Algoritmos e Programação

Aula 06

Representação de Algoritmos utilizando Pseudocódigo

Estrutura Sequencial

aula passada...

- **Tipos primitivos**: Inteiro, Real, Caracter, Lógico.
- **Constantes**: **não sofrem alteração** no tempo ou durante a execução do algoritmo.
- **Variáveis**: **Sofrem alteração** no tempo e execução.
- **Identificadores**: **nomes** para **variáveis**.
- **Declaração de variáveis**: processo de **reservar** as **variáveis** que serão utilizadas no algoritmo.

aula passada...

- **Operadores Aritméticos**: **Operações básicas** da **matemática**. Opera tipos primitivos Inteiro e Real.
  - **Operadores**: $+, -, *, /, \%, div, pow, sqrt$
  - **Resultado** é um valor **inteiro** ou **real**.

- **Operadores Relacionais**: Realiza **comparações** entre dois **valores** do **mesmo tipo primitivo**.
  - **Operadores**: $>, <, >=, <=, ==, != $
  - **Resultado** é um valor **lógico**

- **Operadores Lógicos**: Efetua **avaliações** lógicas entre **valores lógicos**.
  - **Operadores**: $não, e, ou$
  - **Resultado** é um valor **lógico**.

aula passada...

- **Atribuição**: Processo de **associar** um **valor** a uma **variável**.
  - Exemplo: $X = A + B * 10;$
  - OBS: Tipo de dado deve ser compatível com a variável (mesmo tipo primitivo).

- **Entrada de dados**: comando **leia** (atribuir valores a variáveis).
  - Exemplo: $leia(A, B, C);$

- **Saída de dados**: comando **escreva** (aquilo que é mostrado/escrito para o usuário).
  - Exemplo: $escreva(\text{"Resultado} = \text{"}, x);$

Estruturas de Controle

Estruturas de Controle

- Para criar algoritmos utilizamos os conceitos de entrada e saída de dados, variáveis, constantes, atribuições, expressões lógicas, relacionais e aritméticas e comandos (que traduzem esses conceitos).

Estruturas de Controle

- Para criar algoritmos utilizamos os conceitos de entrada e saída de dados, variáveis, constantes, atribuições, expressões lógicas, relacionais e aritméticas e comandos (que traduzem esses conceitos).
- **Fluxo de Execução** é o modo pelo qual os comandos são executados, de maneira que o conjunto de ações executadas seja viável (ou seja, que resolva o problema).

Estruturas de Controle

- Para criar algoritmos utilizamos os conceitos de entrada e saída de dados, variáveis, constantes, atribuições, expressões lógicas, relacionais e aritméticas e comandos (que traduzem esses conceitos).
- **Fluxo de Execução** é o modo pelo qual os comandos são executados, de maneira que o conjunto de ações executadas seja viável (ou seja, que resolva o problema).
- Para solucionar problemas, iremos utilizar estruturas básicas de controle de fluxo de execução: **Sequênciação**, **Seleção**, **Repetição** e a combinação delas.

Algoritmos em Pseudocódigo

Elementos de um algoritmo em pseudocódigo

Pseudocódigo

Todo Algoritmo tem um nome

Algoritmo: exemplo

Declaração:

Algoritmo: NOME

Atenção:

Não pode usar espaço no NOME

Ex: **Algoritmo:** Calcula Media

Pode usar "_" como separador

Exemplos Válidos:

Algoritmo: CalculaMedia

Algoritmo: Calcula_Media

Pseudocódigo

Todo Algoritmo tem início e fim

Algoritmo: exemplo

fim-algoritmo

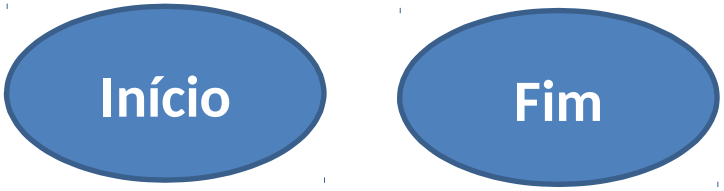
Declaração:

Algoritmo, fim-algoritmo

Atenção:

Todas as operações são realizadas entre o **Algoritmo** e **fim-algoritmo**

Equivale as elipses do fluxograma:



Pseudocódigo

Declaração de Variáveis

```
Algoritmo: exemplo
    tipo var1, var2;
    tipo var3;
    tipo var4, var5, var6;
    tipo var7;

fim-algoritmo
```

Variável:

Valor que é alterado/manipulado internamente pelo algoritmo

Declaração:

tipo IDENTIFICADOR;

- Deve iniciar por caracter alfabético
- Pode ser seguido por mais caracteres alfabéticos ou numéricos
- Não devem ser usados caracteres especiais

Pseudocódigo

Declaração de Variáveis

Algoritmo: exemplo

inteiro Codigo, Contador;
real Preco, Soma;
caracter Nome;
lógico Disponivel;

fim-algoritmo

4 tipos possíveis:

INTEIRO

Números inteiros: 1,2,3,...

REAL

Números reais: 1.3, 22.7,...

CARACTER

Texto entre aspas

Exemplo: “Digite sua senha”

Pode ter espaço entre as aspas

LÓGICO

Indica valor **verdadeiro** ou **falso**

(**V** ou **F**)

Pseudocódigo

Declaração de Constantes

```
Algoritmo: exemplo
    inteiro Codigo, Contador;
    real Preco, Soma;
    caracter Nome;
    lógico Disponivel;
    const inteiro IdadeCerta = 33;
    const caracter NomeCerto = “Juca”;
    const lógico GeneroCerto = V;
    const real Pi = 3.1415;

fim-algoritmo
```

Constante

Valor que não se altera durante a execução do algoritmo.

Não muda nunca.

É fixo!

Declaração:

const tipo IDENTIFICADOR = valorFixo;

Exemplos:

const real pi = 3.1415;

const caracter turno = “noite”;

const lógico resultado = V;

Pseudocódigo

Operações de I/O (*input* e *output*)

Algoritmo: exemplo

inteiro Codigo, Contador;
real Preco, Soma;
caracter Nome;
lógico Disponivel;
const inteiro IdadeCerta = 33;
const caracter NomeCerto = “Juca”;
const lógico GeneroCerto = V;
const real Pi = 3.1415;

OPERAÇÕES DE I/O

fim-algoritmo

Operações de I/O

Comandos utilizados para a realização de leitura de dados e escrita de dados.

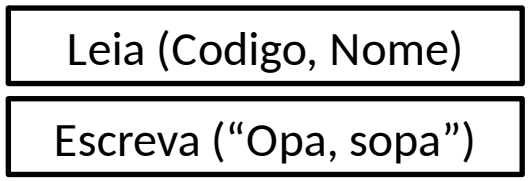
Exemplos:

leia(Codigo, Nome**);**

escreva(“Opa, sopa!”**);**

escreva(NomeCerto**);**

Equivalem aos retângulos no Fluxograma:



Pseudocódigo

Comandos de Atribuição

Algoritmo: exemplo

inteiro Codigo, Contador;
real Preco, Soma;
caracter Nome;
lógico Disponivel;
const inteiro IdadeCerta = 33;
const caracter NomeCerto = “Juca”;
const lógico GeneroCerto = V;
const real Pi = 3.1415;

OPERAÇÕES DE I/O
ATRIBUIÇÕES

fim-algoritmo

Comandos de Atribuição

É o processo de **atribuir** um **valor** a uma **variável** já **declarada**.

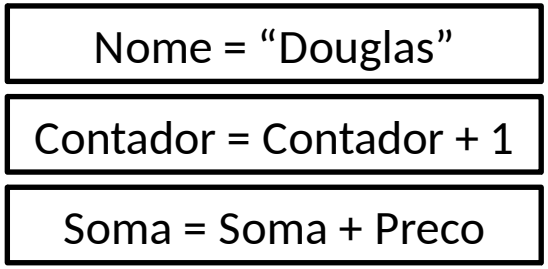
Exemplos:

Nome = “Douglas”;

Contador = Contador + 1;

Soma = Soma + Preco;

Equivale ao retângulos no Fluxograma:



Pseudocódigo

Testes (assunto da próxima aula)

Algoritmo: exemplo

inteiro Codigo, Contador;
real Preco, Soma;
caracter Nome;
lógico Disponivel;
const inteiro IdadeCerta = 33;
const caracter NomeCerto = “Juca”;
const lógico GeneroCerto = V;
const real Pi = 3.1415;

OPERAÇÕES DE I/O
ATRIBUIÇÕES
TESTES

fim-algoritmo

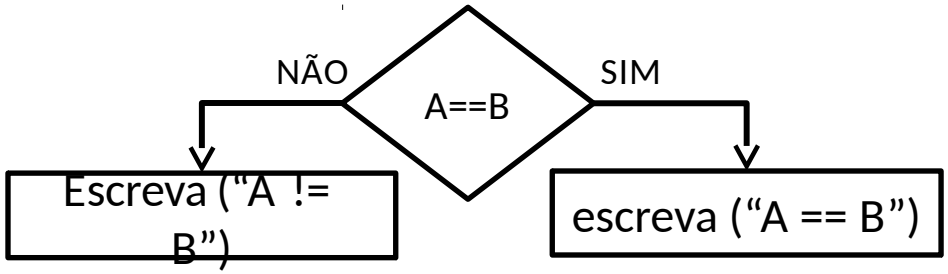
Testes (assunto da próxima aula)

São utilizados para testar uma condição

Exemplo:

SE (A == B) **ENTÃO**
escreva(“A == B”);
SENÃO
escreva(“A != B”);
FIM-SE

Equivale aos losangos



Pseudocódigo

Repetições (assunto de um futuro próximo)

Algoritmo: exemplo
inteiro Codigo, Contador;
real Preco, Soma;
caracter Nome;
lógico Disponivel;
const inteiro IdadeCerta = 33;
const caracter NomeCerto = “Juca”;
const lógico GeneroCerto = V;
const real Pi = 3.1415;

OPERAÇÕES DE I/O
ATRIBUIÇÕES
TESTES
REPETIÇÕES

fim-algoritmo

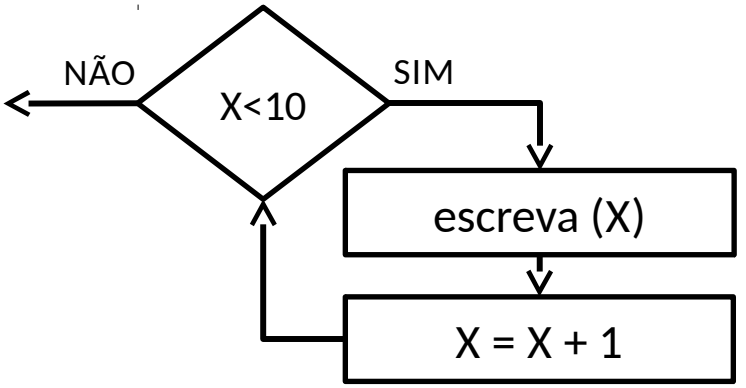
Repetições (assunto de um futuro próximo)

São utilizados para realizar laços de repetição.

Exemplo:

ENQUANTO (X < 10) **FAÇA**
escreva(X);
X = X + 1;
FIM-ENQUANTO

Equivalente em fluxogramas



Pseudocódigo

- Modelo básico que usaremos para descrever algoritmos:

```
Algoritmo: Modelo_Geral //identificação do algoritmo e do início do bloco correspondente ao algoritmo

    //declaração de variáveis/constantes

    //corpo do algoritmo
    ação 1;
    ação 2;
    ação 3;
    ...
    ação n;

Fim_algoritmo // fim do algoritmo
```

- As ações são apenas: Operações de I/O (leia e escreva), atribuições, testes e repetições.
  - Não usaremos NADA além disso.

Algoritmos em Estrutura Sequencial

Estrutura Sequencial

- **Estrutura Sequencial**: em um algoritmo, o conjunto de ações primitivas será executado em uma sequência linear de cima para baixo e da esquerda para a direita.

Estrutura Sequencial

- **Estrutura Sequencial**: em um algoritmo, o conjunto de ações primitivas será executado em uma sequência linear de cima para baixo e da esquerda para a direita.
  - Ações são seguidas por um ponto-e-vírgula (;)
    - objetivo de separar uma ação da outra
    - e auxiliar a organização sequencial das ações
  - Ao encontrar um (;) devemos executar o próximo comando da sequência.

Estrutura Sequencial

- **Estrutura Sequencial**: em um algoritmo, o conjunto de ações primitivas será executado em uma sequência linear de cima para baixo e da esquerda para a direita.
  - Ações são seguidas por um ponto-e-vírgula (;)
    - objetivo de separar uma ação da outra
    - e auxiliar a organização sequencial das ações
  - Ao encontrar um (;) devemos executar o próximo comando da sequência.
  - Para **algoritmos** em **Estrutura Sequencial** **utilizaremos apenas**: Operações de **I/O** (**leia** e **escreva**) e comandos de **atribuição**!

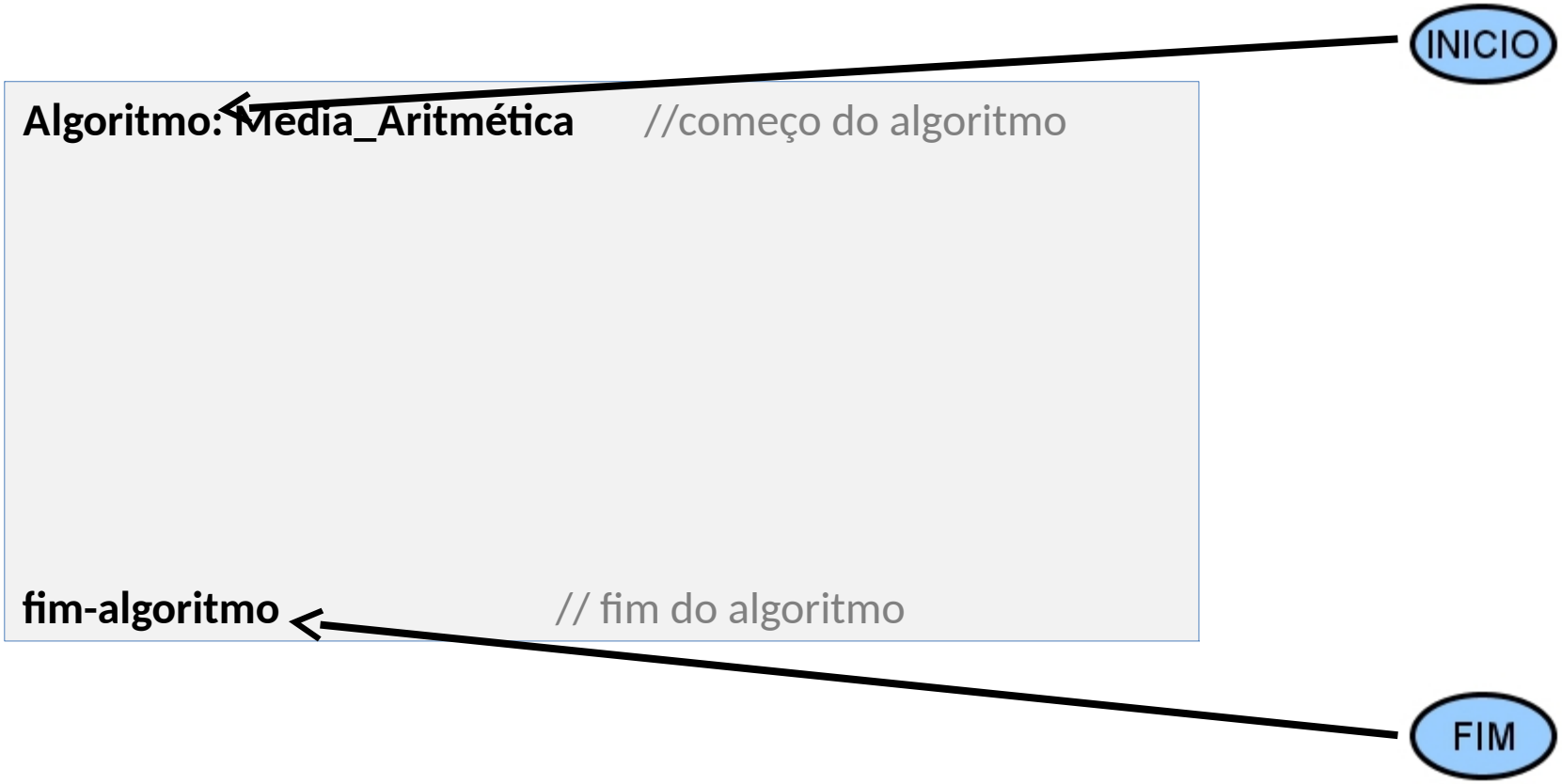
Exemplo

1. Construa um algoritmo que calcule a média aritmética entre quatro notas bimestrais quaisquer fornecidas pelo usuário.

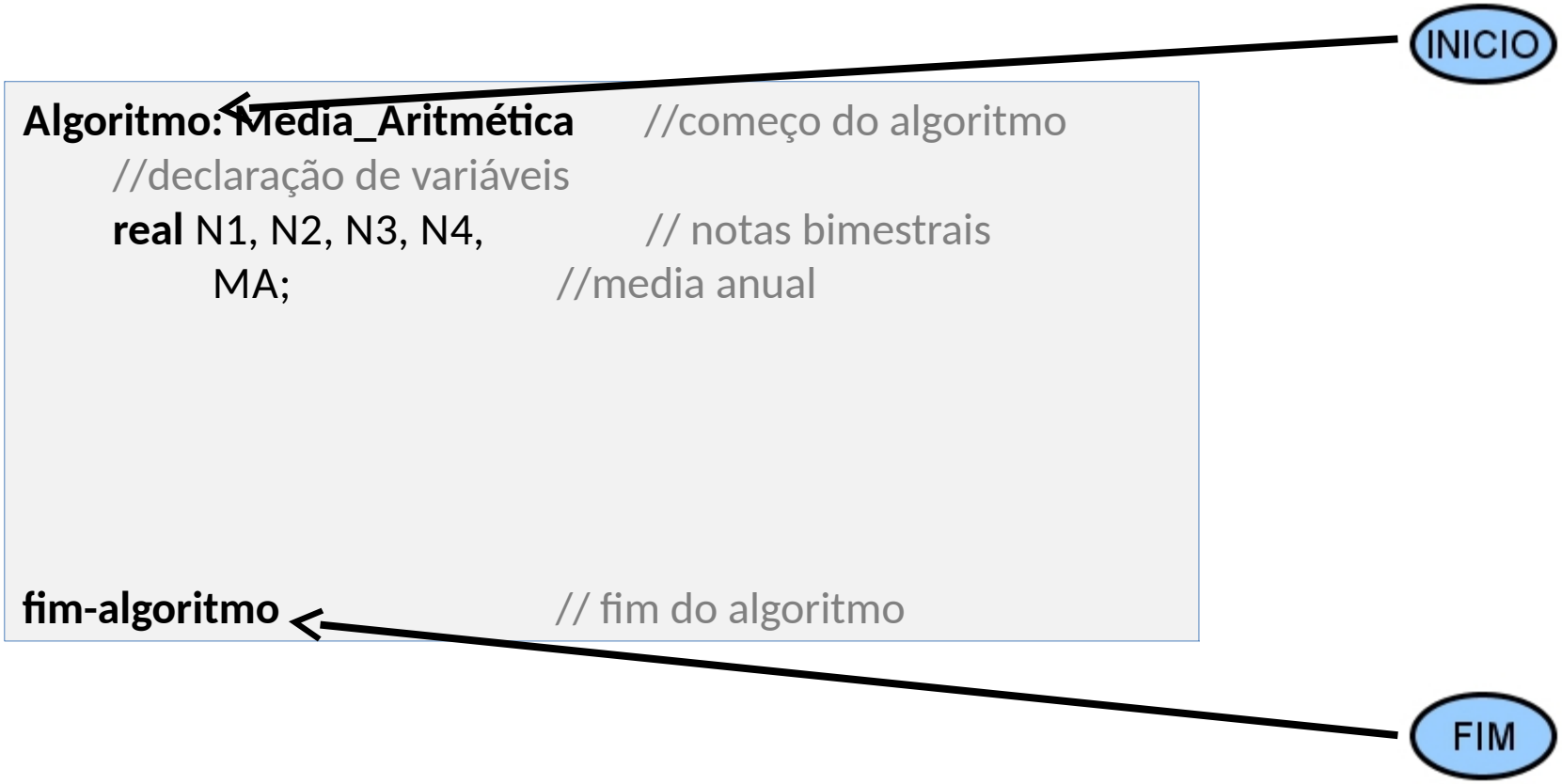
   Dados de entrada: quatro notas bimestrais (N1, N2, N3, N4).

   Dados de Saída: média aritmética anual (MA).

Exemplo – Resposta



Exemplo – Resposta



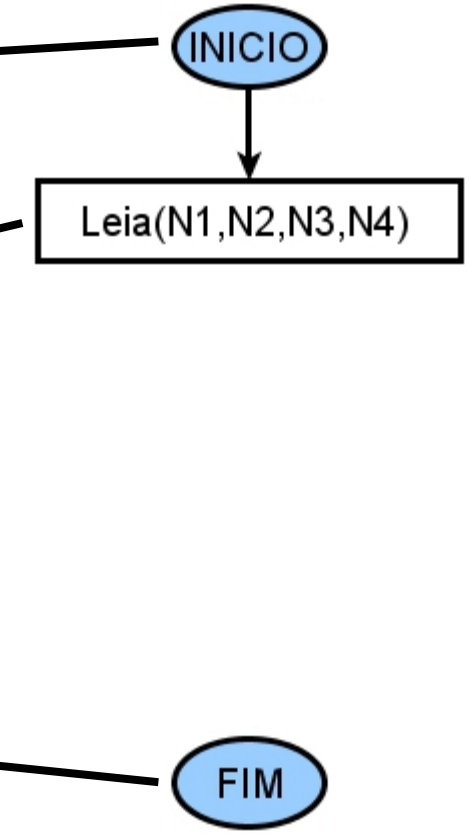
Exemplo – Resposta

```
Algoritmo: Media_Aritmética      //começo do algoritmo
    //declaração de variáveis
    real N1, N2, N3, N4,          // notas bimestrais
         MA;                      //media anual

    leia(N1, N2, N3, N4);         //entrada de dados



fim-algoritmo                     // fim do algoritmo
```

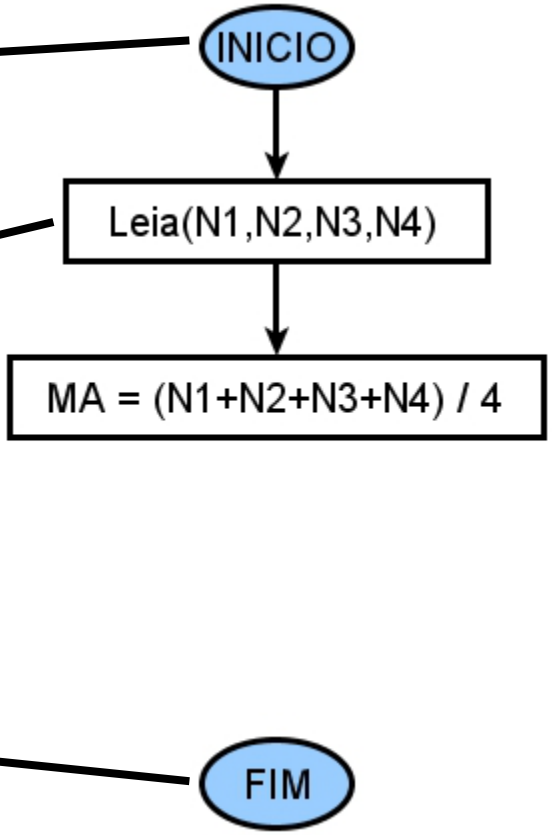


Exemplo – Resposta

```
Algoritmo: Media_Aritmética      //começo do algoritmo
    //declaração de variáveis
    real N1, N2, N3, N4,         // notas bimestrais
         MA;                     //media anual

    leia(N1, N2, N3, N4);        //entrada de dados
    MA = (N1 + N2 + N3 + N4) / 4; //processamento


fim-algoritmo                    // fim do algoritmo
```

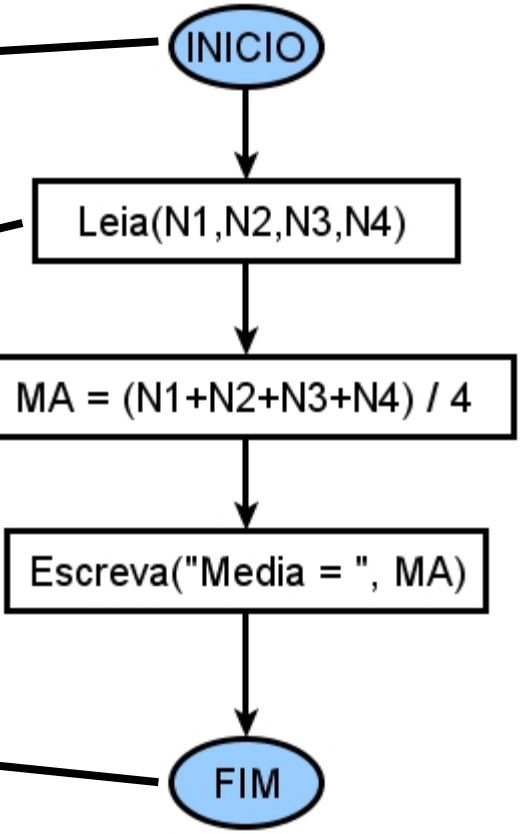


Exemplo – Resposta

```
Algoritmo: Media_Aritmética        //começo do algoritmo
    //declaração de variáveis
    real N1, N2, N3, N4,           // notas bimestrais
         MA;                       //media anual

    leia(N1, N2, N3, N4);          //entrada de dados
    MA = (N1 + N2 + N3 + N4) / 4;  //processamento
    escreva("Média = " , MA);      //saída de dados

fim-algoritmo                      // fim do algoritmo
```



Exercícios

1. Construa um algoritmo que calcule a quantidade de latas de tinta necessárias e o custo para pintar tanques cilíndricos de combustível, em que são fornecidas a altura e o raio desse cilindro (em metros), sabendo que:
   - A lata de tinta custa $ 50,00;
   - Cada lata contem 5 litros;
   - Cada litro de tinta pinta 3 metros quadrados;

   Dados de entrada: altura (H) e raio (R)

   Dados de Saída: custo (C) e a quantidade (QTDE)

   Utilizando o planejamento reverso, sabemos que:
   - O custo é dado pela quantidade de latas * $ 50,00;
   - A quantidade de latas é dada pela quantidade total de litros/5;
   - A quantidade total de litros é dada pela área do cilindro/3;
   - A área do cilindro é dada pela área da base (duas) + área lateral;
   - A área da base é (PI * pow(R,2));
   - A área lateral é comprimento * altura: (2 * PI * R * H)
   - Sendo que R (raio) e H (altura) são dados de entrada e PI é uma constante de valor conhecido: 3.14159265358979323846264338327795...

Exercícios – Resposta 1

```
Algoritmo: Quantidade_latas_custo
    real h, r, custo, qtdLatas, areaTotal, areaBase, areaLateral, litros;
    const real PI = 3.14159265358979323846264338327795;

    leia(h, r);
    areaBase = PI * pow(r,2);
    areaLateral = 2 * PI * r * h;
    areaTotal = 2 * areaBase + areaLateral;
    litros = areaTotal/3;
    qtdLatas = litros/5;
    custo = qtdLatas * 50.00;
    escreva(“Você vai precisar de ”, qtdLatas, “ latas, e gastará $ ”, custo, “ Dilmas”);

fim-algoritmo
```

Exercícios – Resposta 1

```
Algoritmo: Quantidade_latas_custo
    real h, r, custo, qtdLatas, areaTotal, areaBase, areaLateral, litros;
    const real PI = 3.14159265358979323846264338327950;

    leia(h, r);
    areaBase = PI * pow(r,2);
    areaLateral = 2 * PI * r * h;
    areaTotal = 2 * areaBase + areaLateral;
    litros = areaTotal/3;
    qtdLatas = litros/5;
    custo = qtdLatas * 50.00;
    escreva("Você vai precisar de ", qtdLatas, " latas, e gastará $ ", custo, " Dilmas");

fim-algoritmo
```

OBS: este algoritmo não funciona para todos os caros. Encontre a saída do algoritmo para altura (h) e raio (r) iguais a 1 metro.

Exercícios – Resposta 1

```
Algoritmo: Quantidade_latas_custo
    real h, r, custo, qtdLatas, areaTotal, areaBase, areaLateral, litros;
    const real PI = 3.14159265358979323846264338327950;

    leia(h, r);
    areaBase = PI * pow(r,2);
    areaLateral = 2 * PI * r * h;
    areaTotal = 2 * areaBase + areaLateral;
    litros = areaTotal/3;
    qtdLatas = ceil(litros / 5);
    custo = qtdLatas * 50.00;
    escreva(“Você vai precisar de ”, qtdLatas, “ latas, e gastará $ ”, custo, “ Dilmas”);

fim-algoritmo
```

A função **ceil(x)** arredonda x para cima, retorna o menor valor inteiro não menor que x.
A função **floor(x)** arredonda x para baixo, retorna o maior valor inteiro não maior que x.

Exercícios

2. Construa um algoritmo para calcular as raízes de uma equação do 2º grau ($Ax^2 + Bx + C$), sendo que os valores A, B e C são fornecidos pelo usuário (considere que a equação possui duas raízes reais).
3. Construa um algoritmo que, tendo como dados de entrada dois pontos quaisquer no plano $P_1(x_1, y_1)$ e $P_2(x_2, y_2)$, imprima a distância entre eles. A fórmula que efetua tal cálculo é: $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$ que reescrita utilizando os operadores matemáticos adotados fica: $d = sqrt(pow(x_2 - x_1, 2) + pow(y_2 - y_1, 2))$
4. Faça um algoritmo para calcular o volume de uma esfera de raio R, em que R é um dado fornecido pelo usuário. O volume de uma esfera é dado por: $V = \frac{4}{3}\pi R^3$

Exercícios – Resposta 2

```
Algoritmo: Eq_Segundo_grau
    real A, B, C, D, X1, X2;

    leia(A, B, C);
    D = pow(B,2) – 4*A*C;
    X1 = (-B + sqrt(D)) / (2*A);
    X2 = (-B - sqrt(D)) / (2*A);
    escreva(“primeira raiz = ”, X1);
    escreva(“segunda raiz = ”, X2);

fim-algoritmo
```

Exercícios – Resposta 3

```
Algoritmo: distancia_euclidiana
    real D, X1, Y1, X2, Y2;

    leia(X1, Y1, X2, Y2);
    D = sqrt( pow(X2 - X1, 2) + pow(Y2 - Y1,
    2) );
    escreva("Distancia euclidiana =  ", D);

fim-algoritmo
```

Exercícios – Resposta 4

```
Algoritmo: Volume_Esfera
    real R, V;
    const real PI = 3.14159265358979323846264338327795;

    leia(R);
    V = 4/3 * PI * pow(R,3);
    escreva("Volume = ", V);

fim-algoritmo
```