**BNF da Linguagem Bacco**

**Programa**

<inicio> ::= Start<fimDaLinha> (<corpo>)+ End<fimDaLinha>

<corpo> ::= <dvar> <comandos>

<comandos> ::= <attr>

| <cond>

| <read>

| <print>

| <loop>

**Declarations**

<dvar> ::= <var><fimDaLinha> <declaVar> (<virgula><declaVar>)\* <endVar><fimDaLinha>

<declaVar> ::= <tipo> <id>(<virgula><id>)\*<fimDaLinha>

<var>::= Var

<endVar>::= endVar

**Types**

<tipo> ::= <tipoNumerico> | <tipoTexto>

<tipoNumerico> ::= number

<tipoTexto> ::= text

**Blocks and Commands**

<attr> ::= <id> = <attrConteudo><fimDaLinha>

<attrConteudo> := <id> | <tipoIntegral> | <tipoDecimal> | <aspas><texto> <aspas> | <aspas><texto> <aspas><concatenar> <aspas><texto> <aspas><fimDaLinha>

<concatenar>::= #

<aspas>::= “

<cond> ::= <condicional1> | <condicional2>

<condicional1> ::= <if><abreParenteses><expressão><fechaParenteses> <corpoComando>

<condicional2> ::= <if><abreParenteses> <expressão><fechaParenteses> <corpoComando><else> <corpoComando>

<if>::= IsTrue

<else>::= IsNot

<corpoComando>::= <abreChave><comandos><fechaChave>

<abreChave>::= {

<fechaChave>::= }

<abreParenteses>::= (

<fechaParenteses>::= )

<read> ::= <id> = <read><abreParenteses>(<aspas><texto> <aspas>)\* <fechaParenteses> <fimDaLinha>

<read>::= read

<print> ::= <print ><abreParenteses> <aspas><texto> <aspas> | <id> | (<virgula> <id> )\* <fechaParenteses> <fimDaLinha>

<print>::= print

<virgula>::= ,

<loop> ::= <repeat><abreParenteses> <expressão><fechaParenteses> <corpoComando>

<repeat>::= repeat

**Expressions**

<expressão> ::= <expressãoRelacional> | <expressãoMatematica>

<expressãoRelacional>::= <expressãoSimples> (<relação> <expressãoSimples>)\*

<relação> ::= == | != | < | <= | >= | >

<expressãoSimples> ::= <condicionalAnd> | <condicionalOr> | <expressãoSimples>

<expressãoMatematica>::= <tipoNumerico> <operadorMatematico> <expressãoMatematica> | <tipoNumerico> <operadorMatematico> <tipoNumerico>

<operadorMatematico>::= <soma> | <sub> | <divisao> | <multi> | <potencia>

<soma>::= +

<sub>::= -

<divisao>::= /

<multi>::= \*

<potencia>::= ^

<and> ::= &&

<or> ::= ||

<not>::= !

<condicionalAnd> ::= <fator> <and> <fator> | <fator> <and> <expressão>

<condicionalOr> ::= <fator> <or> <fator>| <fator> <or> <expressão>

<fator> ::= <id>

| <tipoIntegral>

| <tipoDecimal>

| <texto>

| <expressão>

| <not><fator>

<tipoIntegral> ::= (["0"-"9"])+

<tipoDecimal> ::= (["0"-"9"])\*(".")?(["0"-"9"])+

<texto> ::= ["a"-"z","A"-"Z"](["0"-"9"]|["a"-"z","A"-"Z"])\*

<listaDeExpressões> ::= <expressão> {<virgula> <expressão>}

<chamadaDeFunção> ::= <id> [ (<listaDeExpressões> ) ]

<fimDaLinha>::= ;

<keyword> ::= IsTrue | IsNot | Start | End | Repeat | Var | endVar | number | text | read | print