**BNF da Linguagem Bacco**

**Programa**

<inicio> ::= Start<fim da linha> (<comandos>)\* End<fim da linha>

<comandos> ::= <dvar>

| <attr>

| <cond>

| <read>

| <print>

| <loop>

| <start>

| <end>

**Declarations**

<dvar> ::= <var><fim da linha> <declavar> (<virgula><declavar>)\* <endvar><fim da linha>

<declavar> ::= <tipo> <id>(<virgula><id>)\*<fim da linha>

<var>::= Var

<endvar>::= EndVar

**Types**

<tipo> ::= <tipo numerico> | <tipo texto>

<tipo numerico> ::= number

<tipo texto> ::= text

**Blocks and Commands**

<attr> ::= <id> = <attrConteudo><fim da linha>

<attrConteudo> := <id> | <tipo integral> | <tipo decimal> | <aspas><texto> <aspas> | <aspas><texto> <aspas><concatenar> <aspas><texto> <aspas>

<concatenar>::= #

<aspas>::= “

<cond> ::= <condicional 1> | <condicional 2>

<condicional 1> ::= <if><abre parenteses><expressão><fecha parenteses> <corpo comando>

<condicional 2> ::= <if><abre parenteses> <expressão><fecha parenteses> <corpo comando>IsNot <corpo comando>

<if>::= IsTrue

<else>::= IsNot

<corpo comando>::= <abre chave><comandos><fecha chave>

<abre chave>::= {

<fecha chave>::= }

<abre parenteses>::= (

<fecha parenteses>::= )

<read> ::= <variável> = <read><abre parenteses>(<aspas><texto> <aspas>)\* <fecha parenteses>

<read>::= read

<print> ::= <print ><abre parenteses> <aspas><texto> <aspas> | <variável> | (<virgula> <variável> )\* <fecha parenteses>

<print>::= print

<virgula>::= ,

<loop> ::= <repeat><abre parenteses> <expressão><fecha parenteses> <corpo comando>

<repeat>::= repeat

**Expressions**

<expressão> ::= <expressão relacional> | <expressão matematica>

<expressão relacional>::= <expressão simples> (<relação> <expressão simples>)\*

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

<expressão simples> ::= <condicional and> | <condicional or> | <expressão simples>

<expressão matematica>::= <tipo numerico> <operador matematico> <expressão matematica> | <tipo numerico> <operador matematico> <tipo numerico>

<operador matematico>::= <soma> | <sub> | <divisao> | <multi> | <potencia>

<soma>::= +

<sub>::= -

<divisao>::= /

<multi>::= \*

<potencia>::= ^

<and> :== &&

<or> ::= ||

<not>::= !,,

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

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

<fator> ::= <variável>

| <tipo integral>

| <tipo decimal>

| <texto>

| ( <expressão> )

| !<fator>

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

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

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

<variável> ::= <id>

<lista de expressões> ::= <expressão> {<virgula> <expressão>}

<chamada de função> ::= <id> [ (<lista de expressões> ) ]

<fim da linha>::= ;

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