Computação I

Professora: Myriam Regattieri Delgado - CPGEI sala 10

O material de apoio está disponível em:

http://www.dainf.cefetpr.br/~myriam/PastaWeb/CompI/

Ementa:

- **Fase 1: Algoritmos :**
  Variáveis, operadores, comandos básicos e estruturas de controle
- **Fase 2: Linguagem de Programação**
  Fase 2.1:
  - Variáveis e Operadores
  - Comandos Básicos
  - Entrada-saída
  - Estruturas de Seleção
  - Estruturas de Repetição

  Fase 2.2
  - Variáveis Indexadas: vetores e matrizes
  - Registros
  - Funções

Bibliografia

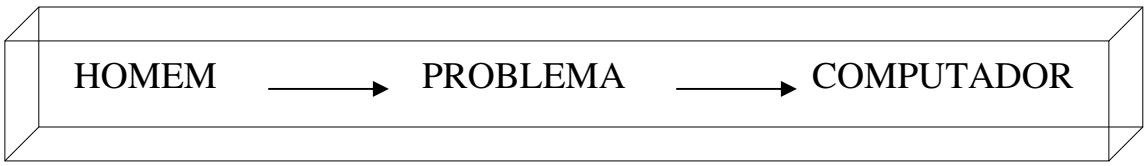
Algoritmos:

- Lógica de Programação – André Forbellone & Henri Eberspächer - Makron Books, 1993
- Algoritmos e Estrutura de Dados – Guimarães e Lages Livros Técnicos e Científicos Editora, 1985

1. Introdução

O que é Programação ?

“ É muito mais do que se sentar na frente de um microcomputador e digitar um programa”



HOMEM

- Criatividade
- Imaginação
- Inferências

PROBLEMA

- **Análise do Problema**
- **Implementação do Algoritmo**
- **Linguagem de Alto Nível**
- **Linguagem Assembler**
- **Linguagem de Máquina**

COMPUTADOR

Execução organizada de instruções em linguagem de máquina

Implementação do Algoritmo

início
receber um valor via teclado
calcular o seu dobro
mostrar o resultado na tela
fim

Independe do computador e da linguagem

↓

Programa em Linguagem C

```
#include <stdio.h>
Int  main()
{       int valor, dobro;
        scanf("%d",&valor);
        dobro = 2*valor;
        printf("%d",dobro);
        return 0;
}
```

Independe (quase sempre) do computador

↓

Linguagem Assembler

```
INICIO:   IN      A,(TECLADO)
          LD      (VALOR),A
          LD      A, 2
          MUL     A, (VALOR)
          LD      (DOBRO), A
          LD      A, (DOBRO)
          OUT     A, (VIDEO)
          JMP     INICIO
```

Cada comando corresponde a um comando em alto nível (não otimizado)

↓

Linguagem de Máquina

```
03
00
30
02
03
01
31
```

Programa pronto para ser executado

1.1 Análise do problema

- **Etapa muito importante**
- **Observação do problema real pra elaborar uma solução independente da linguagem ou computador**

Exemplo: Problema do transporte de carga

Um homem precisa atravessar um rio com um barco que possui capacidade de carregar apenas ele mesmo e mais uma de suas três cargas:
1 lobo
1 bode
1 maço de alfafa

O que o homem deve fazer para conseguir atravessar o rio sem perder suas cargas ?

Análise:
Quais são as restrições do problema ?
1 carga de cada vez
o lobo é predador do bode (não podem ficar juntos)
o bode é predador de alfafa (não podem ficar juntos)

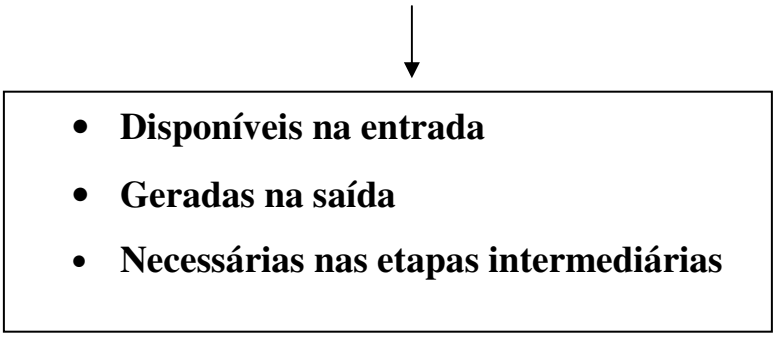
Solução:

