Aula do CCM do dia 8/10/2013

Leonardo Nagami Coregliano

9 de outubro de 2013

Observação. *Os exercícios numerados abaixo são do livro [1] e a sua numeração foi preservada.*

Exercício (1.3.41. Mediana de cinco)**.** *Escreva um programa que recebe cinco inteiros distintos da linha de comando e imprime a mediana deles.*

Extra: Resolva o problema fazendo menos que sete comparações.

Exercício (Fibonacci)**.** *A sequência de Fibonacci é definida recursivamente como*

$$\begin{aligned} a_1 &= 1; \\ a_2 &= 1; \\ a_{n+2} &= a_n + a_{n+1}, \text{ para } n \in \mathbb{N}^*. \end{aligned}$$

Essa sequência pode ser "aproximada" pela sequência

$$\forall n \in \mathbb{N}^*, b_n = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n.$$

Escreva um programa que recebe um argumento N na linha de comando e imprime os termos de 1 a N de ambas as sequências.

Extra: Descubra o porquê das aspas em "aproximada".

O que realmente é um vetor ("array")?

Quando criamos um vetor `a[]`, o nome `a` refere-se ao endereço do primeiro elemento do vetor e não ao vetor em si. Os outros elementos do vetor são obtidos através de deslocamentos fixos (dependendo do tipo dos elementos de `a`).

Em Java, os vetores ainda possuem outras informações embutidas, a mais útil delas é o campo `length`, que contém o comprimento do vetor e pode ser obtido fazendo `a.length`.

Exercício (1.4.9)**.** *Qual é a saída do trecho de código abaixo?*

```
int[] a = {1, 2, 3};
int[] b = {1, 2, 3};
System.out.println(a == b);
```

Exercício (1.4.11)**.** *Escreva fragmentos de código que criam uma matriz* `b[][]` *que é uma cópia de uma matriz* `a[][]` *já existente, sob as seguintes premissas:*

a. `a[][]` *é quadrada;*

b. `a[][]` *é retangular;*

c. `a[][]` *é geral.*

Exercício (Algoritmo de Euclides para m.d.c)**.** *Implemente o algoritmo de Euclides para cálculo de m.d.c. descrito abaixo:*

Dados inteiros não-nulos a_0 e a_1, faça $a_{n+2} = a_n \bmod a_{n+1}$ até que $a_m = 0$ para algum m. Devolva a_{m-1}.

Considere que a_0 e a_1 são dados como entrada na linha de comando.

Exercício (1.4.14)**.** *Escreva um programa que recebe um inteiro N da linha de comando e cria uma matriz $N \times N$ booleana* `a[][]` *tal que* `a[i][j]` *é* `true` *se* `i+1` *e* `j+1` *são relativamente primos e é* `false` *caso contrário.*

Exercício (1.4.20 (EP). Simulação de dados)**.** *O seguinte código computa a distribuição de probabilidade exata da soma de dois dados honestos:*

```
double[] dist = new double[13];
for (int i = 1; i <= 6; i++)
  for (int j = 1; j <= 6; j++)
    dist[i+j] += 1.0;
for (int k = 1; k <= 12; k++)
  dist[k] /= 36.0;
```

O valor `dist[k]` *é a probabilidade de que os dados somem* `k`*. Execute experimentos para validar esse cálculo simulando N jogadas de dados, acompanhando as frequências de ocorrência de cada valor de soma. Quão grande N tem de ser para que os resultados empíricos batam com os calculados com precisão até a terceira casa decimal?*

Exercício (1.4.21 (EP). Plateau mais longo)**.** *Dado um vetor de inteiros, determine o comprimento e a localização da maior sequência contígua de valores iguais onde os valores dos elementos imediatamente antes e depois dessa sequência não são maiores.*

Exercício (1.4.30. Campo Minado)**.** *Escreva um programa que recebe 3 argumentos na linha de comando M, N e p e produz uma matriz $M \times N$ booleana, onde cada entrada é ocupada com probabilidade p (independentemente). No jogo de Campo Minado, células ocupadas representam bombas e células vazias representam células seguras. Imprima a matriz usando um asterisco para bombas e um ponto para células seguras. Em seguida, substitua cada célula vazia com o número de bombas vizinhas (aos lados e diagonais) e imprima o resultado.*

```
* * . . . * * 1 0 0
. . . . . 3 3 2 0 0
. * . . . 1 * 1 0 0
```

Tente escrever seu código de forma que ele tenha poucos casos especiais a serem tratados usando uma matriz $(M + 2) \times (N + 2)$.

Exercício (Ordenação)**.** *Faça um programa que recebe uma sequência de inteiros nos argumentos da linha de comando e imprime-os em ordem crescente.*

Dica. *Um dos algoritmos mais simples de ordenação consiste em repetidamente procurar o i-ésimo menor valor e colocá-lo na i-ésima posição.*

Referências

[1] R. Sedgewick and K.D. Wayne, *Introduction to programming in java: an interdisciplinary approach*, Pearson Addison-Wesley, 2008. (document)