**Instituto Tecnológico y de Estudios Superiores de Monterrey**

**Campus Santa Fe**

Blue text on a black background

AI-generated content may be incorrect.

**Desarrollo de aplicaciones avanzadas de ciencias computacionales**

**ITC (Gpo 501)**

**Proyecto 2: Analizador Sintáctico**

Estudiante:

David Medina Domínguez - A01783155

Docentes:

Dr. Víctor Manuel de la Cueva Hernández

**Fecha de entrega**

28 de Abril del 2025

# Gramática de C- en BNF

|  |  |  |
| --- | --- | --- |
|  | Identificador (No-Terminal) | Valores  (No-Terminal y Terminales) |
| 0. | <program> | <declaration-list> |
| 1. | <declaration-list> | <declaration> <declaration-list>  | λ |
| 2. | <declaration> | <type-specifier> ID <declaration-prime> |
| 3. | <declaration-prime> | ';'  | '[' NUM ']' ';'  | '(' <params> ')' <compound-stmt>  | ‘,’ ID <declaration-prime>  | ‘=’ NUM ‘;’ |
| 4. | <type-specifier> | 'int'  | 'void' |
| 5. | <params> | 'void'  | <param-list> |
| 6. | <param-list> | <param> <param-list-prime> |
| 7. | <param-list-prime> | ',' <param> <param-list-prime>  | λ |
| 8. | <param> | <type-specifier> ID <param-prime> |
| 9. | <param-prime> | '[' ']'  | λ |
| 10. | <compound-stmt> | '{' <local-declarations> <statement-list> '}' |
| 11. | <local-declarations> | <var-declaration> <local-declarations>  | λ |
| 12. | <var-declaration> | <type-specifier> ID <var-declaration-prime> |
| 13. | <var-declaration-prime> | ';'  | '[' NUM ']' ';'  | ‘,’ ID <var-declaration-prime>  | ‘=’ <asg-declaration\_prime> ‘;’ |
| 14. | <asg-declaration\_prime> | ID  | NUM |
| 15. | <statement-list> | <statement> <statement-list>  | λ |
| 16. | <statement> | <expression-stmt>  | <compound-stmt>  | <selection-stmt>  | <iteration-stmt>  | <return-stmt> |
| 17. | <expression-stmt> | <expression> ';'  | ';' |
| 18. | <selection-stmt> | 'if' '(' <expression> ')' <statement> <selection-stmt-prime> |
| 19. | <selection-stmt-prime> | 'else' <statement>  | λ |
| 20. | <iteration-stmt> | 'while' '(' <expression> ')' <statement> |
| 21. | <return-stmt> | 'return' <return-stmt-prime> |
| 22. | <return-stmt-prime> | ';'  | <expression> ';' |
| 23. | <expression> | <simple-expression> <expression-prime> |
| 24. | <expression-prime> | '=' <expression>  | λ |
| 25. | <var> | ID <var-prime> |
| 26. | <var-prime> | '[' <expression> ']'  | λ |
| 27. | <simple-expression> | <additive-expression> <simple-expression-prime> |
| 28. | <simple-expression-prime> | <relop> <additive-expression>  | λ |
| 29. | <relop> | '<'  | '<='  | '>'  | '>='  | '=='  | '!=' |
| 30. | <additive-expression> | <term> <additive-expression-prime> |
| 31. | <additive-expression-prime> | <addop> <term> <additive-expression-prime>  | λ |
| 32. | <addop> | '+'  | '-' |
| 33. | <term> | <factor> <term-prime> |
| 34. | <term-prime> | <mulop> <factor> <term-prime>  | λ |
| 35. | <mulop> | '\*'  | '/' |
| 36. | <factor> | '(' <expression> ')'  | ID <factor-prime>  | NUM |
| 37. | <factor-prime> | '(' <args> ')'  | <var-prime> |
| 38. | <args> | <arg-list>  | λ |
| 39. | <arg-list> | <expression> <arg-list-prime> |
| 40. | <arg-list-prime> | ',' <expression> <arg-list-prime>  | λ |

