用高级语言要求的规则。
- (2) 创建一个使用 LL(1) 方法或 LR(1) 方法的语法分析程序。
  - a) 输入：包含 2° 型文法（上下文无关文法）的产生式集合的文本文档；任务 1 词法分析程序输出的（生成的）token 令牌表。
  - b) 输出：YES 或 NO（源代码字符串符合此 2° 型文法，或者源代码字符串不符合此 2° 型文法）；错误提示文件，如果有语法错标示出错误行号，并给出大致的出错原因。

### 2. 项目基本内容

- (1) 词法分析器
  - a) 输入：自定义正规文法文件、程序源代码
  - b) 输出：Token串（三元组）
  - c) 流程：读入正规文法，生成对应NFA，确定化NFA，根据生成的DFA对输入源程序代码进行分析，产生Token串
- (2) 语法分析器
  - a) 输入：自定义2型文法文件、词法分析产生的Token串
  - b) 输出：LR（1）分析信息（项目集族、状态信息等）、错误信息、分析结果
  - c) 流程：
    - i) 第一步，读入2型文法，构造项目族规范集，根据LR（1）分析法产生对应的DFA、Action表、Goto表。
    - ii) 第二步，读入Token串，根据LR（1）的产物进行语法分析。

### 3. 开发环境

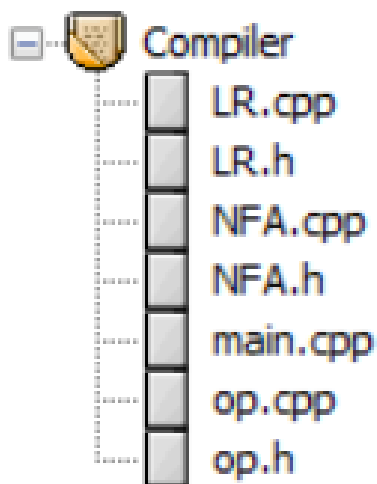
- (1) 语言：C++
- (2) 硬件环境：Windows10 + DevC++5.11

#### 4. 项目结构

(1) 文件说明:

- a) 文件夹
  - i) Compiler 项目文件夹
  - ii) Lexer 子项目词法分析器
  - iii) Parser 子项目语法分析器
- b) .cpp/.h
  - i) LR LR (1) 分析法
  - ii) NFA 不确定、确定的有穷自动机
  - iii) Op 将复杂的文法描述转换成字母符号产生式
- c) .txt (输入、输出文件)
  - i) Input.txt/output.txt: 输入、输出测试
  - ii) Regular grammar.txt: 正规文法
  - iii) Source code.txt: 源程序代码
  - iv) Lexer.txt: Token (行号、内容、类别) 序列
  - v) Describe.txt: 语法描述
  - vi) 2NF grammar.txt: 语法对应2型文法
  - vii) Parser.txt: 程序输出

(2) 项目结构:



- a) Compiler

名称	修改日期	类型	大小
Lexer	2022/5/5 16:39	文件夹	
Parser	2022/5/5 15:58	文件夹	
Compiler.dev	2022/5/5 16:58	Dev-C++ Projec...	2 KB
Compiler.exe	2022/5/5 16:06	应用程序	3,199 KB
Compiler.layout	2022/5/5 16:58	LAYOUT 文件	1 KB
main.cpp	2022/5/5 16:00	C++ Source File	1 KB
main.o	2022/5/5 16:00	O 文件	196 KB
Makefile.win	2022/5/5 16:52	WIN 文件	2 KB
op.cpp.cpp	2022/5/5 15:51	C++ Source File	0 KB
op.cpp.o	2022/5/5 15:51	O 文件	1 KB
op.cpp	2022/5/5 16:06	C++ Source File	3 KB
op.h	2022/5/5 15:47	C Header File	1 KB
op.o	2022/5/5 16:06	O 文件	129 KB
项目说明.txt	2022/5/2 22:09	文本文档	1 KB

#### b) Lexer

名称	修改日期	类型	大小
input.txt	2022/4/30 11:41	文本文档	2 KB
lexer.txt	2022/5/5 16:52	文本文档	2 KB
NFA.cpp	2022/5/5 15:40	C++ Source File	11 KB
NFA.h	2022/5/5 15:40	C Header File	2 KB
NFA.o	2022/5/5 15:51	O 文件	965 KB
output.txt	2022/5/5 16:52	文本文档	8 KB
regular grammar.txt	2022/5/1 23:10	文本文档	2 KB
source code.txt	2022/5/5 14:59	文本文档	1 KB

#### c) Parser

名称	修改日期	类型	大小
2NF grammar.txt	2022/5/5 16:52	文本文档	1 KB
describe.txt	2022/5/5 14:39	文本文档	2 KB
LR.cpp	2022/5/5 15:55	C++ Source File	21 KB
LR.h	2022/5/5 15:58	C Header File	3 KB
LR.o	2022/5/5 15:58	O 文件	1,450 KB
parser.txt	2022/5/5 16:52	文本文档	0 KB

