Introdução à Programação
notas de aula – versão 1

11 de fevereiro de 2011

1 Variáveis, entrada e saída

Suponha que queiramos escrever um programa que calcule a área de um retânglo. É algo trivial e pode parecer estranho criarmos um programa somente para isso, mas ele nos servirá como um exemplo importante.
Primeiro, o programa precisa perguntar ao usuário os valores dos comprimentos da base e da altura; depois, multiplica ambos; finalmente, mostra o valor da área.
O pseudocódigo deste programa é mostrado a seguir:

```
leia base
leia altura
area ← base * altura
mostre area
```

As duas primeiras linhas significam que queremos que o computador pergunte dois valores ao usuário, e que lhes dê os nomes "base" e "altura".
Quando dissemos que o computador deve "dar nome" a um valor, queremos dizer que este valor deve ser armazenado em algum lugar da memória, e que vamos referenciar esse lugar pelo nome. Chamamos estes "lugares com nome" de *variáveis*
A terceira linha diz ao computador para multiplicar os dois valores e que deixe estes valores em um lugar da memória com o nome "area". Damos a isto o nome de *atribuição de valor a uma variável*.
A quarta linha determina que o computador mostre ao usuário o conteúdo da variável "area".
A seguir há um resumo dos conceitos ilustrados com este pequeno programa.

- **Sequenciamento:** realizar várias operações, uma depois da outra.
  Como: Listar as operações, uma por linha.

  ```
  comando-1
  comando-2
  comando-3
  ```

- **Variáveis:** locais da memória onde podemos guardar valores. Estes locais são referenciados pelo nome.
- **Artribuição:** modificar o valor de uma variável.
  Como: após o nome da variável, ←, seguido do valor a ser armazenado na variável. Por exemplo:

  ```
  a ← b + c
  ```

- **Entrada e saída:** Usamos dois comandos, `leia` e `mostra` para realizar *operações de entrada e de saída de dados.*

2 Escolhas

O próximo programa pergunta ao usuário três números (os comprimentos de três segmentos de reta, sendo o primeiro deles o maior), e depois diz se os três podem ou não ser lados de um triângulo.
Se os números são a, b, c, precismos verificar se $b + c > a$. Caso seja verdade, dizemos que é um triângulo; em caso contrário, dizemos que não

```
leia a
leia b
leia b
se ( b+c > a )
    mostre "E um triangulo"
senao
    mostre "Nao podem ser lados de um triangulo"
```

As três primeiras linhas não trazem nada de novo, mas as outras sim. Ao encontrar o comando `se`, o programa verificará a expressão logo em seguida (neste caso, "`b+c > a`"). Se ela for verdadeira, o programa executará o trecho logo depois do `se`, que está "deslocado para a direita" – neste exemplo, somente a linha `mostre "É um triângulo"`. Se a expressão for falsa, o programa executará o trecho logo após o `senão`.
Uma "condição" é uma expressão cujo valor pode ser "verdadeiro" ou "falso". Por exemplo, "$3 > 2$" (verdadeiro), "$x < y$" (depende dos valores de x e y).
A seguir há um resumo dos conceitos ilustrados com este pequeno programa.
Decisão: executar n comandos, apenas se uma determinada condição for verdadeira.
Como: após a palavra "`se`" deve haver:

- Uma expressão condicional entre parênteses;
- Depois da expressão, dois pontos;
- Depois, os comandos a serem executados, indentados à frente;
- Depois, a palavra "`senao`", sem indentação;
- Depois, os comandos a serem executados se a condição for FALSA (indentados à frente).

```
se (CONDICAO):
    comando-1
```

```
    comando-2
    ...
    comando-n
senao:
    comando-a
    comando-b
    ...
    comando-k
```

Quando não for necessária, a parte do "senão" pode ser omitida:

```
se (CONDICAO):
    comando-1
    comando-2
    ...
    comando-n
```

3 Repetições

Agora queremos repetir um trecho de código várias vezes. Por exemplo, um programa que conte de dez a um de depois escreva "Final da contagem". Podemos crir uma variável n e dar-lhe o valor inicial 10. Depois, fazemos o seguinte:

- Mostramos o valor de n;
- Calculamos $n-1$ e guardamos no lugar onde estava n (em outras palavras, mudamos o valor da variável n para $n-1$);
- Continuamos a fazer estas operações enquanto n for positivo;
- Finalmente, mostramos a mensagem "Fim da contagem".

