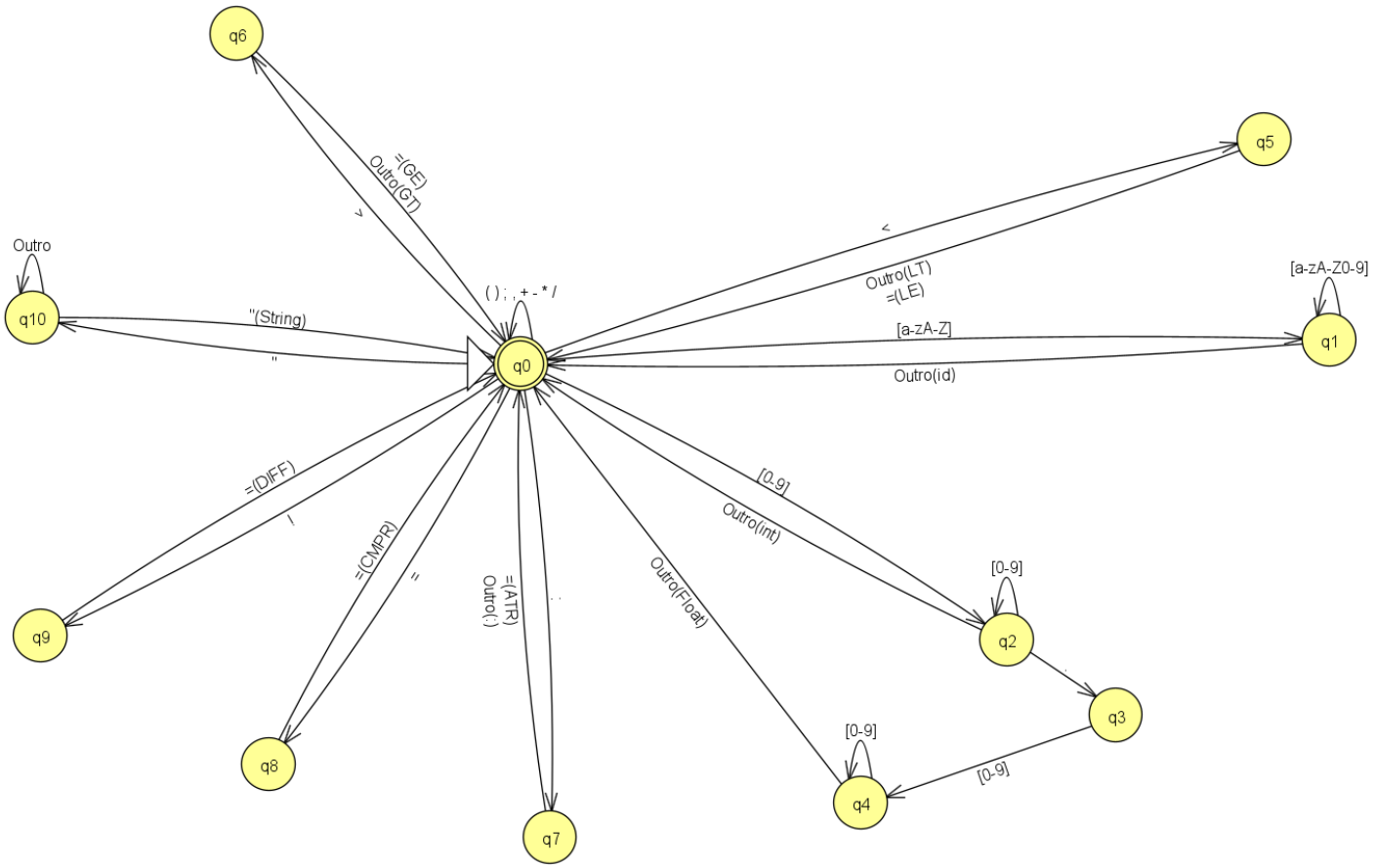
Alunos: Felipe Ferreira Marra e Fabrício Dinali Adão