1.2 Implementação do Algoritmo (Importante)

1.2.1 Estrutura de Dados

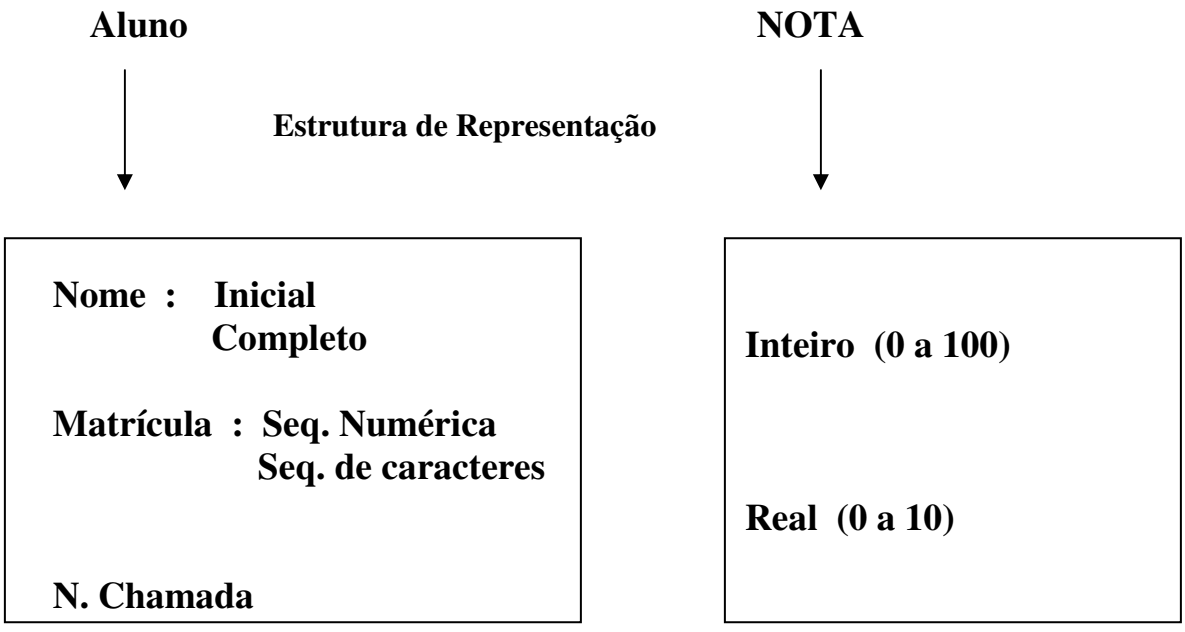
A estrutura do algoritmo depende da estrutura de dados utilizada

As | **estruturas de dados** | **representam as** | **informações** | **do problema**



Exemplo: Problema de ordenação das notas de alunos

Dados:



1.2 Implementação do Algoritmo

1.2.2 Aspectos Estáticos e Dinâmicos

Aspectos Estáticos : Um algoritmo é basicamente um texto com instruções

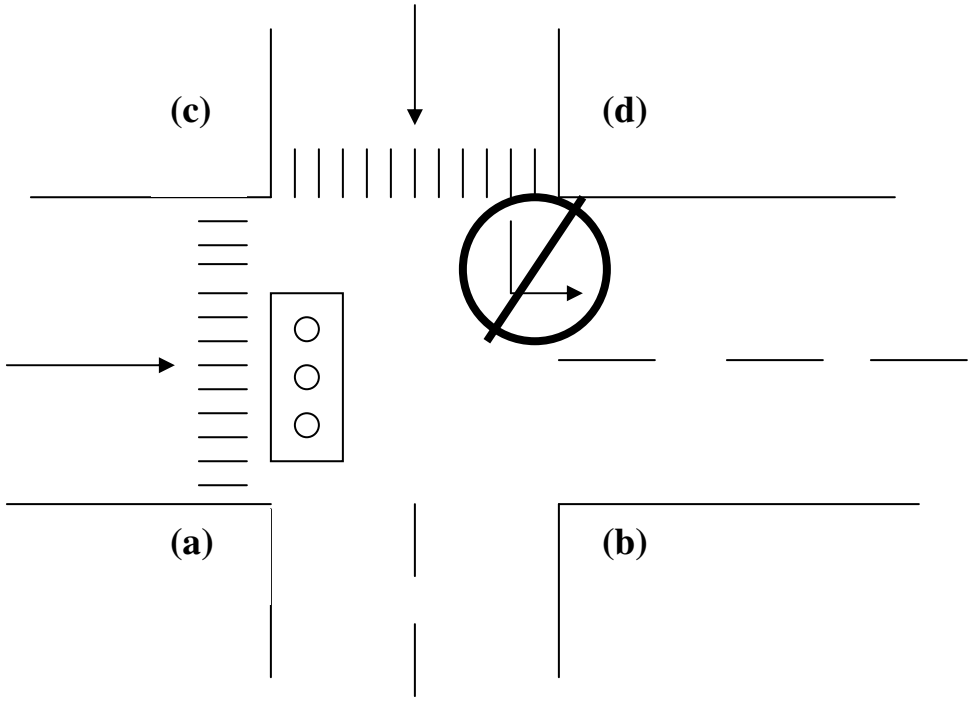
Exemplo: Algoritmo para trocar uma lâmpada