## 二、词法分析器 (Lexer)

### 1. 实现原理

#### (1) 识别类型

类型	具体说明
关键字	"if"、"then"、"else"、"bool"、"char"、"int"、"double"、"const"、 "while"、"do"、"begin"、"end"
标识符	可以由数字、字母、下划线组成。
常量	可以识别整数、小数、科学计数法、复数
运算符	单目运算符: "=", "+", "-", ">", "<", "*", "/", "%"
	双目运算符: "==", ">=", "<="等
界符	";", "(", ")", "{", "}", ",", "

#### (2) 正规文法

```
regular grammar.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
Start -> Token|Operator|Separator|Constant|Key
//Token
Token-> aAD|bAD|cAD|dAD|eAD|fAD|gAD|hAD|iAD|jAD|kAD|lAD|mAD|nAD|oAD|pAD|qAD|rAD|sAD|tAD|uAD|vAD|wAD|xAD|yAD|zAD|_AD
AD-> $|aAD|bAD|cAD|dAD|eAD|fAD|gAD|hAD|iAD|jAD|kAD|lAD|mAD|nAD|oAD|pAD|qAD|rAD|sAD|tAD|uAD|vAD|wAD|xAD|yAD|zAD|_AD|1AD|2AD|3AD|4AD|5AD|6AD|7AD|8AD|9AD|0AD
//Operator
Operator-> +E|-E|*E|/E|%E|=E|<E|>E
E-> $|=
//Separator
Separator-> ,|;|(|)|\t|\n|0
//Key
//Constant
Constant-> A|B|C|D
A-> 1A1|2A1|3A1|4A1|5A1|6A1|7A1|8A1|9A1|0
A1-> 0A1|1A1|2A1|3A1|4A1|5A1|6A1|7A1|8A1|9A1|$
B-> 1B1|2B1|3B1|4B1|5B1|6B1|7B1|8B1|9B1|0B2
B2-> .B3
B3-> 1B3|2B3|3B3|4B3|5B3|6B3|7B3|8B3|9B3|0B3|$
B4-> 1B4|2B4|3B4|4B4|5B4|6B4|7B4|8B4|9B4|0B4
B1-> 1B1|2B1|3B1|4B1|5B1|6B1|7B1|8B1|9B1|0B1|B4
B4-> .B3
C-> 1C1|2C1|3C1|4C1|5C1|6C1|7C1|8C1|9C1
C1-> 0C1|1C1|2C1|3C1|4C1|5C1|6C1|7C1|8C1|9C1|C2|C2
C2-> $|+C
D-> 1D1|2D1|3D1|4D1|5D1|6D1|7D1|8D1|9D1
D1-> 1D1|2D1|3D1|4D1|5D1|6D1|7D1|8D1|9D1|0D1|.D2|$eD3
D2-> 0D2|1D2|2D2|3D2|4D2|5D2|6D2|7D2|8D2|9D2|eD3
D3-> +D4|-D4
D4-> 1D5|2D5|3D5|4D5|5D5|6D5|7D5|8D5|9D5
D5-> 1D5|2D5|3D5|4D5|5D5|6D5|7D5|8D5|9D5|0D5|$
```

说明:

- 内容类别由正规文法第一行产生式右部确定: Token-标识符、Operator-运算符、Separator-界符、Constant-常量、Key-关键字
- 根据此正规文法, 可识别“识别类型”中全部类别
- 文法已由符号代替非终结符, 并作正规化, 该步骤独立与机器手动完成
- 程序读入时自动跳过“//”注释之后该行的内容

### 2. 重要类与结构定义

#### (1) NFA 不确定的有穷自动机

```

class NFA
{
public:
    friend class LA;

    ofstream fout;          /* 文件输出 */
    bool isDefinated;        /* 是否已经确定化 */
    int tot, ftot;           /* 状态总数, 终态总数 */
    string line, VT, VN;     /* 输入行、非终结符、终结符 */
    string Start;            /* 初态 */
    vector<bool> FinalSet;    /* 终态集 */
    set<char> CharSet;       /* VT集合 */
    map<string,int> Map;      /* VN集合 */
    vector<vector<psc>> List; /* 状态转换图 */
    map<string,int> Tag;      /* 令牌集合 */
    map<string,int> Symbol;   /* 各状态令牌表 */

    NFA();
    void Input( string file1 );
    void Print( string file2 );
    void cope_line( string line );
    void cope_formula( string formula );
    string get_first_VN( string line );
    string get_next_VN( string& line );
    string get_next_VT( string& line );
    void get_Symbols( string line );
    void attach( string VN );
    void nfa_2_dfa();
    void e_closure( string status, vector<string>& V );
    void vt_move( char vt, vector<string> V_closure, vector<string>& T );
    void print_state( vector<string> state );
};