Lógica do AFD

- Estado 0 é o inicial e final, aceita as entradas que são caracteres únicos.
- Estado 1 aceita IDs, a transição de 0 para ele são os caracteres que podem dar início ao nome de um ID. Seu self loop contém todos os caracteres que podem dar sequência ao nome de um ID.
- Estado 2 aceita inteiros, tanto a transição de 0 para ele quanto seu self loop são números de 0 a 9, qualquer caractere que não seja um número volta para o estado 0 tendo então aceitado um inteiro
- Estado 3 é um estado de transição da parte inteira para a parte decimal de um float, sendo demarcado pelo caractere '.'
- Estado 4 é a parte decimal do float, qualquer caractere que não seja um número volta para o estado 0 tendo então aceitado um float
- Estado 5 é responsável pelo '<' e '<=', a transição de 0 para o mesmo é o caractere '<', caso venham qualquer caractere diferente de '=' o estado é aceito como apenas o símbolo '<', mas caso venha um '=' o estado é aceito como sendo o símbolo '<='
- Estado 6 é responsável pelo '>' e '>=', a transição de 0 para o mesmo é o caractere '>', caso venham qualquer caractere diferente de '=' o estado é aceito como apenas o símbolo '>', mas caso venha um '=' o estado é aceito como sendo o símbolo '>='
- Estado 7 é responsável pelos símbolos ':=' e ':', a transição de 0 para o mesmo é dada pelo símbolo ':', se o símbolo sucessor for um '=' é gerado o símbolo ':=', caso contrário será aceito como sendo o símbolo ':'
- Estado 8 é responsável pelo símbolo '==', a transição de 0 para o mesmo é dada pelo símbolo '=', se o símbolo sucessor for um '=' é gerado o símbolo '==', caso contrário não será aceito
- Estado 9 é responsável pelo símbolo '!=', a transição de 0 para o mesmo é dada pelo símbolo '!', se o símbolo sucessor for um '=' é gerado o símbolo '!=', caso contrário não será aceito
- Estado 10 é responsável pelas strings literais, a transição de 0 para o mesmo é dada pelo símbolo ", esse Estado recebe então qualquer elemento em um self loop, caso chegue o símbolo " volta pro Estado 0.



Lógica da Tabela de Símbolos

A tabela de símbolos é constituída por um dicionário onde a chave é uma string que representa o lexema do ID e o valor é o token correspondente. Quando há um match no processo de análise sintática o token e o lexema são adicionados à Tabela de Símbolos.

Lógica das Ações Semânticas

Para verificar a declaração de variável, temos que analisar os não terminais que derivam ID. Então, nas funções referentes a estes não terminais é chamada uma função que verifica a presença dos IDs na tabela de símbolos, caso não seja encontrado temos o erro semântico de não declaração.

No momento em que há um match com o token, verifica-se, antes de adicioná-lo à tabela de símbolos, sua presença nesta. Então, caso o token já faça parte da tabela de símbolos temos o erro semântico de redeclaração de variável.
A verificação da compatibilidade de tipos é realizada recuperando os tipos dos componentes que integram a expressão. Então, a cada expressão os tipos recuperados são armazenados em um vetor e, ao fim desta, é verificado se há no vetor tipos incompatíveis, caso haja temos então o erro semântico de compatibilidade de tipos.

Dificuldades (o que foi e não foi implementado)

Enfrentamos certa dificuldade na criação das regras semânticas, de modo que posteriormente seriam aplicadas na análise semântica. Foi possível realizar as devidas verificações, utilizando a tabela de símbolos, mas a passagem pelas regras semânticas não foi muito bem definida e aplicada. Foi possível implementar todos os passos do processo. Alguns problemas relacionados à análise sintática foram enfrentados, mas posteriormente foi possível corrigir tais problemáticas.

Testes:

1:

Código

```
program calc
        var x,y z: real;
        var nome, endereco: string;
        begin
                x = 3.15;
                y := 2.4;
                w := 100;
                if x then
                   z := x - y;
                end
                nome := "skdfkm23235FNFOWOF?";
                endereco := "Rua Fulad_2 139, 234 #";
        end
```

Logs:

ERR LEXICO: Expected Char = after = but char was given

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: calc | Token: ID
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: x | Token: ID
Linha 2 | Lexema: , | Token: COMMA
Linha 2 | Lexema: y | Token: ID