- **Pegue uma escada**
- **Posicione-a embaixo da lâmpada**
- **Busque uma lâmpada nova**
- **Suba na escada**
- **Retire a lâmpada velha**
- **Coloque a lâmpada nova**

Aspectos Dinâmicos: Efeito no tempo dado um conjunto inicial de valores

Exemplo: Problema do Sinal de Trânsito

"Especifique as ações necessárias para que uma pessoa que está no local (a) possa ir, em segurança, até o local (b). Para tal, observe o sentido do tráfego na encruzilhada, a faixa, o semáforo e a placa de trânsito."



2.Algoritmos

O que é um algoritmo?

- **É uma seqüência de passos que visam atingir um objetivo bem definido (solucionar o problema).**

- **Os algoritmos determinísticos apresentam um padrão de comportamento (mesmas entradas sempre produzem mesmas saídas)**

- **Um algoritmo normalmente envolve:**

1. **Estrutura seqüencial**
2. **Estrutura de seleção**
3. **Estrutura de repetição**

Exemplo: Alterar o algoritmo para troca da lâmpada para os seguintes casos:

- **A lâmpada que está no teto não apresenta defeito**
- **A nova lâmpada apresenta defeito**

Solução:

- **Pegue uma escada**

- **Posicione-a embaixo da lâmpada**

- **Busque uma lâmpada nova**

- **Suba na escada**

- **Retire a lâmpada velha**

- **Coloque a lâmpada nova**

2.1 Regras para Construção de Algoritmos

- **Colocar o máximo de comentários possíveis**
  - **identificados por { ... } ou * ... *\ ou **
- **Escolher nomes significativos para as variáveis**
- **Grifar as palavras chaves**
- **ALINHAR os comandos (identação)**

Exemplo:

```
início {algoritmo que determina qual o menor valor de um conjunto de valores
fornecidos pelo usuário (condição de parada = -1)}
|
|     inteiro: MenorValor; {contém a cada instante o menor valor lido}
|              ValorLido; {recebe o valor lido (fornecido pelo usuário)}
|     leia (ValorLido);
|     MenorValor ← ValorLido;
|     enquanto ValorLido ≠ -1 faça
|     |     se ValorLido < MenorValor então
|     |     |     MenorValor ← ValorLido;
|     |     fim se;
|     |     leia(ValorLido);
|     fim enquanto;
|     imprima("O menor valor é ",MenorValor);
fim
```

Expressões:

As expressões envolvem VARIÁVEIS, CONSTANTES E OPERADORES

$$V = \frac{4\pi R^2}{3}$$ → `V = (4*PI*R**2)/3`

4 e 3 são constantes, PI e R são variáveis

* ** / são operadores

2.2 Declaração de Variáveis

- **Toda variável deve ser declarada.**
- **A declaração define o tipo da variável a ser utilizada pelo algoritmo**

Tipos Básicos: inteiro, real, caracter e lógico

Exemplo 2.2.1:

```
inteiro: X1;
real: A,B,PI;
caracter: FRASE, NOME;
lógico: DECISAO;
```

2.3 Comandos Básicos

2.3.1 Comando de Atribuição : ←

NomeDaVariável ← Expressão;

Exemplo

```
X1 ← 10;
B  ← 2.5;
```

2.3.2 Operadores e funções

1. **Funções matemáticas**
   raiz(x); exp(x);
   abs(x) → valor absoluto (módulo) de x
   int(x) → parte inteira de x

sen(x), cos(x), tg(x), arctg(x), etc...

