

TOC Compiler

Especificação da Linguagem TOC

Lucas Machado da Palma

Lucas May Petry

Luiz Henrique Urias de Sousa

A. Sumário

Sumário	2
Introdução	3
Especificação da linguagem	4
1. Identificadores de funções, variáveis e objetos	4
2. Função principal do programa	5
3. Comentários	6
4. Tipos/declarações/atribuições de variáveis	7
5. Arranjos	9
6. Conversão entre tipos	11
7. Operadores matemáticos	13
8. Operadores relacionais e lógicos	14
9. Múltiplos escopos	16
10. Condicionais	18
11. Laços	20
12. Funções	22
13. Função print	25
14. Orientação a objetos	26
15. Tratamento de strings	30
16. Nota do código	31

B. Introdução

Muitos profissionais de tecnologia já se depararam com a situação pouco confortável de trabalhar com códigos de terceiros, ou mesmo com seus próprios códigos em uma formatação deplorável. Más escolhas de indentação, códigos excessivamente longos, ou até mesmo a sintaxe pouco amigável de algumas linguagens podem colaborar para uma baixa produtividade e o desânimo do desenvolvedor.

A linguagem de programação orientada a objetos TOC tem como objetivo estabelecer um padrão simples e compreensivo de codificação. Os pilares da linguagem são:

1. Simplicidade (simpliciTy): suas palavras reservadas são claras e enxutas colaborando, para a compreensão de seus códigos;
2. Organização (Organization): padrões de indentação, nome de funções e classes colaboram para um resultado visualmente elegante;
3. Rapidez (quiCkness): a linguagem permite uma maior produtividade, proporcionada pela simplicidade e organização do código.

Alavancados por esses princípios, todos os códigos desenvolvidos em TOC podem ser facilmente compreendidos por qualquer desenvolvedor interessado. Consequentemente, a colaboração entre desenvolvedores é facilitada.

C. Especificação da linguagem

1. Identificadores de funções, variáveis e objetos

Os identificadores na linguagem TOC são universais para funções, variáveis e objetos. Por exemplo, dentro de um mesmo escopo uma função não pode ter o mesmo nome de uma variável ou objeto.

Os identificadores podem iniciar com um carácter de a até z (minúsculo ou maiúsculo), seguido de outros caracteres, números ou o caractere especial “_”.

Exemplo de código:

```
flt uva = 0

# Método de exemplo
int uva()
    uva = uva + 1
```

[Line 3] Semantic error: Identifier uva is already in use.

2. Função principal do programa

Todo programa em TOC deve conter a função principal *toc*, a qual é o ponto de partida da execução.

Exemplo de código:

```
# Função principal do programa
void toc()
    # Código do programa
```

Tratamento de erros:

- a. Caso a função principal não seja declarada, um erro semântico é emitido.

Exemplo:

```
int abacaxi = 3
```

[Line 1] Semantic error: Main function toc() not found.

3. Comentários

Sobre os dois primeiros pilares do projeto, a linguagem TOC oferece um sistema de comentários simples e organizado. Todas as anotações são feitas com o uso do caractere especial **#** (cerquilha ou sustenido), que estabelecem sintaticamente o início de uma linha de comentário. O comentário é a única estrutura de uma linha.

Exemplo de código:

```
# Apenas um exemplo de comentário ...  
# Continuando o exemplo de comentário ...
```

Representação intermediária:

```
# Apenas um exemplo de comentário ...  
# Continuando o exemplo de comentário ...
```

Todo o comentário deve estar contido em uma única linha de código. Se for necessário mais de uma linha - para extensão do texto ou para facilitar a compreensão - uma nova **#** deve ser adicionada.

Tratamento de erros:

- b. No caso de uma inserção de texto fora do escopo de um comentário, o compilador deve emitir uma mensagem de erro sintático apontando o desconhecimento do caractere ou palavra inicial.

Exemplo:

```
# Apenas um exemplo de comentário ...  
Continuando exemplo
```

[Line 2] Syntax error: Unknown error.