Linha 2 | Lexema: z | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: real | Token: real
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: nome | Token: ID
Linha 3 | Lexema: , | Token: COMMA
Linha 3 | Lexema: endereco | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: string | Token: string
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: begin | Token: begin
Linha 5 | Lexema: x | Token: ID
Linha 5 | Lexema: =3.15 | Token: REAL_CONST
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: y | Token: ID
Linha 6 | Lexema: := | Token: ATR
Linha 6 | Lexema: 2.4 | Token: REAL_CONST
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: w | Token: ID
Linha 7 | Lexema: := | Token: ATR
Linha 7 | Lexema: 100 | Token: INTEGER_CONST
Linha 7 | Lexema: ; | Token: PCOMMA
Linha 8 | Lexema: if | Token: if
Linha 8 | Lexema: x | Token: ID
Linha 9 | Lexema: then | Token: then
Linha 9 | Lexema: z | Token: ID
Linha 9 | Lexema: := | Token: ATR
Linha 9 | Lexema: x | Token: ID
Linha 9 | Lexema: - | Token: MINUS
Linha 9 | Lexema: y | Token: ID
Linha 9 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: end | Token: end
Linha 11 | Lexema: nome | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: skdfkm23235FNFOWOF? | Token: STRING_LITERAL
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: endereco | Token: ID
Linha 12 | Lexema: := | Token: ATR
Linha 12 | Lexema: Rua Fulad_2 139, 234 # | Token: STRING_LITERAL
Linha 12 | Lexema: ; | Token: PCOMMA
Linha 13 | Lexema: end | Token: end

ERR SINTATICO: match: token PCOMMA is NOT a match with var

ERR SINTATICO: match: token TWO_POINTS is NOT a match with ID

ERR SINTATICO: match: token PCOMMA is NOT a match with ID

ERR SINTATICO: match: token begin is NOT a match with ID

ERR SEMANTICO: Variavel z nao declarada na linha 2

2:

Código

```
program calc;
	var x,y,z: real;
	var nome, endereco: string;
	begin
		x := 3.15;
		y := 2.4;
		if x then
		   z := (x - y);
		end
		nome := "skdfkm23235FNFOWOF?";
		endereco := "Rua Fulad_2 139, 234 #";
	end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: calc | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: x | Token: ID
Linha 2 | Lexema: , | Token: COMMA
Linha 2 | Lexema: y | Token: ID
Linha 2 | Lexema: , | Token: COMMA
Linha 2 | Lexema: z | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: real | Token: real
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: nome | Token: ID
Linha 3 | Lexema: , | Token: COMMA
Linha 3 | Lexema: endereco | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: string | Token: string
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: begin | Token: begin

Linha 5 | Lexema: x | Token: ID
Linha 5 | Lexema: := | Token: ATR
Linha 5 | Lexema: 3.15 | Token: REAL_CONST
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: y | Token: ID
Linha 6 | Lexema: := | Token: ATR
Linha 6 | Lexema: 2.4 | Token: REAL_CONST
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: if | Token: if
Linha 7 | Lexema: x | Token: ID
Linha 8 | Lexema: then | Token: then
Linha 8 | Lexema: z | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: (| Token: LBRACKET
Linha 8 | Lexema: x | Token: ID
Linha 8 | Lexema: - | Token: MINUS
Linha 8 | Lexema: y | Token: ID
Linha 8 | Lexema:) | Token: RBRACKET
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 10 | Lexema: end | Token: end
Linha 10 | Lexema: nome | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: skdfkm23235FNFOWOF? | Token: STRING_LITERAL
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: endereco | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: Rua Fulad_2 139, 234 # | Token: STRING_LITERAL
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: end | Token: end

TABELA DE SIMBOLOS:
Chave x | Valor: REAL_CONST
Chave y | Valor: REAL_CONST
Chave z | Valor: REAL_CONST
Chave nome | Valor: STRING_LITERAL
Chave endereco | Valor: STRING_LITERAL

3:

Código

```
program calc;
	var y,z: real;
	var x: string;
	var nome, endereco: string;
	begin
		x := 3.15;
		y := 2.4;
		if x then
```

```
            z := (x - y);
        end
        nome := "skdfkm23235FNFOWOF?";
        endereco := "Rua Fulad_2 139, 234 #";
end
```

Logs:

