

Operadores e Estruturas de Decisão

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br

Aula de hoje...

- Operadores
 - Aritméticos (usados em contas)
 - Relacionais (usados em comparações numéricas)
 - Lógicos (usados em comparações lógicas)
 - De atribuição (armazenamento de valores em variáveis)
- Estruturas de decisão
 - *If...then*
 - *If...then...else*
 - *Switch...case*

Operadores aritméticos

| Operador | Exemplo | Prioridade |
|----------|-----------------------------|------------|
| (expr) | $(1 + 2) * 3 \rightarrow 9$ | 1 |
| var++ | i++ | 2 |
| var-- | j-- | 2 |
| ++var | ++i | 3 |
| --var | --j | 3 |
| +expr | +15 | 3 |
| -expr | $-(5+3) \rightarrow -8$ | 3 |
| * | $5 * 3 \rightarrow 15$ | 4 |
| / | $5 / 3 \rightarrow 1$ | 4 |
| % | $5 \% 3 \rightarrow 2$ | 4 |
| + | $5 + 3 \rightarrow 8$ | 5 |
| - | $5 - 3 \rightarrow 2$ | 5 |

Operadores aritméticos

- Operadores com a mesma prioridade (precedência)
 - Analisados da esquerda para a direita
- Aritmética de inteiros
 - Numerador e denominador inteiros
 - Resultado é somente a parte inteira da divisão
- Aritmética em modo misto
 - Numerador ou denominador real
 - Resultado fracionário

Exemplo

- Considerando

```
int x = 511;
```

```
double y = 9.2 - (++x - 14.0 / 7.0) + 14.0 * 0.1;
```

- Resolução de y

$$y = 9.2 - (512 - 14.0 / 7.0) + 14.0 * 0.1$$

$$y = 9.2 - (512 - 2.0) + 14.0 * 0.1$$

$$y = 9.2 - 510.0 + 14.0 * 0.1$$

$$y = 9.2 - 510.0 + 1.4$$

$$y = - 500.8 + 1.4$$

$$y = - 499.4$$

Type Casting

- Em algumas situações o programador deseja transformar o tipo de uma expressão
 - Para isso, basta preceder a expressão por “(tipo)”
 - *Type Casting* tem prioridade superior a *, / e %
- Passar um real para inteiro
 - float a = 5.1f;
 - int x = (int) a;
 - x vale 5
- Passar inteiro para real
 - int b = 5; int c = 2;
 - float y = (float)b/c;
 - y vale 2.5

Exemplo

- Considerando

```
int x = (int) (3.3 / ( 5/2 ) - 5);
```

```
int y = (int) 3.3 / ( 5/2 ) - 5;
```

- Resolução de x

```
x = (int) (3.3 / ( 2 ) - 5)
```

```
x = (int) (1.65 - 5)
```

```
x = (int) (- 3.35)
```

```
x = - 3
```

- Resolução de y

```
y = (int) 3.3 / ( 2 ) - 5
```

```
y = 3 / 2 - 5
```

```
y = 1 - 5
```

```
y = - 4
```

Funções matemáticas

- A classe Math
 - Contém constantes (PI e número de Euler)
 - Contém diversas funções matemáticas
 - Não é necessário importar o seu pacote, java.lang, pois está sempre disponível
- Constantes
 - $\text{Math.PI} = 3.141592653589793$
 - $\text{Math.E} = 2.718281828459045$