[Line 2] Syntax error: Unknown error.

- c. Para os comentários não iniciados com letra maiúscula o compilador deve emitir uma mensagem de aviso, explicitando a recomendação do uso do padrão de comentários da linguagem.

Exemplo:

```
# apenas um exemplo de comentário ...
```

[Line 1] Warning: TOC recommends that comments initiate with an uppercase character.

4. Tipos/declarações/atribuições de variáveis

Todos os tipos básicos de dados estão disponíveis para uso na linguagem. As palavras reservadas estão limitadas a três letras e são as que seguem:

- `int` : números inteiros.
- `flt` : números de ponto flutuante.
- `boo` : variáveis do tipo booleano, adotam os valores `true` e `false`.
- `str` : textos.

As declarações de variáveis são simples e seguem os padrões de identificador descritos na seção 1. É possível declarar várias variáveis em uma mesma linha, desde que todas elas tenham o mesmo tipo. Para essa forma de declaração há um padrão estabelecido pela linguagem:

id_variável vírgula espaço

Não é possível declarar e inicializar várias variáveis na mesma linha. Se algum padrão de nomenclatura ou espaçamento não for obedecido, mensagens de *warning* deverão ser emitidas.

Exemplo de código:

```
int abacate, banana, caqui
abacate = 0
boo figo, goiaba
flt kiwi = 2
```

Representação intermediária:

```
int abacate
int banana
int caqui
abacate = 0
boo figo
boo goiaba
flt kiwi = 2
```

Tratamento de erros:

- a. No caso de erros léxicos (símbolos e palavras desconhecidas), o compilador deve apenas emitir uma mensagem de erro informando sobre o caractere ou palavra desconhecida, não passando nenhum token para a análise sintática.

Exemplo:

```
int kiwi$$
```

[Line 1] Lexical error: Unknown symbol \$\$

- b. No caso de uso de uma variável ainda não declarada, uma mensagem de erro semântico deve ser impressa.

Exemplo:

```
caqui = 2
```

[Line 1] Semantic error: Undeclared variable caqui

- c. No caso de uma variável ser utilizada mas não inicializada, uma mensagem de erro deve ser emitida.

Exemplo:

```
int abacaxi  
int banana  
banana = abacaxi
```

[Line 3] Semantic error: Variable abacaxi used but not initialized

- d. No caso de variáveis iniciadas com caractere maiúsculo, uma mensagem de *warning* deve ser emitida.

Exemplo:

```
int Abacaxi
```

[Line 1] *Warning*: Variable Abacaxi initiates with an uppercase character. TOC recommends that variable's name initiates with a lowercase character.

- e. No caso de não seguimento dos padrões de espaçamento para atribuição e/ou declaração, uma mensagem de *warning* deve ser emitida.

Exemplo:

```
int b=3  
int a,b
```

[Line 1] *Warning*: TOC recommends that you leave exactly one space in between operands and/or operators, and after commas.

[Line 2] *Warning*: TOC recommends that you leave exactly one space in between operands and/or operators, and after commas.

5. Arranjos

Os nomes de arranjos seguem os padrões de identificador descritos na seção 1. Arranjos simples são tipados com um dos quatro tipos oferecidos pela linguagem TOC. É permitido a declaração de vários arranjos na mesma linha, mas a inicialização deve ser feita de forma individual. Além disso, todos os itens declarados em uma linha devem ser arranjos de um mesmo tipo.

Exemplo de código:

```
# Exemplo de criação de arrays em TOC
str saladas_de_frutas[5]
flt sucos[5], vitaminas[10]
str frutas[2] = {"melancia", "mamão"}
```

Representação intermediária:

```
# Exemplo de criação de arrays em TOC
str saladas_de_frutas[5]
flt sucos[5]
flt vitaminas[10]
str frutas[2] = {"melancia", "mamão"}
```

Tratamento de erros:

- a. No caso de um índice do arranjo ser usado mas não inicializado, o compilador deverá emitir uma mensagem de erro semântico.