```
SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: calc | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: y | Token: ID
Linha 2 | Lexema: , | Token: COMMA
Linha 2 | Lexema: z | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: real | Token: real
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: x | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: string | Token: string
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: var | Token: var
Linha 4 | Lexema: nome | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: endereco | Token: ID
Linha 4 | Lexema: : | Token: TWO_POINTS
Linha 4 | Lexema: string | Token: string
Linha 4 | Lexema: ; | Token: PCOMMA
Linha 5 | Lexema: begin | Token: begin
Linha 6 | Lexema: x | Token: ID
Linha 6 | Lexema: := | Token: ATR
Linha 6 | Lexema: 3.15 | Token: REAL_CONST
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: y | Token: ID
Linha 7 | Lexema: := | Token: ATR
Linha 7 | Lexema: 2.4 | Token: REAL_CONST
Linha 7 | Lexema: ; | Token: PCOMMA
Linha 8 | Lexema: if | Token: if
Linha 8 | Lexema: x | Token: ID
Linha 9 | Lexema: then | Token: then
Linha 9 | Lexema: z | Token: ID
Linha 9 | Lexema: := | Token: ATR
Linha 9 | Lexema: ( | Token: LBRACKET
Linha 9 | Lexema: x | Token: ID
Linha 9 | Lexema: - | Token: MINUS
Linha 9 | Lexema: y | Token: ID
```

Linha 9 | Lexema:) | Token: RBRACKET
Linha 9 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: end | Token: end
Linha 11 | Lexema: nome | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: skdfkm23235FNFOWOF? | Token: STRING_LITERAL
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: endereco | Token: ID
Linha 12 | Lexema: := | Token: ATR
Linha 12 | Lexema: Rua Fulad_2 139, 234 # | Token: STRING_LITERAL
Linha 12 | Lexema: ; | Token: PCOMMA
Linha 13 | Lexema: end | Token: end

ERR SEMANTICO: Incompatible types string and boolean, integer or real | Variavel) na linha 9

TABELA DE SIMBOLOS:
Chave y | Valor: REAL_CONST
Chave z | Valor: REAL_CONST
Chave x | Valor: STRING_LITERAL
Chave nome | Valor: STRING_LITERAL
Chave endereco | Valor: STRING_LITERAL

4:

Código

```
program prog2;
        var flag: boolean;
        var cont: integer;
        begin
                cont := 0;
                flag := true;
                while flag do
                        cont := cont + 1;
                        if cont > 10
                                then flag := false;
                        end
                end
        end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: prog2 | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: flag | Token: ID

Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: boolean | Token: boolean
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: cont | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: integer | Token: integer
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: begin | Token: begin
Linha 5 | Lexema: cont | Token: ID
Linha 5 | Lexema: := | Token: ATR
Linha 5 | Lexema: 0 | Token: INTEGER_CONST
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: flag | Token: ID
Linha 6 | Lexema: := | Token: ATR
Linha 6 | Lexema: true | Token: true
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: while | Token: while
Linha 7 | Lexema: flag | Token: ID
Linha 7 | Lexema: do | Token: do
Linha 8 | Lexema: cont | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: cont | Token: ID
Linha 8 | Lexema: + | Token: PLUS
Linha 8 | Lexema: 1 | Token: INTEGER_CONST
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 9 | Lexema: if | Token: if
Linha 9 | Lexema: cont | Token: ID
Linha 9 | Lexema: > | Token: GT
Linha 9 | Lexema: 10 | Token: INTEGER_CONST
Linha 10 | Lexema: then | Token: then
Linha 10 | Lexema: flag | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: false | Token: false
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: end | Token: end
Linha 12 | Lexema: end | Token: end
Linha 13 | Lexema: end | Token: end

TABELA DE SIMBOLOS:
Chave flag | Valor: boolean
Chave cont | Valor: INTEGER_CONST

5:

Código

```
program prog2;
        var flag: boolean;
```

```
        var cont, input: integer;
    var x,y,z: real;
    var teste: string;
        begin
                cont := 0;
                flag := true;
       x := 3.7;
       y:= 0.45;
       teste := "Estou aqui";
       print x / y;
       read input;
                while flag do
                        cont := cont + 1;
          if cont <= 5 then
             z := x - y;
             z := z/10 + 4 - 3;
                        if cont > 10
                                then flag := false;
                        end
                end
        end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: prog2 | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: flag | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: boolean | Token: boolean
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: cont | Token: ID
Linha 3 | Lexema: , | Token: COMMA
Linha 3 | Lexema: input | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: integer | Token: integer
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: var | Token: var
Linha 4 | Lexema: x | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: y | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: z | Token: ID
Linha 4 | Lexema: : | Token: TWO_POINTS
Linha 4 | Lexema: real | Token: real

Linha 4 | Lexema: ; | Token: PCOMMA
Linha 5 | Lexema: var | Token: var
Linha 5 | Lexema: teste | Token: ID
Linha 5 | Lexema: : | Token: TWO_POINTS
Linha 5 | Lexema: string | Token: string
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: begin | Token: begin
Linha 7 | Lexema: cont | Token: ID
Linha 7 | Lexema: := | Token: ATR
Linha 7 | Lexema: 0 | Token: INTEGER_CONST
Linha 7 | Lexema: ; | Token: PCOMMA
Linha 8 | Lexema: flag | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: true | Token: true
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 9 | Lexema: x | Token: ID
Linha 9 | Lexema: := | Token: ATR
Linha 9 | Lexema: 3.7 | Token: REAL_CONST
Linha 9 | Lexema: ; | Token: PCOMMA
Linha 10 | Lexema: y | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: 0.45 | Token: REAL_CONST
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: teste | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: Estou aqui | Token: STRING_LITERAL
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: print | Token: print
Linha 12 | Lexema: x | Token: ID
Linha 12 | Lexema: / | Token: DIV
Linha 12 | Lexema: y | Token: ID
Linha 12 | Lexema: ; | Token: PCOMMA
Linha 13 | Lexema: read | Token: read
Linha 13 | Lexema: input | Token: ID
Linha 13 | Lexema: ; | Token: PCOMMA
Linha 14 | Lexema: while | Token: while
Linha 14 | Lexema: flag | Token: ID
Linha 14 | Lexema: do | Token: do
Linha 15 | Lexema: cont | Token: ID
Linha 15 | Lexema: := | Token: ATR
Linha 15 | Lexema: cont | Token: ID
Linha 15 | Lexema: + | Token: PLUS
Linha 15 | Lexema: 1 | Token: INTEGER_CONST
Linha 15 | Lexema: ; | Token: PCOMMA
Linha 16 | Lexema: if | Token: if
Linha 16 | Lexema: cont | Token: ID
Linha 16 | Lexema: <= | Token: LE
Linha 16 | Lexema: 5 | Token: INTEGER_CONST

Linha 17 | Lexema: then | Token: then
Linha 17 | Lexema: z | Token: ID
Linha 17 | Lexema: := | Token: ATR
Linha 17 | Lexema: x | Token: ID
Linha 17 | Lexema: - | Token: MINUS
Linha 17 | Lexema: y | Token: ID
Linha 17 | Lexema: ; | Token: PCOMMA
Linha 18 | Lexema: z | Token: ID
Linha 18 | Lexema: := | Token: ATR
Linha 18 | Lexema: z | Token: ID
Linha 18 | Lexema: / | Token: DIV
Linha 18 | Lexema: 10 | Token: INTEGER_CONST
Linha 18 | Lexema: + | Token: PLUS
Linha 18 | Lexema: 4 | Token: INTEGER_CONST
Linha 18 | Lexema: - | Token: MINUS
Linha 18 | Lexema: 3 | Token: INTEGER_CONST
Linha 18 | Lexema: ; | Token: PCOMMA
Linha 19 | Lexema: if | Token: if
Linha 19 | Lexema: cont | Token: ID
Linha 19 | Lexema: > | Token: GT
Linha 19 | Lexema: 10 | Token: INTEGER_CONST
Linha 20 | Lexema: then | Token: then
Linha 20 | Lexema: flag | Token: ID
Linha 20 | Lexema: := | Token: ATR
Linha 20 | Lexema: false | Token: false
Linha 20 | Lexema: ; | Token: PCOMMA
Linha 21 | Lexema: end | Token: end
Linha 22 | Lexema: end | Token: end
Linha 23 | Lexema: end | Token: end

TABELA DE SIMBOLOS:
Chave flag | Valor: boolean
Chave cont | Valor: INTEGER_CONST
Chave input | Valor: INTEGER_CONST
Chave x | Valor: REAL_CONST
Chave y | Valor: REAL_CONST
Chave z | Valor: REAL_CONST
Chave teste | Valor: STRING_LITERAL

6:

Código

```
program teste4;
        var flag: boolean;
        var cont, input: integer;
   var x,y,z: real;
   var teste: string;
        begin
```

