\Programa → Classe **EOF**

Classe → "class" ID ":" ListaFuncao Main "end" "."

DeclaraID → TipoMacro ID ";"

ListaFuncao → ListaFuncao Funcao | ε

Funcao → "def" TipoMacro ID "(" ListaArg ")" ":" (DeclaraID)∗ ListaCmd Retorno "end" ";"

ListaArg → Arg "," ListaArg | Arg | ε

Arg → TipoMacro ID

Retorno → "return" Expressao ";" | ε

Main → "defstatic" "void" "main" "(" "String" "[" "]" ID ")" ":" (DeclaraID)∗ ListaCmd "end" ";"

TipoMacro → TipoPrimitivo "[" "]" | TipoPrimitivo

TipoPrimitivo → "bool" | "integer" | "String" | "double" | "void"

ListaCmd → ListaCmd Cmd | ε

Cmd → CmdIF | CmdWhile | CmdAtribui | CmdFuncao | CmdWrite | CmdWriteln

CmdIF → "if" "(" Expressao ")" ":" ListaCmd "end" ";"

| "if" "(" Expressao ")" ":" ListaCmd "eles" ":" ListaCmd "end" ";"

CmdWhile → "while" "(" Expressao ")" ":" ListaCmd "end" ";"

CmdWrite → "write" "(" Expressao ")" ";"

CmdWriteln → "writeln" "(" Expressao ")" ";"

CmdAtribui → ID "=" Expressao ";"

| ID "[" Expressao "]" "=" Expressao ";"

CmdFuncao → ID "(" (Expressao ("," Expressao)∗ )? ")" ";"

Expressao → Expressao Op Expressao

| ID "[" Expressao"]" | ID

| ID "(" (Expressao ("," Expressao)∗ )? ")"

| ConstInteger | ConstDouble | ConstString | "true" | "false"

| "vector" TipoPrimitivo "[" Expressao "]"

| OpUnario Expressao | "(" Expressao")"

Op → "or" | "and" | "<" | "<=" | ">" | ">=" | "==" | "!=" | "/" | "\*" | "-" | "+"

OpUnario → "-" | "!"

Tratando a expressão regular de Fecho de Kleene

(DeclaraID)\* = FDID🡪DeclaraID FDID | ε

( , Expressao)\* = FEXP 🡪 , Expressao FEXP | ε

(Expressao FEXP)? = FEXP2 🡪Epressao FEXP | ε

Eliminando a ambiguidade sobre precedência de operadores

Expressao → Expressao or Expressao1 | Expressao and Expressao1| Expressao1

Expressao1 →| Expressao1 < Expressao2| Expressao1 <= Expressao2| Expressao1 > Expressao2

| Expressao1 >= Expressao2| Expressao1 == Expressao2| Expressao1 != Expressao2

| Expressao1 = Expressao2| Expressao2

Expressao2 → Expressao2 + Expressao3| Expressao2 – Expressao3 | Expressao3

Expressao3 → Expressao3\* Expressao4| Expressao3 / Expressao4 | Expressao4

Expressao4 → ID "[" Expressao"]" | ID | ID "(" (Expressao ("," Expressao)∗ )? ")"

| ConstInteger | ConstDouble | ConstString | "true" | "false"

| "vector" TipoPrimitivo "[" Expressao "]"| OpUnario Expressao | "(" Expressao")"

Eliminando recursão

ListaFuncao → ListaFuncao Funcao | ε

Como Ficou

ListaFuncao 🡪 ListaFuncao’

ListaFuncao’🡪Funcao ListaFuncao’ | ε

ListaCmd → ListaCmd Cmd | ε

Como Ficou

ListaCmd 🡪 ListaCmd’

ListaCmd’🡪Cmd ListaCmd’ | ε

Expressao → Expressao1 Expressao’

Expressao’ 🡪or Expressao’| and Expressao’ | ε

Expressao1 → Expressao2 Expressao1’

Expressao1’ → < Expressao1’| <= Expressao1’| > Expressao1’| >= Expressao1’| == Expressao1’

| != Expressao1’| = Expressao1’

Expressao2 → Expressao3 Expressao2’

Expressao2’ → + Expressao2’| – Expressao2’| ε

Expressao3 → Expressao4 Expressao3’

Expressao3’ →\* Expressao3’| / Expressao3’| ε

Fatoração á Esquerda

CmdIF → "if" "(" Expressao ")" ":" ListaCmd "end" ";"

| "if" "(" Expressao ")" ":" ListaCmd "eles" ":" ListaCmd "end" ";"

Como ficou

CmdIF →"if" "(" Expressao ")" ":" ListaCmd CmdIF’

CmdIF’🡪 end;| else : ListaCmd end ;

CmdAtribui → ID "=" Expressao ";"

| ID "[" Expressao "]" "=" Expressao ";"

Como ficou

CmdAtribui🡪ID CmdAtribui’

CmdAtribui’🡪"=" Expressao ";" | "[" Expressao "]" "=" Expressao ";"

ListaArg → Arg "," ListaArg | Arg | ε

Como Ficou

ListaArg🡪 Arg ListaArg’ | ε

ListaArg’🡪 “,” ListaArg | ε

TipoMacro → TipoPrimitivo "[" "]" | TipoPrimitivo

Como Ficou

TipoMacro → TipoPrimitivo TipoMacro’

TipoMacro’🡪"[" "]" | ε

Expressao4 → ID "[" Expressao"]" | ID | ID "(" (FEXP2) ")"

| ConstInteger | ConstDouble | ConstString | "true" | "false"

| "vector" TipoPrimitivo "[" Expressao "]"| OpUnario Expressao | "(" Expressao")"

Como ficou

Expressao4 → ID Expressão4’ | ConstInteger | ConstDouble | ConstString | "true" | "false"

| "vector" TipoPrimitivo "[" Expressao "]"| OpUnario Expressao | "(" Expressao")"

Expressao4’🡪”[“Expressao”]” | "(" (FEXP2) ")" | ε

**Gramática Corrigida**

Programa → Classe **EOF**

Classe → "class" ID ":" ListaFuncao Main "end" "."

