编译原理第一次小作业

软件42 张智晴 2014013458

1. 语法分析程序的主要任务

```
* 分析源程序的单词流是否符合语言的语法规则
* 报告语法错误
* 产生源程序的语法分析结果，以语法分析树或与之等价的形式体现出来
```

2. 自顶向下分析思想

```
* 从文法开始符号出发进行推导;每一步推导都获得文法的一个句型;
  直到产生出一个句子,恰好是所期望的终结符串
* 每一步推导是对当前句型中剩余的某个非终结符进行扩展,
  即用该非终结符的一个产生式的右部替换该非终结符
* 如果不存在任何一个可以产生出所期望的终结符串的推导,则表明存在语法错误
```

2. 推导

```
    S -> a | & | (T)
    T -> T,S | S

    (a, (a, a))的最左推导:
    S=>(T)=>(T,S)=>(S,S)=>(a,S)=>(a,(T))=>(a,(T,S))=>(a,(S,S))
    =>(a,(a,S))=>(a,(a,a))

    (((a,a), &, (a)),a)的最左推导:
    S=>(T)=>(T,S)=>(S,S)=>((T),S)=>((T,S),S)=>((T,S,S),S)
    =>(((T),S,S),S)=>(((T,S),S,S),S)=>(((S,S),S,S),S)
    =>(((a,S),S,S),S)=>(((a,a),S,S),S)=>(((a,a),&,S),S)
    =>(((a,a),&,(T)),S)=>(((a,a),&,(S)),S)=>(((a,a),&,(a)),S)
    =>(((a,a),&,(a)),a)
```

3. 判断LL(1)文法

(1)

```
S -> AB | PQx
A -> xy
B -> bc
P -> dP | ϵ
Q -> aQ | ϵ
First(A) = {x}
First(Q) = {a, ϵ}
First(P) = {d, ϵ}
First(B) = {b}
First(S) = {x,a,d}
First(AB) = {x}
First(PQx) = {a,d,x}
First(AB)∩First(PQx) != ∅
故该文法不是LL(1)文法
```

(2)

```
S -> TP
T -> +PT | ϵ
P -> (S) | a

First(P) = {(,a}
First(T) = {+, ϵ}
First(S) = {+,(,a)
First(+PT) = {+}
First((S)) = {(}
First(+PT)∩First(ϵ) = ∅
First((S))∩First(a) = ∅

Follow(S) = {#,)}
Follow(P) = {+,#,),(,a}
Follow(T) = {(,a}
First(+PT)∩Follow(T) = ∅
综上所述，该文法是LL(1)文法
```

4. 验证如下文法是LL(1)文法。若是，则请构造递归下降分析程序

(1)

```
E -> Aa | Bb
A -> cA | eB
B -> bd

First(B) = {b}
First(A) = {c,e}
First(E) = {c,e,b}
First(Aa) = {c,e}
First(Bb) = {b}
First(cA) = {c}
First(eB) = {e}
First(Aa)∩First(Bb) = ∅
First(cA)∩First(eB) = ∅

Follow(E) = {#}
Follow(A) = {a}
Follow(B) = {b,a}

综上所述，该文法是LL(1)文法
```

该文法的递归下降分析程序：

```
void ParseE()
{
    switch(lookahead)
    {
        case c,e:
            ParseA();
            MatchToken(a);
            break;
        case b:
            ParseB();
            MatchToken(b);
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}

void ParseA()
{
    switch(lookahead)
    {
        case c:
            MatchToken(c);
```

```
            ParseA();
            break;
        case e:
            MatchToken(e);
            ParseB();
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}

void ParseB()
{
    switch(lookahead)
    {
        case b:
            MatchToken(b);
            MatchToken(d);
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}
```

(2)

```
    S -> AB
    A -> aA | ϵ
    B -> bB | ϵ

    First(A) = {a,ϵ}
    First(B) = {b,ϵ}
    First(S) = {a,b,ϵ}
    First(aA)∩First(ϵ) = ∅
    First(bB)∩First(ϵ) = ∅

    Follow(S) = {#}
    Follow(A) = {b,#}
    Follow(B) = {#}
    First(aA)∩Follow(A) = ∅
    First(bB)∩Follow(B) = ∅
    综上所述，该文法是LL(1)文法
```

该文法的递归下降分析程序：

```
void ParseS()
{
    switch(lookahead)
    {
        case a,b:
            ParseA();
            ParseB();
            break;
        case #:
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}

void ParseA()
{
    switch(lookahead)
    {
        case a:
            MatchToken(a);
            ParseA();
            break;
        case b,#:
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}

void ParseB()
{
    switch(lookahead)
    {
        case b:
            MatchToken(b);
            ParseB();
            break;
        case #:
            break;
        default:
            printf("syntax error \n");
            exit(0);
    }
}
```

