Fundamentos de Sistemas Operacionais
Trabalho 01

Jonathan Henrique Maia de Moraes - 12/0122553

01/09/2016

Ambiente de Desenvolvimento

- **Sistema Operacional**: Debian Jessie (8.5)
- **Editor de Texto**: Atom 1.0.19
- **Compilador**: gcc 4.9.2
- **Flags do Compilador**: -O2 -lm -Wall

Instruções

Questão 01

- **Diretório**: question_01
- **Comandos do Makefile**:
  - **make**: compila o código, gerando o executável de nome "q01"
  - **make clean**: remove o executável de nome "q01"
- **Comando de Execução**: "./q01"
- **Operações**:
  - **Menu Principal**: Selecione dentre as opções descritas digitando o valor numérico correspondente. Para sair, digite "q"
  - **1) Definir Triângulo**: Escreva 03 pares numéricos inteiros, com espaço ou ENTER entre cada valor numérico. Exemplo: "0 0 0 10 10 0" definirá os valores para os os pontos a, b e c do triângulo:

$$\begin{aligned} a &= (0,0) \\ b &= (0,10) \\ c &= (10,0) \end{aligned} \tag{1}$$

  - **2) Tamanho dos Lados do Triângulo**: É informado a soma de cada lado do triângulo através do Teorema de Pitágoras que

define o lado como a distância entre 02 pontos do triângulo. Para o exemplo da eq. (1), o tamanho de cada lado do triângulo é:

$$\begin{aligned} t &= \sqrt{\Delta x^2 + \Delta y^2} \\ AB = \ t_{a,b} &= \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2} \\ &= \sqrt{(0-0)^2 + (10-0)^2} = 10 \\ BC = \ t_{b,c} &= \sqrt{(10-0)^2 + (0-10)^2} \approx 14.14 \\ CA = \ t_{c,a} &= \sqrt{(0-10)^2 + (0-0)^2} = 10 \end{aligned} \tag{2}$$

– **3) Existência do Triângulo**: É informado a condição de existência de um triângulo através do Teorema da Desigualdade Triangular de Euclídes. Para o exemplo da eq. (2), as afirmações a seguir são verdadeiras:

$$\begin{aligned} |AB - BC| < CA &< AB + BC \\ |10 - 14.14| < 10 &< 10 + 14 \\ 4.14 < 10 &< 24 \\ |CA - AB| < BC &< CA + AB \\ 0 < 14.14 &< 20 \\ |BC - CA| < AB &< BC + CA \\ 4.14 < 10 &< 2 \end{aligned} \tag{3}$$

– **4) Perímetro do Triângulo**: É informado o perímetro do triângulo através da soma de cada lado do mesmo. Se o valor desta soma for 0, certamente os pontos estão definidos na mesma posição. Para o exemplo da eq. (2), o perímetro do triângulo é:

$$\begin{aligned} p &= AB + BC + CA \\ p &= 10 + 14.14 + 10 = 34.14 \end{aligned} \tag{4}$$

– **5) Área do Triângulo**: É informado a área do triângulo através da Fórmula de Heron. Se o valor da área for 0, certamente os

pontos não definem um triângulo. Para o exemplo da eq. (5), a área do triângulo é:

$$\begin{aligned} s &= \frac{p}{2} = 17.07 \\ a &= \sqrt{s(s-AB)(s-BC)(s-CA)} \\ &= \sqrt{17.07(17.07-10)(17.07-14.14)(17.07-10)} \\ &= \sqrt{17.07(7.07)(2.93)(7.07)} \\ &= \sqrt{2500} \\ &= 50 \end{aligned} \tag{5}$$

Questão 02

- **Diretório**: question_02
- **Comandos do Makefile**:
  - **make**: compila o código, gerando o executável de nome "q02"
  - **make tc1**: compila e executa o "q02" com as entradas "1 11 5 21 10 31 15 41 20 51 25". O valor esperado de saída é: "1 5 10 11 15 20 21 25 31 41 51". O objetivo deste teste é avaliar o resultado da ordenação sem entradas repetidas.
  - **make tc2**: compila e executa o "q02" com as entradas "20 20 15 15 30 30 40 5 5 2". O valor esperado de saída é: "2 5 5 15 15 20 20 30 30 40". O objetivo deste teste é avaliar o resultado da ordenação com entradas repetidas.
  - **make tc3**: compila e executa o "q02" com as entradas "-d 1 2 3 4 5 10 9 8 7 6". O valor esperado de saída é: "1 2 3 4 5 6 7 8 9 10". O objetivo deste teste é avaliar o resultado da ordenação com a sinalização "-d".
  - **make tc4**: compila e executa o "q02" com as entradas "-r 1 2 3 4 5 10 9 8 7 6". O valor esperado de saída é: "10 9 8 7 6 5 4 3 2 1". O objetivo deste teste é avaliar o resultado da ordenação com a sinalização "-r".
  - **make tc5**: compila e executa o "q02" com as entradas "-r 5 -d 2 -r 25". O valor esperado de saída é: "25 5 2". O objetivo deste teste

é avaliar o resultado da ordenação com múltiplas sinalizações. A sinalização considerada será sempre a última, neste caso: "-r".

- **make tc6**: compila e executa o "q02" com as entradas "-40 40 -30 30 -20 20 -10 10 0". O valor esperado de saída é: "-40 -30 -20 -10 10 20 30 40". O objetivo deste teste é expor a limitação de uma entrada de valor 0, além do resultado da ordenação com números negativos. O métdo "atoi()" retorna 0 caso a entrada não seja válida. Desta forma, o valor 0 foi considerado como palavra reservada neste projeto

- **Comando de Execução**: "./q02"

- **Operações**:

  - **Ordenação**: Através dos argumentos numéricos fornecidos ao executar no terminal, é ordenado os mesmos em ordem crescente ou decrescente (caso seja inserido o sinalizador "-r" nos argumentos) e informado o resultado de tal ordenação.

Questão 03

- **Diretório**: question_03

- **Comandos do Makefile**:

  - **make**: compila o código, gerando o executável de nome "q03"
  - **make tc1**: compila e executa o "q03" com o arquivo de entrada "tc1.in" com os valores "um q". O objetivo deste teste é avaliar os resultados em um caso típico.
  - **make tc2**: compila e executa o "q03" com o arquivo de entrada "tc2.in" com uma string de 106 caracteres seguido de um espaço e a letra "q". O objetivo deste teste é avaliar os resultados em um caso atípico.

- **Comando de Execução**: "./q03"

- **Operações**:

– **Inserção de uma *string***: Após o passo 9 é esperado o fornecimento de uma *string* para a conclusão dos passos seguintes. As respostas das perguntas descritas no arquivo-guia estão inclusos no programa.