# FIRST y FOLLOW de C-

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Nonterminal** | **FIRST set** | | **FOLLOW set** | |
| 0. | <program> | { 'int', 'void', ε } | { $ } | |
| 1. | <declaration-list> | { 'int', 'void', ε } | { $ } | |
| 2. | <declaration> | { 'int', 'void' } | { 'int', 'void', $ } | |
| 3. | <declaration-prime> | { ';', '[', '(' , ‘,’, ‘=’ } | { 'int', 'void', $ } | |
| 4. | <type-specifier> | { 'int', 'void' } | { ID } | |
| 5. | <params> | { 'void', 'int' } | { ')' } | |
| 6. | <param-list> | { 'int', 'void' } | { ')' } | |
| 7. | <param-list-prime> | { ',', ε } | { ')' } | |
| 8. | <param> | { 'int', 'void' } | { ',', ')' } | |
| 9. | <param-prime> | { '[', ε } | { ',', ')' } | |
| 10. | <compound-stmt> | { '{' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 11. | <local-declarations> | { 'int', 'void', ε } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', '}' } | |
| 12. | <var-declaration> | { 'int', 'void' } | { 'int', 'void', $ } | |
| 13. | <var-declaration-prime> | { ';', '[' } | { 'int', 'void', $ } | |
| 14. | <asg-declaration-prime> | { ID, NUM } | { ‘;’ } | |
| 15. | <statement-list> | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', ε } | { '}' } | |
| 16. | <statement> | { 'if', 'while', 'return', '{', '(', ID, NUM, ';' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 17. | <expression-stmt> | { '(', ID, NUM, ';' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 18. | <selection-stmt> | { 'if' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 19. | <selection-stmt-prime> | { 'else', ε } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 20. | <iteration-stmt> | { 'while' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 21. | <return-stmt> | { 'return' } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 22. | <return-stmt-prime> | { ';', '(', ID, NUM } | { 'if', 'while', 'return', '{', '(', ID, NUM, ';', 'else', '}' } | |
| 23. | <expression> | { '(', ID, NUM } | { ';', ')', ']', ',', } | |
| 24. | <expression-prime> | { '=', ε } | { ';', ')', ']', ',' } | |
| 25. | <var> | { ID } | { '=', ';', ')', ']', '<', '<=', '>', '>=', '==', '!=', '+', '-', '\*', '/' } | |
| 26. | <var-prime> | { '[', ε } | { '=', ';', ')', ']', '<', '<=', '>', '>=', '==', '!=', '+', '-', '\*', '/' } | |
| 27. | <simple-expression> | { '(', ID, NUM } | { ';', ')', ']', ',', } | |
| 28. | <simple-expression-prime> | { '<', '<=', '>', '>=', '==', '!=', ε } | { ';', ')', ']', ',' } | |
| 29. | <relop> | { '<', '<=', '>', '>=', '==', '!=' } | { '(', ID, NUM } | |
| 30. | <additive-expression> | { '(', ID, NUM } | { ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 31. | <additive-expression-prime> | { '+', '-', ε } | { ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 32. | <addop> | { '+', '-' } | { '(', ID, NUM } | |
| 33. | <term> | { '(', ID, NUM } | { '+', '-', ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 34. | <term-prime> | { '\*', '/', ε } | { '+', '-', ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 35. | <mulop> | { '\*', '/' } | { '(', ID, NUM } | |
| 36. | <factor> | { '(', ID, NUM } | { '\*', '/', '+', '-', ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 37. | <factor-prime> | { '(', '[', ε } | { '\*', '/', '+', '-', ';', ')', ']', ',', '<', '<=', '>', '>=', '==', '!=', } | |
| 38. | <args> | { '(', ID, NUM, ε } | { ')' } | |
| 39. | <arg-list> | { '(', ID, NUM } | { ')' } | |
| 40. | <arg-list-prime> | { ',', ε } | { ')' } | |