2. **Operadores aritméticos:**

** ou ^	
*	/
+	-
res	Resto da divisão

3. **Operadores relacionais: =, ≠,>,≥,<,≤**

4. **Operadores lógicos:** 4.1 **conjunção: e**
   4.2 **disjunção: ou**
   4.3 **negação: não**
   4.4 **disjunção exclusiva: x-ou**

Tabela Verdade

Def: Tabela verdade é o conjunto de todas a possíveis combinações entre os valores de duas ou mais variáveis lógicas e um conjunto de operadores lógicos.

A	B	A e B
F	F	F
F	V	F
V	F	F
V	V	V

A	B	A ou B
F	F	F
F	V	V
V	F	V
V	V	V

A	não A
F	V
V	F

Exemplos :

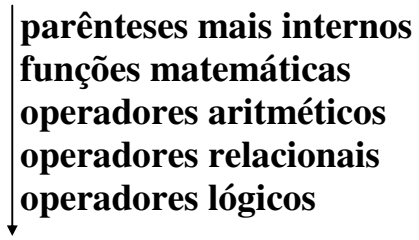
a) (2 < 5) **e** ((15/3) = 5)
(2 < 5) **e** (5 = 5)
V **e** V
V

b) **(não verdadeiro) ou ((3**2)/3) < (15-(35res7)))**
F ou 9/3 < 15 - 0
F ou 3 < 15
F ou V
V

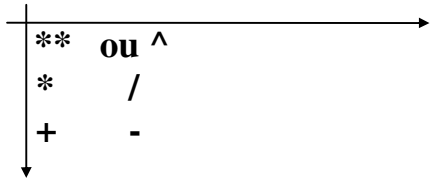
Exemplos de Atribuição e Operadores:

```
X1 ← X1 + 1;
DECISAO ← falso;
PI ← 3.14159;
LETRA ← ´M´;
A ← raiz(B);
RESTO ← X1  res 3;
```

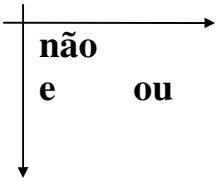
Prioridades entre os Operadores :



operadores aritméticos



operadores lógicos



Exemplo :

a)

```
não  2**3  < 4**2  ou abs(int(15/-2))       <  10
não  2**3  < 4**2  ou abs(int(-7.5))        <  10
não  2**3  < 4**2  ou abs(-7)          <  10
não  2**3  < 4**2  ou   7   <  10
não   8   <   16   ou   7   <  10
não   V             ou         V
      F             ou         V
                    V
```

2.4 Comandos de Entrada-Saída

Comando de Entrada: leia(...);

Comando de Saída: imprima(...);

Exemplo:

```
início {algoritmo que lê duas variáveis e imprime a soma delas}
        real: A,B,SOMA;
        imprima("Entre com os valores de A e B");
        leia(A,B);
        imprima("O Valor da primeira variável é",A);
        imprima("O Valor da segunda variável é",B);
        SOMA ← A+B;
        imprima(" O Valor da soma A+B é",SOMA);
fim
```

2.5 Comandos Básicos de Controle

- **Estrutura seqüencial**
- **Estrutura de seleção**
- **Estrutura de repetição**

Identifique no algoritmo que calcula o menor valor de um conjunto de valores fornecidos pelo usuário (seção 2.1), as estruturas seqüenciais, de seleção e repetição.

3.Algoritmos – Estruturas de Seleção

- **Seleção Simples**
- **Seleção Composta**
- **Seleção Encadeada**
- **Seleção de Múltipla Escolha**

3.1 Seleção Simples

```
se <condição for V> então

        comando1;
        {ou bloco de comandos};
fim se
```

Exemplo

```
leia(A);
se A> 0 então

        imprima("OK");
        A ← A +1;
fim se
```

3.2 Seleção Composta

```
se <condição1 for V> então
        comando1;
        {ou bloco de comandos};
senão
        comando2;
        {ou bloco de comandos};
fim se
```

Exemplo

```
leia(A);
se A> 0 então
        imprima("OK");
        A ← A +1;
senão
        imprima("Erro");
fim se
```

3.3 Seleção Encadeada Homogênea

- **Encadeamento Simples e Composto**

3.3.1 Seleção Encadeada Simples

```
se  <condição1 for  V> então
    se  <condição2 for V> então
        :
          se <condiçãoN for V> então
              comando1; {ou bloco de comandos};
          fim se
    fim se
fim se
```

⇓ **equivalente**

```
se <condição1 for V>  e  <condição 2 for V>  e  ...  e  <condição N for V> então
      comando1;   {ou bloco de comandos};
fim se
```