Este algoritmo é traduzido no seguinte pseudocódigo:

```
n ← 10
enquanto n > 0
    mostre n
    n ← n - 1
mostre "Fim da contagem"
```

O comando `enquanto` é semelhante ao `se`, porque verifica uma condição e executa um trecho de código somente se a condição for verdadeira. No entanto, o `enquanto` continuará executando o trecho interno até que a condição deixe de ser verdade.
Uma variável usada da forma que usamos o n neste exemplo é chamada de *contador*.
A seguir há um resumo dos conceitos ilustrados com este pequeno programa.
Repetição: enquanto uma condição for verdadeira, repetir a execução de n comandos.
Como: após a palavra "`enquanto`":

- uma condição entre parênteses;
- Dois pontos;
- Depois, os n comandos listados, um por linha,indentados à frente.

```
enquanto (CONDICAO):
    comando-1
    comando-2
    ...
    comando-n
```

3.1 Exemplos completos

3.1.1 n é primo?

Para verificar se um número é primo podemos contar de 2 até $n-1$ verificando se algum número divide n. Se encontrarmos divisor, n não é primo. Se não encontarmos, n é primo. Este algoritmo é descrito em pseudocódigo a seguir:

```
leia n

i ← 2
enquanto i < n
    se resto(n,i) = 0
        mostre "n e composto!"
    i ← i + 1
mostre "n e' primo"
```

Podemos melhorar o algoritmo: não precisamos testar todos os números! Se já testamos 2, não há porque tentar divisores pares. Começamos do 3 e contamos de dois em dois:

```
leia n
se n = 2
    mostre "n e' primo!"
    pare
i ← 3
enquanto i < n
    se resto(n,i) = 0
        mostre "n e composto!"
    i ← i + 2
mostre "n e' primo"
```

Ainda melhor – não precisamos contar até depois de $n/2$:

```
leia n
se n = 2
    mostre "n e' primo!"
    pare
i ← 3
enquanto i < n/2
    se resto(n,i) = 0
        mostre "n e' composto!"
    i ← i + 2
mostre "n e' primo"
```

Na verdade só precisamos testar até que $\sqrt{n}$. Suponha que $k^2 = n$. Se há um inteiro X maior que k que divide n, então existe inteiro y tal que $Xy = n$. Mas y precisa ser menor que $\sqrt{n}$, porque se tanto X como y forem maiores que $\sqrt{n}$, então Xy seria maior que n. Mas já teremos testado y antes de chegar a $\sqrt{n}$.

```
leia n
se n = 2
    mostre "n e' primo!"
    pare
i ← 3
enquanto i < √n
    se resto(n,i) = 0
        mostre "n e' composto!"
    i ← i + 2
mostre "n e' primo"
```

3.1.2 MDC

O algoritmo de Euclides para determinar o máximo divisor comum de dois números é descrito a seguir em pseudocódigo. Usamos "`a != b`" com significado "a é diferente de b".

```
leia a
leia b
se (a = 0):
    mostre b
    pare
se (b = 0):
    mostre a
    pare
enquanto (resto a / b  != 0):
    c ← resto a / b
    a ← b
    b ← c
mostre b
```


4 Tipos de dados

(esta seção está incompleta)
As próximas seções mostram a tradução deste exemplo para Java.

5 Pseudocódigo → Java

Este texto é uma pequena ajuda a quem estiver perdido tentando traduzir pseudocódigo para Java. O exemplo do MDC será desenvolvido, um pedaço de cada vez.
Os programas Java que você desenvolverá no laboratório terão a mesma estrutura:

```
public class Euclides {

  public static void main (String[] args) {

    // Seu programa comeca aqui

  }
}
```

- A linha "`import java.io.*;`" avisa o compilador que usaremos métodos de entrada e saída de dados;
- O programa está dentro de "algo" chamado "`public static void main(...)`". Este é sempre o ponto de partida de um programa Java. Aqui, "main" (algo como "principal") significa "o método principal do programa".
- O método `main`, por sua vez, está dentro de algo que se chama "`public class Euclides`". Em um programa Java, não pode haver código fora de alguma "classe" (veremos mais tarde o que isso significa).

