FIRST

<Acessando>={. ,[}

<AcessandoAux>={. ,[,E}

<Acesso>={. ,[}

<Bloco>={ { }

<BlocoAux>= { --, ++, !, (, }, ++, CadeiaDeCaracteres, Digito, false, identificador, if, print, return, scan, struct, true, typ
while }

<Declaracao>={ const, function, procedure, start, typedef, var }

<DeclaracaoDeConst>={ const }

<DeclaracaoDeConstanteCorpo>={ bool, float, identificador, int, string, struct }

<DeclaracaoDeConstanteCorpoAux>={   bool, float, identificador, int, string, struct, E}

<DeclaracaoDeConstanteLinha>= { bool, float, identificador, int, string, struct }

<DeclaracaoDeFuncao>={function }

<DeclaracaoDeInicio>={ start }

<DeclaracaoDeProcedimento>= { procedure }

<DeclaracaoDeStruct>= { struct }

<DeclaracaoDeStructAux>= { {, extends,identificador }

<DeclaracaoDeStructCorpo>= { bool, float, identificador, int, string, struct }

<DeclaracaoDeStructCorpoAux>= { bool, float, identificador, int, string, struct, E }

<DeclaracaoDeStructLinha>= { bool, float, identificador, int, string, struct }

<DeclaracaoDeTypedef>= { typdef}

<DeclaracaoDeTypedefAux>= { bool, float, identificador, int, string, struct }

<DeclaracaoDeVar>= { var }

<DeclaracaoDeVariavelCorpo>={ bool, float, identificador, int, string, struct }

<DeclaracaoDeVariavelCorpoAux>={bool, float, identificador, int, string, struct,E }

<DeclaracaoDeVariavelLinha>={ bool, float, identificador, int, string, struct }

<Entrada>= { identificador}

<Escalar>={ bool, float, int, string }

<EscalarRelacional>= { != , < , <= , == , >=, > }

<EstruturaCondicional>= { if }

<EstruturaCondicionalAux>={else, E }

<Expressao>= { -- , ! , (, ++ , CadeiaDeCaracteres, Digitos, false, identificador, true }

<ExpressaoAux>={ E, || }

<ExpressaoIdentificadorConst>= { Identificador }

<ExpressaoIdentificadoresConst>={ Identificador }

<ExpressaoIdentificadoresConstAux>={ ‘,’ , ; }

<ExpressaoIdentificadoresStruct>={ Identificador }

<ExpressaoIdentificadoresStructAux>={ ‘,’ , ; }

<ExpressaoIdentificadoresVar>= { Identificador}

<ExpressaoIdentificadoresVarAux> = { ‘,’ , ; }

<ExpressaoIdentificadorStruct>= { Identificador }

<ExpressaoIdentificadorVar>= {Identificador }

<ExpressaoIdentificadorVarAux>= { =, E }

<Extends>= { extends, E }

<Final>= { Identificador }

<FuncaoProcedimentoFim>={), bool, float, identificador,int, string, struct }

<FuncID>= { bool, float, identificador,int, string, struct }

<IfThen>={ if }

<Instrucao>={ --, !, (, ++, CadeiadeCaracter, Digitos, false, identificador, if, print, return,scan, struct, true, typedef, var, while }

<InstrucaoDeRetornoAux>= { E,--, ! , (, ++ , CadeiadeCaracter , Digitos, false, identificador, true }

<InstrucaoNormal>={ --, !, (, ++ , CadeiadeCaracter, Digitos, false, identificador, print, return, scan, struct, true }

<ListaDeIntrucoes> = { --, !, (, ++, CadeiaDeCaracter, Digitos, false, identificador, print, return , scan, struct,true, typdef, var,while}

<ListaDeIntrucoesAux> = { --, !, (, ++, CadeiaDeCaracter, Digitos, false, identificador, print, return , scan, struct,true, typdef, var,while, E }

<OpE> ={ --, !, (, ++, CadeiadeCaracter, digitos, false,identificador, true }

<OpEAux> = { &&, E }

<OperacaoDeAtribuicao> = { -- , ! , (, ++ , CadeiadeCaracteres, Digitos, false, identificador, true }

<OpMult> = { --, !, (, ++,CadeiadeCaracteres, Digitos, false, identificador, true }

<OpMultAux> = { E, * , / }

<OpRelacional> = { --, ! , (, ++ , cadeiaDeCaracteres, Digitos, false, identificador, true }

<OpRelacionalAux> = { E, != , < , <=, == , > , >= }

<OpUnary>= { --, ! , (, ++ ++, CadeiadeCaracter, digitos, false, identificador, true }

<OutrasEntradas> = { E, ‘,’ }

<OutrasSaidas> = { E, ‘,’ }

<Parametro> = { bool, float, identificador, int, string,struct }

<Parametros> = { bool, float, identificador, int, string,struct }

<ParametrosAux> ={ E, ‘,’ }

<ParametrosFuncao> = { -- , ! , (, ++, CadeiaDeCaracter, ditigos, false, identificador, true }

<ParametrosFuncaoAux> = { E, ‘,’ }

<Print>= { print }

<Programa> = {const, function, procedure, start, struct, typedef, var }

<ProgramaAux> = { const, function, procedure, start, struct, typedef, var, E }

<Saida> = { -- , !, (, ++ , CadeiaDeCaracter, Digitos, false, identificador, true }

<Scan> = { scan }

<SimboloUnario> = { --, ++, E }

<Tipo> = { bool, float, identificador,int, string, struct }

<TipoAux> = { E, [}

<TipoBase> = { bool, float, identificador,int string,struct }

<TipoVetorDeclarado> = { [}

<TipoVetorDeclarando> = { { }

<TipoVetorDeclarandoAux> = { [, E }

<Valor> = { (, cadeiadeCaracter, Digitos, false, identificador, true }

<ValorAux1> = { (, E }

<ValorAux2> = { --, !, (,) , ++ , CadeiadeCaracter, digitos, false, identificador, true }