Exemplo:

```
int frutas[5]
int ameixa = frutas[4]
```

[Line 2] Semantic error: Element at index 4 used but not initialized.

- b. Quando o número de valores na inicialização de um arranjo difere do tamanho do arranjo, um erro semântico deve ser emitido.

Exemplo:

```
str frutas[3] = {"cereja", "jaca", "limão", "pitanga"}
```

[Line 1] Semantic error: Number of elements in assignment differs from array length.

- c. Quando o usuário tenta atribuir um único valor na declaração de um arranjo de n ($n > 1$) posições, um erro sintático deverá ser emitido.

Exemplo:

```
flt sucos[3] = 1.5
```

[Line 1] Syntax error: Unknown error.

6. Conversão entre tipos

Diferentemente de linguagens conhecidas como C++ ou Java, a linguagem TOC possui uma conversão implícita entre tipos pré-definidos. A conversão entre os tipos segue a tabela abaixo:

de\para	int	flt	boo	str
int	-	int é um flt	!= 0 true = 0 false	Inteiro como texto
flt	Recebe apenas a parte inteira do valor	-	!= 0 true = 0 false	Ponto flutuante como texto
boo	true = 1 false = 0	true = 1 false = 0	-	Valor booleano como texto
str	Equivalente à: str -> flt flt -> int	Conversão válida se str é um flt válido	Equivalente à: str -> flt flt -> int int -> boo Ou "True" = true "False" = false (case insensitive)	-

Obs: O símbolo "-" significa que não é necessário nenhum tipo de conversão.

Exemplo de código:

```
# Exemplo de conversão entre tipos
str laranja = "5"
flt suco = laranja
boo bom = suco
```

Representação intermediária:

```
# Exemplo de conversão entre tipos
str laranja = "5"
flt suco = [flt] (laranja)
boo bom = [boo] (suco)
```

Tratamento de erros:

- a. Para inconsistências na atribuição de variáveis do tipo strings, onde o valor da string é um texto não tratável pelo compilador, um erro semântico deverá ser emitido.

Exemplo:

```
str groselha = "groselha"  
flt suco = groselha
```

[Line 2] Semantic error: Value of groselha is not a number.

7. Operadores matemáticos

Os operadores matemáticos suportados pela linguagem TOC são: adição, subtração, divisão, multiplicação e o operador unário de inversa (-).

As precedências das operações seguem as regras matemáticas usuais, incluindo parênteses.

Precedência	Operação	Símbolo
1	Parênteses	()
2	Multiplicação, Divisão	*, /
3	Adição, Subtração	+, -

Exemplo de código:

```
# Exemplo de operadores matemáticos
int a = 5
a = a+5*3
```

Representação intermediária:

```
# Exemplo de operadores matemáticos
int a = 5
a = a + 5 * 3
```

8. Operadores relacionais e lógicos

A linguagem TOC possui suporte a operadores relacionais, os quais são utilizados sobre valores inteiros ou de ponto flutuante (booleanos e *strings* são convertidos implicitamente):

- > : maior
- >= : maior ou igual
- < : menor
- <= : menor ou igual
- == : igual
- != : diferente

Ainda, possui operadores lógicos que são usados sobre expressões que resultam em valores booleanos:

Precedência	Operação	Símbolo
1	Negação	not
2	E lógico, Ou lógico	and, or

Para a avaliação de operações relacionais de operandos de tipos distintos, ocorre uma conversão implícita para o tipo mais forte dentre os operandos. O tipo mais forte é definido pela seguinte ordem:

- flt : é o tipo mais abrangente
- int : todo inteiro é também um número de ponto flutuante
- boo : um booleano é também um inteiro (true = 1, false = 0)
- str : tentativa de conversão para o tipo mais forte. Caso não seja possível, ela é tratada como um erro de conversão de tipos.