5.1 Variáveis

Agora adicionamos ao programa as declarações de tipo:

```
import java.util.Scanner;

public class Euclides {

  public static void main (String[] args) {
    // Declaramos que a, b e c sao variaveis onde
    // guardaremos numeros inteiros; tambem aproveitamos e
    // colocamos zero em cada uma delas:
    int a = 0;
    int b = 0;
    int c = 0;
   }
}
```

5.2 Entrada de dados

O comando `leia` que usamos em pseudocódigo é traduzido, em Java, da seguinte forma:

```
Scanner sc = new Scanner(System.in);
a = sc.nextInt();
b = sc.nextInt();
```

A variável `sc` (do tipo `Scanner`) pode ser usada para ler dados do teclado:

- `sc.nextInt()` lê um número inteiro;
- `sc.nextDouble()` lê um número real (um `double`);
- `sc.next()` lê a próxima palavra;
- `sc.nextLine()` lê uma linha inteira.

5.3 Decisão

Agora usamos a estrutura de decisão ("`se`" em pseudocódigo) em Java:

```
if (a == 0) {
  System.out.println (b);
  System.exit(0);
}
if (b == 0) {
  System.out.println (a);
  System.exit(0);
}
```

Se a for igual a zero, mostre b e termine o programa. Se b for zero, mostre a e termine o programa.
Note:

- O operador de igualdade é `==`, enquanto `=` significa atribuição de valor! Assim,
  `a == b` significa "a é igual a b?", enquanto
  `a = b` significa "pegue o valor de b e armazene em a";
- `System.out.println(...)` mostra texto na tela (neste caso mostrará as variáveis b e a);
- `System.exit(0)` termina imediatamente o programa.

5.4 Repetição

```
while (a % b != 0) {
  c = a % b;
  a = b;
  b = c;
}
```

Enquanto o resto da divisão de a por b (`a % b`) for diferente de zero, faça:

```
c <- resto de a por b
a <- b
b <- c
```

A próxima seção mostra a tradução completa dos dois exemplos (MDC e teste de primalidade).

6 Os exemplos traduzidos para Java

6.1 n é primo?

A última versão do programa que testa primalidade se traduz em Java da seguinte forma:

```
import java.util.Scanner;

public class Primo {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);

        int n;
        int i;

        n = sc.nextInt();

        if (n == 2) {
            System.out.println("n e' primo!");
            System.exit(0);
        }
        i = 3;
        while (i < Math.sqrt(n)) {
            if (n % i == 0) {
                System.out.println("n e' composto!");
                System.exit(0);
            }
            i = i + 2;
        }
        System.out.println("n e' primo!");
    }
}
```


6.2 MDC

Versão sem janelas, com Scanner:

```
import java.util.Scanner;

public class Euclides {
    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int a = 0;
        int b = 0;
        int c = 0;
        a = sc.nextInt();
        b = sc.nextInt();
        if (a == 0) {
            System.out.println (b);
            System.exit(0);
        }
        if (b == 0) {
            System.out.println (a);
            System.exit(0);
        }
```

```
        while (a % b != 0) {
            c = a % b;
            a = b;
            b = c;
        }
        System.out.println (b);
    }
}
```

Versão com JOptionPane:

```
import javax.swing.JOptionPane;

public class Euclides {
    public static void main (String[] args) {
        String s;
        int a = 0;
        int b = 0;
        int c = 0;
        s = JOptionPane.showInputMessage(null,"Digite a");
        a = Integer.parseInt(s);
        s = JOptionPane.showInputMessage(null,"Digite b");
        b = Integer.parseInt(s);
        if (a == 0) {
            JOptionPane.showMessageDialog (b);
            System.exit(0);
        }
        if (b == 0) {
            JOptionPane.showMessageDialog (a);
            System.exit(0);
        }
        while (a % b != 0) {
            c = a % b;
            a = b;
            b = c;
        }
        JOptionPane.showMessageDialog(null, b);
    }
}
```


6.3 Exercícios

1. Faça um algoritmo para calcular juros compostos (o usuário entra o valor a financiar, o tempo e a taxa de juros, e o programa diz quanto será o montante total pago).

2. Faça um algoritmo que leia três números e diga qual deles é o maior.

3. Faça um algoritmo que leia os três lados de um triângulo e calcule sua área.

4. Faça um algoritmo que leia dois lados A e B de um triângulo, o ângulo entre A e B, e calcule:

   - O comprimento do lado C;

- A área do triângulo.

