Introdução à Programação
notas de aula – versão 1

1 de setembro de 2013

1 Variáveis, entrada e saída

Suponha que queiramos escrever um programa que calcule a área de um retângulo. 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.
- **Atribuição:** 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 criar 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 encontrarmos, 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).

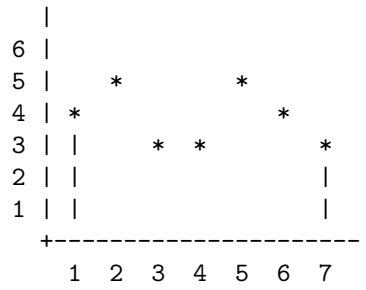
22. Faça um algoritmo que leia números complexos e os vá somando, até que o usuário entre Xr=0 e Yr=0.

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)

7 Um pouco além dos elementos básicos

Tipos: Definimos os tipos dados que cada variável pode guardar.
Como: No começo do programa, declaramos "`real a`", por exemplo, para determinar que a variável `a` é do tipo real.
Quais:

Valores	Tipo (pseudocódigo)	Tipo (Python)	Tipo (Java)
-1, 0, 1, 10	`int`	`int`	`int`
-3.5, 0.0, 2.3, 4.5	`real`	`float`	`double`
'a', 'b', 'x', '$', '*'	`char`	`str`	`char`
"abc", ".....", "x"	`string`	`str`	`String`
`veradeiro` e `falso`	`bool`	`bool`	`bool`

Laço "PARA": usar uma variável para contar de a até b.
Como:

```
para i em (1..10):
    comandos
```

Por exemplo, para imprimir os números de 1 a 10:

```
para i em (1..10):
    mostra i
```

Strings: podemos obter o i-ésimo caracter de uma string; também podemos descobrir o tamanho de uma string e obter um "pedaço de string".
Como: se uma variável `n` é do tipo string,

- `n[i]` é seu i-ésimo caracter;
- `tamanho(n)` é o número total de caracteres em `n`;
- `substring(n,i,j)` é o pedaço de `n` que vai do i-ésimo ao j-ésimo caracter.
- `concat(n,a)` é a concatenação da string `n` com a string `a` (ou seja, uma string maior, que começa com o conteúdo de `n` e termina com o conteúdo de `a`).

Por exemplo,

```
int meio, t
string n
leia n
leia k

primeiro <- n[0]   // primeiro caracter da string
t <- tamanho (n)
meio <- t/2        // meio
c <- n[meio]       // c e' o caracter que esta' no meio da string
n2 <- substring (n,0,meio) // n2 e' o pedaco da string que vai do comeco de n ate' a metade
n3 <- concat(n,k)     // n3 e' n seguida de k
```


7.1 Um exemplo completo

O programa a seguir lê uma string e separa as palavras em linhas:

```
Digite uma frase
> Ouviram do Ipiranga as margens plácidas
Ouviram
do
Ipiranga
as
margens
plácidas
```

```
string frase
int i, n

mostre "Digite uma frase"
leia frase
n <- tamanho (frase)

para i em (0..n-1):
    se (frase[i] = ' '):
        nova_linha
    senao:
        mostra frase[i]
```

O programa lê uma string, guarda na variável "frase", e depois usa `i` para contar de 0 até o tamanho da frase menos um. Durante a contagem, quando `frase[i]` for um espaço, vai para a linha debaixo. Quando não for, mostra o caracter `frase[i]`.
As próximas seções mostram a tradução deste exemplo para Python e Java.

