程序左递归消除

1. program -> declaration-list

2. declaration-list -> declaration-list declaration | declaration

3. declaration -> var-declaration | fun-declaration

4. var-declaration -> type-specifier ID; | type-specifier ID[NUM];

5. type-specifier -> int | void

6. fun-declaration -> type-specifier ID(params) | compound-stmt

7. params -> param-list | void

8. param-list -> param-list, param | param

9. param -> type-specifier ID | type-specifier ID[ ]

10. compound-stmt -> { local-declarations statement-list }

11. local-declarations -> local-declarations var-declaration | empty

12. statement-list -> statement-list statement | empty

13. statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt

14. expression-stmt -> expression ; | ;

15. selection-stmt -> if(expression) statement | if(expression) statement elsestatement

16. iterstions-stmt -> while(expression) statement

17. reutrn-stmt -> return ; |reutrn expression ;

18. expression -> var=expresion | simple-expression

19. var -> ID | ID[expression]

20. simple-expression -> additive-expression relop addtive-expression | additive-expression

21. relop -> <= | < | > | >= | == | !=

22. addtive-expression -> additive-expression addop term | term

23. addop -> + | -

24. term -> term mulop factor | factor

25. mulop -> \* | /

26. factor -> (expression) | var | call | NUM

27. call -> ID(args)

28. args -> arg-list | empty

29. arg-list -> arg-list, expression | expression

消除后

1. program -> declaration-list
2. 直接左递归

declaration-list -> declaration declaration-list-end

declaration-list-end -> declaration declaration-list-end |

1. declaration -> var-declaration | fun-declaration
2. 左公因子提取

var-declaration -> type-specifier ID var-declaration-end

var-declaration-end -> ; | [NUM];

1. type-specifier -> int | void
2. fun-declaration -> type-specifier ID(params) | compound-stmt
3. params -> param-list | void
4. 直接左递归

param-list -> param param-list-end

param-list-end -> , param param-list-end | empty

1. 左公因子提取

param -> type-specifier ID param-end

param-end -> [] | empty

1. compound-stmt -> { local-declarations statement-list }
2. 直接左递归

local-declarations -> local-declarations-end

local-declarations-end -> var-declaration local-declarations-end | empty

1. 直接左递归

statement-list -> statement statement-list-end

statement-list-end -> statement statement-list-end | empty

1. statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt
2. expression-stmt -> expression ; | ;
3. 左公因子提取

selection-stmt -> if(expression) statement selection-stmt-end

selection-stmt-end -> else statement | empty

1. iterstions-stmt -> while(expression) statement
2. 左公因子提取

reutrn-stmt -> return reutrn-stmt-end

reutrn-stmt-end -> ; | expression ;

1. expression -> var=expresion | simple-expression
2. 左公因子提取

var -> ID var-end

var-end -> [expression] | empty

1. 左公因子提取

simple-expression -> additive-expression simple-expression-end

simple-expression-end -> relop addtive-expression | empty

1. 左公因子提取

relop -> < relop-end | > relop-end | == | !=

relop-end -> = | empty

1. 直接左递归

addtive-expression -> term addtive-expression-end

addtive-expression-end -> addop term addtive-expression-end | empty

1. addop -> + | -
2. 直接左递归

term -> factor term-end

term-end -> mulop factor term-end | empty

1. mulop -> \* | /
2. factor -> (expression) | var | call | NUM
3. call -> ID(args)
4. args -> arg-list | empty
5. 直接左递归

arg-list -> expression arg-list-end

arg-list-end -> , expression arg-list-end | empty

扩充BNF：

1. program -> declaration-list

2. declaration-list -> declaration {declaration}

3. declaration -> type-specifier ID (; | [NUM]; | (params) ) | compound-stmt

4. var-declaration -> type-specifier ID (; | [NUM];)

5. type-specifier -> int | void

6. fun-declaration -> type-specifier ID(params) | compound-stmt

7. params -> param{,param} | void

8. param-list -> param{,param}

9. param -> type-specifier ID (empty | [ ] )

10. compound-stmt -> { local-declarations statement-list }

11. local-declarations -> var-declaration{var-declaration} | empty

12. statement-list -> statement { statement }| empty

13. statement -> expression-stmt | compound-stmt | selection-stmt | iteration-stmt | return-stmt

14. expression-stmt -> expression ; | ;

15. selection-stmt -> if(expression) statement [elsestatement]

16. iterstions-stmt -> while(expression) statement

17. reutrn-stmt -> return (;|expression ;)

18. expression -> var = expresion | simple-expression

19. var -> ID ( empty | [expression] )

20. simple-expression -> additive-expression [relop addtive-expression ]

21. relop -> <= | < | > | >= | == | !=

22. addtive-expression -> term {addop term}

23. addop -> + | -

24. term -> factor{ mulop factor }

25. mulop -> \* | /

26. factor -> (expression) | var | call | NUM

27. call -> ID(args)

28. args -> arg-list | empty

29. arg-list ->expression{, expression }