Funções matemáticas

| Método | Descrição | Exemplo |
|-------------------------------------|-----------------------------------|-------------------------------------|
| <code>Math.abs(expr)</code> | Valor absoluto | <code>Math.abs(-5.3) → 5.3</code> |
| <code>Math.round(expr)</code> | Arredonda um número | <code>Math.round(5.3) → 5</code> |
| <code>Math.ceil(expr)</code> | Arredonda para cima | <code>Math.ceil(5.3) → 6.0</code> |
| <code>Math.floor(expr)</code> | Arredonda para baixo | <code>Math.floor(5.3) → 5.0</code> |
| <code>Math.max(expr1, expr2)</code> | Maior de dois números | <code>Math.max(5, 6) → 6</code> |
| <code>Math.min(expr1, expr2)</code> | Menor de dois números | <code>Math.min(5, 6) → 5</code> |
| <code>Math.sqrt(expr)</code> | Raiz quadrada | <code>Math.sqrt(4) → 2.0</code> |
| <code>Math.pow(expr1, expr2)</code> | Potência | <code>Math.pow(2, 3) → 8.0</code> |
| <code>Math.log10(expr)</code> | Logaritmo na base 10 | <code>Math.log10(100) → 2.0</code> |
| <code>Math.log(expr)</code> | Logaritmo natural (base E) | <code>Math.log(Math.E) → 1.0</code> |
| <code>Math.exp(expr)</code> | Exponencial (e^{expr}) | <code>Math.exp(0) → 1.0</code> |

Funções matemáticas

| Função | Descrição | Exemplo |
|-----------------------------------|------------------------------|--|
| <code>Math.sin(expr)</code> | Seno | <code>Math.sin(0) → 0.0</code> |
| <code>Math.asin(expr)</code> | Arco seno | <code>Math.asin(1) → 1.5707963267948966</code> |
| <code>Math.cos(expr)</code> | Cosseno | <code>Math.cos(0) → 1.0</code> |
| <code>Math.acos(expr)</code> | Arco cosseno | <code>Math.acos(-1) → 3.141592653589793</code> |
| <code>Math.tan(expr)</code> | Tangente | <code>Math.tan(1) → 1.5574077246549023</code> |
| <code>Math.atan(expr)</code> | Arco tangente | <code>Math.atan(1) → 0.7853981633974483</code> |
| <code>Math.toDegrees(expr)</code> | Converte radianos para graus | <code>Math.toDegrees(Math.PI) → 180.0</code> |
| <code>Math.toRadians(expr)</code> | Converte graus para radianos | <code>Math.toRadians(180) → 3.141592653589793</code> |

- Funções trigonométricas trabalham com radiano
- Existem algumas outras funções menos usadas

Números aleatórios

- Algumas aplicações necessitam que o computador sorteie um número
 - Método `Math.random()`
 - Gera número pseudo aleatório entre 0 e 1
- A partir desse número, é possível gerar números em outros intervalos
 - $\text{inicio} + (\text{fim} - \text{inicio}) * \text{Math.random}()$

Exemplo

- **Número entre 0 e 1**

```
System.out.println(Math.random());
```

- **Número entre 5 e 6**

```
System.out.println(5 + Math.random());
```

- **Número entre 0 e 10**

```
System.out.println(Math.random() * 10);
```

- **Número entre 50 e 70**

```
System.out.println(50 + Math.random() * 20);
```

Operadores relacionais

| Operador | Exemplo | Prioridade |
|----------------------------------|-------------------------------------|------------|
| $\text{expr1} < \text{expr2}$ | $5 < 3 \rightarrow \text{false}$ | 1 |
| $\text{expr1} \leq \text{expr2}$ | $5 \leq 3 \rightarrow \text{false}$ | 1 |
| $\text{expr1} > \text{expr2}$ | $5 > 3 \rightarrow \text{true}$ | 1 |
| $\text{expr1} \geq \text{expr2}$ | $5 \geq 3 \rightarrow \text{true}$ | 1 |
| $\text{expr1} == \text{expr2}$ | $5 == 3 \rightarrow \text{false}$ | 2 |
| $\text{expr1} != \text{expr2}$ | $5 != 3 \rightarrow \text{true}$ | 2 |

- Prioridade sempre inferior aos operadores aritméticos
- Sempre têm **números como operandos**
- Sempre têm **resultado booleano**

Operadores lógicos

| Operador | Exemplo | Prioridade |
|----------------|-----------------------|------------|
| ! expr | !true → false | 1 |
| expr1 & expr2 | true & false → false | 2 |
| expr1 ^ expr2 | true ^ true → false | 3 |
| expr1 expr2 | true true → true | 4 |
| expr1 && expr2 | true && false → false | 5 |
| expr1 expr2 | True false → true | 6 |

- Prioridade sempre inferior aos operadores relacionais
- Exceção para “!”, com prioridade superior a *, / e %
- Sempre têm **booleanos como operandos**
- Sempre têm **resultado booleano**

Tabela verdade

| a | b | !a | a & b a && b | a ^ b | a b a b |
|-------|-------|-------|-----------------|-------|-----------------|
| true | true | false | true | false | true |
| true | false | false | false | true | true |
| false | true | true | false | true | true |
| false | false | true | false | false | false |

Ou e E otimizados

- `&` e `&&`, assim como `|` e `||` têm a mesma tabela verdade, mas
 - `&` e `|` sempre avaliam os dois operandos
 - `&&` e `||` só avaliam o segundo operando se o primeiro não for conclusivo
- Diferença quando o segundo operando altera valores

`i = 10`

Caso 1: `(i > 3) | (++i < 2) → true` (com `i` valendo 11)

Caso 2: `(i > 3) || (++i < 2) → true` (com `i` valendo 10)

Operadores de atribuição

| Operador | Exemplo |
|-------------|--------------------------|
| var = expr | x = 10 + 5 |
| var += expr | x += 5 → x = x + 5 |
| var -= expr | x -= 5 → x = x - 5 |
| var *= expr | x *= 5 → x = x * 5 |
| var /= expr | x /= 5 → x = x / 5 |
| var %= expr | x %= 5 → x = x % 5 |
| var &= expr | x &= true → x = x & true |
| var ^= expr | x ^= true → x = x ^ true |
| var = expr | x = true → x = x true |

Exemplo

- Considerando

```
double x = 10.0;
```

```
double y = -2.0;
```

```
double z = 5.0;
```

```
boolean w = x * y < z / x || x / y > z * x && z * y < x;
```

- Resolução de w

```
10.0 * -2.0 < 5.0 / 10.0 || 10.0 / -2.0 > 5.0 * 10.0 &&  
5.0 * -2.0 < 10.0
```

```
-20.0 < 0.5 || -5.0 > 50.0 && -10.0 < 10.0
```

```
true || false && true
```

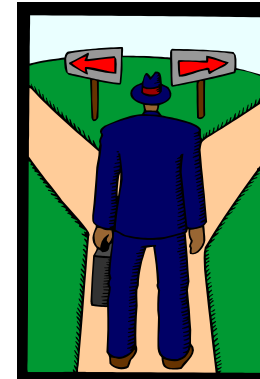
```
true || false
```

```
true
```

Decisão

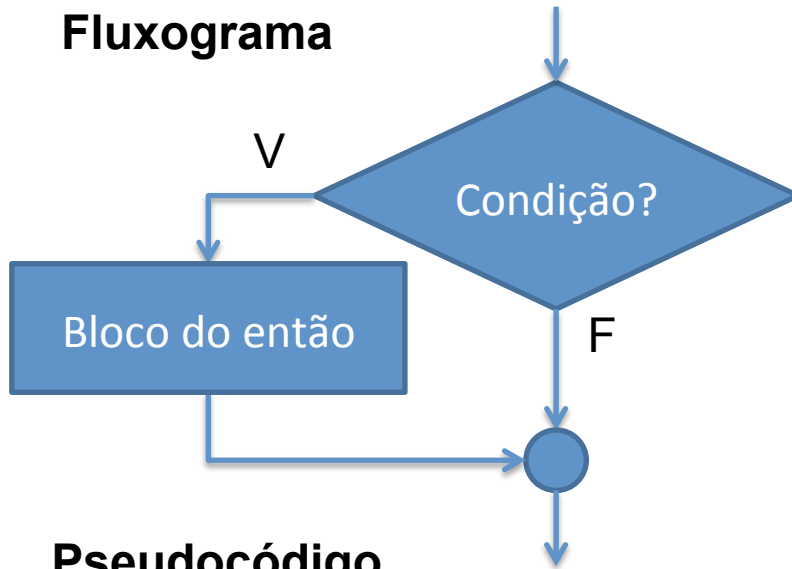
Mecanismos de decisão:

- *If...then*
 - Executa algo somente quando uma condição é verdadeira
- *If...then...else*
 - Bifurca a execução do código em função de uma condição
- *Switch...case*
 - Executa múltiplos trechos de código em função do valor de uma expressão



Decisão do tipo *if...then*

Fluxograma



Pseudocódigo

```

...
Se CONDIÇÃO então
  INSTRUÇÃO 1
  INSTRUÇÃO 2
  ...
  INSTRUÇÃO N
...
  
```

Java

```

...
if (CONDIÇÃO)
  INSTRUÇÃO;
...
  
```

Ou

```

...
if (CONDIÇÃO) {
  INSTRUÇÃO 1;
  INSTRUÇÃO 2;
  ...
  INSTRUÇÃO N;
}
...
  
```

Decisão do tipo *if...then*

- Executa o bloco de instruções somente se a condição for verdadeira
- A condição é uma expressão booleana que pode fazer uso de quaisquer operadores
- A condição deve sempre estar entre parênteses
- Pode omitir { e } caso execute somente uma instrução
 - As variáveis declaradas dentro de um bloco (entre { e }) só valem nesse bloco ou subblocos

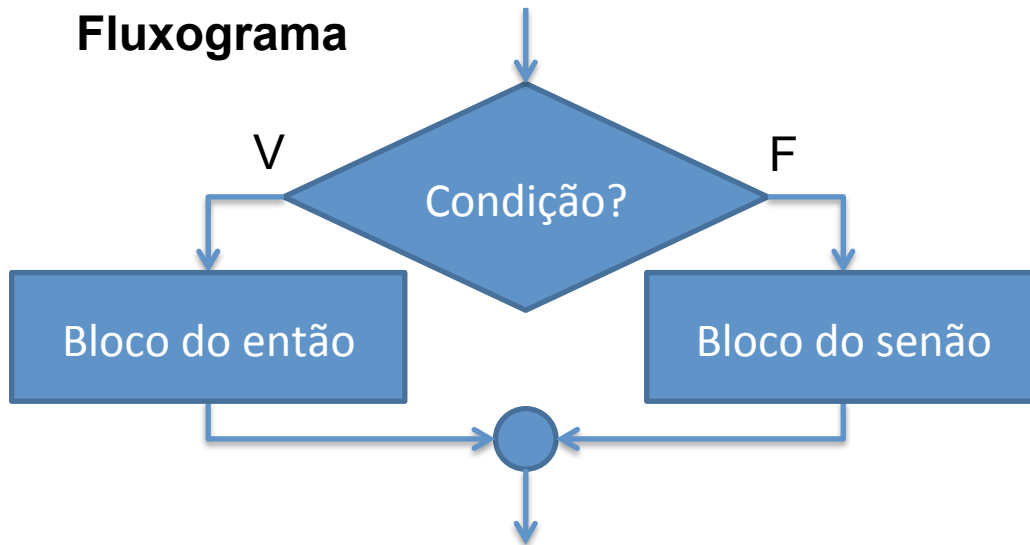
Exemplo de *if...then*

- Programa para informar o valor absoluto de um número:

```
import java.util.Scanner;
public class Absoluto {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um número: ");
        double numero = teclado.nextDouble();
        if (numero < 0)
            numero = -numero;
        System.out.println("Valor absoluto: " + numero);
    }
}
```

Decisão do tipo *if...then...else*

Fluxograma



Pseudocódigo

```

...
Se CONDIÇÃO então
  INSTRUÇÃO 1
  INSTRUÇÃO 2
  ...
  INSTRUÇÃO N
Senão
  INSTRUÇÃO 1
  INSTRUÇÃO 2
  ...
  INSTRUÇÃO N
...
  
```

Decisão do tipo *if..then..else*

Java

```

...
if (CONDIÇÃO)
    INSTRUÇÃO;
else
    INSTRUÇÃO;
...

```

Ou

```

...
if (CONDIÇÃO) {
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
} else {
    INSTRUÇÃO 1;
    INSTRUÇÃO 2;
    ...
    INSTRUÇÃO N;
}
...

```


Decisão do tipo *if...then...else*

- Executa um ou o outro bloco de instruções em função da condição ser verdadeira ou falsa
- Valem as mesmas regras para *if...then*
- Qualquer combinação de instrução individual ou em bloco é aceita no *then* e no *else*
- Podem ser aninhados com outras estruturas do tipo *if...then...else*

Exemplo de *if...then...else*

- Programa para informar se um número é par ou ímpar:

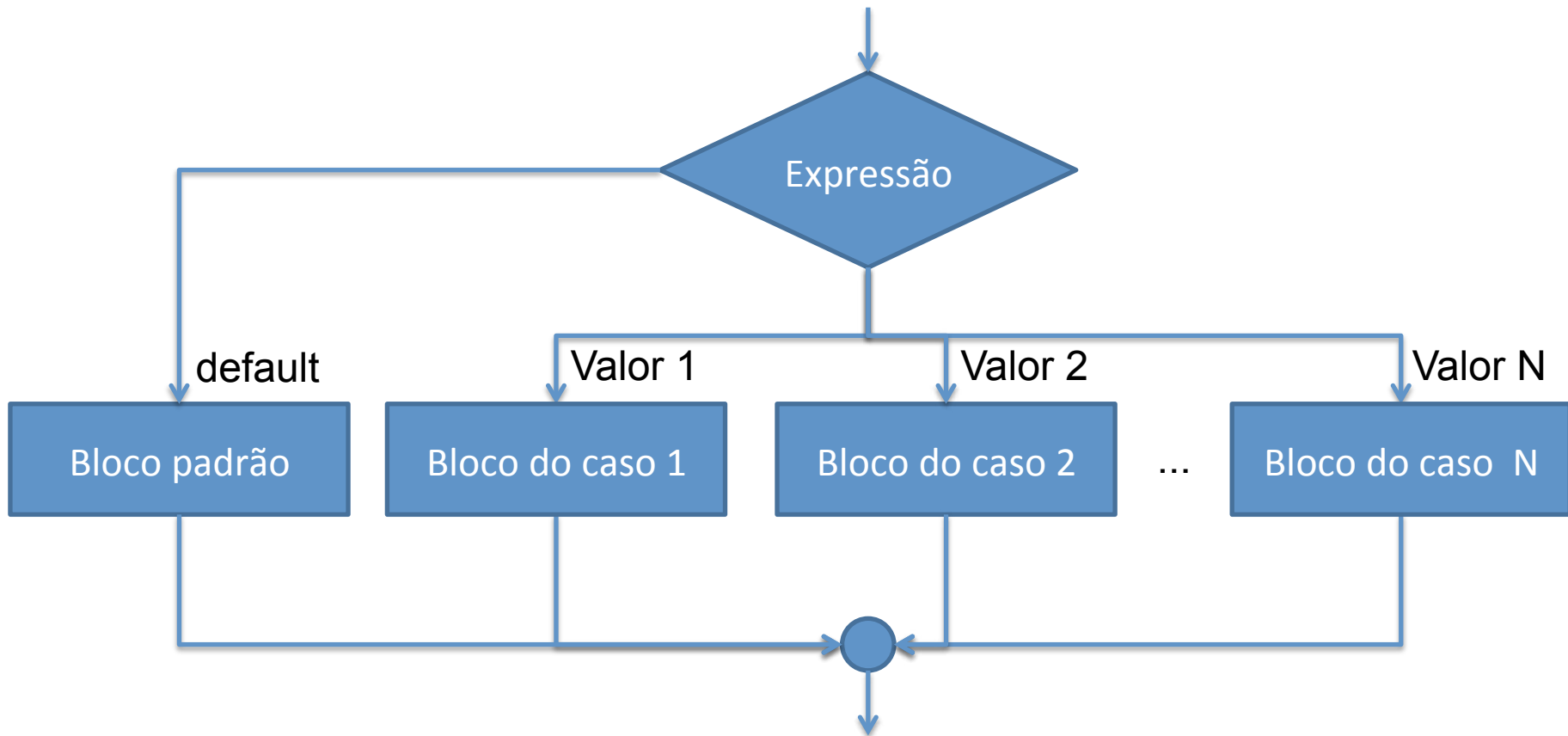
```
import java.util.Scanner;
public class Paridade {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um número: ");
        int numero = teclado.nextInt();
        if (numero % 2 == 0)
            System.out.println("O número é par!");
        else
            System.out.println("O número é ímpar!");
    }
}
```

Exemplo de *if* aninhado

- Programa para informar o número de dias de um mês:

```
import java.util.Scanner;
public class DiasMes {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um mês (1 a 12): ");
        byte mes = teclado.nextByte();
        if ((mes==1) || (mes==3) || (mes==5) || (mes==7) || (mes==8) || (mes==10) || (mes==12))
            System.out.println("Esse mês tem 31 dias!");
        else if ((mes==4) || (mes==6) || (mes==9) || (mes==11))
            System.out.println("Esse mês tem 30 dias!");
        else {
            System.out.print("Entre com o ano (4 dígitos): ");
            short ano = teclado.nextShort();
            if ((ano%400==0) || ((ano%4==0) && (ano%100!=0)))
                System.out.println("Esse mês tem 29 dias!");
            else
                System.out.println("Esse mês tem 28 dias!");
        }
    }
}
```

Decisão do tipo *switch...case*



Decisão do tipo *switch...case*

Java

```

...
switch (EXPRESSÃO) {
    case VALOR 1: INSTRUÇÃO 1;
                ...
                break;
    case VALOR 2: INSTRUÇÃO 1;
                ...
                break;

    ...
    case VALOR N: INSTRUÇÃO 1;
                ...
                break;
    default: INSTRUÇÃO 1;
            ...
}
...

```

Decisão do tipo *switch...case*

- Aceita expressões dos tipos byte, short, int, char e String
- É equivalente a *if* aninhado
 - Escolher o que tem melhor legibilidade
 - *Switch...case* é baseado em valores individuais
 - *If...then...else* pode ser baseado em intervalo de valores
- O uso de *break* é fundamental para a quebra do fluxo
 - A clausula *case* delimita somente o ponto de entrada
 - O programa executará todas as linhas seguintes até encontrar um *break* ou terminar o *switch*

Exemplo de *switch...case*

```
import java.util.Scanner;
public class DiasMes {
    public static void main(String[] args) {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Entre com um mês (1 a 12): ");
        byte mes = teclado.nextByte();
        switch (mes) {
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                System.out.println("Esse mês tem 31 dias!");
                break;
            case 4: case 6: case 9: case 11:
                System.out.println("Esse mês tem 30 dias!");
                break;
```



Exemplo de *switch...case*



case 2:

```

System.out.print("Entre com o ano (4 dígitos): ");
short ano = teclado.nextShort();
if ((ano%400==0) || ((ano%4==0) && (ano%100!=0)))
    System.out.println("Esse mês tem 29 dias!");
else
    System.out.println("Esse mês tem 28 dias!");
break;
default:
    System.out.println("Mês inválido!");
}
}
}

```


Exercícios

- Faça um programa que calcule o IMC de uma pessoa ($\text{IMC} = \text{massa em kg} / \text{altura em metros}^2$) e informe a sua classificação segundo a tabela a seguir, obtida na Wikipédia

| IMC | Classificação |
|-------------|------------------------------|
| < 18,5 | Abaixo do Peso |
| 18,6 – 24,9 | Saudável |
| 25,0 – 29,9 | Peso em excesso |
| 30,0 – 34,9 | Obesidade Grau I |
| 35,0 – 39,9 | Obesidade Grau II (severa) |
| ≥ 40,0 | Obesidade Grau III (mórbida) |

Exercícios

- Faça um programa que leia três coordenadas num espaço 2D e indique se formam um triângulo, juntamente com o seu tipo (equilátero, isósceles e escaleno)
 - Equilátero: todos os lados iguais
 - Isósceles: dois lados iguais
 - Escaleno: todos os lados diferentes

Exercícios

- Faça um programa que leia um número inteiro de 5 dígitos e indique se ele é palíndromo
 - Um número palíndromo é aquele que se lido da esquerda para a direita ou da direita para a esquerda possui o mesmo valor (ex.: 15451)

Exercícios

- Faça um programa que leia um número inteiro entre 0 e 9999 e escreva o seu valor por extenso

Operadores e Estruturas de Decisão

Leonardo Gresta Paulino Murta
leomurta@ic.uff.br