7.2 O exemplo traduzido para Python

```
# coding: utf-8

import sys

frase = raw_input()
n = len(frase)

for i in range(0,n):
    if frase[i] == ' ':
        print
    else:
        sys.stdout.write(frase[i])
```

7.3 O exemplo traduzido para Java

```
import java.util.Scanner;

public class QuebraEmLinhas {
    public static void main (String[] args) {
        Scanner sc = new Scanner (System.in);
        String frase = "";
        int i;
        int n;
        frase = sc.nextLine();

        n = frase.length();
        for (i=0; i<n; i=i+1)
            if (frase.charAt(i) == ' ')
                System.out.println();
            else
                System.out.print(frase.charAt(i));
        System.out.println();
    }
}
```


7.4 Exercícios

1. Escreva um algoritmo que calcule o valor de pi. Há mais de uma maneira de calcular pi:

   $\pi = 4(1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13...)$
   $\pi^2/6 = 1 + 1/2^2 + 1/3^2 + 1/4^2 + ...$

2. Escreva um algoritmo que pergunte ao usuário dois números complexos e os multiplique.

3. Escreva um algoritmo que leia uma sequência de números complexos e apresente o seu produtório. O algoritmo primeiro perguntará ao usuário quantos números ele quer digitar; depois, o algoritmo lerá dois números reais de cada vez, representando as partes real e imaginária de cada número complexo. Quando todos os n complexos tiverem sido lidos (ou seja, 2n números reais lidos), o programa deve parar e apresentar o produtório.

4. Faça um algoritmo que pergunte o raio de uma esfera e calcule seu volume.

5. Faça um algoritmo que leia oito números, representando um triângulo e um ponto, e diga se o ponto está dentro do triângulo.

6. Faça um algoritmo que leia números até que um deles seja maior que a soma de todos os anteriores.

7. (Não é tão difícil quanto parece, mas se não conseguir não tem problema! Este exercício requer um pouco de paciência e inventividade SE VOCÊ NÃO CONSEGUIR JÁ NO COMEÇO DO CURSO, ISSO NÃO SIGNIFICA QUE NÃO ESTEJA INDO BEM NA DISCIPLINA.)

   Faça um programa que leia:

   - Um valor para x
   - Uma sequência de números representando um polinômio (cada dois números são coeficiente e expoente). O algoritmo deve parar quando ler um coeficiente igual a zero.

   e em seguida, calcule a derivada do polinômio no ponto lido.

   Veja o exemplo de como usaríamos o algoritmo:

```
/----
Ponto? 10
Entre com o polinômio.
2
4
4
1
-3
3
0
A derivada da função descrita por este polinômio no ponto x=10 é: 7104
\----
```

   Neste exemplo, demos para o algoritmo um ponto x=10 e o polinômio $2x^4 + 4x^1 - 3x^3$. Ele nos respondeu 7104, que é a derivada do polinômio no ponto x=10 (confira!).

8. Faça um programa que leia uma string e diga se ela é palíndroma.

9. Faça um programa que desenhe um círculo usando asteriscos:

```
    *****
  *********
 ***********
 ***********
  *********
    *****
```

10. Descreva um algoritmo que leia uma frase e um caracter, e em seguida conte o número de vezes que o caracter aparede na frase.

11. Modifique o algoritmo do exercício 2 para que ele também indique em que posição o caracter aparece pela primeira vez.

12. Escreva um algoritmo que leia uma frase e escreva apenas as primeiras palavras, antes de 72 colunas serem usadas

13. Use o que desenvolveu no exercício 4 para escrever um programa que leia um texto e o mostre, uma linha por vez, sem exceder 72 colunas.

14. Faça um algoritmo que conte o número de palavras em uma frase

15. Faça um algoritmo que divida uma frase em 3 partes de igual tamanho (ou quase igual, se o tamanho não for divisível por 3) e mostre uma parte por linha. Seu algoritmo não deve usar repetição (nem "enquanto" e nem "para"). Por exemplo:

    ```
    > The quick brown fox jumps over the lazy dog

    The quick brow
    n fox jumps ov
    er the lazy dog
    ```

16. Faça um algoritmo que inverta as palavras de uma frase (mas não a ordem das palavras!)

    Por exemplo:

    ```
    > homo sapiens non urinat in ventum

    omoh sneipas non taniru ni mutnev
    ```

    (Aliás, esta frase está gravada na entrada da praça Max Euwe em Amsterdam: `http://nl.wikipedia.org/wiki/Bestand:Amsterdam_Homo_sapiens_non_urinat_in_ventum.jpg`)

17. Traduza a frase do item anterior do Latim para o Português, e explique o motivo dela ter sido gravada no portal da praça.