<ValorRelacional> = { --,!, (, ++, CadeiadeCaracter, digitos, false, identificador, true }

<ValorRelacionalAux> = { E, - , + }

<While> = { while }

FOLLOW

<Acessando> = { =, -- , ++, *, /, - , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<AcessandoAux> = { =, -- , ++, *, /, - , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<Acesso> = { =, -- , ++, *, /, - , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<Bloco> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, Identificador, print, return , scan, struct, true, typdef, var, while, else, } , const, function, procedure, start, $ }

<BlocoAux> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, Identificador, print, return , scan, struct, true, typdef, var, while, else, } , const, function, procedure, start, $ }

<Declaracao> = { const, function, procedure, start, struct, typedef, var, $ }

<DeclaracaoDeConst> = { const, function, procedure, start, struct, typedef, var, $ }

<DeclaracaoDeConstanteCorpo> = { } }

<DeclaracaoDeConstanteCorpoAux> = { } }

<DeclaracaoDeConstanteLinha> = { bool, float, identificador, int, string, struct } }

<DeclaracaoDeFuncao> = { const, function, procedure, start, struct, typedef, var, $ }

<DeclaracaoDeInicio> = { const, function, procedure, start, struct, typedef, var, $ }

<DeclaracaoDeProcedimento> = { const, function, procedure, start, struct, typedef, var, $ }

<DeclaracaoDeStruct> = { const, function, procedure, start, struct, typedef, var, $,; , [, Identificador }

<DeclaracaoDeStructAux> = { const, function, procedure, start, struct, typedef, var, $,; , [, Identificador }

<DeclaracaoDeStructCorpo> = { } }

<DeclaracaoDeStructCorpoAux> = { } }

<DeclaracaoDeStructLinha> = { bool, float, identificador, int, string, struct, } }

<DeclaracaoDeTypedef> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, }, const, function, procedure, start, $ }

<DeclaracaoDeTypedefAux> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, }, const, function, procedure, start, $ }

<DeclaracaoDeVar> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, }, const, function, procedure, start, $ }

<DeclaracaoDeVariavelCorpo> = { } }

<DeclaracaoDeVariavelCorpoAux> = { } }

<DeclaracaoDeVariavelLinha> = {bool, float, identificador, int, string, struct, } }

<Entrada> = { ‘,’,) }

<Escalar> = { [, Identificador }

<EscalarRelacional> = { --, ! , (, ++ , CadeiaDeCaracteres, Digitos, false, identificador, true }

<EstruturaCondicional> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, }

<EstruturaCondicionalAux> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, }

<Expressao> = {), ‘,’, ;,] }

<ExpressaoAux> = {), ‘,’, ;,] }

<ExpressaoIdentificadorConst> = { ‘,’ , ; }

<ExpressaoIdentificadoresConst> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadoresConstAux> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadoresStruct> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadoresStructAux> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadoresVar> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadoresVarAux> = { bool, float, identificador, int, string, struct } }

<ExpressaoIdentificadorStruct> = { ‘,’ , ; }

<ExpressaoIdentificadorVar> = { ‘,’ , ; }

<ExpressaoIdentificadorVarAux> = { ‘,’ , ; }

<Extends> = { { }

<Final> = { =, -- , ++, *, /, - , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<FuncaoProcedimentoFim> = { const, function, procedure, start, struct, typedef, var, $ }

<FuncID> = { (}

<IfThen>= { else, --, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true,

typdef, var, while, } }

<Instrucao> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, } }

<InstrucaoDeRetornoAux> = { ; }

<InstrucaoDeRetorno> = { ; }

<InstrucaoNormal> = {--, !, (, ++, CadeiaDeCaracteres, Digitos, false, identificador, print, return , scan, struct, true, typdef, var, while, } }

<ListaDeIntrucoes> = { } }

<ListaDeIntrucoesAux> = { } }

<OpE>= { || ,), ‘,’, ;,] }

<OpEAux> = { || ,), ‘,’, ;,] }

<OperacaoDeAtribuicao> = { ; }

<OpMult> = {- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<OpMultAux> = {- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<OpRelacional> = { &&, || ,), ‘,’, ;,] }

<OpRelacionalAux> = { &&, || ,), ‘,’, ;,] }

<OpUnary> = {* , /,- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<OutrasEntradas> = {) }

<OutrasSaidas> = {) }

<Parametro> = { ‘,’,) }

<Parametros> = {) }

<ParametrosAux> = {) }

<ParametrosFuncao> = {) }

<ParametrosFuncaoAux> = {) }

<Print> = { ; }

<Programa> = { $ }

<ProgramaAux>= { $ }

<Saida> = {‘,’ ,) }

<Scan> = { ; }

<SimboloUnario> = { * , /,- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<Tipo> = { Identificador }

<TipoAux> = { Identificador }

<TipoBase> = { [, Identificador }

<TipoVetorDeclarado> = { [, Identificador }

<TipoVetorDeclarando> = { Identificador }

<TipoVetorDeclarandoAux> = { Identificador }

<Valor>= {--, ++, * , /,- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<ValorAux1>= {--, ++, * , /,- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<ValorAux2> = {--, ++, * , /,- , + != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<ValorRelacional>= { != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<ValorRelacionalAux>= { != , < , <=, == , > , >= , &&, || ,), ‘,’, ;,] }

<While> = { --, !, (, ++, CadeiaDeCaracter, Digitos, false, identificador, print, return , scan, struct,true, typdef, var,while, } }