- **Nos dois casos o comando(s) só é (são) executado(s) se todas as condições forem verdadeiras**

3.3.2 Seleção Encadeada Composta

```
se  <condição1 for V> então
    comando1;  {ou bloco de comandos};
senão
        se <condição2 for V> então
              :
        senão
              se <condiçãoN for V> então
                    comandoN;  {ou bloco de comandos}
              senão   comandoM;  {ou bloco de comandos}
              fim se
        fim se
fim se
```

3.4 Seleção Encadeada Heterogênea

Não há um padrão lógico definido:

```
se <Condição1 forV> então
    se  <Condição2 for V> então
        {inicio do bloco verdade A}
        C1;
        ...
        Cn;
    fim se
senão
    se  <Condição3 forV> então
        {inicio do bloco verdade B}
        C1;
        ...
        Cn;
    senão
        se  <Condição4 for V> então
            se  <Condição 5=V> então
                ComandoX;
            fim se
        senão
            Comando Y;
        fim se
    fim se
fim se
```

Condição1	Condição2	Condição3	Condição4	Condição5	Ação
V	V	--	--	--	Bloco A
F	--	V	--	--	Bloco B
F	--	F	V	V	Comando X
F	--	F	F	--	Comando Y

3.5 Seleção de Múltipla Escolha

escolha X
caso E1: Comando1; {ou bloco de comandos}
caso E2: Comando2; {ou bloco de comandos}
:
caso EN; ComandoN; {ou bloco de comandos}
caso contrário: ComandoM; {ou bloco de comandos}
fim escolha

Exemplos:

```
início
        caractere: op;
        leia(op);
        escolha(op)
                caso 'c' : imprima("copiando arquivo");
                                {comandos necessários para copiar arquivo}
                caso 'a': imprima("apagando arquivo");
                                {comandos necessários para apagar arquivo}
                caso 'd': imprima("criando diretório");
                                {comandos necessários para criar diretório}
                caso 'f': imprima("formatando disquete");
                                {comandos necessários para formatar disquete}
                caso contrário: imprima("saindo do programa");
                                 {comandos para sair do programa}
        fim escolha
fim
```

```
início
        inteiro: A,B;
        leia(A);
        escolha(A)
                caso 10:  B = A**2;
                caso 20: imprima("Erro");
        fim escolha
fim
```

Seleção Encadeada X Múltipla Escolha

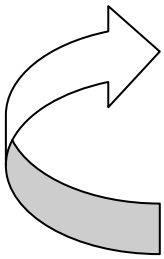
- Toda múltipla escolha pode ser transformada numa seleção encadeada
- O inverso nem sempre é verdadeiro

```
início
        inteiro: A,B;
        leia(A);
        escolha(A)
                caso 10:  B = A**2;
                caso 20: B = A**3;
                caso contrário: imprima("erro");
        fim escolha
fim
```

```
início
        inteiro: A,B;
        leia(A);
        se A = 10 então
                B = A**2;
        senão se A = 20 então
                        B = A**3;
                        senão imprima("erro");
                fim se
        fim se
fim
```

4.Algoritmos – Estruturas de Repetição

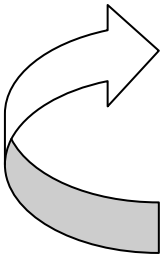
4.2 Repetição com teste no início : enquanto



```
enquanto <condição for V> faça
        comando1;
            :
        comando N;
fim enquanto
```

- A condição representa a condição para executar o laço

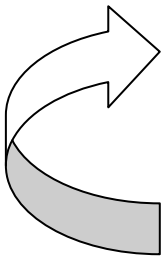
4.3 Repetição com teste no fim : repita



```
repita
        comando1;
            :
        comando N;
até <condição for V>
```

- A condição representa a condição de parar o laço

4.1 Repetição com variável de controle incremental: para



```
para Variável  de  ValInic  até  ValFin  passo P  faça
        comando1;
            :
        comando N;
fim para
```

- Os valores ValInic, ValFin e P definem quantas vezes o laço será executado

Exemplo :
Os 3 algoritmos a seguir imprimem a tabuada do número 5. Cada um utiliza uma estrutura de repetição diferente. Siga passo a passo cada um deles e defina qual seria a saída nos 3 casos :

- **utilizando para**

```
inicio
        inteiro : CON;
        para CON de 1 até 10 passo 1 faça
                imprima (CON, " vezes 5 = ",CON*5);
        fim para
fim
```