18. Faça um algoritmo que leia uma frase e a mostre no formato de ampulheta. Por exemplo:

    ```
    Asdrubal trouxe o trombone
     sdrubal trouxe o trombon
      drubal trouxe o trombo
       rubal trouxe o tromb
    ```

```
    ubal trouxe o trom
     bal trouxe o tro
      al trouxe o tr
       l trouxe o t
         trouxe o
       l trouxe o t
      al trouxe o tr
     bal trouxe o tro
    ubal trouxe o trom
   rubal trouxe o tromb
  drubal trouxe o trombo
 sdrubal trouxe o trombon
Asdrubal trouxe o trombone
```

19. Faça um algoritmo que leia uma frase e a mostre no formato de cálice:

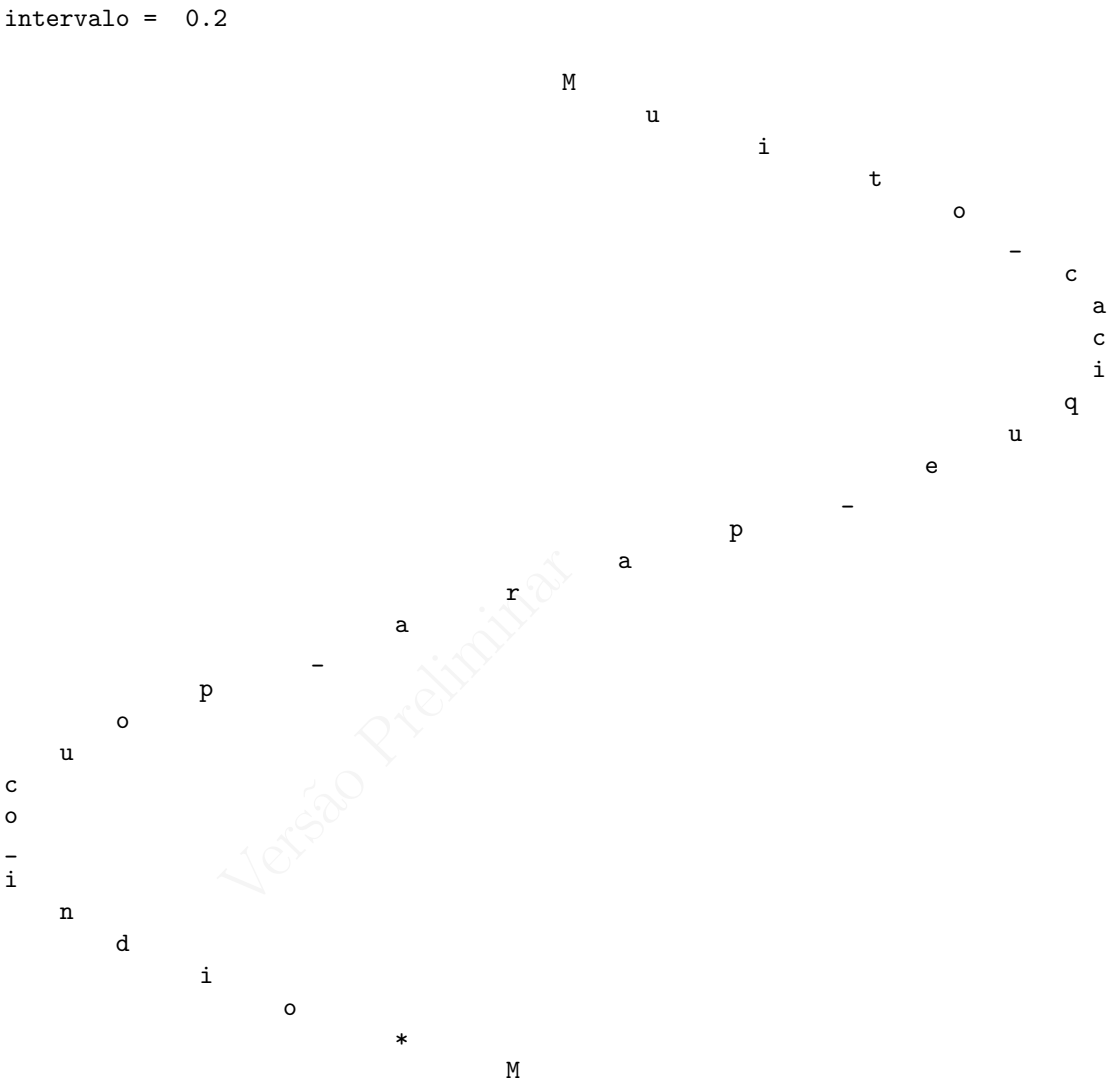
```
Asdrubal trouxe o trombone
 sdrubal trouxe o trombon
  drubal trouxe o trombo
   rubal trouxe o tromb
    ubal trouxe o trom
     bal trouxe o tro
      al trouxe o tr
       l trouxe o t
         trouxe o
         trouxe o
         trouxe o
         trouxe o
         trouxe o
         trouxe o
         trouxe o
```


