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 | e

12. statement\_list 🡪 statement\_list statement | e

13. statement 🡪 assignment\_stmt | call\_stmt | compound \_stmt | selection\_stmt

| iteration \_stmt | return \_stmt | input \_stmt | output \_stmt

14. assignment\_stmt 🡪 var = expression ;

15. call\_stmt 🡪 call ;

16. selection \_stmt 🡪 **if** ( expression ) statement

| **if** ( expression ) statement **else**  statement

17. iteration \_stmt 🡪 **while** ( expression ) statement

18. return \_stmt 🡪 **return ;** | **return** expression ;

19. input \_stmt 🡪 **input** var ;

20. output \_stmt 🡪 **output** expression ;

21. var 🡪 **ID** | **ID** **[** arithmetic\_expression  **]**

22. expression 🡪 arithmetic\_expression relop arithmetic\_expression

| arithmetic\_expression

23. relop 🡪 <= | < | > | >= | == | !=

24. arithmetic\_expression 🡪 arithmetic\_expression addop term | term

25. addop 🡪 + | -

26. term 🡪 term mulop factor | factor

27. mulop 🡪 \* | /

28. factor 🡪 ( arithmetic\_expression ) | var | call | **NUM**

29. call 🡪 **ID** ( args )

30. args 🡪 args\_list | **e**

31. args\_list 🡪 args\_list , arithmetic\_expression | arithmetic\_expression

**1- Errores**

1. program 🡪 declaration\_list void ID(void) | void ID(void)

2. declaration\_list 🡪 declaration\_list declaration | declaration

3. declaration 🡪 var\_declaration | fun\_declaration

**2- Errores**

4. var\_declaration 🡪 **int ID ;** | **int ID [ NUM ] ;**

5. type\_specifier 🡪 **int** | **void**

**3- Errores**

6. fun\_declaration 🡪 type\_specifier **ID** ( params ) compound\_stmt

7. params 🡪 param\_list | **void**

8. param\_list 🡪 param\_list , param | param

9. param 🡪 **int ID** | **int ID [ ]**

**7 recursividades izquierdas**

**5 recursivas**

**2-Producciones sin errores**

1. program 🡪 declaration\_list **void ID(void)** compound\_stmt | **void ID(void)** compound\_stmt

2. declaration\_list 🡪 declaration\_list declaration | declaration

3. declaration 🡪 var\_declaration | fun\_declaration

4. var\_declaration 🡪 **int ID ;** | **int 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 🡪 **int ID** | **int ID [ ]**

10. compound\_stmt 🡪 { local\_declarations statement\_list }

11. local\_declarations 🡪 local\_declarations var\_declaration | e

12. statement\_list 🡪 statement\_list statement | e

13. statement 🡪 assignment\_stmt | call\_stmt | compound \_stmt | selection\_stmt

| iteration \_stmt | return \_stmt | input \_stmt | output \_stmt

14. assignment\_stmt 🡪 var = expression ;

15. call\_stmt 🡪 call ;

16. selection \_stmt 🡪 **if** ( expression ) statement | **if** ( expression ) statement **else**  statement

17. iteration \_stmt 🡪 **while** ( expression ) statement

18. return \_stmt 🡪 **return ;** | **return** expression ;

19. input \_stmt 🡪 **input** var ;

20. output \_stmt 🡪 **output** expression ;

21. var 🡪 **ID** | **ID** **[** arithmetic\_expression  **]**

22. expression 🡪 arithmetic\_expression relop arithmetic\_expression | arithmetic\_expression

23. relop 🡪 <= | < | > | >= | == | !=

24. arithmetic\_expression 🡪 arithmetic\_expression addop term | term

25. addop 🡪 + | -

26. term 🡪 term mulop factor | factor

27. mulop 🡪 \* | /

28. factor 🡪 ( arithmetic\_expression ) | var | call | **NUM**

29. call 🡪 **ID** ( args )

30. args 🡪 args\_list | **e**

31. args\_list 🡪 args\_list , arithmetic\_expression | arithmetic\_expression

**2- Recursividad Izquierda**

1. program 🡪 declaration\_list **void ID(void)** compound\_stmt | **void ID(void)** compound\_stmt

2. declaration\_list 🡪 declaration declaration\_list ‘

3. declaration\_list ‘ 🡪 declaration declaration\_list ‘ | **ε**

4. declaration 🡪 var\_declaration | fun\_declaration

5. var\_declaration 🡪 **int ID ;** | **int ID [ NUM ] ;**

6. type\_specifier 🡪 **int** | **void**

7. fun\_declaration 🡪 type\_specifier **ID** ( params ) compound\_stmt

8. params 🡪 param\_list | **void**

9. param\_list 🡪 param param\_list ‘

10. param\_list ‘ 🡪 , param param\_list ‘ | **ε**

11. param 🡪 **int ID** | **int ID [ ]**

12. compound\_stmt 🡪 { local\_declarations statement\_list }

13. local\_declarations 🡪 local\_declarations ‘

14. local\_declarations ‘ 🡪 var\_declaration local\_declarations ‘| **ε**

15. statement\_list 🡪 statement\_list ‘

16. statement\_list ‘ 🡪 statement statement\_list ‘ | **ε**

17. statement 🡪 assignment\_stmt | call\_stmt | compound \_stmt | selection\_stmt

| iteration \_stmt | return \_stmt | input \_stmt | output \_stmt

18. assignment\_stmt 🡪 var = expression ;

19. call\_stmt 🡪 call ;

20. selection \_stmt 🡪 **if** ( expression ) statement | **if** ( expression ) statement **else**  statement

21. iteration \_stmt 🡪 **while** ( expression ) statement

22. return \_stmt 🡪 **return ;** | **return** expression ;

23. input \_stmt 🡪 **input** var ;

24. output \_stmt 🡪 **output** expression ;

25. var 🡪 **ID** | **ID** **[** arithmetic\_expression  **]**

26. expression 🡪 arithmetic\_expression relop arithmetic\_expression | arithmetic\_expression

27. relop 🡪 <= | < | > | >= | == | !=

28. arithmetic\_expression 🡪 term arithmetic\_expression ‘

29. arithmetic\_expression ‘ 🡪 addop term arithmetic\_expression ‘ | **ε**

30. addop 🡪 + | -

31. term 🡪 factor term ‘

32. term ‘ 🡪 mulop factor term ‘ | **ε**

33. mulop 🡪 \* | /

34. factor 🡪 ( arithmetic\_expression ) | var | call | **NUM**

35. call 🡪 **ID** ( args )

36. args 🡪 args\_list | **ε**

37. args\_list 🡪 arithmetic\_expression args\_list ‘

38. args\_list ‘ 🡪 , arithmetic\_expression args\_list ‘ | **ε**

**3- Factorización Izquierda**

1. program 🡪 declaration\_list **void ID(void)** compound\_stmt | **void ID(void)** compound\_stmt