```

## (2) LA 词法分析

```

class LA /* Lexical Analysis */
{
public:
    ofstream fout;

    string line;          /* 输入行 */
    string word;          /* 当前处理串 */
    string T;             /* 当前状态 */
    int index;            /* 当前状态对应List中下标 */

```

```

int id;          /* 输出编号 */

LA();

void analysis( NFA& nfa, string file3, string file2 );
void read_next_VT( char ch, NFA& nfa, string file2 );
void output( int tag, NFA& nfa, string file2 );
};

```

### (3) isEqualALL 比较类（在map中根据second找key）

```

class isEqualALL
{
public:
    explicit isEqualALL(char c) : ch(c) {}
    bool operator() (const std::pair<string, char>& element) const {
        return element.second == ch;
    }
private:
    const char ch;
};

```

## 3. 重要函数

```

void NFA::nfa_2_dfa()
/* NFA的图转化为 DFA的图 */

void NFA::cope_line( string line )
/* 每次处理一行，可能有多条产生式*/

void NFA::cope_formula( string formula )
/* 每次处理一条产生式 */

void NFA::e_closure( string status, vector<string>& T )
/* 给出一个状态，求$闭包 */

void NFA::vt_move( char vt, vector<string> V, vector<string>& T )
/* 在闭包V中找到vt弧转换， 将新状态放入T中 */

```

```

string NFA::get_first_VN( string formula )
/* 从输入种获取第一个VN，这是起始符 */

string NFA::get_next_VN( string & formula )
/* 从输入中得到下一个VN，并在line中去掉其与其之后的-> */

string NFA::get_next_VT( string & formula )
/* 从输入中得到下一个VT */

void LA::analysis( NFA& nfa, string file3, string file4 )
/* 根据NFA对象进行词法分析 */

void LA::read_next_VT( char ch, NFA& nfa, string file4 )
/* 处理下一个非终结符 */

```

\*具体实现见代码包

## 4. 运行结果

### (1) 输入源程



source code.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```

const int x = 8, y = 7;
int a,b;
begin
a = x + y;
b = x * a;
if a == 3 then b = 10;
while a == 3 do b = b+2;
end

```

### (2) 运行时相关信息

#### a) NFA信息

NFA  
 VN ( nonterminal character )  
 A A1 AD B B1 B2 B3 B4 C C1 C2 Constant D D1 D2 D3 D4 D5 E Final0 Final1 Final10 Final11 Final12 Final13 Final14 Final15 Final16 Final17 Final2 Final5  
 VT ( terminal character )  
 \$ % ( ) \* + , - . / 0 1 2 3 4 5 6 7 8 9 ; < = > \_ a b c d e f g h i j k l m n o p q r s t u v w x y z { }  
 Start ( initial state )  
 Start  
 Final ( final state )  
 Final0 Final1 Final2 Final3 Final4 Final5 Final6 Final7 Final8 Final9 Final10 Final11 Final12 Final13 Final14  
 Symbol  
 Constant 4 Key 5 Operator 2 Separator 3 Token 1  
 A 4 A1 4 AD 1 B 4 B1 4 B2 4 B3 4 B4 0 C 4 C1 4 C2 4 Constant 4 D 4 D1 4



b) NFA 状态-弧 (部分)