DeclaraID → TipoMacro ID ";"

ListaFuncao 🡪 ListaFuncao’

ListaFuncao’🡪Funcao ListaFuncao’ | ε

Funcao → "def" TipoMacro ID "(" ListaArg ")" ":" FDID ListaCmd Retorno "end" ";"

FDID🡪DeclaraID FDID | ε

ListaArg🡪 Arg ListaArg’ | ε

ListaArg’🡪 “,” ListaArg | ε

Arg → TipoMacro ID

Retorno → "return" Expressao ";" | ε

Main → "defstatic" "void" "main" "(" "String" "[" "]" ID ")" ":" FDID ListaCmd "end" ";"

TipoMacro → TipoPrimitivo TipoMacro’

TipoMacro’🡪"[" "]" | ε

TipoPrimitivo → "bool" | "integer" | "String" | "double" | "void"

ListaCmd 🡪 ListaCmd’

ListaCmd’🡪Cmd ListaCmd’ | ε

Cmd → CmdIF | CmdWhile | CmdWrite | CmdWriteln | Cmd’

CmdIF →"if" "(" Expressao ")" ":" ListaCmd CmdIF’

CmdIF’🡪 end “;” | else “:” ListaCmd end” ;”

CmdWhile → "while" "(" Expressao ")" ":" ListaCmd "end" ";"

CmdWrite → "write" "(" Expressao ")" ";"

CmdWriteln → "writeln" "(" Expressao ")" ";"

Cmd’🡪ID Cmd’’

Cmd’’🡪CmdAtribui’ | "(" FEXP2 ")" ";"

CmdAtribui’🡪"=" Expressao ";" | "[" Expressao "]" "=" Expressao ";"

FEXP 🡪” ,” Expressao FEXP | ε

FEXP2 🡪Epressao FEXP | ε

Expressao → Expressao1 Expressao’

Expressao’ 🡪or Expressao1Expressao’| and Expressao1 Expressao’ | ε

Expressao1 → Expressao2 Expressao1’

Expressao1’ → < Expressao2 Expressao1’| <= Expressao2 Expressao1’

| > Expressao2 Expressao1’| >= Expressao2 Expressao1’

| == Expressao2 Expressao1’ | != Expressao2 Expressao1’| ε

Expressao2 → Expressao3 Expressao2’

Expressao2’ → + Expressao3 Expressao2’| – Expressao3 Expressao2’| ε

Expressao3 → Expressao4 Expressao3’

Expressao3’ →\* Expressao4 Expressao3’| / Expressao4 Expressao3’| ε

Expressao4 → ID Expressão4’ | ConstInteger | ConstDouble | ConstString | "true" | "false"

| "vector" TipoPrimitivo "[" Expressao "]"| OpUnario Expressao | "(" Expressao")"

Expressao4’🡪”[“Expressao”]” | "(" FEXP2 ")" | ε

OpUnario → "-" | "!”

|  |  |  |
| --- | --- | --- |
|  | FIRST | FOLLOW |
| Programa | class | $ |
| Classe | class | **EOF** |
| DeclaraID | bool, integer, String, Double, void | bool, integer, String, Double, void, IF, while, ID, write, writeln, Def, Defstatic, end |
| ListaFuncao | Def, ε | Defstatic |
| ListaFuncao’ | Def, ε | Defstatic |
| Funcao | Def | Def, Defstatic |
| FDID | bool, integer, String, Double, void, ε | IF, while, ID, write, writeln, Def, Defstatic, end |
| ListaArg | bool, integer, String, Double, void, ε | “)” |
| ListaArg’ | “,” , ε | “)” |
| Arg | bool, integer, String, Double, void | “,”,“)” |
| Retorno | Return, ε | end |
| Main | Defstatic | end |
| TipoMacro | bool, integer, String, Double, void | ID |
| TipoMacro’ | “[“ , ε | ID |
| TipoPrimitivo | bool, integer, String, Double, void | “[“, ID |
| ListaCmd | IF, while, ID, write, writeln, ε | Return, end, else, Def, Defstatic |
| ListaCmd’ | IF, while, ID, write, writeln, ε | Return, end, else, Def, Defstatic |
| Cmd | IF, while, ID, write, writeln | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| CmdIF | IF | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| CmdIF’ | End,else | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| CmdWhile | While | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| CmdWrite | Write | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| Cmdwriteln | Writeln | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| Cmd’ | ID | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| Cmd’’ | “=”, “[“, “(“ | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| CmdAtribui’ | “=”, “[“ | IF, while, ID, write, writeln, Return, end, else, Def, Defstatic |
| FEXP | “,” , ε | “)” |
| FEXP2 | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, (,ε | “)” |
| Expressao | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao’ | Or, and, ε | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao1 | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao1’ | "<", "<=" , ">" , ">=", "==" , "!=", ε | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao2 | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao2’ | +,-, ε | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao3 | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao3’ | \*, /, ε | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao4 | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| Expressao4’ | “[“ , “(“,ε | \*, /,+,-,"<", "<=" , ">" , ">=", "==" , "!=", Or, and, “;”, “)”, “]”,“,” |
| OpUnario | -, ! | ID, ConstInteger , ConstDouble , ConstString , "true" , "false", vector, -, !, ( |