Título: \**Insert project name\**

Cadeira: Compiladores

Autores: André Gual – nº 99999 🡨 Alterem

Carina Neves – nº 99999 🡨 Alterem

Diogo Silveira – nº 99999 🡨 Alterem

João Ferreira – nº 99999 🡨 Alterem

João Miguel Abrantes Dias – nº 89236

Data: 13/06/2020

**Index**

[Nota Introdutória: 2](#_Toc486947527)

[1. Linguagem Secundária](#_Builder_Pattern) 3

[1.1 introdução da ramática](#_Introduction_1) 3

[1.2 análise semântica](#_Problem_1) 3

[2. Linguagem Principal 7](#_Observer_Pattern)

[2.1 introdução da gramática 7](#_Introdução_da_Gramática)

[2.2 declaração e atribuição de variávies 7](#_Declaração_e_Atribuição)

[2.3 impressão 10](#_Impressão)

[2.4 entradas 11](#_Entradas)

[2.5 operadores 12](#_Operadores_1)

[2.5.1 igualdade e relação 12](#_Igualdade_e_Relação)

[2.5.2 negação 12](#_Negação)

[2.5.3 reunião e interseção 12](#_Reunião_e_Interseção)

[2.5.4 unários 12](#_Unários)

[2.5.5 aritméticos 12](#_Aritméticos)

[2.6 atribuição exponencial 12](#_Atribuição_Exponencial)

[2.7 instruções condicionais 12](#_Instruções_Condicionais)

[2.8 instruções iterativas 12](#_Instruções_Iterativas)

[2.9 análise semântica 12](#_Análise_Semântica)

[3. Geração de código](#_Geração_de_Código) 3

[4. programas Modelo](#_Programas_Exemplo) 3

[5. notas finais](#_Notas_Finais) 3

[6. Referências](#_Referências) 3

**Nota Introdutória:**

**#TODO**

# Linguagem Secundária

## Introdução da gramática

**#TODO**

## Análise Semântica

**#TODO**

Erros Semânticos:

- Não são permitidas declarações de grandezas já anteriormente declaradas.

- Não são permitidas declarações de grandezas em função de outras ainda não

declaradas.

- Não são permitidas unidades repetidas em grandezas diferentes.

- Não são permitidas unidades repetidas para a mesma grandeza.

- Não são permitidas relações em função de mais do que uma unidade.

- Não são permitidas relações com unidades não declaradas.

- Não são permitidas unidades de grandezas compostas definidas com unidades não declaradas.

Speed = Distance/Time SE;

add SA to Speed: m/h; - m e h têm de estar declarados no Distance e Time

# Linguagem Primária

## Introdução da Gramática

#TODO

## Declaração e Atribuição de variáveis

A declaração e atribuição de variáveis pode ser feita das seguintes maneiras:

* < Grandeza > variável ;
* nome\_da\_variável = expr ;
* < Grandeza > variável = expr ;

Apenas as grandezas e unidades definidas na linguagem secundária e os tipos já existentes (interger, float, boolean, string) é que são aceites.

**Exemplos** **corretos**:

* Distância d ;
* d = 10 m ;

Esteja Tempo definido na lingua secundária como Tempo = int h ;

* Tempo tempo = 5 h ;
* Float float ;

**Exemplos** **incorretos**:

* Tempo tempo ; *(Se a grandeza não estiver definida na linguagem secundária)*
* Distância d ; Peso p ; d = p *; (As variáveis teriam de ser da mesma grandeza)*

## Impressão

A impressão é feita da seguinte maneira:

* print expr ;

**Exemplos corretos:**

* print “ Exemplo ” ;
* print “ 5 ” ;
* print “ valor de x = ” x ; (*Seja x uma variável previamente definida*)

## Entradas

As entrades de informação feitas pelo utilizador são feitas da seguinte maneira:

* <Grandeza> nome\_da\_variável = input “informação adicional” unidades ;

**Exemplos** **corretos**:

* Tempo x = input “ Insira o valor de x ” h ;
* Speed x = input “ Insira o valor de y ” km/h ;

**Exemplos** **incorretos**:

* Tempo x = input k/m ; (*é obrigatório o uso de uma String de informação adicional*)

## Operadores

Na nossa linguagem temos definidos alguns operadores de comparação, operadores de igualdade, relaçáo, negação, unários e aritméticos.

### Igualdade e Relação

Os operadores de igualdade (==, !=) e de relação (>, <, <=, >=) permitem comparar dois valores numéricos de grandezas iguais.