```
            cont := 0;
            flag := true;
    x := 3.7;
    y:= 0.45;
    teste := "Estou aqui";
    print x / y;
    read input;
            while flag do
                    cont := cont + 1;
       if cont == 5 then
          z := x - y;
          z := z/10 + 4 - 3;
                    end
                    if cont != 10
                            then flag := false;
                    end
            end
       end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: teste4 | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: flag | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: boolean | Token: boolean
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: cont | Token: ID
Linha 3 | Lexema: , | Token: COMMA
Linha 3 | Lexema: input | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: integer | Token: integer
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: var | Token: var
Linha 4 | Lexema: x | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: y | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: z | Token: ID
Linha 4 | Lexema: : | Token: TWO_POINTS
Linha 4 | Lexema: real | Token: real
Linha 4 | Lexema: ; | Token: PCOMMA
Linha 5 | Lexema: var | Token: var
Linha 5 | Lexema: teste | Token: ID

Linha 5 | Lexema: : | Token: TWO_POINTS
Linha 5 | Lexema: string | Token: string
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: begin | Token: begin
Linha 7 | Lexema: cont | Token: ID
Linha 7 | Lexema: := | Token: ATR
Linha 7 | Lexema: 0 | Token: INTEGER_CONST
Linha 7 | Lexema: ; | Token: PCOMMA
Linha 8 | Lexema: flag | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: true | Token: true
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 9 | Lexema: x | Token: ID
Linha 9 | Lexema: := | Token: ATR
Linha 9 | Lexema: 3.7 | Token: REAL_CONST
Linha 9 | Lexema: ; | Token: PCOMMA
Linha 10 | Lexema: y | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: 0.45 | Token: REAL_CONST
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: teste | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: Estou aqui | Token: STRING_LITERAL
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: print | Token: print
Linha 12 | Lexema: x | Token: ID
Linha 12 | Lexema: / | Token: DIV
Linha 12 | Lexema: y | Token: ID
Linha 12 | Lexema: ; | Token: PCOMMA
Linha 13 | Lexema: read | Token: read
Linha 13 | Lexema: input | Token: ID
Linha 13 | Lexema: ; | Token: PCOMMA
Linha 14 | Lexema: while | Token: while
Linha 14 | Lexema: flag | Token: ID
Linha 14 | Lexema: do | Token: do
Linha 15 | Lexema: cont | Token: ID
Linha 15 | Lexema: := | Token: ATR
Linha 15 | Lexema: cont | Token: ID
Linha 15 | Lexema: + | Token: PLUS
Linha 15 | Lexema: 1 | Token: INTEGER_CONST
Linha 15 | Lexema: ; | Token: PCOMMA
Linha 16 | Lexema: if | Token: if
Linha 16 | Lexema: cont | Token: ID
Linha 16 | Lexema: == | Token: CMPR
Linha 16 | Lexema: 5 | Token: INTEGER_CONST
Linha 17 | Lexema: then | Token: then
Linha 17 | Lexema: z | Token: ID
Linha 17 | Lexema: := | Token: ATR

Linha 17 | Lexema: x | Token: ID
Linha 17 | Lexema: - | Token: MINUS
Linha 17 | Lexema: y | Token: ID
Linha 17 | Lexema: ; | Token: PCOMMA
Linha 18 | Lexema: z | Token: ID
Linha 18 | Lexema: := | Token: ATR
Linha 18 | Lexema: z | Token: ID
Linha 18 | Lexema: / | Token: DIV
Linha 18 | Lexema: 10 | Token: INTEGER_CONST
Linha 18 | Lexema: + | Token: PLUS
Linha 18 | Lexema: 4 | Token: INTEGER_CONST
Linha 18 | Lexema: - | Token: MINUS
Linha 18 | Lexema: 3 | Token: INTEGER_CONST
Linha 18 | Lexema: ; | Token: PCOMMA
Linha 19 | Lexema: end | Token: end
Linha 20 | Lexema: if | Token: if
Linha 20 | Lexema: cont | Token: ID
Linha 20 | Lexema: != | Token: DIFF
Linha 20 | Lexema: 10 | Token: INTEGER_CONST
Linha 21 | Lexema: then | Token: then
Linha 21 | Lexema: flag | Token: ID
Linha 21 | Lexema: := | Token: ATR
Linha 21 | Lexema: false | Token: false
Linha 21 | Lexema: ; | Token: PCOMMA
Linha 22 | Lexema: end | Token: end
Linha 23 | Lexema: end | Token: end
Linha 24 | Lexema: end | Token: end

