

## Instruções para o trabalho

Este documento descreve os requisitos do trabalho prático.

### 1. Resumo:

Este trabalho consiste em desenvolver um simulador para a Máquina de Turing, doravante MT, conforme as especificações que seguem. A sintaxe dos comandos da MT foi inspirada no formado adotado no simulador disponível em <http://morphett.info/turing/turing.html>, acrescentada da noção de bloco de comandos para permitir a implementação modular de componentes na programação.

### 2. Entrada:

O simulador será iniciado na linha de comando e todas as saídas serão impressas na tela em modo texto. A sintaxe da linha de comando será: **simturing** <opções> <programa>, com <programa> denotando o nome do arquivo (padrão \*.MT) que contém o código da MT. As <opções> podem ser:

- **-resume** (ou -r), executa o programa até o fim e depois imprime o conteúdo final na fita.
- **-verbose** (ou -v), mostra a execução passo a passo do programa até o fim.
- **-step <n>** (ou -s <n>), mostra n linhas de execução passo a passo na tela, depois abre *prompt* e aguarda nova opção (r, v, s). Caso não seja fornecida nova opção (entrada em branco), o padrão é repetir a última opção.

Iniciado o programa, será apresentado *prompt* para o usuário fornecer a palavra inicial que será escrita na fita antes de iniciar a simulação. Exemplo:

```
1 prompt> simturing -r palindromo.MT
2
3 Simulador de Máquina de Turing ver 1.0
4 Desenvolvido como trabalho prático para a disciplina de Teoria da Computação
5 autor, IFMG, 2016.
6
7 Forneça a palavra inicial:
```

### 3. Saída:

A execução passo a passo será apresentada com a impressão da configuração instantânea da MT antes e depois de cada transição. Cada configuração instantânea será apresentada em modo texto numa linha da tela no formato:

<bloco>.<estado>: <fita à esquerda><cabeçote><fita à direita>

Descrição dos campos:

- <bloco> identificador do bloco em execução, máximo de 16 caracteres significativos.
- <estado> número inteiro positivo, máximo de 9999 por bloco.
- <fita à esquerda> os 20 caracteres mais próximos do cabeçote (ou espaços) pela esquerda
- <cabeçote> 3 caracteres: delimitador da esquerda, símbolo na fita, delimitador da direita.
- <fita à direita> os 20 caracteres mais próximos do cabeçote (ou espaços) pela direita.

Os delimitadores do cabeçote são ( e ) por padrão, podendo ser alterados pela opção **-head delim** que modifica os dois caracteres utilizados como delimitadores esquerdo e direito do cabeçote da fita. Exemplo: “-head <>” vai definir como delimitadores os caracteres < e >.

Exemplo de saída para o código que será apresentado na sequência:

```

1  prompt> simturing -s 12 palindromo.MT
2
3  Simulador de Máquina de Turing ver 1.0
4  Desenvolvido como trabalho prático para a disciplina de Teoria da Computação
5  autor, IFMG, 2016.
6
7  Forneça a palavra inicial: aba
8
9  ..... main.0001: ..... ( a )ba
10 ..... main.0010: ..... ( A )ba
11 ..... moveFim.0001: ..... ( A )ba
12 ..... moveFim.0001: ..... A ( b )a
13 ..... moveFim.0001: ..... Ab ( a )
14 ..... moveFim.0001: ..... Aba ( _ )
15 ..... moveFim.0001: ..... Ab ( a )
16 ..... main.0011: ..... Ab ( a )
17 ..... iniEsq.0001: ..... Ab ( a )
18 ..... main.0012: ..... Ab ( a )
19 ..... main.0030: ..... Ab ( A )
20 ..... moveIni.0001: ..... Ab ( A )
21
22 Forneça opção ( r , v , s ) :
```

## 4. Linguagem da MT:

A sintaxe foi inspirada no formado adotado no simulador disponível em <http://morphett.info/turing/turing.html>, mas foi modificada em vários aspectos para contemplar a utilização de blocos que permitam a programação modular. Basicamente, existem 3 tipos de comandos: básicos, de criação de bloco e de chamada de bloco.