- **utilizando enquanto**

```
inicio
        inteiro : CON;
        CON←1;
        enquanto  CON <= 10
                imprima (CON, " vezes 5 = ",CON*5);
                CON←CON+1;
        fim enquanto
fim
```

- **utilizando repita**

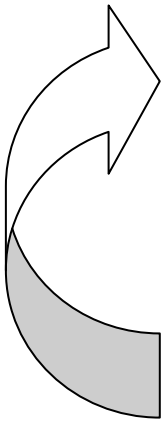
```
inicio
        inteiro : CON;
        CON←1;
        repita
                imprima (CON, " vezes 5 = ",CON*5);
                CON←CON+1;
        até CON > 10
fim
```

Exemplos de Algoritmos com Estruturas de Repetição

início {Algoritmo para o Cálculo da Média Final de uma Classe de 40 Alunos Repetição com **variável de controle** }

```
        real:P1, P2, P3, MF;
        inteiro: CONT;
        caractere : NOME

        para CONT=1 até CONT<=40 passo 1 faça
                leia(NOME);
                leia(P1,P2,P3);
                MF←(P1+P2+P3)/3.0;
                se MF>=7.0 então
                        imprima(NOME, "Aprovado");
                        imprima("Media Final = ",MF);
                fim se
        fim para
fim
```

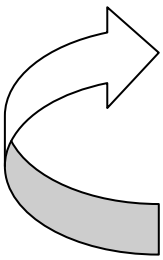


---

inicio {Algoritmo para o calculo da média de valores lidos até que se digite o valor –1 (Repetição com **teste no início do laço**)}

```
        inteiro: VALOR,ACUMVAL,TOTLID;
        real : MEDIAVAL;

        ACUMVAL ← 0;
        TOTLID← 0;
        leia(VALOR);
        enquanto VALOR≠ -1 faça
                ACUMVAL←ACUMVAL+VALOR;
                TOTLID←TOTLID + 1;
                leia(VALOR);
        fim enquanto
        se TOTLID≠ 0 então
                MEDIAVAL←ACUMVAL/TOTLID;
                imprima(MEDIAVAL,TOTLID);
        fim se
fim
```



```
inicio {Algoritmo para o cálculo de vinhos branco, tinto e rose fornecidos pelo usuário
(Repetição com teste no fim do laço )}
        caractere : TV;  { tipo do vinho}
        inteiro :    CONV, {contador de vinhos}
                     CT,   {contador de tinto}
                     CB,   {contador de branco}
                     CR;   {contador de rose}
        real :  PT, PB, PR  { porcentagem de tinto, branco e rose}
        CONV ← 0;          CT ← 0;
        CB ← 0;            CR ← 0;

        repita
                imprima("(T)into");
                imprima("(B)ranco");
                imprima("(R)ose");
                imprima("(F)im");
                imprima("Entre com a Opcao");
                leia(TV);
                CONV ← CONV + 1;
                escolha(TV)
                        caso 'T' : CT  ← CT + 1;
                        caso 'B' : CB ← CB + 1;
                        caso 'R' : CR  ← CR + 1;
                        caso 'F': CONV ← CONV - 1;
                fim escolha
        até TV = 'F'

        se CONV > 0 então

                        PT ← (CT *100) / CONV;
                        PB ← (CB *100) / CONV;
                        PR ← (CR *100) / CONV;
                        imprima ("Porcentagem de Tintos = ", PT);
                        imprima ("Porcentagem de Brancos = ", PB);
                        imprima ("Porcentagem de Roses = ", PR);
        senão
                        imprima("Nenhum tipo foi fornecido");
        fim se
fim
```

IMPORTANTE

- **O laço para-faça é mais apropriado quando já se sabe (através do usuário ou definindo-se o total através de uma constante) o número de iterações (repetições).**
- **O laço enquanto-faça é mais apropriando quando a condição de parada é inesperada , isto é, não se sabe previamente o total de repetições. (Ex. Condição de parada : entrada de um valor -1)**
- **O laço repita-até também é mais apropriado quando a condição de parada é inesperada, mas o laço precisa ser executado ao menos 1 vez.**

Conversão de Estruturas de Repetição :

- **toda estrutura para-faça pode ser convertida em enquanto-faça ou repita-até quando o número de iterações é previamente conhecido**

- **Quando o número de iterações não é previamente conhecido, não é possível converter enquanto-faça ou repita-até em para-faça**

- **toda estrutura enquanto-faça pode ser convertida em repita-até e vice-versa utilizando-se a negação da condição de parada.**