Original grammar in BNF form after removing all EBNF grammars:

|  |  |
| --- | --- |
| <prog> | 🡪**program** <identifier>, **var** <dec-list> **begin** <stat-list> **end.** |
| <identifier> | 🡪<letter> <post-identifier> |
| <post-identifier> | 🡪<letter> <post-identifier> |
| <post-identifier> | 🡪 <digit> <post-identifier> |
| <post-identifier> | 🡪λ |
| <dec-list> | 🡪 <dec> : <type> ; |
| <dec> | 🡪<identifier>, <dec> |
| <dec> | 🡪<identifier> |
| <type> | 🡪**integer** |
| <stat-list> | 🡪<stat> |
| <stat-list> | 🡪 <stat> <stat-list> |
| <stat> | 🡪 <write> |
| <stat> | 🡪 <assign> |
| <write> | 🡪 write( <str> <identifier> ); |
| <str> | 🡪 “value=” , |
| <str> | 🡪λ |
| <assign> | 🡪<identifier> = <expr>; |
| <expr> | 🡪<expr> + <term> |
| <expr> | 🡪<expr> - <term> |
| <expr> | 🡪<term> |
| <term> | 🡪<term> \* <factor> |
| <term> | 🡪 <term> / <factor> |
| <term> | 🡪 <factor> |
| <factor> | 🡪<identifier> |
| <factor> | 🡪 <number> |
| <factor> | 🡪 ( <expr> ) |
| <number> | 🡪 <sign> <digit> <post-number> |
| <post-number> | 🡪 <digit> <post-number> |
| <post-number> | 🡪λ |
| <sign> | 🡪 + |
| <sign> | 🡪- |
| <sign> | 🡪λ |
| <digit> | 🡪0 |
| <digit> | 🡪1 |
| <digit> | 🡪2 |
| <digit> | 🡪3 |
| <digit> | 🡪4 |
| <digit> | 🡪5 |
| <digit> | 🡪6 |
| <digit> | 🡪7 |
| <digit> | 🡪8 |
| <digit> | 🡪9 |
| <letter> | 🡪p |
| <letter> | 🡪q |
| <letter> | 🡪r |
| <letter> | 🡪s |

Final BNF grammar for predictive parsing method after removing left-recursive rules:

|  |  |
| --- | --- |
| <prog> | 🡪**program** <identifier>, **var** <dec-list> **begin** <stat-list> **end.** |
| <identifier> | 🡪<letter> <post-identifier> |
| <post-identifier> | 🡪<letter> <post-identifier> |
| <post-identifier> | 🡪 <digit> <post-identifier> |
| <post-identifier> | 🡪λ |
| <dec-list> | 🡪 <dec> : <type> ; |
| <dec> | 🡪<identifier>, <dec> |
| <dec> | 🡪<identifier> |
| <type> | 🡪**integer** |
| <stat-list> | 🡪<stat> |
| <stat-list> | 🡪 <stat> <stat-list> |
| <stat> | 🡪 <write> |
| <stat> | 🡪 <assign> |
| <write> | 🡪 write( <str> <identifier> ); |
| <str> | 🡪 “value=” , |
| <str> | 🡪λ |
| <assign> | 🡪<identifier> = <expr>; |
| <expr> | 🡪<term> <post-expr> |
| <post-expr> | 🡪+ <term> <post-expr> |
| <post-expr> | 🡪 - <term> <post-expr> |
| <post-expr> | 🡪λ |
| <term> | 🡪<factor> <post-term> |
| <post-term> | 🡪\* <factor> <post-term> |
| <post-term> | 🡪 / <factor> <post-term> |
| <post-term> | 🡪 λ |
| <factor> | 🡪<identifier> |
| <factor> | 🡪 <number> |
| <factor> | 🡪 ( <expr> ) |
| <number> | 🡪 <sign> <digit> <post-number> |
| <post-number> | 🡪 <digit> <post-number> |
| <post-number> | 🡪λ |
| <sign> | 🡪 + |
| <sign> | 🡪- |
| <sign> | 🡪λ |
| <digit> | 🡪0 |
| <digit> | 🡪1 |
| <digit> | 🡪2 |
| <digit> | 🡪3 |
| <digit> | 🡪4 |
| <digit> | 🡪5 |
| <digit> | 🡪6 |
| <digit> | 🡪7 |
| <digit> | 🡪8 |
| <digit> | 🡪9 |
| <letter> | 🡪p |
| <letter> | 🡪q |
| <letter> | 🡪r |
| <letter> | 🡪s |

List all terminals and non-terminals:

|  |  |
| --- | --- |
| **list of non-terminals** | **list of terminals** |
| |  |  | | --- | --- | | old name | new name | | <prog> | P | | <identifier> | I | | <post-identifier> | PI | | <dec-list> | DL | | <dec> | D | | <type> | T | | <stat-list> | SL | | <stat> | S | | <write> | W | | <str> | St | | <assign> | A | | <expr> | E | | <post-expr> | PE | | <term> | Te | | <post-term> | PT | | <factor> | F | | <number> | N | | <post-number> | PN | | <sign> | Sn | | <digit> | Di | | <letter> | Le | | λ 0 1 2 3 4 5 6 7 8 9 p q r s ( ) \* / + - = ; “value=” , write integer : program var begin end. |

Find members of FIRST and FOLLOW:

|  |  |  |  |
| --- | --- | --- | --- |
| Non-terminals | | FIRST | FOLLOW |
| P | <prog> | program | $ |
| I | <identifier> | p q r s | , : ) = \* / + - ; ) |
| PI | <post-identifier> | p q r s 0 1 2 3 4 5 6 7 8 9 λ | , : ) = \* / + - ; ) |
| DL | <dec-list> | p q r s | begin |
| D | <dec> | p q r s | : |
| T | <type> | integer | ; |
| SL | <stat-list> | write p q r s | end. |
| S | <stat> | write p q r s | end. write p q r s |
| W | <write> | write | end. write p q r s |
| St | <str> | “value=” λ | p q r s |
| A | <assign> | p q r s | end. write p q r s |
| E | <expr> | p q r s + - 0 1 2 3 4 5 6 7 8 9 ( | ; ) |
| PE | <post-expr> | + - λ | ; ) |
| Te | <term> | p q r s + - 0 1 2 3 4 5 6 7 8 9 ( | + - ; ) |
| PT | <post-term> | \* / λ | + - ; ) |
| F | <factor> | p q r s + - 0 1 2 3 4 5 6 7 8 9 ( | \* / + - ; ) |
| N | <number> | + - 0 1 2 3 4 5 6 7 8 9 | \* / + - ; ) |
| PN | <post-number> | 0 1 2 3 4 5 6 7 8 9 λ | \* / + - ; ) |
| Sn | <sign> | + - λ | 0 1 2 3 4 5 6 7 8 9 |
| Di | <digit> | 0 1 2 3 4 5 6 7 8 9 | p q r s 0 1 2 3 4 5 6 7 8 9 , : ) = \* / + - ; ) |
| Le | <letter> | p q r s | p q r s 0 1 2 3 4 5 6 7 8 9 , ; ) = \* / + - ; ) |

Predictive Parsing table: