



Gabarito dos Exercícios de Fixação do Livro

Sumário

1 - Informações Gerais	4
1.1 - Base Tecnológica para o Ensino de Lógica de Programação	4
1.2 - Competências e Habilidades	4
1.3 - Conteúdo de Programação de Computadores Abordado na Obra.....	4
2 - Gabarito dos Exercícios	6
3 - Exercícios de Fixação do Capítulo 3.....	6
4 - Exercícios de fixação do capítulo 4	14
5 - Exercícios de fixação do capítulo 5.....	23
5.1 - Laço de repetição pré-teste - controle condicional verdadeiro	23
5.2 - Laço de repetição pré-teste - controle condicional falso.....	35
5.3 - Laço de repetição pós-teste - controle condicional falso	47
5.4 - Laço de repetição pós-teste - controle condicional verdadeiro.....	58
5.5 - Laço de Repetição Condicional Seletivo	70
5.6 - Laço de Repetição Incondicional	81
6 - Exercícios de fixação do capítulo 6	87
7 - Exercícios de fixação do capítulo 7	103
8 - Exercícios de fixação do capítulo 8	128
9 - Exercícios de fixação do capítulo 9	149
10 - Exercícios de fixação do capítulo 10	162

1 - Informações Gerais

1.1 - Base Tecnológica para o Ensino de Lógica de Programação

Normalmente a base tecnológica da lógica de programação para computadores é ministrada em vários cursos da área de Tecnologia da Informação (também denominada por algumas pessoas como Área de Informática), seja em nível técnico (cursos técnicos de nível médio), nível tecnológico (cursos técnicos de nível superior), bacharelado (cursos de nível superior, tais como Sistemas de Informação, Ciência da Computação, Engenharia da Computação, entre outros), cursos de treinamento empresarial e cursos livres, com algumas denominações diferentes, a saber:

- ▶ Algoritmos e Lógica de Programação;
- ▶ Algoritmos e Pseudocódigos;
- ▶ Algoritmos;
- ▶ Lógica Computacional;
- ▶ Lógica de Programação de Computadores;
- ▶ Lógica de Programação;
- ▶ Lógica;
- ▶ Técnicas de Programação - Lógica;
- ▶ Estudo de Algoritmos Computacionais;
- ▶ Introdução à Lógica de Programação;
- ▶ Introdução ao Algoritmo;
- ▶ Linguagem e Técnica de Programação;
- ▶ Lógica Aplicada à Programação de Computadores;
- ▶ Lógica de Programação - Algoritmos;
- ▶ Técnicas Básicas de Programação de Computadores;
- ▶ Técnicas de Programação - Algoritmos;
- ▶ Entre outras possíveis denominações.

Independentemente da nomenclatura utilizada para designá-la, fica evidente a necessidade de desenvolver as habilidades de raciocínio lógico aplicado à programação de computadores, por meio da utilização e implementação de algoritmos computacionais. Neste sentido, a obra à qual este gabarito pertence busca atender seu público, respeitando as estruturas definidas de programação de acordo com a norma internacional ISO 5807:1985 no que tange à elaboração de diagramas de blocos (e não fluxogramas) e com as estruturas de PDL (Program Design Language) propostas por Caine e Gordon na elaboração dos códigos em português estruturado.

1.2 - Competências e Habilidades

O aprendizado da lógica de programação de computadores é baseado na obtenção preliminar das seguintes competências:

- ▶ Desenvolver algoritmos por meio da divisão estruturada com sub-rotinas e também com orientação a objetos;
- ▶ Desenvolver algoritmos por meio da ação de refinamento sucessivo;
- ▶ Interpretar pseudocódigos (algoritmos textuais) como base de documentação do raciocínio lógico para o desenvolvimento de software;
- ▶ Interpretar algoritmos gráficos baseados na norma ISO 5807:1985 (E);
- ▶ Interpretar algoritmos textuais baseados na técnica PDL;
- ▶ Avaliar resultados de teste dos algoritmos desenvolvidos;
- ▶ Integrar módulos de programas desenvolvidos separadamente;
- ▶ Desenvolver algoritmos segundo paradigma da orientação a objetos.

O aprendizado da lógica de programação de computadores é baseado na obtenção preliminar das seguintes habilidades:

- ▶ Selecionar e utilizar técnicas de programação estruturada e/ou orientada a objetos na resolução de problemas computacionais;
- ▶ Fazer uso de modelos, pseudocódigos, diagramas de blocos na representação da solução de problemas computacionais;
- ▶ Executar procedimentos de testes de programas.

1.3 - Conteúdo de Programação de Computadores Abordado na Obra

Capítulo 1. Introdução à Computação

Capítulo 2. Mercado Computacional

Capítulo 3. Linguagens de Programação

Capítulo 4. Algoritmos Computacionais

Capítulo 5. Lógica de Programação de Computadores

Capítulo 6. Compiladores, Interpretadores e Tradutores

Capítulo 7. Programação com Estrutura Sequencial

7.1. Etapas de Ação de um Computador

7.2. Tipos de Dados Primitivos

7.3. O Uso de Variáveis

7.4. O Uso de Constantes

7.5. Os Operadores Aritméticos

7.6. Expressões Aritméticas

7.7. Instruções e Comandos

7.8. Exercício de Aprendizagem

7.9. Exercícios de Fixação

Capítulo 8. Programação com Tomada de Decisão

8.1. Decisões, Condições e Operadores Relacionais

8.2. Tomada de Decisão Simples

8.3. Desvio Condicional Composto

8.4. Outras Formas de Tomada de Decisão

8.5. Operadores Lógicos

8.6. Divisibilidade: Múltiplos e Divisores

Capítulo 9. Programação com Laços

9.1. Laços ou Malhas de Repetição (Loopings ou Loops)

9.2. Laço de Repetição Condicional Pré-Teste

9.3. Laço de Repetição Condicional Pós-Teste

9.4. Laço de Repetição Condicional Seletivo

9.5. Laço de Repetição Incondicional

9.6. Considerações entre Tipos de Laços

Capítulo 10. Estruturas de Dados Homogêneas

10.1. Matrizes de Uma Dimensão (Vetores)

10.2. Classificação de Elementos

10.3. Métodos de Pesquisa de Elementos

10.4. Matrizes com Mais de Uma Dimensão (Tabelas)

10.5. Tipo de Dado Derivado: Estrutura de Registro

10.6. Estrutura de Registro de Matriz

10.7. Estrutura de Matriz de Registros

Capítulo 11. Subprogramas

11.1. Modularidade

11.2. Métodos Top-Down e Bottom-Up

11.3. Procedimentos

11.4. Escopo de Variáveis

11.5. Passagens de Parâmetros

11.6. Funções e Recursividade

Capítulo 12. Introdução à Programação Orientada a Objetos

12.1. Origem da Programação Orientada a Objetos

12.2. Programação Estruturada versus Programação Orientada a Objetos

12.3. Classes, Objetos, Atributos, Métodos, Herança e Encapsulamento

12.4. Poliformismo (Polimorfismo)

2 - Gabarito dos Exercícios

O material apresentado é de uso exclusivo do professor, com o objetivo de ajudar na preparação de aulas, bem como na escolha dos exercícios a serem aplicados durante o curso ministrado, tanto para fixação como para avaliação.

São apresentadas as respostas dos exercícios de fixação do livro "Algoritmos - Lógica para o Desenvolvimento de Programação de Computadores". Para cada exercício de programação são descritos o diagrama de bloco [de acordo com norma ISO 5807:1985 (E)] e o código de programa correspondente escrito em PDL - Program Design Language (Português Estruturado).

As soluções fornecidas neste gabarito caracterizam-se por serem as mais genéricas possíveis, a fim de que fique fácil implementar qualquer algoritmo em qualquer linguagem de programação de computadores. Não há intenção ou pretensão de que este gabarito contenha a melhor resposta para um determinado exercício ou problema, mas apenas uma ideia de como conseguir determinada resposta. Fica a critério do professor trabalhar as questões de refinamento algorítmico no sentido de buscar a melhor solução para um determinado problema apresentado na obra.

As respostas dos exercícios seguem como nomenclatura de nomes de programas a estrutura **Cap03_Ex03b_Pg057**, em que **Cap** indica o número do capítulo, **Ex** o exercício em uso e **Pg** a página onde o exercício está posicionado no livro.

3 - Exercícios de Fixação do Capítulo 3

Tópico 3.9 - Exercício 1 - Página 56

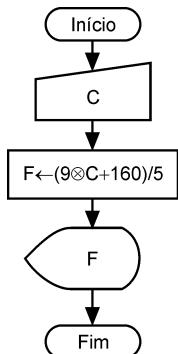
-456	INTEIRO	0	INTEIRO
.F.	LÓGICO	1.56	REAL
.Falso.	LÓGICO	-1.56	REAL
.V.	LÓGICO	34	INTEIRO
"0.87"	CADEIA	45.8976	REAL
"0"	CARACTERE	-465	INTEIRO
"-9.12"	CADEIA	678	INTEIRO
"-900"	CADEIA	-678	INTEIRO
"Casa 8"	CADEIA	-99.8	REAL
"Cinco"	CADEIA	.V.	LÓGICO
"V"	CARACTERE	1000	INTEIRO

Tópico 3.9 - Exercício 2 - Página 57

- | | |
|------------------|----------------|
| (X) ENDEREÇO | () END*A-6 |
| () 21BRASIL | (X) CIDADE3 |
| () FONE\$COM | () #CABEC |
| (X) NAMEUSER | () REAL |
| (X) NOME_USUÁRIO | () REAL\$ |
| () NOME*USUÁRIO | () SOBRE NOME |

Tópico 3.9 - Exercício 3 - Página 57

a) Diagrama de bloco

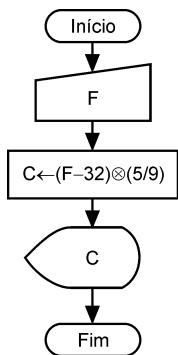


Português estruturado

```

programa Cap03_Ex3a_Pg057
var
  C, F : real
início
  leia C
  F ← (9 * C + 160) / 5
  escreva F
fim
  
```

b) Diagrama de bloco

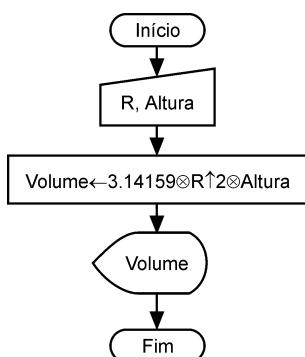


Português estruturado

```

programa Cap03_Ex3b_Pg057
var
  C, F : real
início
  leia F
  C ← (F - 32) * (5 / 9)
  escreva C
fim
  
```

c) Diagrama de bloco

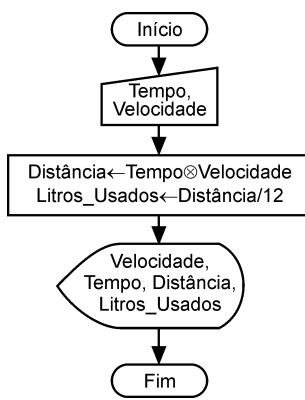


Português estruturado

```

programa Cap03_Ex3c_Pg057
var
  VOLUME, R, ALTURA : real
início
  leia R, ALTURA
  VOLUME ← 3.14159 * R ↑ 2 * ALTURA
  escreva VOLUME
fim
  
```

d) Diagrama de bloco

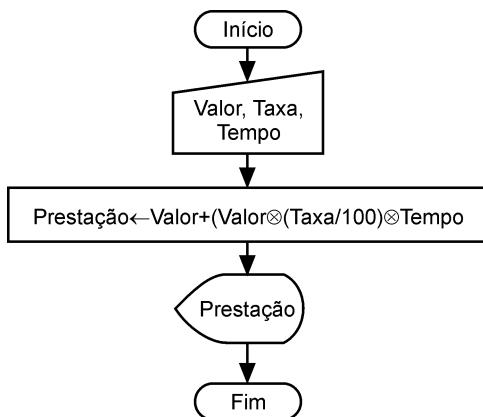


Português estruturado

```

programa Cap03_Ex3d_Pg057
var
  DISTÂNCIA, TEMPO, VELOCIDADE, LITROS_USADOS : real
início
  leia TEMPO, VELOCIDADE
  DISTÂNCIA ← TEMPO * VELOCIDADE
  LITROS_USADOS ← DISTÂNCIA / 12
  escreva VELOCIDADE, TEMPO, DISTÂNCIA, LITROS_USADOS
fim
  
```

e) Diagrama de bloco

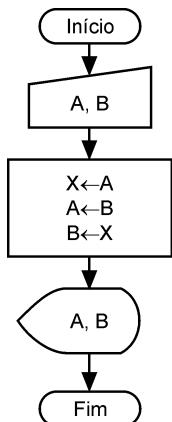


Português estruturado

```

programa Cap03_Ex3e_Pg057
var
  PRESTAÇÃO, VALOR, TAXA, TEMPO : real
início
  leia VALOR, TAXA, TEMPO
  PRESTAÇÃO ← VALOR + (VALOR * (TAXA / 100) * TEMPO)
  escreva PRESTAÇÃO
fim
  
```

f) Diagrama de bloco



Português estruturado

Este programa também pode ser escrito da seguinte forma:

```

programa
Cap03_Ex3f_Pg057_versão_1
var
  A, B, X : inteiro
início
  leia A, B
  X ← A
  A ← B
  B ← X
  escreva A, B
fim
  
```

```

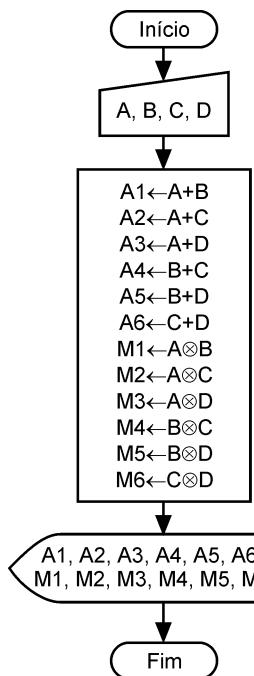
programa
Cap03_Ex3f_Pg057_versão_2
var
  A, B, X : real
início
  leia A, B
  X ← A
  A ← B
  B ← X
  escreva A, B
fim
  
```

Este programa também pode ser escrito da seguinte forma:

```

programa
Cap03_Ex3f_Pg057_versão_3
var
  A, B, X : cadeia
início
  leia A, B
  X ← A
  A ← B
  B ← X
  escreva A, B
fim
  
```

g) Diagrama de bloco



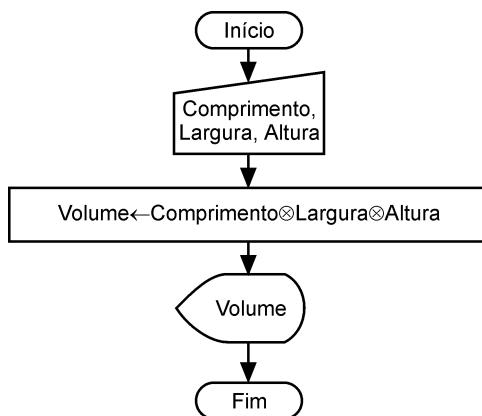
Português estruturado

```

programa Cap03_Ex3g_Pg057
var
    A, B, C, D : inteiro
    A1, A2, A3, A4, A5, A6 : inteiro
    M1, M2, M3, M4, M5, M6 : inteiro
início
    leia A, B, C, D
    A1 ← A + B
    A2 ← A + C
    A3 ← A + D
    A4 ← B + C
    A5 ← B + D
    A6 ← C + D
    M1 ← A * B
    M2 ← A * C
    M3 ← A * D
    M4 ← B * C
    M5 ← B * D
    M6 ← C * D
    escreva A1, A2, A3, A4, A5, A6
    escreva M1, M2, M3, M4, M5, M6
fim

```

h) Diagrama de bloco



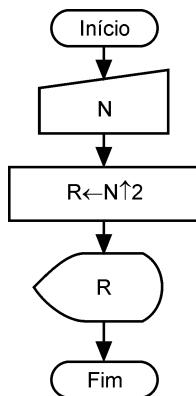
Português estruturado

```

programa Cap03_Ex3h_Pg057
var
    VOLUME, COMPRIMENTO, LARGURA, ALTURA : real
início
    leia COMPRIMENTO, LARGURA, ALTURA
    VOLUME ← COMPRIMENTO * LARGURA * ALTURA
    escreva VOLUME
fim

```

i) Diagrama de bloco



Português estruturado

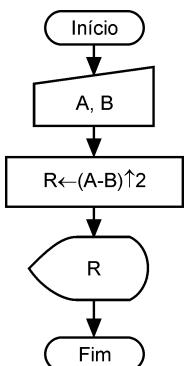
```

programa Cap03_Ex3i_Pg057
var
    N, R : inteiro
início
    leia N
    R ← N * N (ou R ← N ↑ 2)
    escreva R
fim

```

Tópico 3.9 - Exercício - Página 58

j) Diagrama de bloco

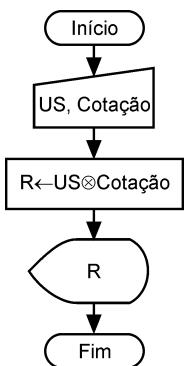


Português estruturado

```

programa Cap03_Ex3j_Pg058
var
  A, B, R : inteiro
início
  leia A, B
  R ← (A - B) ↑ 2
  escreva R
fim
  
```

k) Diagrama de bloco

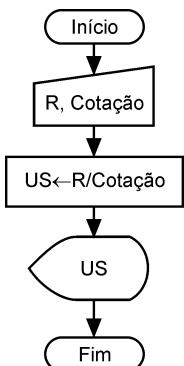


Português estruturado

```

programa Cap03_Ex3k_Pg058
var
  R, US, COTAÇÃO : real
início
  leia US, COTAÇÃO
  R ← US * COTAÇÃO
  escreva R
fim
  
```

l) Diagrama de bloco

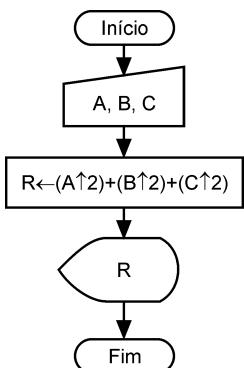


Português estruturado

```

programa Cap03_Ex3l_Pg058
var
  R, US, COTAÇÃO : real
início
  leia R, COTAÇÃO
  US ← R / COTAÇÃO
  escreva US
fim
  
```

m) Diagrama de bloco

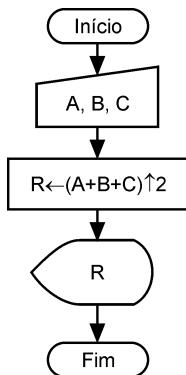


Português estruturado

```

programa Cap03_Ex3m_Pg058
var
  R, A, B, C : inteiro
início
  leia A, B, C
  R ← (A ↑ 2) + (B ↑ 2) + (C ↑ 2)
  escreva R
fim
  
```

n) Diagrama de bloco

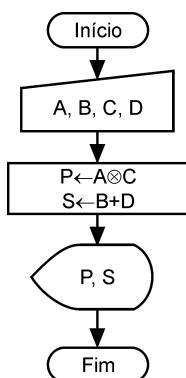


Português estruturado

```

programa Cap03_Ex3n_Pg058
var
    R, A, B, C : inteiro
início
    leia A, B, C
    R ← (A + B + C) ↑ 2
    escreva R
fim
    
```

o) Diagrama de bloco

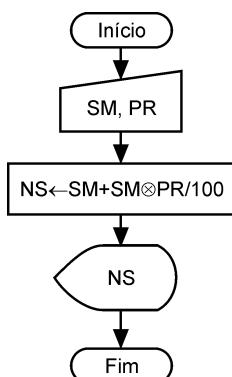


Português estruturado

```

programa Cap03_Ex3o_Pg058
var
    A, B, C, D, P, S : inteiro
início
    leia A, B, C, D
    P ← A * C
    S ← B + D
    escreva P, S
fim
    
```

p) Diagrama de bloco

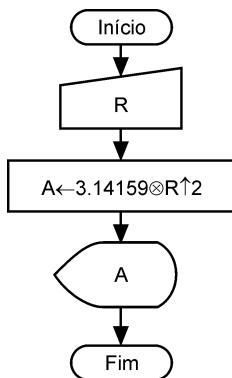


Português estruturado

```

programa Cap03_Ex3p_Pg058
var
    SM, PR, NS : real
início
    leia SM, PR
    NS ← SM + SM * PR / 100
    escreva NS
fim
    
```

q) Diagrama de bloco

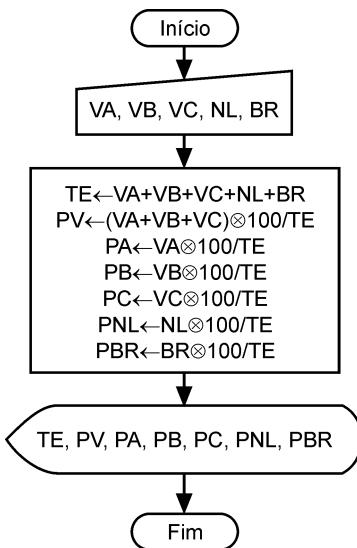


Português estruturado

```

programa Cap03_Ex3q_Pg058
var
    A, R : real
início
    leia R
    A ← 3.14159 * R ↑ 2
    escreva A
fim
    
```

r) Diagrama de bloco

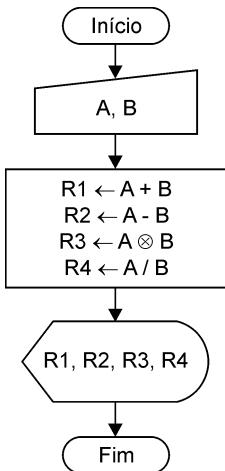


Português estruturado

```

programa Cap03_Ex3r_Pg058
var
  VA, VB, VC, NL, BR, TE : inteiro
  PV, PA, PB, PC, PNL, PBR : real
início
  leia VA, VB, VC, NL, BR
  TE ← VA + VB + VC + NL + BR
  PV ← (VA + VB + VC) * 100 / TE
  PA ← VA * 100 / TE
  PB ← VB * 100 / TE
  PC ← VC * 100 / TE
  PNL ← NL * 100 / TE
  PBR ← BR * 100 / TE
  escreva TE, PV, PA, PB, PC, PNL, PBR
fim
  
```

s) Diagrama de bloco

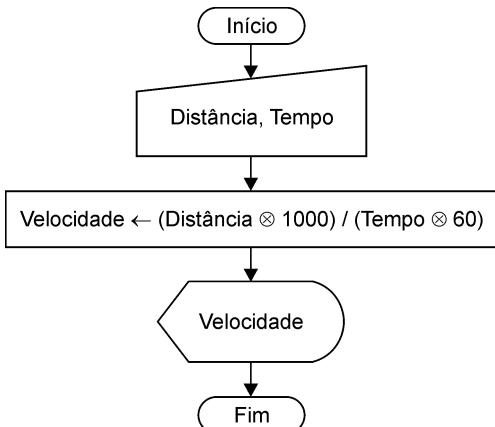


Português estruturado

```

programa Cap03_Ex3s_Pg058
var
  A, B, R1, R2, R3, R4 : real
início
  leia A, B
  R1 ← A + B
  R2 ← A - B
  R3 ← A * B
  R4 ← A / B
  escreva R1, R2, R3, R4
fim
  
```

t) Diagrama de bloco

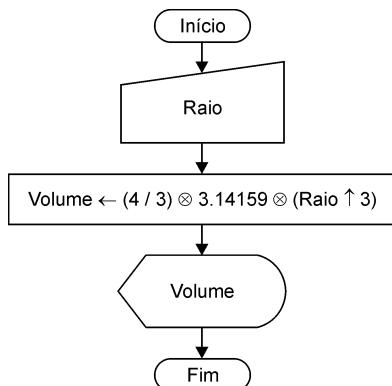


Português estruturado

```

programa Cap03_Ex3t_Pg058
var
  VELOCIDADE, DISTÂNCIA, TEMPO : real
início
  leia DISTÂNCIA, TEMPO
  VELOCIDADE ← (DISTÂNCIA * 1000) / (TEMPO * 60)
  escreva VELOCIDADE
fim
  
```

u) Diagrama de bloco

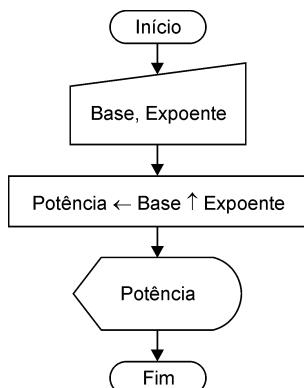


Português estruturado

```

programa Cap03_Ex3u_Pg058
var
  VOLUME, RAIO : real
início
  leia RAIO
  VOLUME ← (4 / 3) * 3.14159 * (RAIO ↑ 3)
  escreva VOLUME
fim
  
```

v) Diagrama de bloco

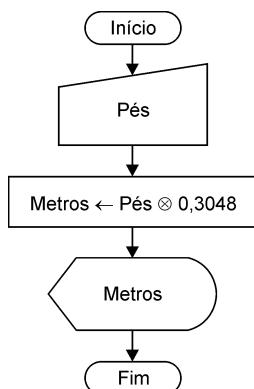


Português estruturado

```

programa Cap03_Ex3v_Pg058
var
  BASE, EXPOENTE, POTÊNCIA : inteiro
início
  leia BASE, EXPOENTE
  POTÊNCIA ← BASE ↑ EXPOENTE
  escreva POTÊNCIA
fim
  
```

w) Diagrama de bloco

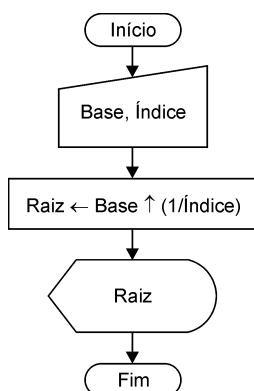


Português estruturado

```

programa Cap03_Ex3x_Pg058
var
  PÉS, METROS : real
início
  leia PÉS
  METROS ← PÉS * 0,3048
  escreva METROS
fim
  
```

x) Diagrama de bloco

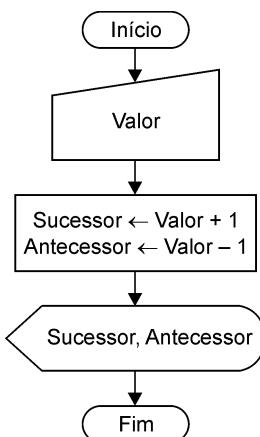


Português estruturado

```

programa Cap03_Ex3x_Pg058
var
  RAIZ, BASE, ÍNDICE : REAL
início
  leia BASE, ÍNDICE
  RAIZ ← BASE ↑ (1 / ÍNDICE)
  escreva RAIZ
fim
  
```

y) Diagrama de bloco



Português estruturado

```

programa Cap03_Ex3y_Pg058
var
  VALOR, SUCESSOR, ANTECESSOR : INTEIRO
início
  leia VALOR
  SUCESSOR ← VALOR + 1
  ANTECESSOR ← VALOR - 1
  escreva SUCESSOR, ANTECESSOR
fim
  
```

4 - Exercícios de fixação do capítulo 4

Tópico 4.7 - Exercício 1 - Páginas 97 e 98

- a) Verdadeiro
- b) Falso
- c) Verdadeiro
- d) Verdadeiro
- e) Verdadeiro
- f) Falso
- g) Falso
- h) Verdadeiro
- i) Falso
- j) Falso

É ideal que nestes exercícios o aluno perceba a posição das condições *verdadeira* e *falsa*. Deve ficar claro que sempre a ação da condição *verdadeira* será executada entre as declarações de instrução **se...senão** e a ação da condição *falsa* será executada entre as declarações de instrução **senão...fim_se**, nos casos de decisões compostas.

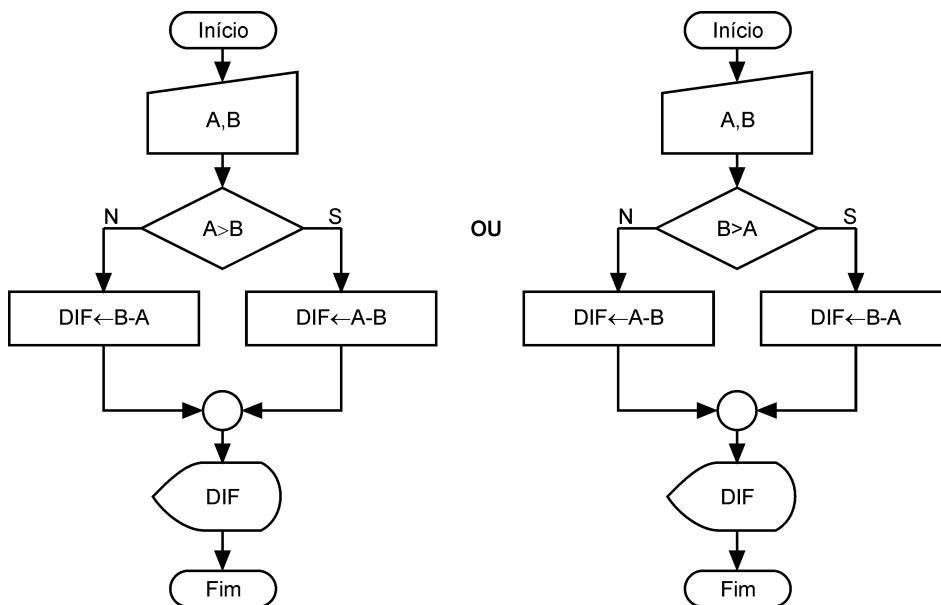
Neste exercício não é necessário fazer o aluno efetuar o cálculo do valor das equações; basta apenas apontar se a condição é *verdadeira* ou *falsa* e indicar qual equação deve ser calculada.