20. (ESTE É UM DAQUELES QUE VOCÊ NÃO PRECISA FAZER SE NÃO CONSEGUIR) Faça um algoritmo que plote uma função (30sen(x)+30, por exemplo – como o da lsta 1) – mas ao invés de asteriscos, use letras de uma frase entrada pelo usuário.

    Por exemplo:

```
0         // aqui entrei a
6.28      // entrei b (aprox. 2*pi)
0.2       // quero que plote de 0.2 em 0.2
Muito cacique para pouco indio

a =  0.0
b =  6.28
```



8 Abstração: funções

ATENÇÃO: este texto é só um resumo! Você pode aprender mais sobre funções no livro-texto: leia o capítulo 6 (ele chama funções de "módulos").

Função: Um nome dado a uma sequência de comandos que, após serem executados, produzem um resultado. Pode-se passar argumentos para a função. Por exemplo, o "fatorial de n" é uma função com o argumento "n" e que retorna um valor (o fatorial de n).
Como: Escrevemos "`funcao`", e em seguida:

- Uma lista de argumentos entre parênteses:
- Dois pontos
- Declaração de variáveis locais (que só são usadas dentro da função);
- O código da função
- "`retorne X`" é usado para dizer que o valor da função é X.

Por exemplo, se definirmos a função fatorial:

```
funcao fatorial (n):
    int i, f
    f <- 1
    para i em (1..n):
        f <- f * i
    retorne f
```

Note que declaramos os tipos das variáveis i e f, que são usadas dentro da função. A declaração ficou dentro da função. Estas variáveis *não* podem ser usadas fora desta função.
Veja também a última linha: ela diz qual deve ser o *valor de retorno* da função: sempre que, no programa, usarmos "fatorial(x)", u
Podemos usá-la depois no programa:

```
// Este programa calcula combinacoes
leia n
leia k
c <- fatorial(n) / fatorial(k) * fatorial (n-k)
```

Note que se nao tivéssemos definido a função "fatorial", teríamos que escrever o código "para i ..." três vezes (porque usamos o fatorial três vezes no programa). Esta é uma finalidade da abstração: reusar código
Outro efeito da abstração é a simplificação do programa. A linha

```
c <- fatorial(n) / fatorial(k) * fatorial (n-k)
```

é fácil de ler e lembra a fórmula que usaríamos ($C_{n,k} = \frac{n!}{k!(n-k)!}$).
Uma função pode não retornar valor algum. Por exemplo,

```
funcao pontos (n):
   int i
   para i em (1..n):
       mostre "."
```

O que importa é que a função `pontos` mostra n pontos na tela; ela não precis retornar valor algum. Podemos usá-la assim:

```
mostre nome
pontos (10)
mostre telefone
```

