Lista de Exercícios de Aquecimento

UC - Estruturas de Dados e Análise de Algoritmos

Universidade São Judas Tadeu

Esta primeira série de exercícios serve como base para uma visão geral dos conceitos apresentados sobre a linguagem Java a ser usada durante as aulas de Estruturas de Dados e Análise de Algoritmos. Note que o foco dos exercícios não está na linguagem usada nem construção da interface, e sim, nos algoritmos propostos. Há vários exercícios pequenos e simples. Todos eles são baseados na mesma dinâmica do primeiro programa básico, mostrado a seguir:

```
import java.io.*;
import java.util.Scanner;

public class Lista1a
{
  public static void main ( String args [] )
  {
    /* cria mecanismo para fazer a leitura dos dados pelo console*/
    Scanner entrada = new Scanner(System.in);

    /* Declare aqui as variáveis necessárias */

    int n1,n2,tot;

    /* Coloque aqui as instruções para leitura dos dados */

    try{
      System.out.print("Entre com dois valores inteiros:\n");
      n1 = entrada.nextInt();  // entrada.nextFloat() para reais
      n2 = entrada.nextInt();  // entrada.nextLine() para strings

      /* Coloque aqui o código para resolver o problema.
       * Novas variáveis podem ser criadas, se necessário. */
      tot = n1+n2;

      /* Coloque aqui o código para mostrar os resultados */

      String saida = "Resposta:\n";
      saida = saida + n1 + " + " + n2 + " = " + tot;
      saida = saida + "\nFim.";
      System.out.print(saida);

      } catch (Exception e) {
         System.out.println("Ocorreu algum erro!");
    }
  }
}
```

Este programa bem simples faz a leitura de dois números inteiros e retorna na tela o valor da soma deles. Tanto a entrada quanto a saída é feita pelo console que, no JGrasp, é apresentado na janela inferior à janela de código na aba “Run I/O”.

Em primeiro lugar é apresentada uma mensagem na tela solicitando que se digite dois números. Em seguida, após a leitura dos números, é feita a soma e apresentado o resultado da soma.

Detalhando mais a descrição do programa, podemos dizer que logo de início são declarados os *imports* necessários para o programa relacionados a entrada e saída de dados. Após a declaração da classe sempre necessária em um programa Java e logo em seguida a declaração padrão da rotina principal (*main*) por onde o programa começará a executar. Logo de início, dentro da rotina *main*, é criado o mecanismo pelo qual se fará a leitura de dados com a declaração do *Scanner* e em seguida são criadas as variáveis n1 e n2 onde serão colocadas as entradas e a variável tot que receberá o cálculo da soma.

As instruções seguintes estão sob a “guarda” de um “try/catch” para que qualquer erro ocorrido (principalmente com a entrada de dados) seja tratada e não termine simplesmente em encerramento anormal do programa sem uma mensagem de erro.

Logo em seguida temos uma instrução que apresenta no console a solicitação dos dois números, as duas instruções de entrada de dados em n1 e n2, a instrução do cálculo da soma e, por fim, as instruções que constroem a String de saída e a saída em si com o *System.out.println*.

- Exercícios básico só com estrutura seqüencial:

01) Faça um programa que receba um valor que é o valor pago, um segundo valor que é o preço do produto e retorne o troco a ser dado.

02) Faça um programa que receba o valor do quilo de um produto e a quantidade de quilos do produto consumida calculando o valor final a ser pago.

Aqui você pode usar o valor da quantidade do produto como um float. A leitura fica assim:

valor = entrada.nextFloat();

- Exercícios básicos com estrutura de decisão:

```
if  (condição1) {
   if (condição2) {
      instruçõesA
   } else {
      instruçõesB
   }
} else {
      instruçõesC
}
```

```
switch (variável) {
      case opção1 : instruçõesA;
            [break;]
      case opção2 : instruçõesB;
            [break;]
      case opção3 : instruçõesC;
            [break;]
      default: instruçõesD;
}
```

03) Faça um programa que receba 2 valores e retorne o maior entre eles.

04) Faça um programa que receba 4 valores e retorne o menor entre eles.

05) Faça um programa que verifique se um número é impar.

06) Faça um programa que receba 3 valores que representarão os lados de um triângulo e verifique se os valores formam um triângulo e classifique esse triângulo como:

- eqüilátero (3 lados iguais);
- isósceles (2 lados iguais);
- escaleno (3 lados diferentes).