2. declaration\_list 🡪 declaration declaration\_list ‘

3. declaration\_list ‘ 🡪 declaration declaration\_list ‘ | **ε**

4. declaration 🡪 var\_declaration | fun\_declaration

5. var\_declaration 🡪 **int ID** var\_declaration ‘

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

7. type\_specifier 🡪 **int** | **void**

8. fun\_declaration 🡪 type\_specifier **ID** ( params ) compound\_stmt

9. params 🡪 param\_list | **void**

10. param\_list 🡪 param param\_list ‘

11. param\_list ‘ 🡪 , param param\_list ‘ | **ε**

12. param 🡪 **int ID** param ‘

13. param ‘ 🡪 **ε | [ ]**

14. compound\_stmt 🡪 { local\_declarations statement\_list }

15. local\_declarations 🡪 local\_declarations ‘

16. local\_declarations ‘ 🡪 var\_declaration local\_declarations ‘| **ε**

17. statement\_list 🡪 statement\_list ‘

18. statement\_list ‘ 🡪 statement statement\_list ‘ | **ε**

19. statement 🡪 assignment\_stmt | call\_stmt | compound \_stmt | selection\_stmt

| iteration \_stmt | return \_stmt | input \_stmt | output \_stmt

20. assignment\_stmt 🡪 var = expression ;

21. call\_stmt 🡪 call ;

22. selection \_stmt 🡪 **if** ( expression ) statement selection\_stmt ‘

23. selection\_stmt ‘ 🡪 **else**  statement | **ε**

24. iteration \_stmt 🡪 **while** ( expression ) statement

25. return \_stmt 🡪 **return** return\_stmt ‘

26. return\_stmt ‘ 🡪 **;** | expression ;

27. input \_stmt 🡪 **input** var ;

28. output \_stmt 🡪 **output** expression ;

29. var 🡪 **ID** var ‘

30. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

31. expression 🡪 arithmetic\_expression expression ‘

32. expression ‘ 🡪 relop arithmetic\_expression | **ε**

33. relop 🡪 <= | < | > | >= | == | !=

34. arithmetic\_expression 🡪 term arithmetic\_expression ‘

35. arithmetic\_expression ‘ 🡪 addop term arithmetic\_expression ‘ | **ε**

36. addop 🡪 + | -

37. term 🡪 factor term ‘

38. term ‘ 🡪 mulop factor term ‘ | **ε**

39. mulop 🡪 \* | /

40. factor 🡪 ( arithmetic\_expression ) | var | call | **NUM**

41. call 🡪 **ID** ( args )

42. args 🡪 args\_list | **ε**

43. args\_list 🡪 arithmetic\_expression args\_list ‘

44. args\_list ‘ 🡪 , arithmetic\_expression args\_list ‘ | **ε**

**4- Eliminacion de producciones-ε**

**Estados que derivan a épsilon {** declaration\_list ‘ , param\_list ‘ , local\_declarations ‘ , statement\_list ‘  **,** selection\_stmt ‘ , var ‘ , expression ‘ , arithmetic\_expression ‘ , term ‘ , args , args\_list ‘ **}**

1. program 🡪 declaration\_list **void ID(void)** compound\_stmt | **void ID(void)** compound\_stmt

2. declaration\_list 🡪 declaration declaration\_list ‘

3. declaration\_list ‘ 🡪 declaration declaration\_list ‘ | **ε**

4. declaration 🡪 var\_declaration | fun\_declaration

5. var\_declaration 🡪 **int ID** var\_declaration ‘

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

7. type\_specifier 🡪 **int** | **void**

8. fun\_declaration 🡪 type\_specifier **ID** ( params ) compound\_stmt

9. params 🡪 param\_list | **void**

10. param\_list 🡪 param param\_list ‘

11. param\_list ‘ 🡪 , param param\_list ‘ | **ε**

12. param 🡪 **int ID** param ‘

13. param ‘ 🡪 **ε | [ ]**

14. compound\_stmt 🡪 { local\_declarations statement\_list }

15. local\_declarations 🡪 local\_declarations ‘

16. local\_declarations ‘ 🡪 var\_declaration local\_declarations ‘ | **ε**

17. statement\_list 🡪 statement\_list ‘

18. statement\_list ‘ 🡪 statement statement\_list ‘ | **ε**

19. statement 🡪 assignment\_stmt | call\_stmt | compound \_stmt | selection\_stmt

| iteration \_stmt | return \_stmt | input \_stmt | output \_stmt

20. assignment\_stmt 🡪 var = expression ;

21. call\_stmt 🡪 call ;

22. selection \_stmt 🡪 **if** ( expression ) statement selection\_stmt ‘

23. selection\_stmt ‘ 🡪 **else**  statement | **ε**

24. iteration \_stmt 🡪 **while** ( expression ) statement

25. return \_stmt 🡪 **return** return\_stmt ‘

26. return\_stmt ‘ 🡪 **;** | expression ;

27. input \_stmt 🡪 **input** var ;

28. output \_stmt 🡪 **output** expression ;

29. var 🡪 **ID** var ‘

30. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

31. expression 🡪 arithmetic\_expression expression ‘

32. expression ‘ 🡪 relop arithmetic\_expression | **ε**

33. relop 🡪 <= | < | > | >= | == | !=

34. arithmetic\_expression 🡪 term arithmetic\_expression ‘

35. arithmetic\_expression ‘ 🡪 addop term arithmetic\_expression ‘ | **ε**

36. addop 🡪 + | -

37. term 🡪 factor term ‘

38. term ‘ 🡪 mulop factor term ‘ | **ε**

39. mulop 🡪 \* | /

40. factor 🡪 ( arithmetic\_expression ) | var | call | **NUM**

41. call 🡪 **ID (** call ‘

42. call ‘ 🡪 args) | )

43. args 🡪 args\_list

44. args\_list 🡪 arithmetic\_expression args\_list ‘

45. args\_list ‘ 🡪 , arithmetic\_expression args\_list ‘ | **ε**

**5- Eliminacion de producciones unitarias**

1. program 🡪 declaration\_list **void ID(void)** compound\_stmt | **void ID(void)** compound\_stmt

2. declaration\_list 🡪 declaration declaration\_list ‘

3. declaration\_list ‘ 🡪 declaration declaration\_list ‘ | **ε**

4. declaration 🡪 **int ID** var\_declaration ‘ | type\_specifier **ID** ( params ) compound\_stmt

5. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

6. type\_specifier 🡪 **int** | **void**

7. params 🡪 param param\_list ‘ | **void**

8. param\_list ‘ 🡪 **,** param param\_list ‘ | **ε**

9. param 🡪 **int ID** param ‘

10. param ‘ 🡪 **ε | [ ]**

11. compound\_stmt 🡪 **{** local\_declarations statement\_list **}**

12. local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations | **ε**

13. statement\_list 🡪 statement statement\_list | **ε**

14. statement 🡪 **ID** statement ‘| **{** local\_declarations statement\_list **}** | **if** ( expression ) statement selection\_stmt ‘ | **while** **(** expression **)** statement | **return** return\_stmt ‘ | **input**  **ID** var ‘ **;** | **output** expression **;**