Exemplo de código:

```
# Exemplo de operadores relacionais e lógicos
int abacaxi = 5
flt banana = 5.7
boo carambola = abacaxi > banana
boo sabor = carambola > abacaxi
boo damasco = not carambola
boo figo = carambola and damasco
```

Representação intermediária:

```
# Exemplo de operadores relacionais e lógicos
int abacaxi = 5
flt banana = 5.7
boo carambola = [flt] (abacaxi > anana)
```

```
boo sabor = [int] (carambola > abacaxi)
boo damasco = not carambola
boo figo = carambola and damasco
```

9. Múltiplos escopos

Múltiplos escopos são uma característica importante da linguagem. Escopos são diferenciados pela indentação de cada linha. Condicionais, laços, funções e objetos possuem seus próprios escopos. A relação entre diferentes escopos está sujeita às seguintes regras:

- O escopo global (externo à função principal) é acessível em todos os outros escopos do programa.
- No escopo global só é possível declarar funções e objetos.
- Um escopo mais interno têm acesso aos objetos, variáveis e funções de escopos mais externos.
- O escopo de objetos pode ser acessado por escopos externos à ele, conforme especificado na seção 13.

Exemplo de código:

```
# Função principal
void toc()
    # Novo escopo, pertencente ao escopo global
    print helloWorld()

# Função Olá Mundo
str helloWorld()
    # Novo escopo, pertencente ao escopo global
    ret "Hello World"
```

Representação intermediária:

```
# Função principal
void toc()
    # Novo escopo, pertencente ao escopo global
    print helloWorld()

# Função Olá Mundo
str helloWorld()
    # Novo escopo, pertencente ao escopo global
    ret "Hello World"
```

Tratamento de erros:

- a. Tentativa de acesso a objeto, variável ou função de escopo interno gera erro semântico.

Exemplo:


```
# Método que mistura por um determinado tempo
void misturar(int tempo)
    flt mistura = 0
    mistura = tempo * 3

flt copo = mistura
```

[Line 5] Semantic error: Undeclared variable mistura.

- b. Ao indentar uma linha com mais espaços do que o escopo atual possui, um erro semântico é gerado.

Exemplo:

```
# Faz litros de suco de laranja
flt fazerSuco(flt litros)
    str sabor = "laranja"
    flt gelo = litros/2
        flt mistura = 0
```

[Line 5] Semantic error: Unexpected scope.

- c. Ao indentar as linhas de um escopo em mais de 2 espaços, um erro sintático é gerado.

Exemplo:

```
# Faz litros de suco de laranja
flt fazerSuco(flt litros)
    str sabor = "laranja"
        flt gelo = litros/2
```

[Line 3] Syntax error: Indentation must be 2 spaces only.

[Line 4] Syntax error: Indentation must be 2 spaces only.

- d. Ao criar uma estrutura que requer um novo escopo, o compilador gera um erro semântico caso o usuário esqueça de indentar o código corretamente.

Exemplo:

```
# Faz litros de suco de laranja
flt fazerSuco(flt litros)
ret 6.5
```

[Line 3] Semantic error: Expected new scope.

10. Condicionais

As estruturas condicionais da linguagem TOC incluem o operador “*if*” e o conjunto “*if else*”. O teste da expressão condicional deve ser um valor booleano ou variável. A indentação é usada para definir o início de um bloco de *if* ou *else* (escopo). Não pode haver nada além do *if* ou do *else* nas linhas em que são declarados.

Exemplo de código:

```
# Exemplo de condicionais
int amora = 2
if(amora == 3)
    amora = 3
else
    amora = -5
```

Representação intermediária:

```
# Exemplo de condicionais
int amora = 2
if(amora == 3)
    amora = 3
else
    amora = -5
```

Tratamento de erros:

- Em todas as situações em que não houver nenhuma condição, regra ou comparação a ser feita em um “*if*”, ou seja, o condicional estiver vazio, o compilador deverá emitir um erro sintático.

Exemplo:

```
int amora = 4
if()
    amora = 3
```

[Line 2] Syntax error: Unknown error.

- Quando identificado a falta do uso dos parênteses entre o início e fim da condição, o compilador deverá emitir um erro sintático apontando a necessidade da adição dos mesmos.

Exemplo:

```
int amora = 4
int jabuticaba = 2
if amora > jabuticaba
    amora = amora - 1
```

[Line 3] Syntax error: Unknown error.

- c. Um erro sintático deve ser emitido se as operações realizadas no condicional não resultarem em um valor booleano (mesmo com a conversão implícita no caso de *strings*).

Exemplo:

```
int amora = 4
int jabuticaba = 2
if(amora = jabuticaba)
    banana = 10
```

[Line 3] Syntax error: Unknown error.

11. Laços

Laços são implementados seguindo a estrutura do exemplo abaixo. O escopo de um laço é definido pela indentação. Além do laço comum (*declaração;teste;iteração*) a linguagem também aceita laços sobre vetores. Não pode haver nada além da declaração do *for* na linha em que ele é declarado.

Exemplo de código:

```
# Exemplo de laços comum
for(int amora = 0; amora < 10 ; amora = amora + 1)
    print amora

# Exemplo de laço para/cada
int vector[3] = {1, 2, 3}
for(a in vector)
    print a
```

Representação intermediária:

```
# Exemplo de laços comum
for(int amora = 0; amora < 10 ; amora = amora + 1)
    print amora

# Exemplo de laço para/cada
int vector[3] = {1, 2, 3}
for(a in vector[3])
    print a
```

Tratamento de erros:

- Em todas as situações em que não houver uma das três operações do cabeçalho do laço o compilador deverá emitir um erro sintático.

Exemplo:

```
int amora = 4
for(;;)
    amora = 3
```

[Line 2] Syntax error: Unknown error.

- Quando identificado a falta do uso dos parênteses entre o início e fim da do laço, o compilador deverá emitir um erro sintático apontando a necessidade da adição dos mesmos.

Exemplo:

```
for int amora = 0; amora < 10 ; amora = amora + 1  
    print amora
```

[Line 1] Syntax error: Unknown error.

- c. Um erro sintático deve ser emitido se as operações realizadas na condição de execução do laço não resultarem em um valor booleano (mesmo com a conversão implícita no caso de *strings*).

Exemplo:

```
for(int amora = 0; amora = 4 ; amora = amora + 1)  
    print amora
```

[Line 1] Syntax error: Unknown error.

12. Funções

No âmbito da programação o uso de funções se tornou com o passar dos anos um hábito irrevogável, com elas podemos criar procedimentos específicos e claros através de parâmetros, execução de um conjunto de passos e possivelmente um retorno. Em TOC as funções podem tomar um dos quatro tipos básico, obrigando um retorno do mesmo tipo ou podem adotar tipo nenhum, que é o caso do “void”, onde não há nenhum retorno.

As declarações de funções seguem os padrões de identificador descritos na seção 1. Assim como os laços e condicionais, a indentação é o que define o escopo de uma função. Recomenda-se que os nomes de funções iniciem com a letra minúscula e que separem as palavras posteriores com maiúsculas. O retorno deve ser sempre a última instrução no escopo.

Exemplo de código:

```
# Exemplo de funções
str tempo = "chuva"

# Define se uma banana está madura
boo bananaEstaMadura()
    boo madura = false

    if(tempo == "sol")
        madura = true

    ret madura

int bananas = 0
if(bananaEstaMadura())
    bananas = 10
```

Representação intermediária:

```
# Exemplo de funções
str tempo = "chuva"

# Define se uma banana está madura
boo bananaEstaMadura()
    boo madura = false

    if(tempo == "sol")
        madura = true

    ret madura
```

```
int bananas = 0
if(bananaEstaMadura())
    bananas = 10
```

Tratamento de erros:

- a. Quando uma função for chamada mas não for declarada, o compilador deverá emitir uma mensagem de erro semântico.

Exemplo:

```
flt sucoDeLaranja = espremerLaranjas(10)
```

[Line 1] Semantic error: Function espremerLaranjas(int) used but not declared.