```

A: (1, A1); (2, A1); (3, A1); (4, A1); (5, A1); (6, A1); (7, A1); (8, A1); (9, A1); (0, Final12);
A1: (0, A1); (1, A1); (2, A1); (3, A1); (4, A1); (5, A1); (6, A1); (7, A1); (8, A1); (9, A1); ($, Final13);
AD: ($, Final0); (a, AD); (b, AD); (c, AD); (d, AD); (e, AD); (f, AD); (g, AD); (h, AD); (i, AD); (j, AD); (k, AD); (l, AD); (m, AD); (n, AD); (o, AD); (p, AD); (q, AD); (r, AD); (s, AD); (t, AD); (u, AD); (v, AD); (w, AD); (x, AD); (y, AD); (z, AD); (0, B1); (1, B1); (2, B1); (3, B1); (4, B1); (5, B1); (6, B1); (7, B1); (8, B1); (9, B1); (0, B2);
B1: (1, B1); (2, B1); (3, B1); (4, B1); (5, B1); (6, B1); (7, B1); (8, B1); (9, B1); (0, B1); ($, B4);
B2: (., B3);
B3: (1, B3); (2, B3); (3, B3); (4, B3); (5, B3); (6, B3); (7, B3); (8, B3); (9, B3); (0, B3); ($, Final14);
B4: (1, B4); (2, B4); (3, B4); (4, B4); (5, B4); (6, B4); (7, B4); (8, B4); (9, B4); (0, B4); (., B3);
C: (1, C1); (2, C1); (3, C1); (4, C1); (5, C1); (6, C1); (7, C1); (8, C1); (9, C1); (0, C1); (1, C1); (2, C1); (3, C1); (4, C1); (5, C1); (6, C1); (7, C1); (8, C1); (9, C1); (0, C1); (1, C1); (2, C1); (3, C1); (4, C1); (5, C1); (6, C1); (7, C1); (8, C1); (9, C1); (i, C2); ($, C2);
C2: ($, Final15); (+, C);
Constant: ($, A); ($, B); ($, C); ($, D);
D: (1, D1); (2, D1); (3, D1); (4, D1); (5, D1); (6, D1); (7, D1); (8, D1); (9, D1); (0, D1); (1, D1); (2, D1); (3, D1); (4, D1); (5, D1); (6, D1); (7, D1); (8, D1); (9, D1); (0, D1); (., D2); ($, Final16); (e, D3);
D2: (0, D2); (1, D2); (2, D2); (3, D2); (4, D2); (5, D2); (6, D2); (7, D2); (8, D2); (9, D2); (e, D3);
D3: (+, D4); (-, D4);
D4: (1, D5); (2, D5); (3, D5); (4, D5); (5, D5); (6, D5); (7, D5); (8, D5); (9, D5);
D5: (1, D5); (2, D5); (3, D5); (4, D5); (5, D5); (6, D5); (7, D5); (8, D5); (9, D5); (0, D5); ($, Final17);
E: ($, Final1); (=, Final2);
Final0: NULL
Final1: NULL
Final6: NULL
Final7: NULL
Final8: NULL
Final9: NULL
Key: NULL
Operator: (+, E); (-, E); (*, E); (/, E); (% , E); (=, E); (<, E); (>, E);
Separator: (., Final3); (:, Final4); ({, Final5); (}, Final6); ((, Final7); (}), Final8); ( , Final9); ( , Final10); ( , Final11);
Start: ($, Token); ($, Operator); ($, Separator); ($, Constant); ($, Key);
Token: (a, AD); (b, AD); (c, AD); (d, AD); (e, AD); (f, AD); (g, AD); (h, AD); (i, AD); (j, AD); (k, AD); (l, AD); (m, AD); (n, AD); (o, AD); (p, AD); (q, AD); (r, AD); (s, AD); (t, AD); (u, AD); (v, AD); (w, AD); (x, AD); (y, AD); (z, AD); (0, B1); (1, B1); (2, B1); (3, B1); (4, B1); (5, B1); (6, B1); (7, B1); (8, B1); (9, B1); (0, B2);

```

### c) DFA信息

DFA																							
VN ( nonterminal character )																							
	T0	T1	T10	T11	T12	T13	T14	T15	T16	T17	T18	T19	T2	T20	T21	T22	T23	T4	T5	T6	T7	T8	T9
VT ( terminal character )																							
	\$ % ( ) * + , . - / 0 1 2 3 4 5 6 7 8 9 ; < = > _ a b c d e f g h i j k l m n o p q r s t u v w x y z { }																						
Start ( initial state )																							
	T0																						
Final ( final state )																							
Symbol	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11	T12	T13	T14	T15	T18	T19	T20					
Constant 4	T1 4	T10 3	Key 5 T11 3	Operator 2 T12 3	T13 1	Separator 3 T14 4	T15 4	Token 1 T16 4	T17 4	T18 4	T19 2	T2 4	T20 4	T21 4	T22 4	T3 2	T4 3	T5 3					

### d) DFA 状态-弧