Factor izq

15. statement ‘ 🡪 var ‘ = expression **;** |  **(** call ‘ **;**

16. selection\_stmt ‘ 🡪 **else**  statement | **ε**

17. return\_stmt ‘ 🡪 **;** | expression **;**

18. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

19. expression 🡪 arithmetic\_expression expression ‘

20. expression ‘ 🡪 relop arithmetic\_expression | **ε**

21. relop 🡪 **<=** | **<** | **>** | **>=** | **==** | **!=**

22. arithmetic\_expression 🡪 term arithmetic\_expression ‘

23. arithmetic\_expression ‘ 🡪 addop term arithmetic\_expression ‘ | **ε**

24. addop 🡪 **+** | **-**

25. term 🡪 factor term ‘

26. term ‘ 🡪 mulop factor term ‘ | **ε**

27. mulop 🡪 **\*** | **/**

29. factor 🡪 **(** arithmetic\_expression **)** | **ID** factor ‘| **NUM**

Factor izq

29. factor ‘ 🡪 var ‘ | **(** call ‘

30. call ‘ 🡪 args) | **)**

31. args 🡪 arithmetic\_expression args\_list ‘

32. args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ | **ε**

**6- Sustitución de producciones**

1. program 🡪 **int ID** declaration ‘declaration\_list ‘ | **void ID** **(** program ‘

2. program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **void)** **{** local\_declarations statement\_list **}** program ‘’

3. program ‘’🡪 declaration\_list ‘ | **ε**

program ‘ **🡪 int ID** param ‘ param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘  **void ID(void)** **{** local\_declarations statement\_list **}** | **void)** **{** local\_declarations statement\_list **}** program ‘’

program ‘’ 🡪 **int ID** declaration ‘declaration\_list ‘ **void ID(void)** **{** local\_declarations statement\_list **} | void ID** **(** program ‘’’

Factor izq

program ‘’’ 🡪 param param\_list ‘**)** **{** local\_declarations statement\_list **}** declaration\_list ‘ **void ID(void)** **{** local\_declarations statement\_list **}** | **void** **)** **{** local\_declarations statement\_list **}** program ‘’’’

Factor izq

program ‘’’’ 🡪 declaration\_list ‘ **void ID(void)** **{** local\_declarations statement\_list **}** | **ε**

4. declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘ **| void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **ε**

5. declaration ‘ 🡪 **;** |  **[ NUM ] ;**  | **(** params **)** **{** local\_declarations statement\_list **}**

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

7. params 🡪 param param\_list ‘ | **void**

8. param\_list ‘ 🡪 **,** param param\_list ‘ | **ε**

9. param 🡪 **int ID** param ‘

10. param ‘ 🡪 **ε | [ ]**

11. local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations | **ε**

12. statement\_list 🡪  **ID** statement ‘ statement\_list | **{** local\_declarations statement\_list **}** statement\_list | **if** ( expression ) statement selection\_stmt ‘ statement\_list | **while** **(** expression **)** statement statement\_list | **return** return\_stmt ‘ statement\_list | **input** **ID** var ‘ **;** statement\_list | **output** expression **;** statement\_list | **ε**

13. statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** | = expression **;** |  **(** call ‘ **;**

14. statement 🡪 **ID** statement ‘| **{** local\_declarations statement\_list **}** | **if** ( expression ) statement selection\_stmt ‘ | **while** **(** expression **)** statement | **return** return\_stmt ‘ | **input**  **ID** var ‘ **;**| **output** expression **;**

15. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

16. selection\_stmt ‘ 🡪 **else**  statement | **ε**

17. return\_stmt ‘ 🡪 **;** | arithmetic\_expression expression ‘ **;**

18. expression 🡪 arithmetic\_expression expression ‘

19. expression ‘ 🡪 **<=** arithmetic\_expression | **<** arithmetic\_expression | **>** arithmetic\_expression | **>=** arithmetic\_expression | **==** arithmetic\_expression | **!=** arithmetic\_expression | **ε**

20. arithmetic\_expression 🡪 **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ | **ID** factor ‘ term ‘ arithmetic\_expression ‘ | **NUM** term ‘ arithmetic\_expression ‘

21. arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ | **-** term arithmetic\_expression ‘ | **ε**

22. term 🡪 **(** arithmetic\_expression **)** term ‘ | **ID** factor ‘ term ‘ | **NUM** term ‘

23. term ‘ 🡪 mulop factor term ‘ | **ε**

24. mulop 🡪 **\*** | **/**

25. factor 🡪 **(** arithmetic\_expression **)** | **ID** factor ‘| **NUM**

26. factor ‘ 🡪 **[** arithmetic\_expression  **]** | **ε** | **(** call ‘

27. call ‘ 🡪 args**)** | **)**

28. args 🡪 arithmetic\_expression args\_list ‘

29. args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ | **ε**

**7- First**

1. program 🡪 **int ID** declaration ‘declaration\_list ‘ | **void ID** **(** program ‘

First(program) = { **int** , **void** }

2. program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **void)** **{** local\_declarations statement\_list **}** program ‘’

First(program ‘) = {First(param)-{ **ε** }} U { **void** } = { **int** , **void** }

3. program ‘’🡪 declaration\_list ‘ | **ε**

First(program ‘’) = {First(declaration\_list ‘)-{ **ε** }} U { **ε** } = { **int** , **void** , **ε**}

4. declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘ **| void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **ε**

First(declaration\_list ‘) = { **int** , **void** , **ε**}

5. declaration ‘ 🡪 **;** |  **[ NUM ] ;**  | **(** params **)** **{** local\_declarations statement\_list **}**

First(declaration ‘) = { **;** , **[** , **(** }

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

First(var\_declaration ‘) = { **;** , **[** }

7. params 🡪 param param\_list ‘ | **void**

First(params) = {First(param) – { **ε** }} U { **void** } = { **int** , **void** }

8. param\_list ‘ 🡪 **,** param param\_list ‘ | **ε**

First(param\_list ‘) = { **,** , **ε** }

9. param 🡪 **int ID** param ‘

First(param) = { **int** }

10. param ‘ 🡪 **ε | [ ]**

First(param ‘) = { **[** , **ε** }

11. local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations | **ε**

First(local\_declarations) = { **int** , **ε** }

12. statement\_list 🡪  **ID** statement ‘ statement\_list | **{** local\_declarations statement\_list **}** statement\_list | **if** ( expression ) statement selection\_stmt ‘ statement\_list | **while** **(** expression **)** statement statement\_list | **return** return\_stmt ‘ statement\_list | **input** **ID** var ‘ **;** statement\_list | **output** expression **;** statement\_list | **ε**

First(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **ε** }

13. statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** | **=** expression **;** |  **(** call ‘ **;**

First(statement ‘) = { **[** , **=** , **(** }

14. statement 🡪 **ID** statement ‘| **{** local\_declarations statement\_list **}** | **if** ( expression ) statement selection\_stmt ‘ | **while** **(** expression **)** statement | **return** return\_stmt ‘ | **input**  **ID** var ‘ **;**| **output** expression **;**

First(statement ) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** }

15. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

First(var ‘) = { **[** , **ε** }

16. selection\_stmt ‘ 🡪 **else**  statement | **ε**

First(selection\_stmt ‘) = { **else** , **ε** }

17. return\_stmt ‘ 🡪 **;** | arithmetic\_expression expression ‘ **;**

First(return\_stmt ‘) = { **;** } U {First(arithmetic\_expression)-{ **ε** }} U {First(expression ‘)-{ **ε** }} = { **(** , **ID** , **NUM** , **;** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