5.验证如下文法是LL(1)文法。若是，则请构造文法的LL(1)分析表

(1)

```
D -> T L
T -> int | real
L -> id R
R -> , id R | ϵ

First(R) = {,, ϵ}
First(L) = {id}
First(T) = {int,real}
First(D) = {int,real}
First(int)∩First(real) = ∅
First(, id R)∩First(ϵ) = ∅

Follow(D) = {#}
Follow(T) = {id}
Follow(L) = {#}
Follow(R) = {#}
First(, id R)∩Follow(R) = ∅

综上所述，该文法是LL(1)文法
```

该文法的LL(1)分析表：

	int	real	id	,	#
D	D -> T L	D -> T L			
T	T -> int	T -> real			
L			L -> id R		
R				R ->,id R	R -> ε

(2)

```
E  -> T E'
E' -> + E | ϵ
T  -> F T'
T' -> T | ϵ
F  -> P F'
F' -> * F | ϵ
P  -> (E) | a | &

First(P) = {(,a,&}
First(F') = {*,ϵ}
First(F) = {(,a,&}
First(E') = {+,ϵ}
First(T) = {(,a,&}
First(T') = {(,a,&,ϵ}
First(E) = {(,a,&}

First(+E)∩First(ϵ) = ∅
First(T)∩First(ϵ) = ∅
First(*F)∩First(ϵ) = ∅
First((E))∩First(a) = ∅
First((E))∩First(&) = ∅
First(a)∩First(&) = ∅

Follow(E) = {#,)}
Follow(E') = {#,)}
Follow(T) = {+，#,)}
Follow(T') = {+，#,)}
Follow(F) = {(,a,&,+，#,)}
Follow(F') = {(,a,&,+，#,)}
Follow(P) = {*,(,a,&,+，#,)}

First(+ E)∩Follow(E') = ∅
First(T)∩Follow(T') = ∅
First(* F)∩Follow(F') = ∅

综上所述，该文法是LL(1)文法
```

该文法的LL(1)分析表：

	*	(	a	&	+	)	#
E		E -> T E'	E -> T E'	E -> T E'			
E'					E' -> + E	E' -> ε	E' -> ε
T		T -> F T'	T -> F T'	T -> F T'			
T'		T' -> T	T' -> T	T' -> T	T' -> ε	T' -> ε	T' -> ε
F		F -> P F'	F -> P F'	F -> P F'			
F'	F' -> * F	F' -> ε	F' -> ε	F' -> ε	F' -> ε	F' -> ε	F' -> ε
P		P -> (E)	P -> a	P -> &			

6.消除如下文法的左递归

(1)

```
A -> Ba | Aa | c
B -> Bb | Ab | d
```

排列方法：A,B

对A： 消除A规则的直接左递归:

```
A -> Aa | Ba | c
改写为
A -> BaA' | cA'
A' -> aA' | ϵ
```

对B： 消除左递归:

```
B -> Ab
使用A -> BaA' | cA'替换
得到
B -> BaA'b | cA'b | Bb | d

消除直接左递归
B -> BaA'b | Bb | cA'b | d
改写为
B -> cA'bB' | dB'
B' -> aA'bB' | bB' | ϵ
```

综上所述，消除左递归后文法为

```
A -> BaA' | cA'
A' -> aA' | ϵ
B -> cA'bB' | dB'
B' -> aA'bB' | bB' | ϵ
```

(2)

```
A -> Bx | Cz | w
B -> Ab | Bc
C -> Ax | By | Cp
```

排列方法：A,B,C

对B： 消除左递归:

```
由A -> Bx | Cz | w
得到B -> Bxb | Czb | wb | Bc
即B -> Bxb | Bc | Czb | wb
消除直接左递归
B -> CzbB' | wbB'
B' -> xbB' | cB' | ϵ
```

对C： 消除左递归:

```
由A -> Bx | Cz | w
得到C -> Bxx | Czx | wx | By | Cp
即C -> Bxx | By | Czx | wx | Cp
由B -> CzbB' | wbB'
得到C -> CzbB'xx | wbB'xx | CzbB'y | wbB'y | Czx | wx | Cp
即C -> CzbB'xx | CzbB'y | Czx | Cp | wbB'xx | wbB'y | wx
消除直接左递归
C -> wbB'xxC' | wbB'yC' | wxC'
C' -> zbB'xxC' | zbB'yC' | zxC' | pC' | ϵ
```

综上所述，消除左递归后文法为

```
A -> Bx | Cz | w
B -> CzbB' | wbB'
B' -> xbB' | cB' | ϵ
C -> wbB'xxC' | wbB'yC' | wxC'
C' -> zbB'xxC' | zbB'yC' | zxC' | pC' | ϵ
```