```

T0: (1, T1); (2, T1); (3, T1); (4, T1); (5, T1); (6, T1); (7, T1); (8, T1); (9, T1); (0, T2); (+, T3); (-, T3); (*, T3); (/, T3); (% , T3); (=, T3); (<, T3); (>, T3); (., T4); (:, T5); (|, T6);
, T11); (:, T12); (a, T13); (b, T13); (c, T13); (d, T13); (e, T13); (f, T13); (g, T13); (h, T13); (i, T13); (j, T13); (k, T13); (l, T13); (m, T13); (n, T13); (o, T13); (p, T13); (q, T13); (r
T1: (0, T1); (1, T1); (2, T1); (3, T1); (4, T1); (5, T1); (6, T1); (7, T1); (8, T1); (9, T1); (., T14); (i, T15); (+, T16); (e, T17);
T10: NULL
T11: NULL
T12: NULL
T13: (a, T13); (b, T13); (c, T13); (d, T13); (e, T13); (f, T13); (g, T13); (h, T13); (i, T13); (j, T13); (k, T13); (l, T13); (m, T13); (n, T13); (o, T13); (p, T13); (q, T13); (r, T13); (s, T
T14: (1, T14); (2, T14); (3, T14); (4, T14); (5, T14); (6, T14); (7, T14); (8, T14); (9, T14); (0, T14); (e, T17);
T15: (+, T16);
T16: (1, T20); (2, T20); (3, T20); (4, T20); (5, T20); (6, T20); (7, T20); (8, T20); (9, T20);
T17: (+, T21); (-, T21);
T18: (1, T18); (2, T18); (3, T18); (4, T18); (5, T18); (6, T18); (7, T18); (8, T18); (9, T18); (0, T18);
T19: NULL
T2: (., T18);
T20: (0, T20); (1, T20); (2, T20); (3, T20); (4, T20); (5, T20); (6, T20); (7, T20); (8, T20); (9, T20); (i, T15); (+, T16);
T21: (1, T22); (2, T22); (3, T22); (4, T22); (5, T22); (6, T22); (7, T22); (8, T22); (9, T22);
T22: (1, T22); (2, T22); (3, T22); (4, T22); (5, T22); (6, T22); (7, T22); (8, T22); (9, T22); (0, T22);
T3: (=, T19);
T4: NULL
T5: NULL
T6: NULL

```

### (3) 运行结果

(部分截图)

lexer.txt - 记事本  
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
(0,const,Token)
(1, ,Separator)
(2,int,Token)
(3, ,Separator)
(4,x,Token)
(5, ,Separator)
(6,=,Operator)
(7, ,Separator)
(8,8,Constant)
(9,,Separator)
(10, ,Separator)
(11,y,Token)
(12, ,Separator)
(13,=,Operator)
(14, ,Separator)
(15,7,Constant)
(16,,Separator)
(17,int,Token)
(18, ,Separator)
(19,a,Token)
(20,,Separator)
(21,b,Token)
(22,,Separator)
(23,begin,Token)
(24,a,Token)
(25, ,Separator)
(26,=,Operator)
(27, ,Separator)
(28,x,Token)
(29, ,Separator)
(30,+,Operator)
(31, ,Separator)
(32,y,Token)
(33,,Separator)
```

## 三、语法分析器（Parser）

### 1. 实现原理

- (1) 原理：先根据2型文法，利用LR（1）分析法构造项目集，进而得到DFA、Action表和Goto表。再利用词法分析的结果进行语法分析，输出相应信息

- (2) 文法

```
describe.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
<程序>→<语句>|<常量说明><语句>|<变量说明><语句>|<常量说明><变量说明><语句>|<语句><程序>|<常量说明><语句><程序>|<变量说明><语句><程序>|<常量说明><变
<常量说明>→const <常量定义><常量定义1>;|const <常量定义>;
<常量定义1>→,<常量定义>|,<常量定义><常量定义1>
<常量定义>→<变量类型>Token=Constant|Token=Constant
<变量说明>→<变量类型>Token<Token1>;|<变量类型>Token;
<Token1>→,Token|,Token<Token1>
<变量类型>→(bool|char|int|double)
<语句>→<赋值语句>|<条件语句>|<当循环语句>|<复合语句>
<赋值语句>→Token=<表达式>;
<表达式>→<项>|<项><项1>
<项1>→<加法运算符><项>|<加法运算符><项><项1>
<项>→<因子>|<因子><因子1>
<因子1>→<乘法运算符><因子>|<乘法运算符><因子><因子1>
<因子>→Constant|Token
<加法运算符>→(+|-)
<乘法运算符>→(*|/)
<条件语句>→if<条件>then<语句>|if<条件>then<语句>else<语句>
<条件>→<表达式><关系运算符><表达式>
<关系运算符>→(==|<|>|>=)
<当循环语句>→while<条件>do<语句>
<复合语句>→begin<语句>end|begin<语句><复合1>end
<复合1>→<语句>|<语句><复合1>
```

### 对应2型文法

```
2NF grammar.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
A->B|CB|DB|CDB|BA|CBA|DBA|CDBA
C->aEFb|aEb
F->c|cEF
E->Gdef|def
D->GdHb|Gdb
H->cd|cdH
G->g
B->|I|J|K|L
I->deMb
M->N|NO
O->PN|PNO
N->Q|QR
R->SQ|SQR
Q->f|d
P->h
S->i
J->jTkB|jTkB|B
T->MUM
U->m
K->nToB
L->pBq|pBVq
V->B|BV

*i :+h ;c -:h /i ;b <:m <=:m =:e ==:m >:m >=:m Constant:f Token:d Token1:H begin:p bool:g char:g const:a do:o double:g else:l end:q if:j int:g then:k while:n 变量类型:G
```