18. expression 🡪 arithmetic\_expression expression ‘

First(expression) = {First(arithmetic\_expression)- { **ε** }} U {First(expression ‘)- { **ε** }} = { **(** , **ID** , **NUM** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

19. expression ‘ 🡪 **<=** arithmetic\_expression | **<** arithmetic\_expression | **>** arithmetic\_expression | **>=** arithmetic\_expression | **==** arithmetic\_expression | **!=** arithmetic\_expression | **ε**

First(expression ‘) = { **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

20. arithmetic\_expression 🡪 **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ | **ID** factor ‘ term ‘ arithmetic\_expression ‘ | **NUM** term ‘ arithmetic\_expression ‘

First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

21. arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ | **-** term arithmetic\_expression ‘ | **ε**

First(arithmetic\_expression ‘) = { **+** , **-** , **ε** }

22. term 🡪 **(** arithmetic\_expression **)** term ‘ | **ID** factor ‘ term ‘ | **NUM** term ‘

First(term) = { **(** , **ID** , **NUM**}

23. term ‘ 🡪 mulop factor term ‘ | **ε**

First(term ‘) = {First(mulop) – { **ε** }} U {First(factor) – { **ε** }} U {First(term ‘) – { **ε** }} U { **ε** } = { **\*** , **/** , **ε** ,  **(** , **ID** , **NUM** }

24. mulop 🡪 **\*** | **/**

First(mulop) = { **\*** , **/** }

25. factor 🡪 **(** arithmetic\_expression **)** | **ID** factor ‘| **NUM**

First(factor) = { **(** , **ID** , **NUM**}

26. factor ‘ 🡪 **[** arithmetic\_expression  **]** | **ε** | **(** call ‘

First(factor ‘) = { **[** , **ε** , **(**}

27. call ‘ 🡪 args**)** | **)**

First(call ‘) = {First(args)-{ **ε** }} U { **)** } U { **)** } = { **(** , **)** , **ID** , **NUM** }

28. args 🡪 arithmetic\_expression args\_list ‘

First(args) = {First(arithmetic\_expression)-{ **ε** }} U {First(args\_list ‘)-{ **ε** }} = { **(** , **ID** , **NUM** , **,** , **ε** }

29. args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ | **ε**

First(args\_list ‘) = { **,** , **ε** }

**8 - Follow**

1. program 🡪 **int ID** declaration ‘declaration\_list ‘ | **void ID** **(** program ‘

First(program) = { **int** , **void** }

Follow(program) = { **$** }

2. program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **void)** **{** local\_declarations statement\_list **}** program ‘’

First(program ‘) = {First(param)-{ **ε** }} U { **void** } = { **int** , **void** }

Follow(program ‘) = Follow(program ‘) U Follow(program) = { **$** }

3. program ‘’🡪 declaration\_list ‘ | **ε**

First(program ‘’) = {First(declaration\_list ‘)-{ **ε** }} U { **ε** } = { **int** , **void** , **ε**}

Follow(program ‘’) = Follow(program ‘’) U Follow(program ‘) = { **$** }

4. declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘ **| void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **ε**

First(declaration\_list ‘) = { **int** , **void** , **ε**}

Follow(declaration\_list ‘) = Follow(program) U Follow(program ‘) U Follow(program ‘’) U Follow(declaration\_list ‘) U Follow(declaration\_list ‘) = { **$** }

5. declaration ‘ 🡪 **;** |  **[ NUM ] ;**  | **(** params **)** **{** local\_declarations statement\_list **}**

First(declaration ‘) = { **;** , **[** , **(** }

Follow(declaration ‘) = First(declaration\_list ‘) – { **ε** } = { **int** , **void** } U Follow(program) U First(declaration\_list ‘) – { **ε** } = { **int** , **void** } U Follow(declaration\_list ‘) = { **int** , **void** , **$** }

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

First(var\_declaration ‘) = { **;** , **[** }

Follow(var\_declaration ‘) = First(local\_declarations) – { **ε** } = { **int** } U Follow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **int** }

7. params 🡪 param param\_list ‘ | **void**

First(params) = {First(param) – { **ε** }} U { **void** } = { **int** , **void** }

Follow(params) = First( **)** ) = { **)** }

8. param\_list ‘ 🡪 **,** param param\_list ‘ | **ε**

First(param\_list ‘) = { **,** , **ε** }

Follow(param\_list ‘) = First( **)** ) = { **)** } U Follow(params) U Follow(param\_list ‘) = { **)** }

9. param 🡪 **int ID** param ‘

First(param) = { **int** }

Follow(param) = First(param\_list ‘) – { **ε** } = { **,** } U = { **,** } U Follow(param\_list ‘) U Follow(params) First( **)** ) = { **)** , **,** }

10. param ‘ 🡪 **ε | [ ]**

First(param ‘) = { **[** , **ε** }

Follow(param ‘) = Follow(param ‘) U Follow(param) = { **)** , **,** }

11. local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations | **ε**

First(local\_declarations) = { **int** , **ε** }

Follow(local\_declarations) = First(statement\_list) – { **ε** } = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U First( **}** ) U Follow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** }

12. statement\_list 🡪  **ID** statement ‘ statement\_list | **{** local\_declarations statement\_list **}** statement\_list | **if** ( expression ) statement selection\_stmt ‘ statement\_list | **while** **(** expression **)** statement statement\_list | **return** return\_stmt ‘ statement\_list | **input** **ID** var ‘ **;** statement\_list | **output** expression **;** statement\_list | **ε**

First(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **ε** }

Follow(statement\_list) = Follow(statement\_list) U First( **}** ) = { **}** }

13. statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** | **=** expression **;** |  **(** call ‘ **;**

First(statement ‘) = { **[** , **=** , **(** }

Follow(statement ‘) = Follow(statement) U Follow(statement ‘) U First(statement\_list)-{ **ε** } ={ **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

14. statement 🡪 **ID** statement ‘| **{** local\_declarations statement\_list **}** | **if** ( expression ) statement selection\_stmt ‘ | **while** **(** expression **)** statement | **return** return\_stmt ‘ | **input**  **ID** var ‘ **;**| **output** expression **;**

First(statement ) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** }

Follow(statement) = Follow(selection\_stmt ‘) U Follow(statement) U First(selection\_stmt ‘) – { **ε** } = { **else** } U First(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

15. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

First(var ‘) = { **[** , **ε** }

Follow(var ‘) = { **;** }

16. selection\_stmt ‘ 🡪 **else**  statement | **ε**

First(selection\_stmt ‘) = { **else** , **ε** }

Follow(selection\_stmt ‘) = Follow(statement) U Follow(selection\_stmt ‘) U First(statement\_list) – { **ε** } = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

17. return\_stmt ‘ 🡪 **;** | arithmetic\_expression expression ‘ **;**

First(return\_stmt ‘) = { **;** } U {First(arithmetic\_expression)-{ **ε** }} = { **(** , **ID** , **NUM** , **;** }

Follow(return\_stmt ‘) = Follow(statement) U Follow(return\_stmt ‘) U First(statement\_list-{ **ε** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

18. expression 🡪 arithmetic\_expression expression ‘

First(expression) = {First(arithmetic\_expression)- { **ε** }} = { **(** , **ID** , **NUM** }

Follow(expression) = { **;** , **)** }

19. expression ‘ 🡪 **<=** arithmetic\_expression | **<** arithmetic\_expression | **>** arithmetic\_expression | **>=** arithmetic\_expression | **==** arithmetic\_expression | **!=** arithmetic\_expression | **ε**

First(expression ‘) = { **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

Follow(expression ‘) = Follow(expression) U Follow(expression ‘) = { **;** , **)** }

20. arithmetic\_expression 🡪 **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ | **ID** factor ‘ term ‘ arithmetic\_expression ‘ | **NUM** term ‘ arithmetic\_expression ‘

First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

Follow(arithmetic\_expression) U Follow(expression ‘) U First(expression ‘) – { **ε** } = { **<=** , **<** , **>** , **>=** , **==** , **!=** } U Follow(expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** }

21. arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ | **-** term arithmetic\_expression ‘ | **ε**

First(arithmetic\_expression ‘) = { **+** , **-** , **ε** }

Follow(arithmetic\_expression ‘) = Follow(arithmetic\_expression ‘) U Follow(arithmetic\_expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** }

22. term 🡪 **(** arithmetic\_expression **)** term ‘ | **ID** factor ‘ term ‘ | **NUM** term ‘

First(term) = { **(** , **ID** , **NUM**}

Follow(term) = First(arithmetic\_expression ‘)-{ **ε** } ={ **+** , **-** } U Follow(arithmetic\_expression ‘) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** }

23. term ‘ 🡪 mulop factor term ‘ | **ε**

First(term ‘) = {First(mulop) – { **ε** }} U { **ε** } = { **\*** , **/** , **ε** }

Follow(term ‘) = Follow(term) U First(arithmetic\_expression ‘) – { **ε** } = { **+** , **-** } U Follow(term ‘) U Follow(arithmetic\_expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** }

24. mulop 🡪 **\*** | **/**

First(mulop) = { **\*** , **/** }

Follow(mulop) = First(factor)-{ **ε** }={ **(** , **ID** , **NUM**} = {  **(** , **ID** , **NUM** }

25. factor 🡪 **(** arithmetic\_expression **)** | **ID** factor ‘| **NUM**

First(factor) = { **(** , **ID** , **NUM**}

Follow(factor) = First(term ‘)-{ **ε** }={ **\*** , **/** } U Follow(term ‘) = {**;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** }

26. factor ‘ 🡪 **[** arithmetic\_expression  **]** | **ε** | **(** call ‘

First(factor ‘) = { **[** , **ε** , **(** }

Follow(factor ‘) = Follow(factor) U Follow(factor ‘) U First(term ‘) = { **\*** , **/** } U Follow(term) = {**;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** } U First(arithmetic\_expression ‘) = { **+** , **-** } = {**;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** }

27. call ‘ 🡪 args **)** | **)**

First(call ‘) = {First(args)-{ **ε** }} U { **)** } U { **)** } = { **(** , **)** , **ID** , **NUM** }

Follow(call ‘) = Follow(factor ‘) U Follow(call ‘) U First( **;** ) = {**;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** }

28. args 🡪 arithmetic\_expression args\_list ‘

First(args) = {First(arithmetic\_expression)-{ **ε** }} = { **(** , **ID** , **NUM** }

Follow(args) = { **)** }

29. args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ | **ε**

First(args\_list ‘) = { **,** , **ε** }

Follow(args\_list ‘) = Follow(args\_list ‘) U Follow(args) = { **)** }

**8 - First +**

1. program 🡪 **int ID** declaration ‘declaration\_list ‘ | **void ID** **(** program ‘

First(program) = { **int** , **void** }

Follow(program) = { **$** }

First+(program 🡪 **int ID** declaration ‘declaration\_list ‘) = { **int** }

First+(program 🡪 **void ID** **(** program ‘) = { **void** }

2. program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **void)** **{** local\_declarations statement\_list **}** declaration\_list ‘

First(program ‘) = {First(param)-{ **ε** }} U { **void** } = { **int** , **void** }

Follow(program ‘) = Follow(program ‘) U Follow(program) = { **$** }

First+(program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ ) =

First(param) = { **int** }

First+(program ‘ 🡪 **void)** **{** local\_declarations statement\_list **}** program ‘’ ) =

{ **void** }

3. program ‘’🡪 declaration\_list ‘ | **ε**

program ‘’🡪 **int ID** declaration ‘declaration\_list ‘ **| void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **ε**

**Produccion inutil – Tiene lo mismo que declaration\_list ‘**

4. declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘ **| void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ | **ε**

First(declaration\_list ‘) = { **int** , **void** , **ε**}

Follow(declaration\_list ‘) = Follow(program) U Follow(program ‘) U Follow(program ‘’) U Follow(declaration\_list ‘) U Follow(declaration\_list ‘) = { **$** }

First+(declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘) = { **int** }

First+(declaration\_list ‘ 🡪 **void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘) = { **void** }

First+(declaration\_list ‘ 🡪 **ε**) = First(**ε**) U Follow(declaration\_list ‘) = { **ε** , **$** }

5. declaration ‘ 🡪 **;** |  **[ NUM ] ;**  | **(** params **)** **{** local\_declarations statement\_list **}**

First(declaration ‘) = { **;** , **[** , **(** }

Follow(declaration ‘) = First(declaration\_list ‘) – { **ε** } = { **int** , **void** } U Follow(program) U First(declaration\_list ‘) – { **ε** } = { **int** , **void** } U Follow(declaration\_list ‘) = { **int** , **void** , **$** }

First+(declaration ‘ 🡪 **;** ) = { **;** }

First+(declaration ‘ 🡪 **[ NUM ] ;**  ) = { **[** }

First+(declaration ‘ 🡪 **(** params **)** **{** local\_declarations statement\_list **}** ) = { **(** }

6. var\_declaration ‘ 🡪 **;** |  **[ NUM ] ;**

First(var\_declaration ‘) = { **;** , **[** }

Follow(var\_declaration ‘) = First(local\_declarations) – { **ε** } = { **int** } U Follow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **int** }

First+(var\_declaration ‘ 🡪 **;** ) = { **;** }

First+(var\_declaration ‘ 🡪 **[ NUM ] ;** ) = { **[** }

7. params 🡪 param param\_list ‘ | **void**

First(params) = {First(param) – { **ε** }} U { **void** } = { **int** , **void** }

Follow(params) = First( **)** ) = { **)** }

First+(params 🡪 param param\_list ‘ ) = First(param) = { **int** }

First+(params 🡪 **void** ) = { **void** }

8. param\_list ‘ 🡪 **,** param param\_list ‘ | **ε**

First(param\_list ‘) = { **,** , **ε** }

Follow(param\_list ‘) = First( **)** ) = { **)** } U Follow(params) U Follow(param\_list ‘) = { **)** }

First+(param\_list ‘ 🡪 **,** param param\_list ‘ ) = { **,** }

First+( param\_list ‘ 🡪 **ε**) = First(**ε**) U Follow(param\_list ‘) = { **)** , **ε** }