O trecho de programa acima produzirá a saída:

```
Joao.........3322-4595
```

8.1 Um exemplo comleto

O exemplo a seguir lê uma sequência de n strings, depois lê um número k e diz ao usuário quantas combinações podem ser feitas com as n strings, tomadas k de cada vez.

```
int q, c, i
string s

funcao fatorial (int n):
    int i, f
    f <- 1
    para i em (1..n):
        f <- f * i
    retorne f

funcao combinacao (int n, int k):
    retorne fatorial(n) / fatorial(k) * fatorial (n-k)

mostre "Digite a lista de strings. Quando quiser parar, digite
        uma string vazia"
i <- 0
leia s
enquanto (s != ""):
    i <- i + 1
    leia s

mostre "Quantas strings de cada vez?"
leia q

c <- combinacao (i, q)
mostre "Ha' ", c, " combinacoes."
```


9 Exercícios

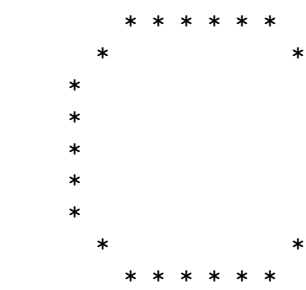
1. Desenvolva um algoritmo que leia um número e o transforme em binário. É fácil encontrar este algoritmo pronto na Internet; no entanto, lembre-se de que você nada ganharia com isso (esta lista não "vale nota", e ver o algoritmo pronto só elimina a sua oportunidade de aprender a desenvolvê-lo sozinho/a).

2. Refaça o algoritmo que calcula dias de vida usando funções.

3. (Fácil) Descreva um algoritmo que leia duas strings representando dois DNAs obtidos de um sequenciador automático. As posições da string podem conter A, C, G ou T (apenas). O algoritmo deve se recusar a analisar os DNAs se eles tiverem comprimento diferente. Se tiverem o mesmo comprimento, deve dizer quão similares eles são (verifique apenas se cada posição de um é igual à mesma posição do outro, e diga a porcentagem de posições similares).

4. Faça um programa que leia o raio e desenhe uma CIRCUNFERÊNCIA (e não um círculo).

```
Raio?
5
```



(Como dá pra notar, ela não ficará muito bonita. Com raio maior o desenho fica um pouco melhor...)

5. (ESTE É UM DAQUELES QUE VOCÊ NÃO PRECISA FAZER SE NÃO CONSEGUIR) Faça um algoritmo que leia duas strings representando fitas de DNA, mas desta vez algumas posições podem conter não apenas A, C, T ou G. Podem conter:

   - R = G A (purina)
   - Y = T C (pirimidina)
   - K = G T (ceto)
   - M = A C (amino)

   (Representando ambiguidade)

   O algoritmo deve informar:

   - Quanto os DNAs seriam similares se, cada vez que houver ambiguidade, considerarmos que as posições não casam;
   - Quanto os DNAs seriam similares se, cada vez que houver ambiguidade, considerarmos que as posições casam (se for possível).

   Por exemplo, 'R' com 'A' contaria como não em (i), mas como sim em (ii). 'R' com 'Y' contaria sempre como não.

10 Vetores

Vetores: regiões da memória onde armazenamos vários valores de um mesmo tipo. Por exemplo, `[ 2, 3, 5 ]` é um vetor de inteiros com três posições; `[ 1.0, 10.5, -0.1, 1.0 ]` é um vetor de reais com quatro posições.
Como: em nosso pseudocódigo, declaramos "`vetor(TIPO) nome`" ou "`(TIPO, TAMANHO) nome`". A i-ésima posição de um vetor v é `v[i]`.

```
int i
real soma
vetor(real, 5) a
para i em (0..4)
    leia a[i]
soma <- 0
para i em (0..tamanho(a)-1)
    soma <- soma + a[i]
mostre soma
```

10.1 Um exemplo completo

```
funcao media (vetor(real) v): real
    int i
    real soma <- 0
    para i em (0..tamanho(v)-1):
        soma <- soma + v[i]
    retorne soma / tamanho(v)

int i, n
mostre "Quantos elementos?"
leia n
vetor(real, n) a

para i em (0..n-1)
    leia a[i]

mostre "Media = ", media(a)
```