说明：

- i) 语法由describe.txt给出，由op转为2型文法并输出到2NF grammar.txt中。
- ii) 每个大写字母对应一个或一类非终结符，每个小写字母对应一个或一类终结符。
- iii) 最终分析的结果再由文件最后一行字典转换为字符对应的状态

### 2. 重要类与结构定义

- (1) LR（0）

```

class LR
{
public:
    char Start;           /* 起始符 */
    vector<char> VtSet;     /* 终结符集 */
    vector<char> VnSet;     /* 非终结符集 */
    vector<string> Productions; /* 产生式 */
    vector<vector<pair<int, char> > > DFA; /* DFA */
    vector<vector<pair<char, int> > > Action; /* Action表 */
    vector<vector<pair<char, int> > > Goto; /* Goto表 */

    virtual void input( string File );
    void analysis( string FileIn, string FileOut );
    void enclosure( vector<psi>& Closure );
    void collection_2_dfa();
    int verify_conflict( vector<psi> Closure );
    void report( int Error );
    void print_state( vector<psi> C );
    void build_table( vector<vector<psi> >& States );
};

```

## (2) LR (1)

```

class LR1 : public LR
{
public:
    map<char, string> First; /* 每个非终结符的First集的集合 */
    map<string, char> Dic;

    void generate_firstset();
    bool contains( string a, string b );
    string first( string remainder, string lookahead );

    virtual void input( string File );
    void analysis( string FileIn, string FileOut );
    void build_DFA();
    void build_table( vector<clcn>& Collections );
    void enclosure( clcn& C );
    int verify_conflict( clcn C );
};

```

```
void print_state( clcn C );

};
```

### 3. 重要函数

```
void LR::collection_2_dfa()
/* 根据项目集规范组得到DFA、Action、Goto */

void LR::build_table( vector<vector<psi> >& States )
/* 根据DFA构造Action表和Goto表 */
/* 规定0状态遇初态结束 */

void LR::enclosure( vector<psi>& Closure )
/* 把闭包根据规则扩充，不改变其状态 */

int LR::verify_conflict( vector<psi> Closure )
/* 检测闭包中是否存在冲突 */

void LR::report( int error )
/* 根据error的类型报告错误 */

void LR::input( string File )
/* 输入2型文法产生式 */
/* 因为是2型文法，故规定每个Vt、Vn长度都为1 */
/* Production第0位是产生式左部，后面是产生式右部 */

void LR1::analysis( string FileIn, string FileOut )
/* 读入词法分析结果，进行语法分析，输出 */

void LR1::generate_firstset()
/* 生成每个非终结符的First集的集合 */

void LR1::print_state( clcn C )
/* 输出项目集族 */

string LR1::first( string remainder, string lookahead )
/* 输入项目分隔符右边部分以及该项目向前搜索符，求给出字串的first集 */

void LR1::build_DFA()
/* 重写LR1中项目集规范族的构造，产生DFA、Action、Goto */

string UTF8ToGB(const char* str);
/* 读入一行包含中文字符的字符串，使其能够存入map中不产生乱码 */
```

```
void turn_to_2NF( string In, string Out );
/* 逐字符处理语法，生成2NF */
```

\*具体实现见代码包

#### 4. 运行结果

##### (1) 运行时相关信息

##### a) 各项目集规范族内容（部分截图）

序号即代表状态号，从结果来看，共186个状态

```
26 :
3->·I , djnp
3->·I , q
3->·J , djnp
3->·J , q
3->·K , djnp
3->·K , q
3->·L , djnp
3->·L , q
->·deMb , djnp
->·deMb , q
I->·jTkB , djnp
I->·jTkB , q
I->·jTkBlB , djnp
I->·jTkBlB , q
<->·nToB , djnp
<->·nToB , q
.->·pBVq , djnp
.->·pBVq , q
.->·pBq , djnp
.->·pBq , q
/->·B , q
/->B· , q
/->·BV , q
/->B·V , q
[ERROR] Shift-Reduce Conflict found while compiling.
Thu May 12 23:36:35 2022
-----
27 :
/->BV· , q
-----
28 :
```

36 :

Q->d· , h

Q->d· , i

Q->d· , m

[ERROR] Reduce-Reduce Conflict found while compling.

Thu May 12 23:36:35 2022

---

37 :

Q->f· , h

Q->f· , i

Q->f· , m

[ERROR] Reduce-Reduce Conflict found while compling.

Thu May 12 23:36:35 2022

---

38 :

T->M·UM , o

U->·m , df

---

39 :

M->·N , o

M->·NO , o

N->·Q , h

N->·Q , o

N->·QR , h

N->·QR , o

Q->·d , h

Q->·d , i

Q->·d , o

Q->·f , h

Q->·f , i

Q->·f , o

T->MU·M , o

139 :

L->pBV·q , #

L->pBV·q , adgjn

L->pBV·q , l

140 :

L->pBq· , #

L->pBq· , adgjn

L->pBq· , l

[ERROR] Reduce-Reduce Conflict found while compiling.