Tópico 4.7 - Exercício 2 - Página 98 e 99

- a) Falso, neste caso executa-se a equação $X \leftarrow (A - B) / C$, $X = -0,20$
- b) Falso, neste caso executa-se a equação $X \leftarrow (A + B) / D * (C + D)$, $X = 7,78$
- c) Verdadeiro, neste executa-se a equação $X \leftarrow (A + 2) * (B - 2)$, $X = 4$
- d) Falso, neste executa-se a equação $X \leftarrow A - B$, $X = -1$
- e) Verdadeiro, neste executa-se a equação $X \leftarrow A + B$, $X = 5$
- f) Falso, neste executa-se a equação $X \leftarrow D / B$, $X = 3$
- g) Verdadeiro, neste executa-se a equação $X \leftarrow (A + D) / 2$, $X = 5,50$
- h) Verdadeiro, neste executa-se a equação $X \leftarrow (A + D) / 2$, $X = 5,50$

Tópico 4.7 - Exercício 3 - Página 99

a) Diagrama de blocos



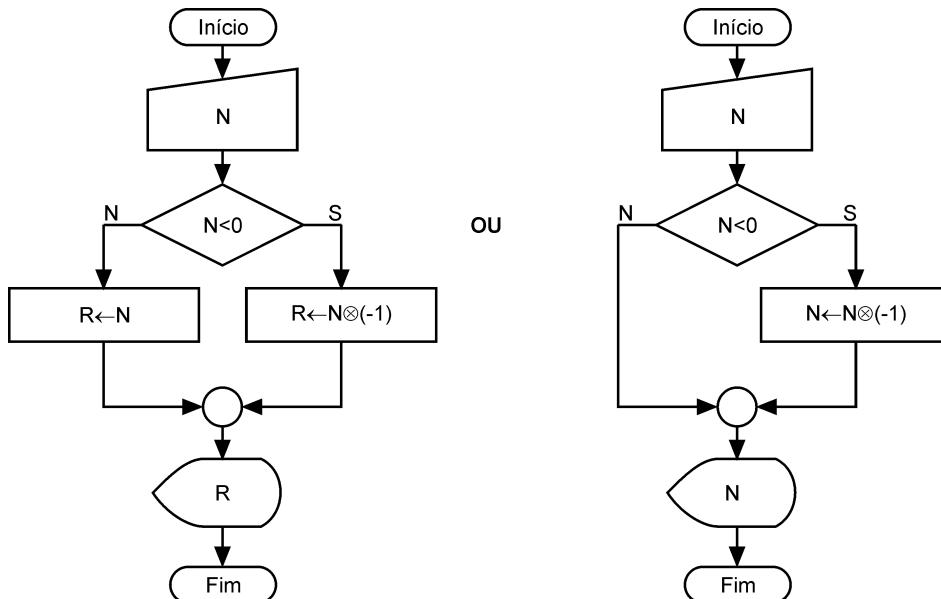
Português estruturado

```
programa Cap04_Ex3a_Pg099_versão_1
var
    A, B, DIF : inteiro
início
    leia A, B
    se (A > B) então
        DIF ← A - B
    senão
        DIF ← B - A
    fim_se
    escreva DIF
fim
```

Este programa também pode ser escrito da seguinte forma:

```
programa Cap04_Ex3a_Pg099_versão_2
var
    A, B, DIF : inteiro
início
    leia A, B
    se (B > A) então
        DIF ← B - A
    senão
        DIF ← A - B
    fim_se
    escreva DIF
fim
```

b) Diagrama de blocos

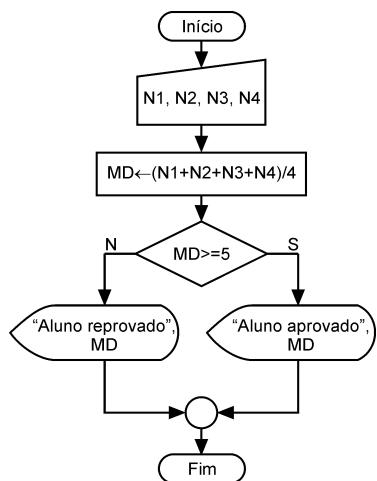


Português estruturado

```

programa Cap04_Ex3b_Pg099_versão_1
var
    N, R : inteiro
início
    leia N
    se (N < 0) então
        R ← N * (-1)
    senão
        R ← N
    fim_se
    escreva R
fim
  
```

c) Diagrama de bloco



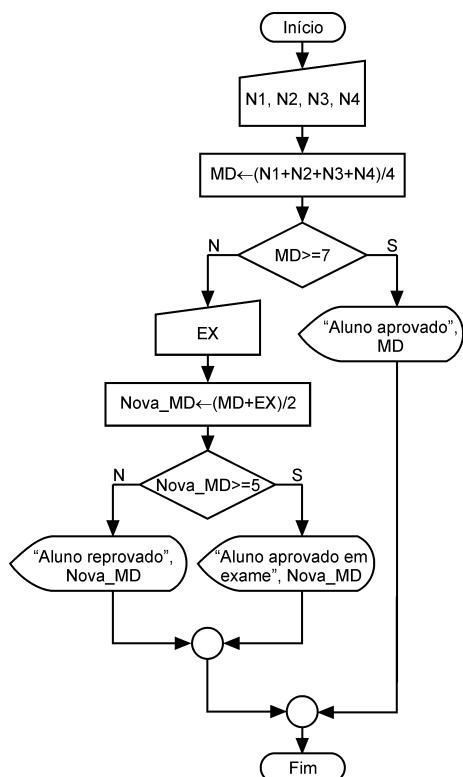
Este programa também pode ser escrito da seguinte forma:

```

programa Cap04_Ex3b_Pg099_versão_2
var
    N : inteiro
início
    leia N
    se (N < 0) então
        N ← N * (-1)
    fim_se
    escreva N
fim
  
```

Tópico 4.7 - Exercício 3 - Página 100

d) Diagrama de bloco



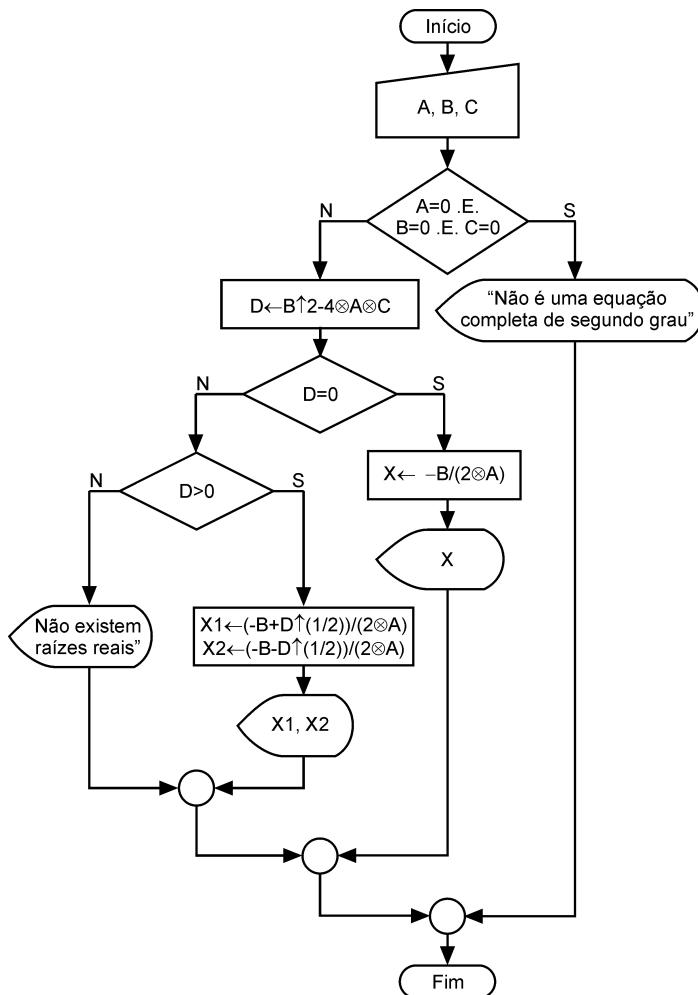
Português estruturado

```

programa Cap04_Ex3d_Pg100
var
    N1, N2, N3, N4, MD, NOVA_MD, EX : real
início
    leia N1, N2, N3, N4
    MD ← (N1 + N2 + N3 + N4) / 4
    se (MD ≥ 7) então
        escreva "Aluno Aprovado", MD
    senão
        leia EX
        NOVA_MD ← (MD + EX) / 2
        se (NOVA_MD ≥ 5) então
            escreva "Aluno Aprovado em Exame", NOVA_MD
        senão
            escreva "Aluno Reprovado", NOVA_MD
        fim_se
    fim_se
fim
  
```

Um ponto a ser observado neste exercício é a necessidade de extrair a raiz quadrada de Δ (delta), que está sendo representado pela variável D. Muitos alunos esquecem-se de que para extraír uma raiz quadrada de um número, basta elevá-lo ao inverso de seu índice. Ao se considerar a extração da raiz quadrada de D, basta fazer $D \uparrow (1/2)$ ou $D \uparrow 0.5$. Essa forma de extração de raiz torna-se adequada por proporcionar ao aluno a visão necessária para extraír raízes de qualquer outro índice. Assim sendo, para uma raiz cúbica, basta elevar a $1/3$; para uma raiz de índice quatro, basta elevar a $1/4$ e assim por diante.

e) Diagrama de bloco

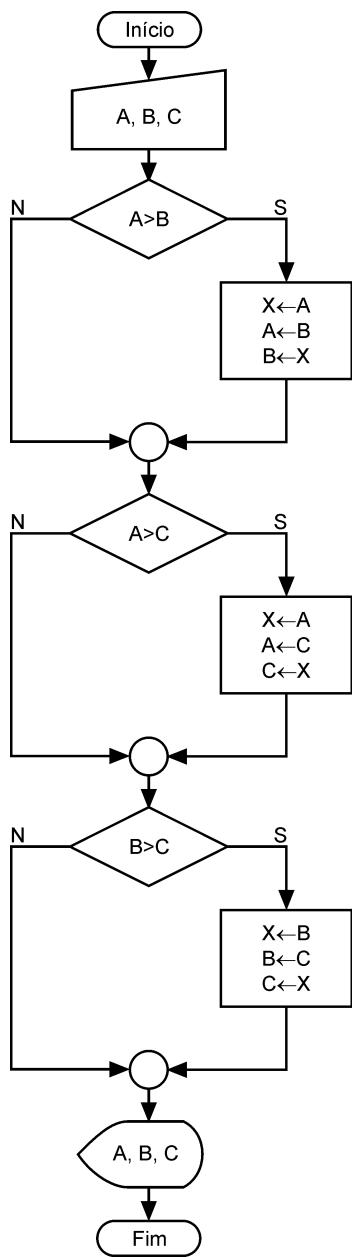


Português estruturado

```

programa Cap04_Ex3e_Pg100
var
    A, B, C, D, X, X1, X2 : real
início
    leia A, B, C
    se (A = 0) .e. (B = 0) .e. (C = 0) então
        escreva "Não é uma equação completa de segundo grau."
    senão
        D ← B ↑ 2 - 4 * A * C
        se (D = 0) então
            X ← -B / (2 * A)
            escreva X
        senão
            se (D > 0) então
                X1 ← (-B + D ↑ (1 / 2)) / (2 * A)
                X2 ← (-B - D ↑ (1 / 2)) / (2 * A)
                escreva X1, X2
            senão
                escreva "Não existem raízes reais."
            fim_se
        fim_se
    fim_se
fim
    
```

f) Diagrama de bloco



Português estruturado

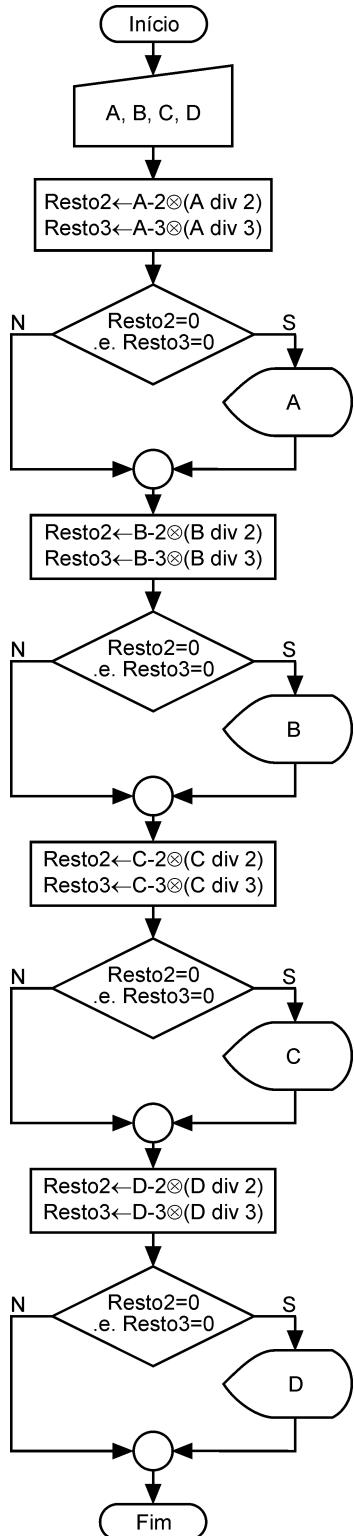
```

programa Cap04_Ex3f_Pg100
var
  A, B, C, X : inteiro
início
  leia A, B, C
  se (A > B) então
    X ← A
    A ← B
    B ← X
  fim_se
  se (A > C) então
    X ← A
    A ← C
    C ← X
  fim_se
  se (C > B) então
    X ← C
    C ← B
    B ← X
  fim_se
  escreva A, B, C
fim
  
```

Tópico 4.7 - Exercício 3g - Página 100

Neste exercício é enfatizado o conceito de trabalhar com valores numéricos inteiros que sejam divisíveis entre dois valores (seus divisores). Para determinar se um valor é divisível por outro, basta verificar se o resto da divisão entre os valores (dividendos e divisores) é igual a zero, desde que o quociente da divisão seja um valor inteiro. É por esta razão que está sendo utilizado o operador de divisão div e não o operador / (barra). O operador / (barra) deve ser usado quando se deseja obter um quociente real. Observe que não é usado o operador mod ou % (operadores que determinam o valor do resto de uma divisão de inteiros quando usados nas linguagens de programação PASCAL e C, C++, C#, Java, etc.), por serem exclusivos das linguagens de programação. É importante que o aluno aprenda a pensar na solução do problema na forma algorítmica e não com o operador da linguagem de programação em si.

g) Diagrama de bloco



Português estruturado

```

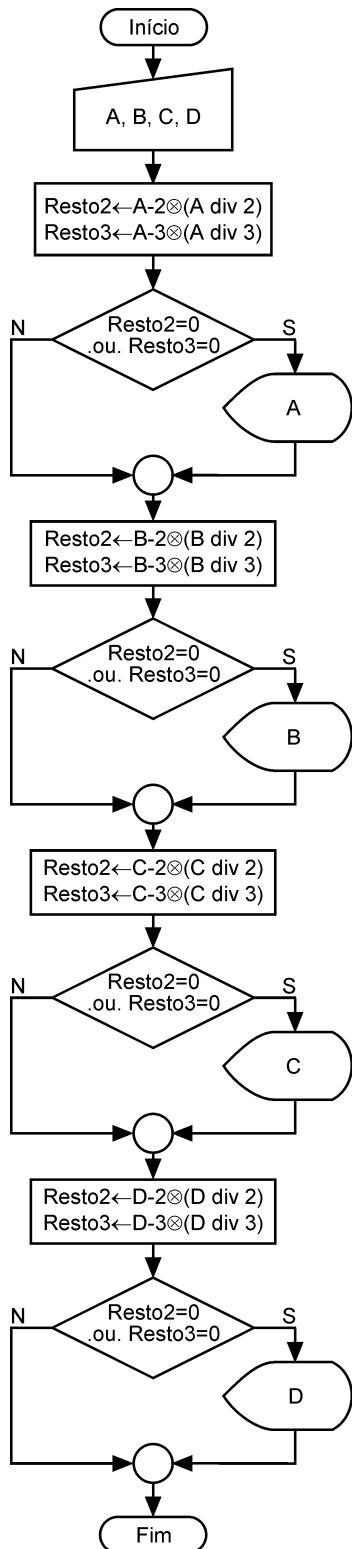
programa Cap04_Ex3g_Pg100
var
    A, B, C, D, RESTO2, RESTO3 : inteiro
início
    leia A, B, C, D
    RESTO2 ← A - 2 * (A div 2)
    RESTO3 ← A - 3 * (A div 3)
    se (RESTO2 = 0) .e. (RESTO3 = 0) então
        escreva "O valor ", A, " é divisível por 2 e 3"
    fim_se
    RESTO2 ← B - 2 * (B div 2)
    RESTO3 ← B - 3 * (B div 3)
    se (RESTO2 = 0) .e. (RESTO3 = 0) então
        escreva "O valor ", B, " é divisível por 2 e 3"
    fim_se
    RESTO2 ← C - 2 * (C div 2)
    RESTO3 ← C - 3 * (C div 3)
    se (RESTO2 = 0) .e. (RESTO3 = 0) então
        escreva "O valor ", C, " é divisível por 2 e 3"
    fim_se
    RESTO2 ← D - 2 * (D div 2)
    RESTO3 ← D - 3 * (D div 3)
    se (RESTO2 = 0) .e. (RESTO3 = 0) então
        escreva "O valor ", D, " é divisível por 2 e 3"
    fim_se
fim

```

Tópico 4.7 - Exercício 3 - Página 100

Este exercício também enfatiza o conceito de trabalhar com valores que sejam divisíveis entre dois valores. Neste caso é utilizado o operador lógico .ou. nas referências condicionais.

h) Diagrama de bloco

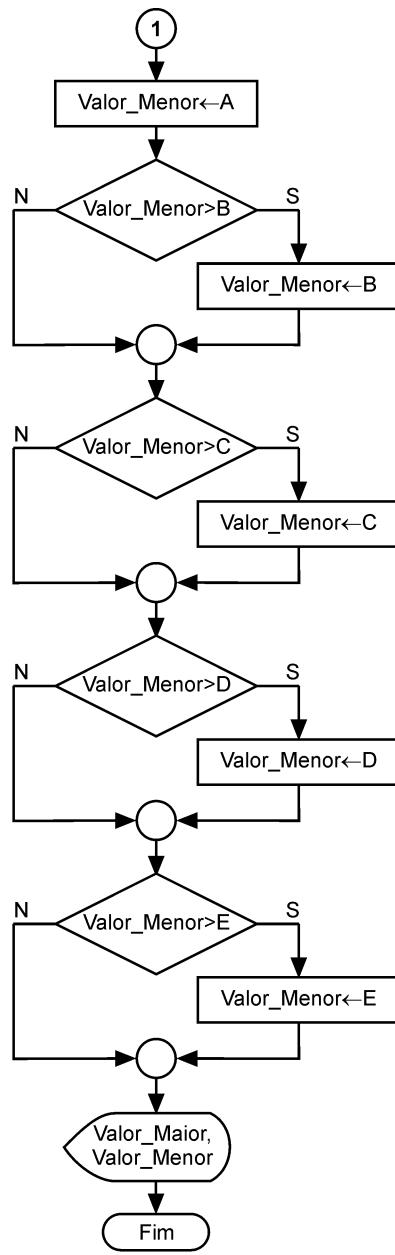
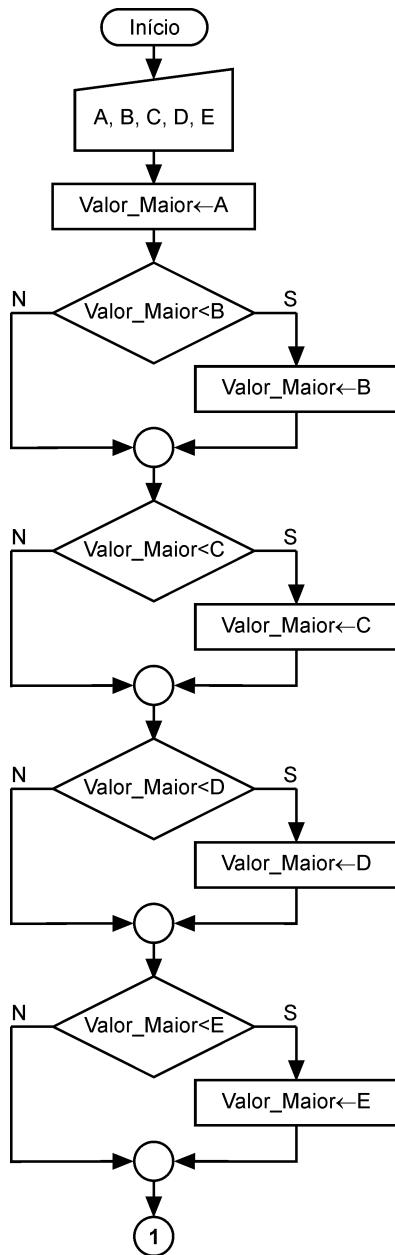


Português estruturado

```

programa Cap04_Ex3h_Pg100
var
  A, B, C, D, RESTO2, RESTO3 : inteiro
início
  leia A, B, C, D
  RESTO2 ← A - 2 * (A div 2)
  RESTO3 ← A - 3 * (A div 3)
  se (RESTO2 = 0) .ou. (RESTO3 = 0) então
    escreva "O valor ", A, " é divisível por 2 ou 3"
  fim_se
  RESTO2 ← B - 2 * (B div 2)
  RESTO3 ← B - 3 * (B div 3)
  se (RESTO2 = 0) .ou. (RESTO3 = 0) então
    escreva "O valor ", B, " é divisível por 2 ou 3"
  fim_se
  RESTO2 ← C - 2 * (C div 2)
  RESTO3 ← C - 3 * (C div 3)
  se (RESTO2 = 0) .ou. (RESTO3 = 0) então
    escreva "O valor ", C, " é divisível por 2 ou 3"
  fim_se
  RESTO2 ← D - 2 * (D div 2)
  RESTO3 ← D - 3 * (D div 3)
  se (RESTO2 = 0) .ou. (RESTO3 = 0) então
    escreva "O valor ", D, " é divisível por 2 ou 3"
  fim_se
fim
  
```

i) Diagrama de blocos



Português estruturado

```

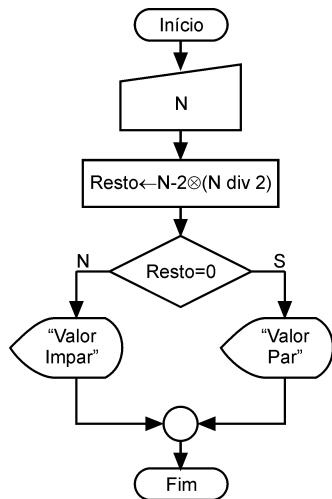
programa Cap04_Ex3i_Pg100
var
  A, B, C, D, E, VALOR_MAIOR, VALOR_MENOR : inteiro
início
  leia A, B, C, D, E
  VALOR_MAIOR ← A
  se (VALOR_MAIOR < B) então
    VALOR_MAIOR ← B
  fim_se
  se (VALOR_MAIOR < C) então
    VALOR_MAIOR ← C
  fim_se
  se (VALOR_MAIOR < D) então
    VALOR_MAIOR ← D
  fim_se
  se (VALOR_MAIOR < E) então
    VALOR_MAIOR ← E
  fim_se
  VALOR_MENOR ← A
  se (VALOR_MENOR > B) então
    VALOR_MENOR ← B
  
```

```

fim_se
se (VALOR_MENOR > C) então
    VALOR_MENOR ← C
fim_se
se (VALOR_MENOR > D) então
    VALOR_MENOR ← D
fim_se
se (VALOR_MENOR > E) então
    VALOR_MENOR ← E
fim_se
escreva VALOR_MAIOR, VALOR_MENOR
fim

```

j) Diagrama de bloco



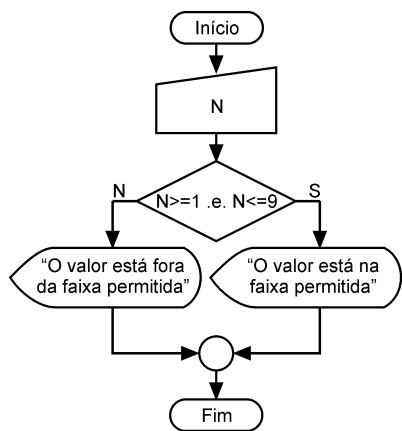
Português estruturado

```

programa Cap04_Ex3j_Pg100
var
    N, RESTO : inteiro
início
    leia N
    RESTO ← N - 2 * (N div 2)
    se (RESTO = 0) então
        escreva "O valor ", N, " é PAR."
    senão
        escreva "O valor ", N, " é IMPAR."
    fim_se
fim

```

k) Diagrama de bloco



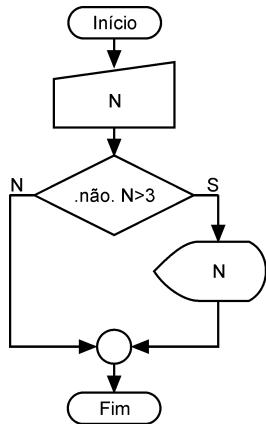
Português estruturado

```

programa Cap04_Ex3k_Pg100
var
    N : inteiro
início
    leia N
    se (N ≥ 1) .e. (N ≤ 9) então
        escreva "O valor está na faixa permitida."
    senão
        escreva "O valor está fora da faixa permitida."
    fim_se
fim

```

l) Diagrama de bloco



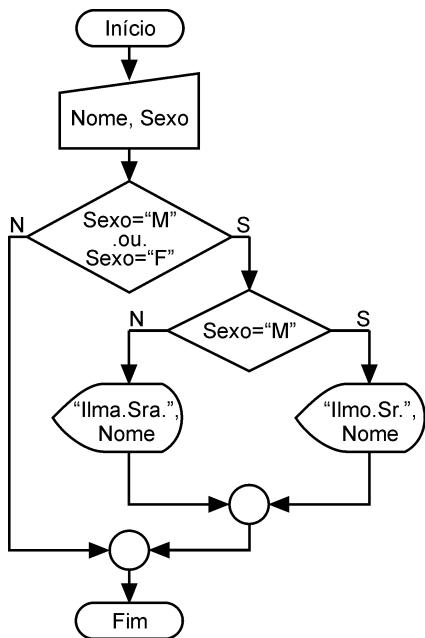
Português estruturado

```

programa Cap04_Ex3l_Pg100
var
    N : inteiro
início
    leia N
    se .não. (N > 3) então
        escreva N
    fim_se
fim

```

m) Diagrama de bloco



Português estruturado

```

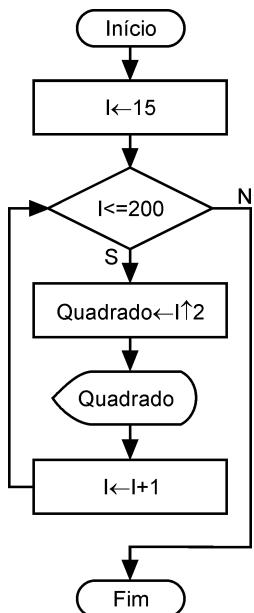
programa Cap04_Ex3m_Pg100
var
  NOME : cadeia
  SEXO : caractere
início
  leia NOME, SEXO
  se (SEXO = "M") .ou. (SEXO = "F") então
    se (SEXO = "M") então
      escreva "Ilmo. Sr. ", NOME
    senão
      escreva "Ilma. Sra. ", NOME
    fim_se
  fim_se
fim
  
```

5 - Exercícios de fixação do capítulo 5

5.1 - Laço de repetição pré-teste - controle condicional verdadeiro

Tópico 5.9 - Exercício 1 - Página 131

a) Diagrama de bloco

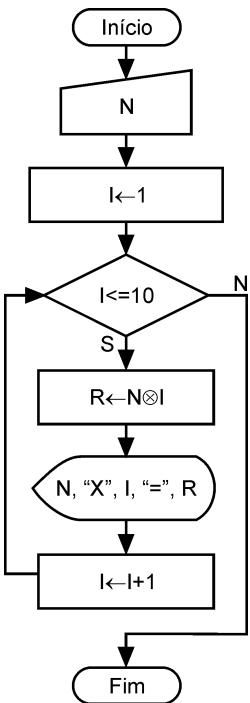


Português estruturado

```

programa Cap05_Ex1a_Pg131
var
  QUADRADO, I : inteiro
início
  I ← 15
  enquanto (I <= 200) faça
    QUADRADO ← I ↑ 2
    escreva QUADRADO
    I ← I + 1
  fim_enquanto
fim
  
```

b) Diagrama de bloco

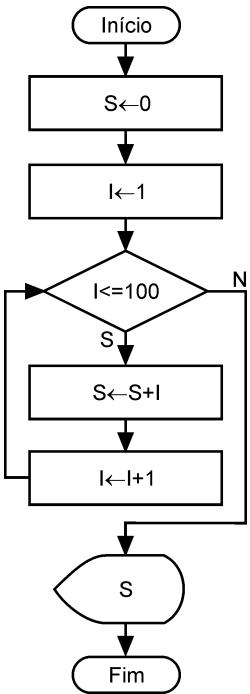


Português estruturado

```

programa Cap05_Ex1b_Pg131
var
  N, I, R : inteiro
início
  leia N
  I ← 1
  enquanto (I <= 10) faça
    R ← N * I
    escreva N, " X ", I, " = ", R
    I ← I + 1
  fim_enquanto
fim
  
```

c) Diagrama de bloco

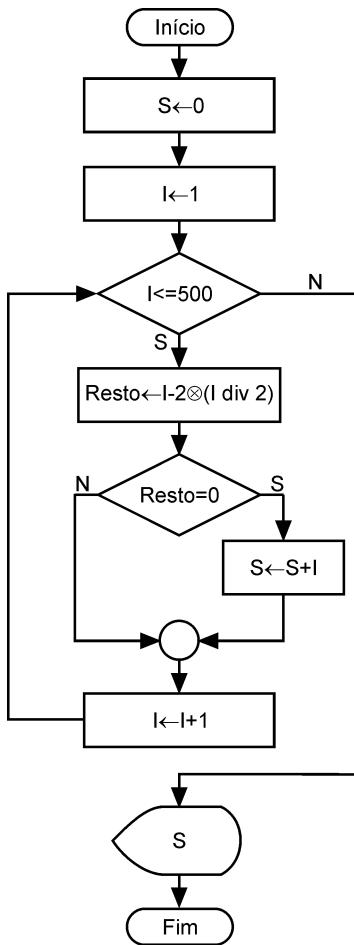


Português estruturado

```

programa Cap05_Ex1c_Pg131
var
  S, I : inteiro
início
  S ← 0
  I ← 1
  enquanto (I <= 100) faça
    S ← S + I
    I ← I + 1
  fim_enquanto
  escreva S
fim
  
```

d) Diagrama de bloco

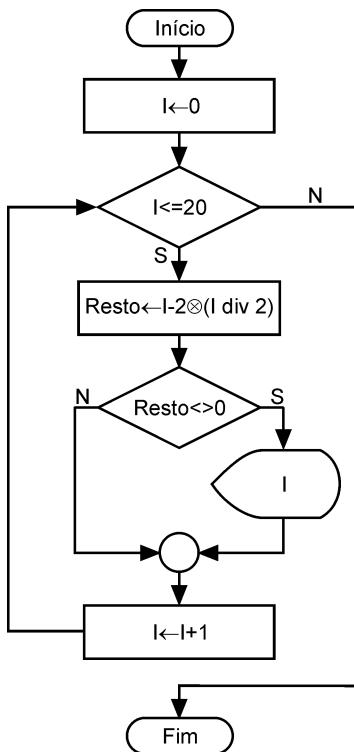


Português estruturado

```

programa Cap05_Ex1d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  I ← 1
  enquanto (I <= 500) faça
    RESTO ← I - 2 * (I div 2)
    se (RESTO = 0) então
      S ← S + I
    fim_se
    I ← I + 1
  fim_enquanto
  escreva S
fim
  
```

e) Diagrama de bloco

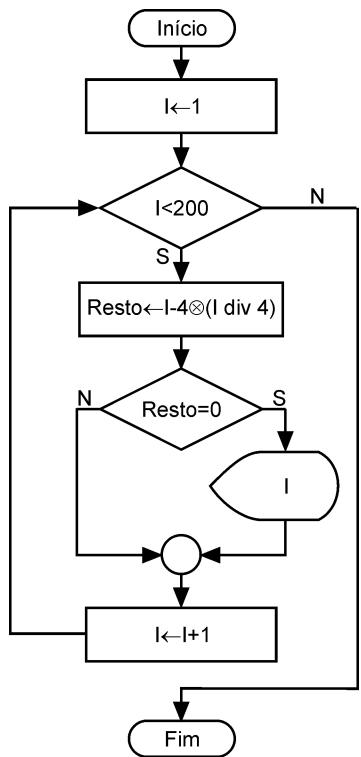


Português estruturado

```

programa Cap05_Ex1e_Pg131
var
  I, RESTO : inteiro
início
  I ← 0
  enquanto (I <= 20) faça
    RESTO ← I - 2 * (I div 2)
    se (RESTO <> 0) então
      escreva I
    fim_se
    I ← I + 1
  fim_enquanto
fim
  
```

f) Diagrama de bloco

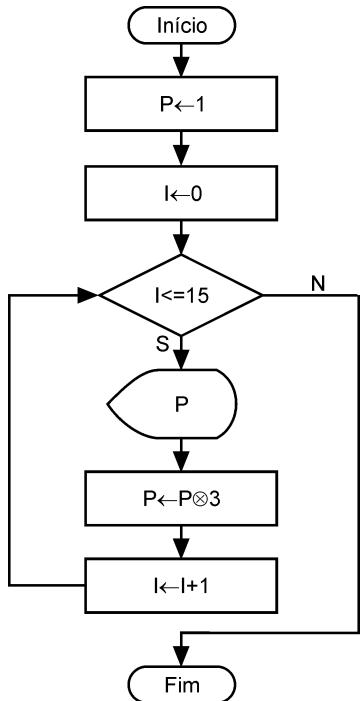


Português estruturado

```

programa Cap05_Ex1f_Pg131
var
  I, RESTO : inteiro
início
  I ← 1
  enquanto (I < 200) faça
    RESTO ← I - 4 * (I div 4)
    se (RESTO = 0) então
      escreva I
    fim_se
    I ← I + 1
  fim_enquanto
fim
  
```

g) Diagrama de bloco

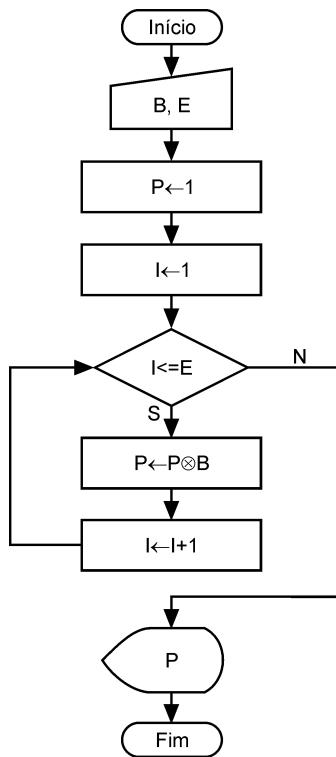


Português estruturado

```

programa Cap05_Ex1g_Pg131
var
  P, I : inteiro
início
  P ← 1
  I ← 0
  enquanto (I <= 15) faça
    escreva P
    P ← P * 3
    I ← I + 1
  fim_enquanto
fim
  
```

h) Diagrama de bloco



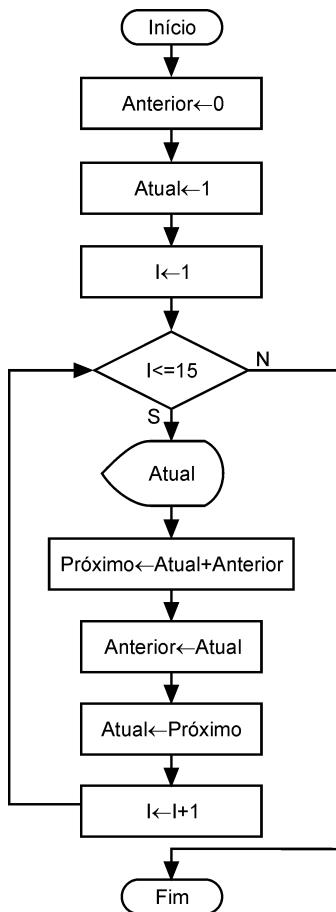
Português estruturado

```

programa Cap05_Ex1h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  I ← 1
  enquanto (I ≤ E) faça
    P ← P * B
    I ← I + 1
  fim_enquanto
  escreva P
fim
  
```

Tópico 5.9 - Exercício 1 - Página 132

i) Diagrama de bloco

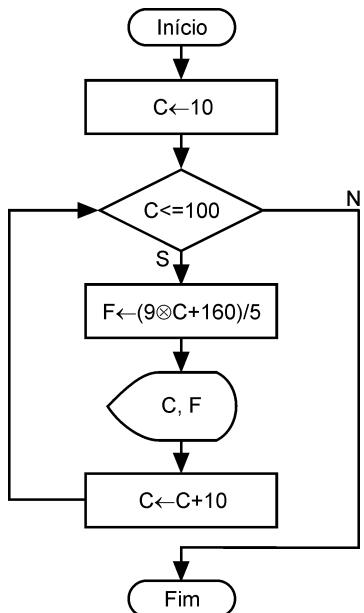


Português estruturado

```

programa Cap05_Ex1i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  I ← 1
  enquanto (I ≤ 15) faça
    escreva ATUAL
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
    I ← I + 1
  fim_enquanto
fim
  