Lembre-se que para formar um triângulo:

- nenhum dos lados pode ser igual a zero;
- um lado não pode ser maior do que a soma dos outros dois;

07) Utilize a estrutura if para fazer um programa que retorne o nome de um produto a partir do código do mesmo. Considere os seguintes códigos:

1 – Parafuso;
2 – Porca;
3 – Prego;

Para qualquer outro código indicar “Diversos”.

08) Refaça o exercício anterior usando a estrutura switch.

- Exercícios básicos com estrutura de repetição:

```
while (condição) {
    instruções
}
```

```
do {
    instruções
} while (condição)
```

```
for( variável = valorinicial; condição; incremento) {
    instruções
}
```

09) Faça um programa que utilize a estrutura ***while*** para ler 50 números e calcule e exiba a média aritmética deles.

10) Refaça o programa anterior utilizando a estrutura ***do while***.

11) Refaça novamente o exercício usando a estrutura ***for***.

- Exercícios que utilizam vetores:

tipo [ ] *identificador* = new *tipo* [*quantidade*];

ou

tipo identificador [ ] = new *tipo* [*quantidade*];

12) Faça um programa que receba 10 valores inteiros e os coloque em um vetor. Em seguida exiba-os em ordem inversa à ordem de entrada.

13) Faça um programa que utilize uma estrutura de repetição para ler 50 números armazenando-os em um vetor e calcule e exiba a média aritmética deles. Em seguida o programa deve apresentar todos os valores armazenados no vetor que sejam menores que a média.

Antes de continuar a lista de exercício considere o programa apresentado na página seguinte e sua explicação...

Os próximos exercícios utilizam vetores. Há vários exercícios pequenos e simples. Todos eles são baseados no mesmo programa básico, mostrado a seguir e servem para relembrar os algoritmos relacionados:

```
public class Lista1b
{
   public static void main ( String args [] )
   {
      /* Vetor de "entrada", que já é alocado e inicializado.
      Neste caso, o new é dispensado, pois o compilador já
     deduz o número de posições que devem ser alocadas. */

      int a [] =
     {32,45,89,66,12,35,10,96,38,15,13,11,65,81,35,64,16,89,54,19};

      /* A variável n conterá sempre o tamanho do vetor a.
      Isto irá facilitar novos testes caso queira
      mudar o conteúdo do vetor a */

      int n = a.length;

      /* Declaração e alocação do vetor b, em que será escrita a saída.
      Não sabemos quantos elementos serão necessários, mas sabemos
      que n serão suficientes. O objetivo dos exercícios é mudar o
      conteúdo do vetor b. A variável m declarada a seguir também deve
      ser alterada, indicando quantos elementos de b são realmente
      importantes para a resposta. */

      int b [] = new int [n];
      int m = 0;

      /* A variável a seguir é usada como índice de laços. */

      int i;

      /* Coloque aqui o código necessário para resolver o problema.
      Novas variáveis podem ser criadas, se necessário. No
      final, b e m devem ser modificados.*/

      /* O código abaixo mostra o resultado. Não deve ser alterado */

      String saida = "Resposta:\n";

      for (i = 0; i < m; i++)
         saida = saida + b[i] + " ";
      saida = saida + "\nFim.";

      System.out.println(saida);

      //System.exit (0);
   }
}
```

Este programa utiliza dois vetores a e b. O vetor a já é inicializado pelo próprio programa. A variável n indica o número de dados escritos em a.

Todos os exercícios a seguir envolvem modificar o conteúdo do vetor **b,** que contém 20 posições, assim como **a**, mas há exercícios em que nem todas as posições de **b** são utilizadas. A variável **m** deve indicar quantas posições de **b** foram realmente usadas.

Assim, em cada um dos exercícios, deve-se fazer um programa que processe os vetores **a** e **b** como pedido. Note que o programa deve ser feito de forma que funcione para quaisquer valores de **n** e dos elementos de **a**. Não serão válidos programas que só funcionem para os dados usados nesta inicialização!

Em cada exercício a seguir, é dado o resultado esperado em tela. Comparem-no com o resultado que obtiveram na prática, mas lembrem-se: às vezes um programa errado, em algumas situações, pode dar a resposta certa! Se a resposta do programa for a mesma que é a apresentada, é provável que esteja no caminho certo, mas sem garantias de que o programa esteja inteiramente correto.