9. param 🡪 **int ID** param ‘

First(param) = { **int** }

Follow(param) = First(param\_list ‘) – { **ε** } = { **,** } U = { **,** } U Follow(param\_list ‘) U Follow(params) First( **)** ) = { **)** , **,** }

First+(param 🡪 **int ID** param ‘ ) = { **int** }

10. param ‘ 🡪 **ε | [ ]**

First(param ‘) = { **[** , **ε** }

Follow(param ‘) = Follow(param ‘) U Follow(param) = { **)** , **,** }

First+(param ‘ 🡪 **ε** ) = First(**ε**) U Follow(param ‘) = { ) , **,** , **ε** }

First+(param ‘ 🡪 **[ ]**  ) = { **[** }

11. local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations | **ε**

First(local\_declarations) = { **int** , **ε** }

Follow(local\_declarations) = First(statement\_list) – { **ε** } = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U First( **}** ) U Follow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** }

First+(local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations) = { **int** }

First+(local\_declarations 🡪 **ε**) = First(**ε**)UFollow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **ε** }

12. statement\_list 🡪  **ID** statement ‘ statement\_list | **{** local\_declarations statement\_list **}** statement\_list | **if** ( expression ) statement selection\_stmt ‘ statement\_list | **while** **(** expression **)** statement statement\_list | **return** return\_stmt ‘ statement\_list | **input** **ID** var ‘ **;** statement\_list | **output** expression **;** statement\_list | **ε**

First(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **ε** }

Follow(statement\_list) = Follow(statement\_list) U First( **}** ) = { **}** }

First+(statement\_list 🡪 **ID** statement ‘ statement\_list ) = { **ID** }

First+(statement\_list 🡪 **{** local\_declarations statement\_list **}** statement\_list ) = { **{** }

First+(statement\_list 🡪 **if** ( expression ) statement selection\_stmt ‘ statement\_list ) = { **if** }

First+(statement\_list 🡪 **while** **(** expression **)** statement statement\_list ) = { **while** }

First+(statement\_list 🡪 **return** return\_stmt ‘ statement\_list ) = { **return** }

First+(statement\_list 🡪 **input** **ID** var ‘ **;** statement\_list ) = { **input** }

First+(statement\_list 🡪 **output** expression **;** statement\_list ) = { **output** }

First+(statement\_list 🡪 **ε** ) = First(**ε**) U Follow(statement\_list) = { **}** , **ε** }

13. statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** | **=** expression **;** |  **(** call ‘ **;**

First(statement ‘) = { **[** , **=** , **(** }

Follow(statement ‘) = Follow(statement) U Follow(statement ‘) U First(statement\_list)-{ **ε** } ={ **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

First+(statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** ) = { **[** }

First+(statement ‘ 🡪 **=** expression **;** ) = { **=** }

First+(statement ‘ 🡪 **(** call ‘ **;** ) = { **(** }

14. statement 🡪 **ID** statement ‘ | **{** local\_declarations statement\_list **}** | **if** ( expression ) statement selection\_stmt ‘ | **while** **(** expression **)** statement | **return** return\_stmt ‘ | **input**  **ID** var ‘ **;**| **output** expression **;**

First(statement ) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** }

Follow(statement) = Follow(selection\_stmt ‘) U Follow(statement) U First(selection\_stmt ‘) – { **ε** } = { **else** } U First(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

First+(statement 🡪 **ID** statement ‘ ) = { **ID** }

First+(statement 🡪 **{** local\_declarations statement\_list **}** ) = { **{** }

First+(statement 🡪 **if** ( expression ) statement selection\_stmt ‘ ) = { **if** }

First+(statement 🡪 **while** **(** expression **)** statement ) = { **while** }

First+(statement 🡪 **return** return\_stmt ‘ ) = { **return** }

First+(statement 🡪 **input**  **ID** var ‘ **;** ) = { **input** }

First+(statement 🡪 **output** expression **;** ) = { **output** }

15. var ‘ 🡪 **[** arithmetic\_expression  **]** | **ε**

First(var ‘) = { **[** , **ε** }

Follow(var ‘) = { **;** }

First+(var ‘ 🡪 **[** arithmetic\_expression  **]** ) = { **[** }

First+(var ‘ 🡪 **ε** ) = First(**ε**) U Follow(var ‘) = { **;** , **ε** }

16. selection\_stmt ‘ 🡪 **else**  statement | **ε**

First(selection\_stmt ‘) = { **else** , **ε** }

Follow(selection\_stmt ‘) = Follow(statement) U Follow(selection\_stmt ‘) U First(statement\_list) – { **ε** } = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

First+(selection\_stmt ‘ 🡪 **else**  statement) = { **else** }

First+(selection\_stmt ‘ 🡪 **ε** ) = First(**ε**) U Follow(selection\_stmt ‘) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** , **ε** }

17. return\_stmt ‘ 🡪 **;** | arithmetic\_expression expression ‘ **;**

First(return\_stmt ‘) = { **;** } U {First(arithmetic\_expression)-{ **ε** }} U {First(expression ‘)-{ **ε** }} = { **(** , **ID** , **NUM** , **;** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

Follow(return\_stmt ‘) = Follow(statement) U Follow(return\_stmt ‘) U First(statement\_list-{ **ε** } U Follow(statement\_list) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** }

First+(return\_stmt ‘ 🡪 **;** ) = { **;** }

First+(return\_stmt ‘ 🡪 arithmetic\_expression expression ‘ **;** ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

18. expression 🡪 arithmetic\_expression expression ‘

First(expression) = {First(arithmetic\_expression)- { **ε** }} U {First(expression ‘)- { **ε** }} = { **(** , **ID** , **NUM** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

Follow(expression) = { **;** , **)** }

First+(expression 🡪 arithmetic\_expression expression ‘ ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

19. expression ‘ 🡪 **<=** arithmetic\_expression | **<** arithmetic\_expression | **>** arithmetic\_expression | **>=** arithmetic\_expression | **==** arithmetic\_expression | **!=** arithmetic\_expression | **ε**

First(expression ‘) = { **<=** , **<** , **>** , **>=** , **==** , **!=** , **ε** }

Follow(expression ‘) = Follow(expression) U Follow(expression ‘) = { **;** , **)** }

First+(expression ‘ 🡪 **<=** arithmetic\_expression ) = { **<=** }

First+(expression ‘ 🡪 **<** arithmetic\_expression ) = { **<** }

First+(expression ‘ 🡪 **>** arithmetic\_expression ) = { **>** }

First+(expression ‘ 🡪 **>=** arithmetic\_expression ) = { **>=** }

First+(expression ‘ 🡪 **==** arithmetic\_expression ) = { **==** }

First+(expression ‘ 🡪 **!=** arithmetic\_expression ) = { **!=** }

First+(expression ‘ 🡪 **ε** ) = First(**ε**) U Follow(expression ‘) ={ **;** , **)** , **ε** }

20. arithmetic\_expression 🡪 **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ | **ID** factor ‘ term ‘ arithmetic\_expression ‘ | **NUM** term ‘ arithmetic\_expression ‘