**Exemplos** **corretos**:

* 5 > 3 ;
* 5 == 3 ;
* 2 < 3 ;
* 4 >= 7 ;
* 5 <=1 ;

Seja Distância d e Tempo t :

* d != t ;
* d == t ;

**Exemplos** **incorretos**:

* 5 == “string” ;
* “string1” == “string2” ;
* 6 == True ;
* True > 6 ;

### Negação

No que toca à negação, usamos o operador ‘!’ para inverter o valor de um booleano.

**Exemplos corretos:**

* !(true) ;
* !(2 < 6) ;

**Exemplos incorretos:**

* ! ”string” ;

### Reunião e Interseção

No que toca aos operadores de reunião ‘&&’ e interseção ‘||’ só podem ser utilizados para comparar expressões booleanas.

**Exemplos corretos:**

* 1 < 2 && 2 <= 3 ;
* !(false) || true ;

**Exemplos incorretos:**

* 5 && variável\_tempo ;
* “string” || “string” ;
* variável\_distância || “string” ;

### Unários

Operadores unários apenas necessitam de um valor para operar, apesar de não ser algo essecial para a nossa linguagem, é algo que facilita e simplifica vários cálculos, dado isto, decidimos implementar seguindo a seguinte estrutura.

* < expr > = +/- < expr >

Onde ambos ‘expr’ são iguais.

### Aritméticos

Operações aritméticas são a base dos cálculos matemáticos e físicos, dada a sua importância, também implemamos este tipo de operações seguindo as seguintes estruturas.

* < expr > + < expr >
* < expr > - < expr >
* < expr > \* < expr >
* < expr > / < expr >

A expressão ‘expr’ tem de ser numérica.

**Exemplos corretos:**

* Distance d = 5 m ;
* Distance t = 3 cm ;
* Distance r ;
* Int var = 2 ;
* r = d + t ;
* r = d – t ;
* r = d / var ;
* var = d / t ;

**#TODO MULTIPLICATION**

**Exemplos incorretos:**

* Peso m = 50 kg ;
* Peso x ;
* Peso y = 2 kg ;
* int a = 2 ;
* x = 2/m ; (erro sintático, não podem haver divisões de tipos primitivos por grandezas)
* x = m/y ; (erro de atribuição, o resultado não tem uma grandeza atribuida)

**#TODO MULTIPLICATION**

## Atribuição Exponencial

Expoentes têm uma grande importância em cálculos físicos, dado isto, foi algo essecial que fosse implementado, Segue então a seguinte estrutura:

* < expr > ^ < expr > (expr é um valor numérico)

**Exemplo correto:**

Dado Distância d;

* d ^ 6
* 6 ^ 9

**Exemplo incorreto:**

Dado Distância d;

* 5 ^ d (só são permitidos expoentes adimensionais)

## Instruções Condicionais

Para a implementação de instruções condicionais usamos ‘if’ para inicializar a condição, ‘elif’ para uma condição nova e ‘else’ caso nenhuma das condições prévia tenha sido cumprida, tudo isto segue a seguinte estrutura:

if ( < exp > ) {

<statement>

}

elif ( < expr > ) {

< statement >

}

else {

< statement >

}

**Exemplo correto:**

Seja Distância d = 20 m;

if (d < 6) {

print “distância menor que 6” ;

}

elif (d > 6) {

print “distância maior que 6” ;

}

else {

print “distância é igual a 6” ;

}

**Exemplo incorreto:**

if (“hello”) {

print “world” ; (*a condição tem de ser uma expressão booleana*)

}

## Instruções Iterativas

Para as instruções iterativas e dado os objetivos da nossa linguagem criámos dois tipos de ciclos, o ‘while’ e o ‘for <expr> in range’, sendo possível acrescentar o incremento ao ciclo for seguindo a seguinte estrutura:

for i in range(i,f) {

< statement >

}

for I in range(i,f) incr(j) {

< statement >

}

while < expr > {

< statement >

}

Sendo “expr” uma expressão booleana.

**Exemplos corretos:**

Integer i ;

for i in range(0,7) {

print “working” ;

}

Distância d ;

for I in range((3+2), d) incr(2) {

print “working too” ;

}

Integer i = 2;

while I < 10 {

print “working also” ;

}

**Exemplos incorretos:**

Distância d = 5 m ;

for i in range(0,d) { (d tem de ser uma expressão com resultado numérico e adimensional)

print “working” ;

}

## Análise Semântica

**#TODO**

# Geração de Código

**#TODO**

# Programas Modelo

**#TODO**

# Notas Finais

**#TODO**

# Referências

**#TODO**