14) O vetor **b** deve se tornar uma cópia do vetor **a**.

Resposta: 32 45 89 66 12 35 10 96 38 15 13 11 65 81 35 64 16 89 54 19

15) O vetor **b** deve se tornar uma cópia revertida do vetor **a** (a ordem dos elementos deve ser trocada).

Resposta: 19 54 89 16 64 35 81 65 11 13 15 38 96 10 35 12 66 89 45 32

16) **b[0]** deve receber o valor do maior elemento (conteúdo) de **a**.

Resposta: 96

17) **b[0]** deve receber o índice (posição) do menor elemento (conteúdo) de **a**. Em caso de empate, o índice indicado deve ser o menor.

Resposta: 6

(Note que **a[6] = 10** é o menor elemento (conteúdo) presente no vetor **a**.)

Variante: modifique o programa para que, em caso de empate entre dois índices (posições), indique-se o maior índice (posição).

18) **b** deve receber a lista dos números que estão nos índices (posições) pares de **a**.

Resposta: 32 89 12 10 38 13 65 35 16 54

19) **b** deve receber a lista dos números pares de **a**.

Resposta: 32 66 12 10 96 38 64 16 54

20) **b** deve receber a lista dos índices (posições) de **a** que contém elementos maiores do que 50.

Resposta: 2 3 7 12 13 15 17 18

21) **b[0]** deve receber a média aritmética dos elementos de **a** (arredondada para baixo).

Resposta: 44

22) **b[0]** deve receber o total dos elementos ímpares de **a**.

Resposta: 497

23) **b[0]** deve receber o maior elemento de **A** que seja inferior a 50 (se não houver números inferiores a 50, a resposta deve ser 0). Considere que nunca haverá elementos negativos em **a**.

Resposta: 45

24) **b[0]** deve receber o índice do primeiro elemento ímpar de **a** (se não houver números ímpares em **a,** a resposta deve ser **n**).

Resposta: 1

25) **b[0]** deve receber o índice do último elemento par de **a** (se não houver números pares em **a,** a resposta deve ser -1).
Resposta: 18

26) **b** deve receber a lista decrescente dos índices de **a** que contenham elementos menores que 50.

Resposta: 19 16 14 11 10 9 8 6 5 4 1 0

27) **b** deve receber a lista dos índices de **a** em que aparecem elementos menores do que os que estão no índice seguinte. O último índice do vetor não deve ser considerado.

Resposta: 0 1 4 6 11 12 14 16

28) **b** deve receber a lista dos índices de **a** em que aparecem elementos que são a média aritmética dos seus vizinhos à esquerda e à direita. O primeiro e o último índice não devem ser considerados.

Resposta: 10 18

29) **b** deve receber a lista dos índices de **a** que contém o mesmo elemento que está no índice "simétrico": O primeiro elemento deve ser comparado com o último, o segundo com o penúltimo e assim por diante. Um par de números só deve ser comparado uma vez, ou seja, se **a[3]** = **a[16]** apenas o **3** deve aparecer na lista.

Resposta: 2 5

30) **b** deve receber **a** " filtrado". O primeiro e o último índice se mantém iguais, mas os índices internos devem ser modificados da seguinte maneira: cada índice de **b** conterá a média aritmética do número na posição correspondente em **b** e dos números vizinhos.

Resposta: 32 55 66 55 37 19 47 48 49 22 13 29 52 60 60 38 56 53 54 19

31) **b[0]** deve receber o maior elemento de **a**, enquanto que **b[1]** deve receber o segundo maior elemento de **a**. Você pode supor que **a** tem pelo menos dois elementos.

Resposta: 96 89

OBS: os próximos exercícios podem exigir dois laços, além de comandos condicionais.

32) **b** deve receber a lista dos elementos de **a** que são primos.

Resposta: 89 13 11 89 19

33) **b** deve receber **a** ordenado de forma crescente ou " ordem não-decrescente", já que poderá haver números repetidos. Este é um problema de solução mais complicada, para a qual haverá soluções clássicas, que veremos nesta disciplina. Veja o que consegue sozinho!

Resposta: 10 11 12 13 15 16 19 32 35 35 38 45 54 64 65 66 81 89 89 96

34) **b** deve receber os elementos de **a**, removendo-se os que aparecem apenas uma vez. Os que aparecem mais de uma vez devem aparecer tantas vezes quantas apareciam em **a**.

Resposta: 89 35 35 89