|  |  |  |
| --- | --- | --- |
|  | First | Follow |
| Program | { Category } | {$} |
| Program` | { Category , Ɛ } | {$} |
| ClassDeclaration | { Category} | {End} |
| CD’ | { { , Derive } | {End} |
| Class\_Implementation | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , Ɛ } | { } } |
| Var\_Method | { ID } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| Var\_Method ` | {, , (} | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| MethodDeclaration | { ( } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| Me\_Declaration | { ; , { } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| Func Decl | { ( } | { ; , { } |
| Type | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical } | {ID} |
| ParameterList | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, Ɛ } | { ) } |
| Non-Empty List | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical } | { ) } |
| Non-Empty List` | {, , Ɛ } | { ) } |
| VarDeclaration | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,Ɛ } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write ,} } |
| ID\_List | {,} | { ; } |
| ID\_List` | {, , Ɛ } | { ; } |
| Statements | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write ,Ɛ } | { } } |
| Statement | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical ,ID,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Assignment | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical ,ID,= } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Func \_Call | {ID} | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| Argument\_List | { ID,Number , Ɛ } | { ) } |
| NonEmpty\_Argument\_List | { ID,Number } | { ) } |
| NonEmpty\_Argument\_List` | { , , Ɛ } | { ) } |
| Block Statements | { { } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write ,} } |
| If \_Statement | {if} | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Condition \_Expression | { ID,Number } | { ) } |
| Co\_Expression | { and,or , ε } | { ) } |
| Condition \_Op | {and,or} | { ID,Number } |
| Condition | { ID,Number } | { and,or , ) } |
| Comparison \_Op | {== , != , > , >= , < , <=} | { ID,Number } |
| Rotatewhen \_Statement | { Rotatewhen } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Continuewhen \_Statement | { Continuewhen } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Replywith \_Statement | { Replywith ,return} | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| terminatethis \_Statement | { terminatethis } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical,ID ,= ,if, Rotatewhen , Continuewhen, Replywith ,return, terminatethis, read, write , } } |
| Expression | { ID,Number } | { ) , ; , , , ==,!= , > , >= , < , <=} |
| Expression` | {+ , -, ε } | { ) , ; , , , ==,!= , > , >= , < , <=} |
| Add\_Op | {+ , -} | { ID,Number } |
| Term | { ID,Number } | {+ , -, ) , ; , , , ==,!= , > , >= , < , <=} |
| Term` | {\* , /, ε } | {+ , -, ) , ; , , , ==,!= , > , >= , < , <= } |
| Mul\_Op | {\* , /} | {ID,Number } |
| Factor | {ID,Number } | {\* , /, + , -, ) , ; , , , ==,!= , > , >= , < , <= } |
| Comment | {< , -} | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| using\_command | { using } | { Ilap , Silap , Clop , Series , Ilapf , Silapf , None , Logical, < , -, using, ID , } } |
| F\_name | { STR } | {.txt} |