First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

Follow(arithmetic\_expression) = { **)** , **]** } U Follow(expression ‘) U First(expression ‘) – { **ε** } = { **<=** , **<** , **>** , **>=** , **==** , **!=** } U Follow(expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** }

First+(arithmetic\_expression 🡪  **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ ) = { **(** }

First+(arithmetic\_expression 🡪  **ID** factor ‘ term ‘ arithmetic\_expression ‘ ) = { **ID** }

First+(arithmetic\_expression 🡪  **NUM** term ‘ arithmetic\_expression ‘ ) = { **NUM** }

21. arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ | **-** term arithmetic\_expression ‘ | **ε**

First(arithmetic\_expression ‘) = { **+** , **-** , **ε** }

Follow(arithmetic\_expression ‘) = Follow(arithmetic\_expression ‘) U Follow(arithmetic\_expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** }

First+(arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ ) = { **+** }

First+(arithmetic\_expression ‘ 🡪 **-** term arithmetic\_expression ‘ ) = { **-** }

First+(arithmetic\_expression ‘ 🡪 **ε** ) = First(**ε**) **U** Follow(arithmetic\_expression ‘ ) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **ε** }

22. term 🡪 **(** arithmetic\_expression **)** term ‘ | **ID** factor ‘ term ‘ | **NUM** term ‘

First(term) = { **(** , **ID** , **NUM**}

Follow(term) = First(arithmetic\_expression ‘)-{ **ε** } ={ **+** , **-** } U Follow(arithmetic\_expression ‘) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** }

First+(term 🡪 **(** arithmetic\_expression **)** term ‘ ) = { **(** }

First+(term 🡪 **ID** factor ‘ term ‘ ) = { **ID** }

First+(term 🡪 **NUM** term ‘ ) = { **NUM** }

23. term ‘ 🡪 mulop factor term ‘ | **ε**

First(term ‘) = {First(mulop) – { **ε** }} U {First(factor) – { **ε** }} U {First(term ‘) – { **ε** }} U { **ε** } = { **\*** , **/** , **ε** ,  **(** , **ID** , **NUM** }

Follow(term ‘) = Follow(term) U First(arithmetic\_expression ‘) – { **ε** } = { **+** , **-** } U Follow(term ‘) U Follow(arithmetic\_expression) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** }

First+(term ‘ 🡪 mulop factor term ‘ ) = First(mulop) = { **\*** , **/** }

First+(term ‘ 🡪 **ε** ) = First(**ε**) U Follow(term ‘) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **ε** }

24. mulop 🡪 **\*** | **/**

First(mulop) = { **\*** , **/** }

Follow(mulop) = First(factor)-{ **ε** }={ **(** , **ID** , **NUM**} U First(term ‘)-{ **ε** }={ **\*** , **/** ,  **(** , **ID** , **NUM** }

= { **\*** , **/** ,  **(** , **ID** , **NUM** }

First+(mulop 🡪 **\*** ) = { **\*** }

First+(mulop 🡪 **/** ) = { **/** }

25. factor 🡪 **(** arithmetic\_expression **)** | **ID** factor ‘| **NUM**

First(factor) = { **(** , **ID** , **NUM**}

Follow(factor) = First(term ‘)-{ **ε** }={ **\*** , **/** ,  **(** , **ID** , **NUM** } U Follow(term ‘) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** ,  **(** , **ID** , **NUM** }

First+(factor 🡪 **(** arithmetic\_expression **)** ) = { **(** }

First+(factor 🡪 **ID** factor ‘ ) = { **ID** }

First+(factor 🡪 **NUM** ) = { **NUM** }

26. factor ‘ 🡪 **[** arithmetic\_expression  **]** | **ε** | **(** call ‘

First(factor ‘) = { **[** , **ε** , **(** }

Follow(factor ‘) = Follow(factor) U Follow(factor ‘) U First(term ‘) = { **\*** , **/** ,  **(** , **ID** , **NUM** } U Follow(term) U First(arithmetic\_expression ‘) = { **+** , **-** } = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** ,  **(** , **ID** , **NUM** }

First+(factor ‘🡪 **[** arithmetic\_expression  **]**  ) = { **[** }

First+(factor ‘🡪 **(** call ‘ ) = { **(** }

First+(factor ‘🡪 **ε** ) = First(**ε**) U Follow(factor ‘) = { **+** , **-** } = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** }

27. call ‘ 🡪 args **)** | **)**

First(call ‘) = {First(args)-{ **ε** }} U { **)** } U { **)** } = { **(** , **)** , **ID** , **NUM** }

Follow(call ‘) = Follow(factor ‘) U Follow(call ‘) U First( **;** ) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** ,  **(** , **ID** , **NUM** }

First+(call ‘ 🡪 args **)** ) = First(args) = { **(** , **ID** , **NUM** , **,** , **ε** } = { **(** , **ID** , **NUM** , **,** , **ε** }

First+(call ‘ 🡪 **)** ) = { **)** }

28. args 🡪 arithmetic\_expression args\_list ‘

First(args) = {First(arithmetic\_expression)-{ **ε** }} U {First(args\_list ‘)-{ **ε** }} = { **(** , **ID** , **NUM** , **,** , **ε** }

Follow(args) = { **)** }

First+(args 🡪 arithmetic\_expression args\_list ‘ ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

29. args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ | **ε**

First(args\_list ‘) = { **,** , **ε** }

Follow(args\_list ‘) = Follow(args\_list ‘) U Follow(args) = { **)** }

First+(args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ ) = { **,** }

First+(args\_list ‘ 🡪 **ε** ) = First(**ε**) U Follow(args\_list ‘) = { **)** , **ε** }

**8 – Producciones Finales**

1. First+(program 🡪 **int ID** declaration ‘declaration\_list ‘ ) = { **int** }

2. First+(program 🡪 **void ID** **(** program ‘ ) = { **void** }

3. First+(program ‘ 🡪 param param\_list ‘ **)** **{** local\_declarations statement\_list **}** declaration\_list ‘ ) =

First(param) = { **int** }

4. First+(program ‘ 🡪 **void)** **{** local\_declarations statement\_list **}** declaration\_list ‘ ) =

{ **void** }

5. First+(declaration\_list ‘ 🡪 **int ID** declaration ‘declaration\_list ‘) = { **int** }

6. First+(declaration\_list ‘ 🡪 **void ID** **(** params **)** **{** local\_declarations statement\_list **}** declaration\_list ‘) = { **void** }

7. First+(declaration\_list ‘ 🡪 **ε**) = First(**ε**) U Follow(declaration\_list ‘) = { **ε** , **$** }

8. First+(declaration ‘ 🡪 **;** ) = { **;** }

9. First+(declaration ‘ 🡪 **[ NUM ] ;**  ) = { **[** }

10. First+(declaration ‘ 🡪 **(** params **)** **{** local\_declarations statement\_list **}** ) = { **(** }

11. First+(var\_declaration ‘ 🡪 **;** ) = { **;** }

12. First+(var\_declaration ‘ 🡪 **[ NUM ] ;** ) = { **[** }

13. First+(params 🡪 param param\_list ‘ ) = First(param) = { **int** }

14. First+(params 🡪 **void** ) = { **void** }

15. First+(param\_list ‘ 🡪 **,** param param\_list ‘ ) = { **,** }

16. First+( param\_list ‘ 🡪 **ε**) = First(**ε**) U Follow(param\_list ‘) = { **)** , **ε** }

17. First+(param 🡪 **int ID** param ‘ ) = { **int** }

18. First+(param ‘ 🡪 **ε** ) = First(**ε**) U Follow(param ‘) = { ) , **,** , **ε** }

19. First+(param ‘ 🡪 **[ ]**  ) = { **[** }

20. First+(local\_declarations 🡪 **int ID** var\_declaration ‘ local\_declarations) = { **int** }

21. First+(local\_declarations 🡪 **ε**) = First(**ε**)UFollow(local\_declarations) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **ε** }