11 Em Java

Em Java, declaramos uma variável do tipo vetor da seguinte forma:

```
double[] a;
```

O código acima determina que `a` é um vetor de valores double, mas não informa ainda quantos valores o vetor comporta.
Para reservar espaço para N elementos em um vetor, usamos

```
double[] a = new double[n];
```

Em Java, um vetor pode ser passado por parâmetro assim:

```
double produto (double[] a, double[] b)  {
    ...
}
```

11.1 Exemplo completo em Java

Além deste exemplo, veja o produto escalar desenvolvido na aula de 9/11 (também disponível no site da disciplina).

```
import java.util.Scanner;
```

```
public class Media {

    public static double media (double[] v) {
        int i;
        double m = 0.0;
        int k = v.length;
        for (i=0; i<k; i++)
            m = m + v[i];
        return (m / k);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i;
        int n = sc.nextInt();

        double[] dados = new double[n];

        for (i=0; i<n; i++)
            dados[i] = sc.nextDouble();

        System.out.printf ("Media = %f\n", media(dados));
    }
}
```


12 Em Python

Novamente lembro que você não precisa aprender Python nesta disciplina! Esta seção é apenas para quem tiver vontade de conhecer um pouco mais.
Para usar vetores em Python, use o pacote numpy:

```
from numpy import *
```

Para criar um vetor inicialmente cheio de zeros, use

```
dados = zeros(n)
```

O acesso aos elementos do vetor é feito da mesma forma que no pseudocódigo e em Java ("`dados[i]`").

12.1 Exemplo completo em Python

```
# coding: utf-8

from numpy import *

def media (a):
    m = 0.0
    k = size(a)
    for i in range(0,k):
        m = m + a[i]
    return m / k

print "Quantos elementos?"
```

```
n = int(raw_input())

dados = zeros(n)

for i in range(0,n):
    dados[i] = float(raw_input())

print "Media = ", media(dados)
```

13 Exercícios

1. Faça um programa que calcule não só a média, mas a moda e a mediana de n elementos (o programa primeiro pergunta ao usuário quantos elementos quer digitar, depois os lê, e em seguida calcula média, moda e mediana).

2. Faça um programa que leia vários números, parando de ler quando o usuário digitar zero. Depois o programa deve informar quantos números eram reais, quantos eram inteiros e quantos eram naturais (lembre-se de que N é subconjunto de Z, que é subconjunto de R).

3. Faça um programa que leia uma frase (ou texto) e diga a média do tamanho das palavras do texto.

4. O programa da outra lista que reconhecia palíndromas na verdade tem um problema. Ele leva espaços em consideração:

   Refaça este algoritmo para que ele ignore espaços, e vírgulas e pontos:

   "Assim a aia vai a missa" é palíndroma (mas o algoritmo da solução anterior diria que não)

   (Não precisa se preocupar com "à", "á"e "a")

5. Faça algoritmos para calcular:

   - Soma de matrizes.
   - Determinate de matriz 3x3.
   - Multiplicação de matrizes.

14 Arquivos

Um arquivo pode ser visto como uma sequência de caracteres, como uma string; a diferença entre strings e arquivos é a maneira como são armazenados:

- Uma string é uma região da memória RAM do computador, usada (normalmente) dentro de um único programa. Como o acesso a qualquer posição de memória é simples e rápido, podemos selecionar qualquer posição da string, como já vimos:
  ```
  a <- "uma string"
  mostra a[4]  // mostrara' 's', porque a[4]='s'
  ```

- Um arquivo é uma sequência de caracteres que fica armazenada em disco ou outra forma de memória não-volátil (HD, DVD, pendrive etc). O acesso a muitas destas mídias é lento, especialmente quando se quer selecionar uma posição qualquer do arquivo. Por isso, normalmente usamos arquivos de maneira diferente de strings: usarems um arquivo como uma sequência de caracteres que lemos do início ao fim, sem retrocesso, ou como uma sequência de caracteres que gravamos, sem podermos apagar algo que já foi escrito.

Antes de usar um arquivo, precisamos "abrí-lo". Faremos isto com o comando `abre`:

```
arquivo arq
arq <- abre "dados.txt"
```

O trecho de código acima faz o seguinte:

-

Usaremos os mesmos comandos de leitura e escrita que já definimos para teclado e tela (mostra e leia).

```
arquivo arq
arq <- abre "meuarquivo.txt"
mostra (arq) "uma linha"
para i em (0..5):
    mostra (arq) i
fecha arq
```


Este agoritmo criará um arquivo `meuarquivo.txt`, e escreverá nele o conteúdo:

```
uma linha
1
2
3
4
5
```

O próximo exemplo usa leitura e gravação em arquivos:

```
arquivo entrada, saida
entrada <- abre "arquivo-entrada.txt"
saida <- abre "arquivo-saida.txt"

leia (entrada) n
vetor(real, n) vet

para i em (0..n):
    leia (entrada) vet[i]

para i em (0..n):
    mostra (saida) i, 2.0 * vet[i]
fecha entrada
fecha saida
```

Este programa lerá de um arquivo "arquivo-entrada.txt"um inteiro n. Depois, lerá n números reais e os armazenará em um vetor. Em seguida, escreverá em outro arquivo (arquivo-saida.txt) uma tabela com o índice (i) de cada valor, e o dobro do valor na posição i do vetor:
arquivo-entrada.txt:

```
3
1.0
2.5
4.0
```

arquivo-saida.txt:

```
0 2.0
1 5.0
2 8.0
```

14.1 Em Java

Em Java, declaramos variáveis do tipo arquivo:

```
File arqentra = new File("meuarquivo.txt");
File arqsai   = new File("outroarquivo.txt");
```

Mas precisamos também criar um Scanner se o arquivo for de saída (e depois poderemos usar `sc.next(), sc.nextLine(), sc.nextInt(), sc.nextDouble()`:

```
Scanner sc = new Scanner(arqentra);
```

Se o arquivo for de entrada, primeiro é necessário criar alguma variável (neste exemplo, "s") do tipo `FileOutputStream` a partir do arquivo, e depois uma outra ("sai", neste exemplo) do tipo `PrintStream`. Assim poderemos usar `sai.printf(), sai.print(), sai.println()`:

```
FileOutputStream s = new FileOutputStream (arqsai);
PrintStream sai = new PrintStream (s, true);
```

Para um exemplo completo em Java, veja o link na página do curso.

15 Exercícios

1. Descreva um algoritmo que abra dois arquivos, "arquivo1.txt"e "arquivo2.txt", e grave em um terceiro arquivo, "arquivo3.txt", o conteúdo dos outros arquivos, intercalando as linhas dos dois.

2. Mostre um algoritmo que leia um arquivo no formato:

   ```
   STRING NUM NUM NUM ...
   ```

Ou seja, cada linha tem uma string e uma sequência de números. A string é o nome de uma figura (quadrado, circulo, retangulo ou triangulo). Os números devem ser ou o raio (para o círculo) ou o comprimento de cada lado.

```
quadrado 2
circulo 3
retangulo 2 4.5
circulo 2
triangulo 3 4 1
```

Em seguida o programa deve descrever a figura com a maior área, dizndo, por exemplo,

"De todas as figuras, a maior área (28.274333882308138) é a do círculo de raio 3."

3. Faça um algoritmo que leia 10 números reais e depois verifique se eles foram digitados em ordem; se tiverem sido digitados fora da ordem, o programa deve dizer exatamente onde está o primeiro elemento fora da ordem. Por exemplo,

```
2 3.1 4 5.5 8 20
Os números estão ordenados

2 3.1 4 8 5.5 20
Os números não estão ordenados: veja o 5o elemento digitado, de
valor 5.5
```

4. Faça uma função que aceite como argumento um vetor de números inteiros e retorne um outro vetor, com os mesmos elementos, mas em ordem diferente: todos os ímpares à esquerda e todos os pares à direita.

5. Desenvolva um algoritmo que leia um texto e guarde em um vetor as frequências de cada letra do alfabeto no texto lido, e depois as mostre.