TABELA DE SIMBOLOS:
Chave flag | Valor: boolean
Chave cont | Valor: INTEGER_CONST
Chave input | Valor: INTEGER_CONST
Chave x | Valor: REAL_CONST
Chave y | Valor: REAL_CONST
Chave z | Valor: REAL_CONST
Chave teste | Valor: STRING_LITERAL

7:

Código

```
program teste5;
        var flag: boolean;
        var cont, input: integer;
   var x,y,z: real;
   var teste: string;
   var teste: integer;
        begin
                cont := 0;
```

```
            flag := true;
    x := 3.7;
    y:= 0.45;
    teste := "Estou aqui";
    print x / teste;
    read input;
            while flag do
                    cont := cont + 1;
       if cont == flag then
          z := x - teste;
          z := z/10 + 4 - 3;
                    end
                    if cont != 10
                            then flag := false;
                    end
            end
       end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: teste5 | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: flag | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: boolean | Token: boolean
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: cont | Token: ID
Linha 3 | Lexema: , | Token: COMMA
Linha 3 | Lexema: input | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: integer | Token: integer
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: var | Token: var
Linha 4 | Lexema: x | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: y | Token: ID
Linha 4 | Lexema: , | Token: COMMA
Linha 4 | Lexema: z | Token: ID
Linha 4 | Lexema: : | Token: TWO_POINTS
Linha 4 | Lexema: real | Token: real
Linha 4 | Lexema: ; | Token: PCOMMA
Linha 5 | Lexema: var | Token: var
Linha 5 | Lexema: teste | Token: ID
Linha 5 | Lexema: : | Token: TWO_POINTS

Linha 5 | Lexema: string | Token: string
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: var | Token: var
Linha 6 | Lexema: teste | Token: ID
Linha 6 | Lexema: : | Token: TWO_POINTS
Linha 6 | Lexema: integer | Token: integer
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: begin | Token: begin
Linha 8 | Lexema: cont | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: 0 | Token: INTEGER_CONST
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 9 | Lexema: flag | Token: ID
Linha 9 | Lexema: := | Token: ATR
Linha 9 | Lexema: true | Token: true
Linha 9 | Lexema: ; | Token: PCOMMA
Linha 10 | Lexema: x | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: 3.7 | Token: REAL_CONST
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: y | Token: ID
Linha 11 | Lexema: := | Token: ATR
Linha 11 | Lexema: 0.45 | Token: REAL_CONST
Linha 11 | Lexema: ; | Token: PCOMMA
Linha 12 | Lexema: teste | Token: ID
Linha 12 | Lexema: := | Token: ATR
Linha 12 | Lexema: Estou aqui | Token: STRING_LITERAL
Linha 12 | Lexema: ; | Token: PCOMMA
Linha 13 | Lexema: print | Token: print
Linha 13 | Lexema: x | Token: ID
Linha 13 | Lexema: / | Token: DIV
Linha 13 | Lexema: teste | Token: ID
Linha 13 | Lexema: ; | Token: PCOMMA
Linha 14 | Lexema: read | Token: read
Linha 14 | Lexema: input | Token: ID
Linha 14 | Lexema: ; | Token: PCOMMA
Linha 15 | Lexema: while | Token: while
Linha 15 | Lexema: flag | Token: ID
Linha 15 | Lexema: do | Token: do
Linha 16 | Lexema: cont | Token: ID
Linha 16 | Lexema: := | Token: ATR
Linha 16 | Lexema: cont | Token: ID
Linha 16 | Lexema: + | Token: PLUS
Linha 16 | Lexema: 1 | Token: INTEGER_CONST
Linha 16 | Lexema: ; | Token: PCOMMA
Linha 17 | Lexema: if | Token: if
Linha 17 | Lexema: cont | Token: ID
Linha 17 | Lexema: == | Token: CMPR

Linha 17 | Lexema: flag | Token: ID
Linha 18 | Lexema: then | Token: then
Linha 18 | Lexema: z | Token: ID
Linha 18 | Lexema: := | Token: ATR
Linha 18 | Lexema: x | Token: ID
Linha 18 | Lexema: - | Token: MINUS
Linha 18 | Lexema: teste | Token: ID
Linha 18 | Lexema: ; | Token: PCOMMA
Linha 19 | Lexema: z | Token: ID
Linha 19 | Lexema: := | Token: ATR
Linha 19 | Lexema: z | Token: ID
Linha 19 | Lexema: / | Token: DIV
Linha 19 | Lexema: 10 | Token: INTEGER_CONST
Linha 19 | Lexema: + | Token: PLUS
Linha 19 | Lexema: 4 | Token: INTEGER_CONST
Linha 19 | Lexema: - | Token: MINUS
Linha 19 | Lexema: 3 | Token: INTEGER_CONST
Linha 19 | Lexema: ; | Token: PCOMMA
Linha 20 | Lexema: end | Token: end
Linha 21 | Lexema: if | Token: if
Linha 21 | Lexema: cont | Token: ID
Linha 21 | Lexema: != | Token: DIFF
Linha 21 | Lexema: 10 | Token: INTEGER_CONST
Linha 22 | Lexema: then | Token: then
Linha 22 | Lexema: flag | Token: ID
Linha 22 | Lexema: := | Token: ATR
Linha 22 | Lexema: false | Token: false
Linha 22 | Lexema: ; | Token: PCOMMA
Linha 23 | Lexema: end | Token: end
Linha 24 | Lexema: end | Token: end
Linha 25 | Lexema: end | Token: end

ERR SEMANTICO: Variavel teste redeclarada na linha 6

8:

Código