- b. Na falta ou excesso de parâmetros na chamada de uma função o compilador deverá emitir uma mensagem de erro semântico.

Exemplo:

```
# Litros de suco, dada uma quantidade de laranjas
flt espremerLaranjas(int laranjas)
    flt litros = laranjas*0.050
    ret litros

flt suco = espremerLaranjas()
```

[Line 5] Semantic error: Function espremerLaranjas(int) expected 1 parameter but received 0.

- c. Em toda função não “void” em que não houver especificação de retorno através da palavra reservada “ret” o compilador deverá emitir uma mensagem de erro semântico.

Exemplo:

```
# Espreme laranjas
flt espremerLaranjas(int laranjas)
    flt litros = laranjas*0.050
```

[Line 2] Semantic error: Return value for function espremerLaranjas(int) is missing.

- d. Para funções que não são precedidas de comentário o compilador deverá emitir uma mensagem de alerta sobre o uso do padrão da linguagem.

Exemplo:

```
flt espremerLaranjas(int laranjas)
    flt litros = laranjas*0.050
    ret litros
```

[Line 1] *Warning*: Function espremerLaranjas(int) is not explained. Please do it right above the function declaration with a comment.

13. Função print

A linguagem TOC implementa uma função de impressão na tela. É uma função global da linguagem que não faz o uso de parênteses.

Exemplo de código:

```
# Exemplo do uso de print
void minhasLaranjas()
    int laranjas = 3
    print "Eu tenho "
    print laranjas
    print " laranjas.\n"

# Função principal
void toc()
    minhasLaranjas()
```

14. Orientação a objetos

Na linguagem TOC o paradigma de orientação a objetos está representado de forma simplificada. Podemos declarar classes com atributos e métodos em dois diferentes encapsulamentos: privado e público. Além disso, os objetos podem ser instanciados ou acessados em diferentes escopos, podendo assumir inclusive o papel de atributo de outros objetos.

Os nomes de classes, atributos e métodos seguem os padrões de identificador descritos na seção 1.

Exemplo de código:

```
# Exemplo de objetos em TOC
obj Arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

# Essa função calcula o número de frutas colhidas
pub int colher()
  for(int a = galhos; a >= 0; a = a-1)
    frutas = frutas + a * 6
  ret frutas
```

Representação intermediária:

```
# Exemplo de objetos em TOC
obj Arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

# Esse método calcula o número de frutas removidas
pub int colher()
  for(int a = galhos; a >= 0; a = a-1)
    frutas = frutas + a * 6
  ret frutas
```

Tratamento de erros:

- Na hora de utilizar atributos que não foram inicializados na classe - ou não recebem um valor pela inicialização por parâmetro da classe (construtor) - uma mensagem de erro semântico é gerada.

Exemplo:

```
# Exemplo de objetos não inicializado em TOC
obj Arvore()
  pub int galhos

void toc()
  Arvore a()
  int b = a.galhos
```

[Line 3] Semantic error : Attribute galhos was not initialized.

- b. Caso o usuário tente acessar métodos ou atributos privados externamente à classe o compilador deve emitir uma mensagem de erro semântico.

Exemplo:

```
# Exemplo de objetos em TOC
obj Arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

  # Esse método calcula o número de frutas colhidas
  pub int colher()
    for(int a = galhos; a >= 0; a = a-1)
      frutas = frutas + a * 6
    ret frutas

Arvore jabuticaba(3)
jabuticaba.galhos
```

[Line 12] Semantic error: Attribute jabuticaba is a private attribute.

- c. Se um objeto de uma classe é instanciado sem o número exato de parâmetros especificados pelo construtor da classe, um erro semântico é emitido.

Exemplo:

```
# Exemplo de objetos em TOC
obj Arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

  # Esse método calcula o número de frutas colhidas
  pub int colher()
    for(int a = galhos; a >= 0; a = a-1)
      frutas = frutas + a * 6
    ret frutas
```

```
Arvore jabuticaba()  
jabuticaba.colher()
```

[Line 11] Semantic error: Constructor of class Arvore expected 1 parameter but received 0.