### Sintaxe dos comandos básicos:

- Cada linha do programa fonte contém uma tupla na forma:  
**<estado atual> <símbolo atual> - - <novo símbolo> <movimento> <novo estado>**
- Para denotar <estado atual> ou <novo estado> você pode utilizar um inteiro de até 4 dígitos.
- Alternativamente, para denotar o <novo estado> você pode utilizar os identificadores “retorne” ou “pare”. Identificadores são *case-sensitive*.
- Para denotar o <símbolo atual> e o <novo símbolo> você pode usar qualquer caractere, ou use ‘\_’ para representar o branco (espaço).
- O <movimento> denota a ação do cabeçote na fita, indicada por um caractere: ‘e’ denota movimento para a esquerda, ‘d’ denota movimento para a direita, ‘i’ denota ausência de movimento (imóvel).
- Tudo depois de um ‘;’ é tratado como comentário e ignorado.
- A execução termina quando a MT alcança o estado “pare” ou ocorre um erro.

### Extensões dos comandos básicos:

- ‘\*’ pode ser usado como coringa em **<estado corrente>** e **<símbolo corrente>** para denotar qualquer estado ou caractere, respectivamente.
- ‘\*’ pode ser usado como coringa em **<novo estado>** e **<novo símbolo>** para significar ausência de mudança.
- ‘!’ pode ser usado no final da linha para criar um *breakpoint*. Durante a execução do programa, a máquina vai pausar automaticamente depois de executar essa linha, abrir *prompt* e aguardar nova opção.

### Criação e chamada de blocos:

- Para iniciar um novo bloco a sintaxe é:  
**bloco** <identificador> <estado inicial>
- Para finalizar um bloco a sintaxe é:  
**fim**
- Para chamar um bloco a sintaxe é:  
<estado atual> <identificador de bloco> <estado de retorno>
- Os estados dentro de um bloco são independentes e não conflitam com estados de outros blocos. A execução do bloco vai iniciar no estado inicial fornecido na declaração. O identificador “retorne” é utilizado para saída do bloco, denotando o estado de retorno fornecido na chamada.
- A execução do programa sempre inicia no bloco especial “main”.

## 5. Informações gerais:

O trabalho pode ser executado em grupo de até dois alunos. A data de entrega será combinada em sala de aula, bem como as apresentações dos trabalhos. A próxima página traz o código fonte de um programa como exemplo.

```

1      ; Detector de palindromos
2      bloco main 1
3          01 a — A i 10
4          01 b — B i 20
5          10 moveFim 11
6          20 moveFim 21
7
8      ; leu a
9          11 iniEsq 12
10         12 a — A i 30
11         12 b — * i 70
12         12 _ — * i 60
13
14     ; leu b
15         21 iniEsq 22
16         22 a — * i 70
17         22 b — B i 30
18         22 _ — * i 60
19
20         30 moveIni 31
21         31 iniDir 32
22         32 _ — * e 60
23         32 * — * i 01
24
25         60 sim pare
26         70 nao pare
27 fim ; main
28
29 ; move para ultimo caractere da palavra
30 bloco moveFim 1
31     01 _ — * e retorne
32     01 * — * d 01
33 fim ; moveFim
34
35 ; move para primeiro caractere da palavra
36 bloco moveIni 1
37     01 _ — * d retorne
38     01 * — * e 01
39 fim ; moveIni
40
41 ; recua ate caractere minusculo ou _
42 bloco iniEsq 1
43     01 _ — * i retorne
44     01 a — * i retorne
45     01 b — * i retorne
46     01 * — * e 01
47 fim ; iniEsq
48
49 ; avanca ate caractere minusculo ou _
50 bloco iniDir 1
51     01 _ — * i retorne
52     01 a — * i retorne
53     01 b — * i retorne
54     01 * — * d 01
55 fim ; iniDir
56
57 ; palavra eh palindromo
58 bloco sim 1
59     01 moveFim 02
60     02 * — * d 03
61     03 * — S d 04
62     03 * — I d 04
63     03 * — M d retorne
64 fim ; sim
65
66 ; palavra nao eh palindromo
67 bloco nao 1
68     01 moveFim 02
69     02 * — * d 03
70     03 * — N d 04
71     03 * — A d 04
72     03 * — O d retorne
73 fim ; nao

```