```
program prog2;
	var flag: boolean;
	var cont: string;
	begin
		cont := 0;
		flag := true;
		while flag do
			cont := cont + 1;
			if cont > 10
				then flag := false;
			end
```

```
        end
    end
```

Logs:

SAIDA LEXICO:
Linha 1 | Lexema: program | Token: program
Linha 1 | Lexema: prog2 | Token: ID
Linha 1 | Lexema: ; | Token: PCOMMA
Linha 2 | Lexema: var | Token: var
Linha 2 | Lexema: flag | Token: ID
Linha 2 | Lexema: : | Token: TWO_POINTS
Linha 2 | Lexema: boolean | Token: boolean
Linha 2 | Lexema: ; | Token: PCOMMA
Linha 3 | Lexema: var | Token: var
Linha 3 | Lexema: cont | Token: ID
Linha 3 | Lexema: : | Token: TWO_POINTS
Linha 3 | Lexema: string | Token: string
Linha 3 | Lexema: ; | Token: PCOMMA
Linha 4 | Lexema: begin | Token: begin
Linha 5 | Lexema: cont | Token: ID
Linha 5 | Lexema: := | Token: ATR
Linha 5 | Lexema: 0 | Token: INTEGER_CONST
Linha 5 | Lexema: ; | Token: PCOMMA
Linha 6 | Lexema: flag | Token: ID
Linha 6 | Lexema: := | Token: ATR
Linha 6 | Lexema: true | Token: true
Linha 6 | Lexema: ; | Token: PCOMMA
Linha 7 | Lexema: while | Token: while
Linha 7 | Lexema: flag | Token: ID
Linha 7 | Lexema: do | Token: do
Linha 8 | Lexema: cont | Token: ID
Linha 8 | Lexema: := | Token: ATR
Linha 8 | Lexema: cont | Token: ID
Linha 8 | Lexema: + | Token: PLUS
Linha 8 | Lexema: 1 | Token: INTEGER_CONST
Linha 8 | Lexema: ; | Token: PCOMMA
Linha 9 | Lexema: if | Token: if
Linha 9 | Lexema: cont | Token: ID
Linha 9 | Lexema: > | Token: GT
Linha 9 | Lexema: 10 | Token: INTEGER_CONST
Linha 10 | Lexema: then | Token: then
Linha 10 | Lexema: flag | Token: ID
Linha 10 | Lexema: := | Token: ATR
Linha 10 | Lexema: false | Token: false
Linha 10 | Lexema: ; | Token: PCOMMA
Linha 11 | Lexema: end | Token: end
Linha 12 | Lexema: end | Token: end

Linha 13 | Lexema: end | Token: end

ERR SEMANTICO: Incompatible types string and boolean, integer or real | Variavel ; na linha 8

ERR SEMANTICO: Incompatible types string and boolean, integer or real | Variavel then na linha 10

TABELA DE SIMBOLOS:
Chave flag | Valor: boolean
Chave cont | Valor: STRING_LITERAL