Thu May 12 23:36:36 2022

141 :

L->pBVq· , #

L->pBVq· , adgjn

L->pBVq· , l

[ERROR] Reduce-Reduce Conflict found while compiling.

Thu May 12 23:36:36 2022

142 :

K->nT·oB , #

K->nT·oB , adgjn

K->nT·oB , l

## b) DFA状态-弧 （部分截图）

DFA

0 : ( B,1 ) ( C,2 ) ( D,3 ) ( l,4 ) ( J,5 ) ( K,6 ) ( L,7 ) ( a,8 ) ( G,9 ) ( g,10 ) ( d,11 ) ( j,12 )

1 : ( B,1 ) ( A,186 ) ( C,2 ) ( D,3 ) ( l,4 ) ( J,5 ) ( K,6 ) ( L,7 ) ( a,8 ) ( G,9 ) ( g,10 ) ( d,11 )

2 : ( B,181 ) ( D,182 ) ( l,4 ) ( J,5 ) ( K,6 ) ( L,7 ) ( G,9 ) ( g,10 ) ( d,11 ) ( j,12 ) ( n,13 )

3 : ( B,179 ) ( l,4 ) ( J,5 ) ( K,6 ) ( L,7 ) ( d,11 ) ( j,12 ) ( n,13 ) ( p,14 )

4 :

5 :

6 :

7 :

8 : ( E,165 ) ( G,166 ) ( d,167 ) ( g,10 )



23 : ( l,15 ) ( J,16 ) ( K,17 ) ( L,18 ) ( d,19 ) ( j,20 ) ( n,21 ) ( p,22 ) ( V,24 ) ( q,25 ) (

24 : ( q,28 )

25 :

26 : ( l,15 ) ( J,16 ) ( K,17 ) ( L,18 ) ( d,19 ) ( j,20 ) ( n,21 ) ( p,22 ) ( B,26 ) ( V,27 )

### c) Action表

#### Action表

状态: 0: ( a, 8 ) ( g, 10 ) ( d, 11 ) ( j, 12 ) ( n, 13 ) ( p, 14 )  
 状态: 1: ( a, 8 ) ( g, 10 ) ( d, 11 ) ( j, 12 ) ( n, 13 ) ( p, 14 ) ( #, -1 )  
 状态: 2: ( g, 10 ) ( d, 11 ) ( j, 12 ) ( n, 13 ) ( p, 14 )  
 状态: 3: ( d, 11 ) ( j, 12 ) ( n, 13 ) ( p, 14 )  
 状态: 4: ( #, -20 ) ( a, -20 ) ( d, -20 ) ( g, -20 ) ( j, -20 ) ( n, -20 ) ( p, -20 )  
 状态: 5: ( #, -21 ) ( a, -21 ) ( d, -21 ) ( g, -21 ) ( j, -21 ) ( n, -21 ) ( p, -21 )  
 状态: 6: ( #, -22 ) ( a, -22 ) ( d, -22 ) ( g, -22 ) ( j, -22 ) ( n, -22 ) ( p, -22 )  
 状态: 7: ( #, -23 ) ( a, -23 ) ( d, -23 ) ( g, -23 ) ( j, -23 ) ( n, -23 ) ( p, -23 )  
 状态: 8: ( d, 167 ) ( g, 10 )  
 状态: 9: ( d, 158 )  
 状态: 10: ( d, -19 )  
 状态: 11: ( e, 155 )  
 状态: 12: ( d, 36 ) ( f, 37 )  
 状态: 13: ( d, 36 ) ( f, 37 )  
 状态: 14: ( d, 19 ) ( j, 20 ) ( n, 21 ) ( p, 22 )  
 状态: 15: ( d, -20 ) ( j, -20 ) ( n, -20 ) ( p, -20 ) ( q, -20 )  
 状态: 16: ( d, -21 ) ( j, -21 ) ( n, -21 ) ( p, -21 ) ( q, -21 )  
 状态: 17: ( d, -22 ) ( j, -22 ) ( n, -22 ) ( p, -22 ) ( q, -22 )  
 状态: 18: ( d, -23 ) ( j, -23 ) ( n, -23 ) ( p, -23 ) ( q, -23 )  
 状态: 19: ( e, 121 )  
 状态: 20: ( d, 36 ) ( f, 37 )  
 状态: 21: ( d, 36 ) ( f, 37 )  
 状态: 22: ( d, 19 ) ( j, 20 ) ( n, 21 ) ( p, 22 )  
 状态: 23: ( d, 19 ) ( j, 20 ) ( n, 21 ) ( p, 22 ) ( q, 25 )  
 状态: 24: ( q, 28 )  
 状态: 25: ( #, -42 ) ( a, -42 ) ( d, -42 ) ( g, -42 ) ( j, -42 ) ( n, -42 ) ( p, -42 )  
 状态: 26: ( d, 19 ) ( j, 20 ) ( n, 21 ) ( p, 22 ) ( q, -44 )  
 状态: 27: ( q, -45 )  
 状态: 28: ( #, -43 ) ( a, -43 ) ( d, -43 ) ( g, -43 ) ( j, -43 ) ( n, -43 ) ( p, -43 )