5. Faça um algoritmo para ler oito pontos, representando dois retêngulos, e depois dizer se os retângulos tem área em comum (se eles se "interceptam").

6. faça um algoritmo que leia dois pontos e dê a distância entre eles no plano.

7. Faça um algoritmo para ler posição e raio de duas circunferências e dizer se elas se interceptam.

8. Faça um algoritmo que leia:

   - Um ponto (a,b) (o centro de uma circunferência);
   - O raio da circunferência;
   - Outro ponto (x,y);

   e em seguida diga se o ponto faz parte da circunferência ou não.

9. Escreva um algoritmo que pergunte os coeficientes de uma equação do segundo grau e depois mostre suas raízes. O algoritmo deve verificar se a equação tem solução. Se houver uma só, mostre só uma. Se houver duas, mostre as duas.

10. Escreva um algoritmo que leia os três lados de um triângulo e o classifique em escaleno, isósceles ou equilátero.

11. Crie algoritmos que contem:

    - De 1 a 50
    - De 1 a 50, contando só os ímpares
    - De 50 a 1, contando só os ímpares
    - De 1 a N (onde N é determinado pelo usuário)
    - De 1 a N, contando só os números divisíveis por K (N e K são determinados pelo usuário).
    - De A a B (dois números determinados pelo usuário). Se $A < B$ a contagem é crescente; se $B < A$ é decrescente.

12. Faça um algoritmo que conte de 1 a N (dado pelo usuário) mostrando tanto o contador como seu quadrado. Por exemplo:

```
digite N: 5

1  1
2  4
3  9
4  16
5  25
```

13. Crie um algoritmo que leia uma sequência de números e pare quando dois números consecutivos forem iguais.

14. Modifique o exercício anterior para que depois de parar, o algoritmo mostre o maior número da sequência.

15. Modifique o algoritmo do exercício anterior para que ele também mostre, ao terminar, o menor elemento da sequência.

16. Faça um algoritmo que leia vários números, e que pare somente quando ler um número que é maior que 100.

17. Faça um algoritmo que pergunte ao usuário um valor a financiar, uma taxa de juros e o prazo máximo. Depois o programa deve mostrar uma tabela que mostra, para cada possibilidade de prazo, o montante total a ser pago. Por exemplo,

```
valor? 500
jutos? 0.3
prazo máximo? 6

prestacoes -- valor a pagar
---------------------------
1    500.0
2    515.0
3    530.45
4    546.3635
5    562.754405
6    579.63703715
```


18. Faça um algoritmo que leia vários números, e que pare somente quando ler um número que é maior que a soma de todos os números digitados antes dele.

19. Faça um algoritmo que leia dois números, K e N. O algoritmo deverá dizer quantos números de 2 até N são divisíveis por K.

20. Faça um algoritmo que leia dois números racionais: o usuário digirará x1, x2, y1, y2. Os dois números serão x1/x2 e y1/y2. O programa deve dizer se os dois números são iguais (por exemplo, $1/3 = 3/9$).

21. Faça um algoritmo que leia dois números complexos e os multiplique. (O usuário deverá informar Xr, Xi, Yr, Yi).

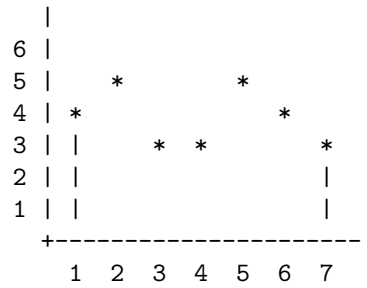
22. Faça um algoritmo que leia números complexos e os vá somando, até que o usuário entre Xr=0 e Yr=0.

23. Faça um algoritmo que leia dois números e verifique quantos números primos existem entre eles.

24. (Um pouco difícil) Fazer um algoritmo que:

    - Leia uma sequência de números; cada número é valor de uma função no ponto i, onde "i" vale 1 para o primeiro número, 2 para o segundo, e assim por diante.
    - Parar ao ler um número negativo
    - Calcular a área abaixo da linha determinada pelos pontos dados.

    Por exemplo, se o algoritmo ler 4, 5, 3, 3, 5, 4, 3, -1:



    A área que queremos é a do polígono (1,0) – (1,4) – (2,5) – (3,3) – (4,3) – (5,5) – (6,4) – (7,3) – (7,0) – (1,0)