- d. Atributos declarados sem um encapsulamento (público ou privado) geram mensagens de erro semântico.

Exemplo:

```
# Exemplo de objetos em TOC  
obj Arvore(int galhos)  
    int galhos = galhos  
    pub int frutas = 0
```

[Line 2] Semantic error: Declaration of variable galhos expected encapsulation.

- e. No caso de uma chamada de método inexistente naquele objeto, um erro semântico é emitido pelo compilador.

Exemplo:

```
# Exemplo de objetos em TOC  
obj Arvore(int galhos)  
    prv int galhos = galhos  
    pub int frutas = 0  
  
Arvore pinheiro(3)  
pinheiro.colher()
```

[Line 7] Semantic error: Object pinheiro has no method called colher().

- f. No caso de atributos e/ou métodos iniciados com caractere maiúsculo, uma mensagem de *warning* deve ser emitida.

Exemplo:

```
# Exemplo de objetos em TOC  
obj Arvore(int galhos)  
    prv int Galhos = galhos  
    pub int frutas = 0  
  
# Esse método calcula o número de frutas colhidas  
pub int Colher()  
    for(int a = galhos; a >= 0; a = a-1)  
        my.frutas = my.frutas + a * random(int)
```

```
ret my.frutas
```

[Line 3] *Warning*: TOC recommends that variable, function and object's names initiate with a lowercase character.

[Line 7] *Warning*: TOC recommends that variable, function and object's names initiate with a lowercase character.

- g. No caso em que a declaração do identificador da classe (nome da classe) for iniciado com uma letra minúscula, uma mensagem de *warning* deve ser emitida.

Exemplo:

```
# Exemplo de objetos em TOC
obj arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

Arvore jabuticaba(3)
jabuticaba.colher()
```

[Line 2] *Warning*: Object arvore initiates with a lowercase character. TOC recommends that class' name initiates with an uppercase character.

- h. Para métodos que não são precedidos de comentário o compilador deverá emitir uma mensagem de alerta sobre o uso do padrão da linguagem.

Exemplo:

```
# Exemplo de objetos em TOC
obj Arvore(int galhos)
  prv int galhos = galhos
  pub int frutas = 0

  pub int colher()
    for(int a = galhos; a >= 0; a = a-1)
      my.frutas = my.frutas + a * random(int)
    ret my.frutas
```

[Line 6] *Warning*: Method colher() is not explained. Please do it right above the function declaration with a comment.

15. Tratamento de strings

Além da declaração e atribuição de variáveis do tipo *string* a linguagem TOC também oferece suporte a concatenação de *strings*.

Exemplo de código:

```
# Exemplo de operações com strings
str uva = "uva"
str passa = "passa"
str arroz = "arroz"
print arroz + uva + passa

str arrozBranco = arroz + " branco"
```

Representação intermediária:

```
# Exemplo de operações com strings
str uva = "uva"
str passa = "passa"
str arroz = "arroz"
print arroz + uva + passa

str arrozBranco = arroz + " branco"
```

16. Nota do código

Ao final do processo de compilação, o compilador atribui uma nota ao código fonte com base nos erros e *warnings* de compilação. Além disso, o compilador gera um gráfico baseado na quantidade de erros e/ou *warnings*. O resultado final segue as seguintes regras:

Resultado do processo de compilação	Nota	Cálculo da nota
Não compilado	[0, 6)	Percentual dos erros léxicos, sintáticos e semânticos em relação ao tamanho do código fonte.
Compilado	[6, 10]	Percentual de <i>warnings</i> em relação ao tamanho do código fonte.

Obs: No cálculo do percentual, os erros são ponderados de acordo com o seu tipo.

Tipo de erro	Peso
Erro léxico	2
Erro sintático	5
Erro semântico	8
<i>Warning</i>	1