22. First+(statement\_list 🡪 **ID** statement ‘ statement\_list ) = { **ID** }

23. First+(statement\_list 🡪 **{** local\_declarations statement\_list **}** statement\_list ) = { **{** }

24. First+(statement\_list 🡪 **if** ( expression ) statement selection\_stmt ‘ statement\_list ) = { **if** }

25. First+(statement\_list 🡪 **while** **(** expression **)** statement statement\_list ) = { **while** }

26. First+(statement\_list 🡪 **return** return\_stmt ‘ statement\_list ) = { **return** }

27. First+(statement\_list 🡪 **input** **ID** var ‘ **;** statement\_list ) = { **input** }

28. First+(statement\_list 🡪 **output** expression **;** statement\_list ) = { **output** }

29. First+(statement\_list 🡪 **ε** ) = First(**ε**) U Follow(statement\_list) = { **}** , **ε** }

30. First+(statement ‘ 🡪 **[** arithmetic\_expression  **]**  = expression **;** ) = { **[** }

31. First+(statement ‘ 🡪 **=** expression **;** ) = { **=** }

32. First+(statement ‘ 🡪 **(** call ‘ **;** ) = { **(** }

33. First+(statement 🡪 **ID** statement ‘ ) = { **ID** }

34. First+(statement 🡪 **{** local\_declarations statement\_list **}** ) = { **{** }

35. First+(statement 🡪 **if** ( expression ) statement selection\_stmt ‘ ) = { **if** }

36. First+(statement 🡪 **while** **(** expression **)** statement ) = { **while** }

37. First+(statement 🡪 **return** return\_stmt ‘ ) = { **return** }

38. First+(statement 🡪 **input**  **ID** var ‘ **;** ) = { **input** }

39. First+(statement 🡪 **output** expression **;** ) = { **output** }

40. First+(var ‘ 🡪 **[** arithmetic\_expression  **]** ) = { **[** }

41. First+(var ‘ 🡪 **ε** ) = First(**ε**) U Follow(var ‘) = { **;** , **ε** }

42. First+(selection\_stmt ‘ 🡪 **else**  statement) = { **else** }

43. First+(selection\_stmt ‘ 🡪 **ε** ) = First(**ε**) U Follow(selection\_stmt ‘) = { **ID** , **{** , **if** , **while** , **return** , **input** , **output** , **}** , **else** , **ε** }

44. First+(return\_stmt ‘ 🡪 **;** ) = { **;** }

45. First+(return\_stmt ‘ 🡪 arithmetic\_expression expression ‘ **;** ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

46. First+(expression 🡪 arithmetic\_expression expression ‘ ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

47. First+(expression ‘ 🡪 **<=** arithmetic\_expression ) = { **<=** }

48. First+(expression ‘ 🡪 **<** arithmetic\_expression ) = { **<** }

49. First+(expression ‘ 🡪 **>** arithmetic\_expression ) = { **>** }

50. First+(expression ‘ 🡪 **>=** arithmetic\_expression ) = { **>=** }

51. First+(expression ‘ 🡪 **==** arithmetic\_expression ) = { **==** }

52. First+(expression ‘ 🡪 **!=** arithmetic\_expression ) = { **!=** }

53. First+(expression ‘ 🡪 **ε** ) = First(**ε**) U Follow(expression ‘) ={ **;** , **)** , **ε** }

54. First+(arithmetic\_expression 🡪  **(** arithmetic\_expression **)** term ‘ arithmetic\_expression ‘ ) = { **(** }

55. First+(arithmetic\_expression 🡪  **ID** factor ‘ term ‘ arithmetic\_expression ‘ ) = { **ID** }

56. First+(arithmetic\_expression 🡪  **NUM** term ‘ arithmetic\_expression ‘ ) = { **NUM** }

57. First+(arithmetic\_expression ‘ 🡪 **+** term arithmetic\_expression ‘ ) = { **+** }

58. First+(arithmetic\_expression ‘ 🡪 **-** term arithmetic\_expression ‘ ) = { **-** }

59. First+(arithmetic\_expression ‘ 🡪 **ε** ) = First(**ε**) **U** Follow(arithmetic\_expression ‘ ) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **ε** }

60. First+(term 🡪 **(** arithmetic\_expression **)** term ‘ ) = { **(** }

61. First+(term 🡪 **ID** factor ‘ term ‘ ) = { **ID** }

62. First+(term 🡪 **NUM** term ‘ ) = { **NUM** }

63. First+(term ‘ 🡪 mulop factor term ‘ ) = First(mulop) = { **\*** , **/** }

64. First+(term ‘ 🡪 **ε** ) = First(**ε**) U Follow(term ‘) = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **ε** }

65. First+(mulop 🡪 **\*** ) = { **\*** }

66. First+(mulop 🡪 **/** ) = { **/** }

67. First+(factor 🡪 **(** arithmetic\_expression **)** ) = { **(** }

68. First+(factor 🡪 **ID** factor ‘ ) = { **ID** }

69. First+(factor 🡪 **NUM** ) = { **NUM** }

70. First+(factor ‘🡪 **[** arithmetic\_expression  **]**  ) = { **[** }

71. First+(factor ‘🡪 **(** call ‘ ) = { **(** }

72. First+(factor ‘🡪 **ε** ) = First(**ε**) U Follow(factor ‘) = { **+** , **-** } = { **;**  , **)** , **<=** , **<** , **>** , **>=** , **==** , **!=** , **,** , **]** , **+** , **-** , **\*** , **/** }

73. First+(call ‘ 🡪 args **)** ) = First(args) = { **(** , **ID** , **NUM** , **,** , **ε** } = { **(** , **ID** , **NUM** , **,** , **ε** }

74. First+(call ‘ 🡪 **)** ) = { **)** }

75. First+(args 🡪 arithmetic\_expression args\_list ‘ ) = First(arithmetic\_expression) = { **(** , **ID** , **NUM** }

76. First+(args\_list ‘ 🡪 **,** arithmetic\_expression args\_list ‘ ) = { **,** }

77. First+(args\_list ‘ 🡪 **ε** ) = First(**ε**) U Follow(args\_list ‘) = { **)** , **ε** }

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Entrada |  | Var | Function | Scope |
| 28 🡪 Identificador |  |  |  |  |
| 29 🡪 Numero |  |  |  |  |
|  |  |  |  |  |
| Entrada |  | Var | Function | Scope |
| 28 🡪 Identificador |  |  |  |  |
| 29 🡪 Numero |  |  |  |  |