INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA MINAS GERAIS Campus Formiga

Graduação em Ciência da Computação

Disciplina: Teoria da Computação
Professores: Walace Rodrigues & Mário Luiz
Atividade: 2o Trabalho Prático

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 quanto 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.

```
; Detector de palindromos
bloco main 1
   01 a -- A i 10
   01 b -- B i 20
   10 moveFim 11
   20 moveFim 21

   ; leu a
   11 iniEsq 12
   12 a -- A i 30
   12 b -- * i 70
   12 _ -- * i 60

   ; leu b
   21 iniEsq 22
   22 a -- * i 70
   22 b -- B i 30
   22 _ -- * i 60

   30 moveIni 31
   31 iniDir 32
   32 _ -- * e 60
   32 * -- * i 01

   60 sim pare
   70 nao pare
fim ; main

; move para ultimo caractere da palavra
bloco moveFim 1
   01 _ -- * e retorne
   01 * -- * d 01
fim ; moveFim

; move para primeiro caractere da palavra
bloco moveIni 1
   01 _ -- * d retorne
   01 * -- * e 01
fim ; moveIni

; recua ate caractere minusculo ou _
bloco iniEsq 1
   01 _ -- * i retorne
   01 a -- * i retorne
   01 b -- * i retorne
   01 * -- * e 01
fim ; iniEsq

; avanca ate caractere minusculo ou _
bloco iniDir 1
   01 _ -- * i retorne
   01 a -- * i retorne
   01 b -- * i retorne
   01 * -- * d 01
fim ; iniDir

; palavra eh palindromo
bloco sim 1
   01 moveFim 02
   02 * -- * d 03
   03 * -- S d 04
   03 * -- I d 04
   03 * -- M d retorne
fim ; sim

; palavra nao eh palindromo
bloco nao 1
   01 moveFim 02
   02 * -- * d 03
   03 * -- N d 04
   03 * -- A d 04
   03 * -- O d retorne
fim ; nao
```