#### d) Goto表（部分截图）

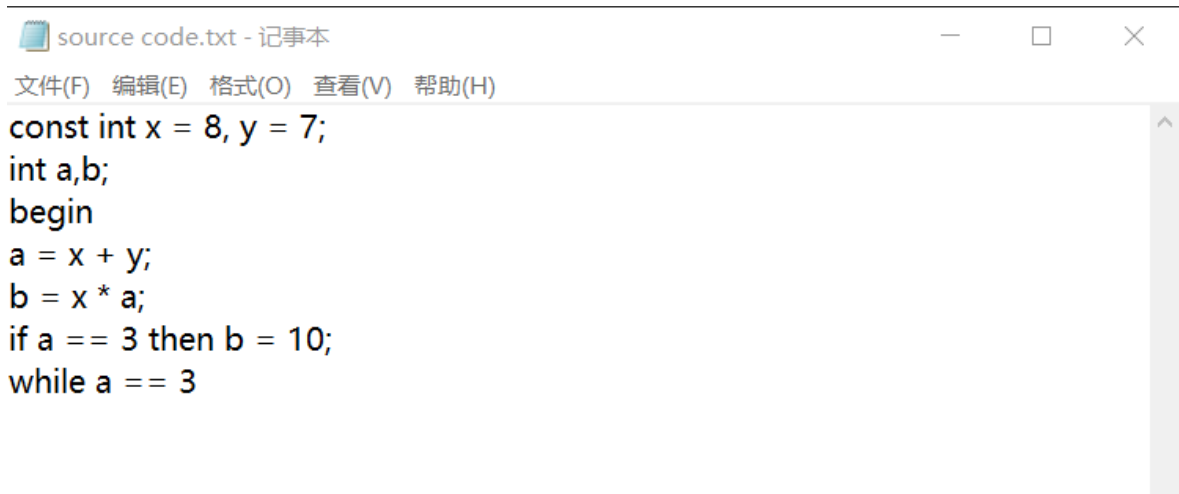
Goto表

```
状态: 0: (B, 1) (C, 2) (D, 3) (I, 4) (J, 5) (K, 6) (L, 7) (G, 9)
状态: 1: (B, 1) (A, 186) (C, 2) (D, 3) (I, 4) (J, 5) (K, 6) (L, 7) (G, 9)
状态: 2: (B, 181) (D, 182) (I, 4) (J, 5) (K, 6) (L, 7) (G, 9)
状态: 3: (B, 179) (I, 4) (J, 5) (K, 6) (L, 7)
状态: 4:
状态: 5:
状态: 6:
状态: 7:
状态: 8: (E, 165) (G, 166)
状态: 9:
状态: 10:
状态: 11:
状态: 12: (T, 127) (N, 34) (Q, 35) (M, 67)
状态: 13: (T, 124) (N, 34) (Q, 35) (M, 38)
状态: 14: (I, 15) (J, 16) (K, 17) (L, 18) (B, 23)
状态: 15:
状态: 16:
状态: 17:
状态: 18:
状态: 19:
状态: 20: (T, 66) (N, 34) (Q, 35) (M, 67)
状态: 21: (T, 33) (N, 34) (Q, 35) (M, 38)
状态: 22: (I, 15) (J, 16) (K, 17) (L, 18) (B, 29)
状态: 23: (I, 15) (J, 16) (K, 17) (L, 18) (V, 24) (B, 26)
状态: 24:
状态: 25:
状态: 26: (I, 15) (J, 16) (K, 17) (L, 18) (B, 26) (V, 27)
状态: 27:
状态: 28:
状态: 29: (I, 15) (J, 16) (K, 17) (L, 18) (V, 30) (B, 26)
状态: 30:
...
```

#### (2) 运行结果

```
79 # Acc
Yes
-----
Process exited after 3.215 seconds with return value 0
请按任意键继续. . .
```

将输入远程序改写后（去掉while-do语句的后半部分），重新执行：



```
source code.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
const int x = 8, y = 7;
int a,b;
begin
a = x + y;
b = x * a;
if a == 3 then b = 10;
while a == 3
```

输出

```
[ERROR] Wrong state visited while compling.
Thu May 12 23:46:21 2022
Informations in State Stack(top to bottom): 44 39 38 21 26 26 23 14 182 2 0
No
```