```

j) Diagrama de bloco

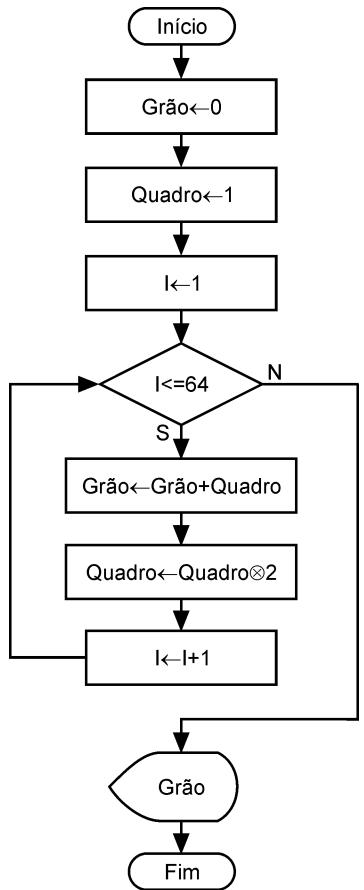


Português estruturado

```

programa Cap05_Ex1j_Pg132
var
  C, F : real
início
  C ← 10
  enquanto (C ≤ 100) faça
    F ← (9 * C + 160) / 5
    escreva C, F
    C ← C + 10
  fim_enquanto
fim
  
```

k) Diagrama de bloco

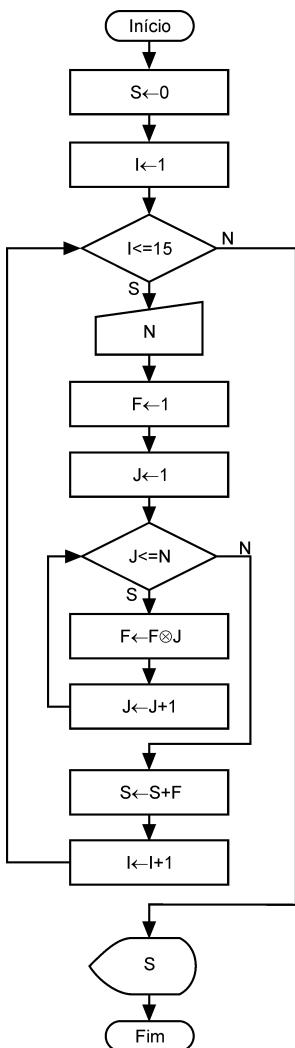


Português estruturado

```

programa Cap05_Ex1k_Pg132
var
  I, GRÃO, QUADRO : inteiro
início
  GRÃO ← 0
  QUADRO ← 1
  I ← 1
  enquanto (I ≤ 64) faça
    GRÃO ← GRÃO + QUADRO
    QUADRO ← QUADRO * 2
    I ← I + 1
  fim_enquanto
  escreva GRÃO
fim
  
```

I) Diagrama de bloco

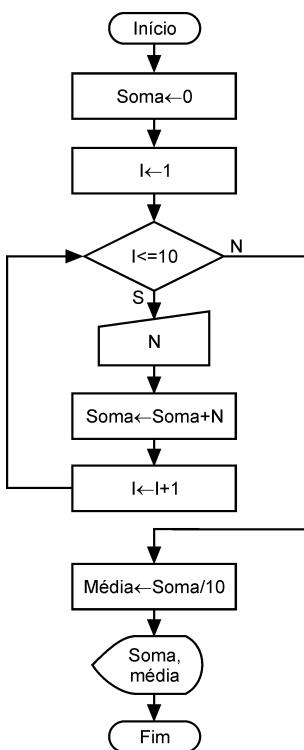


Português estruturado

```

programa Cap05_Ex11_Pg132
var
  I, J, N, S, F : inteiro
início
  S ← 0
  I ← 1
  enquanto (I ≤ 15) faça
    leia N
    F ← 1
    J ← 1
    enquanto (J ≤ N) faça
      F ← F * J
      J ← J + 1
    fim_enquanto
    S ← S + F
    I ← I + 1
  fim_enquanto
  escreva S
fim
  
```

m) Diagrama de bloco

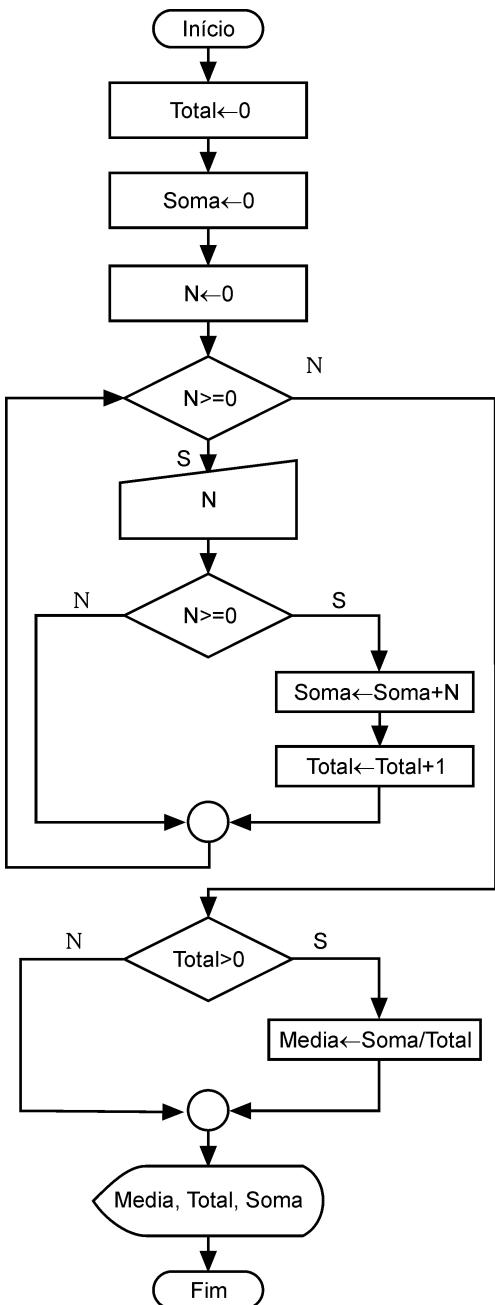


Português estruturado

```

programa Cap05_Ex1m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  I ← 1
  enquanto (I ≤ 10) faça
    leia N
    SOMA ← SOMA + N
    I ← I + 1
  fim_enquanto
  MÉDIA ← SOMA / 10
  escreva SOMA, MÉDIA
fim
  
```

n) Diagrama de bloco

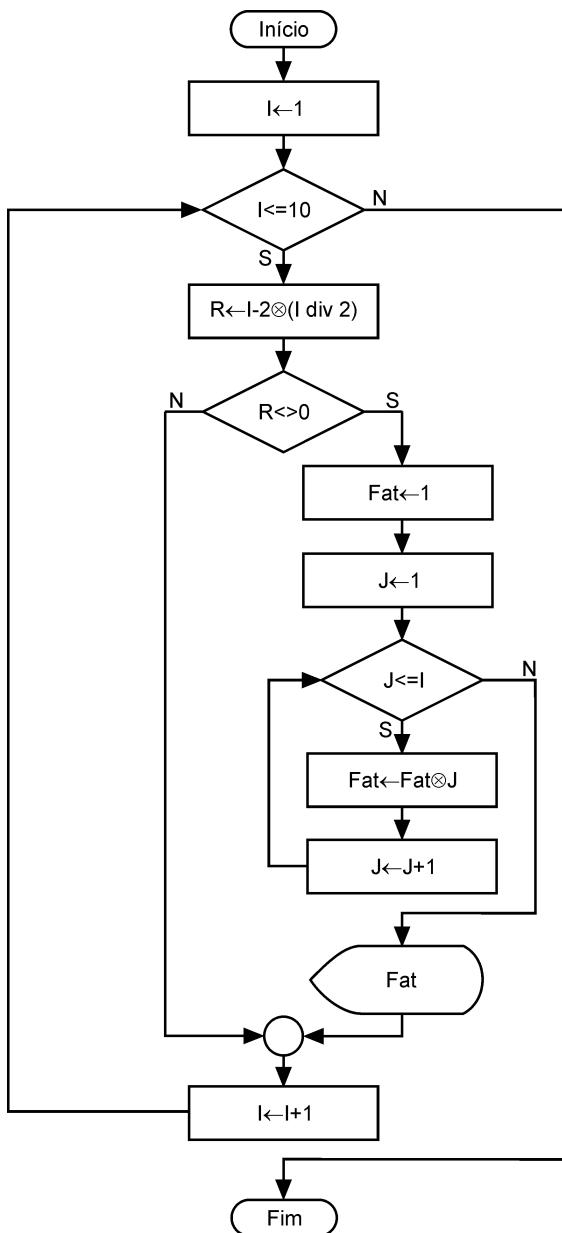


Português estruturado

```

programa Cap05_Ex1n_Pg132
var
  TOTAL : inteiro
  SOMA, MÉDIA, N : real
início
  TOTAL ← 0
  SOMA ← 0
  N ← 0
  enquanto (N >= 0) faça
    leia N
    se (N >= 0) então
      SOMA ← SOMA + N
      TOTAL ← TOTAL + 1
    fim_se
  fim enquanto
  se (TOTAL > 0) então
    MÉDIA ← SOMA / TOTAL
  fim_se
  escreva MÉDIA, TOTAL, SOMA
fim
  
```

o) Diagrama de bloco

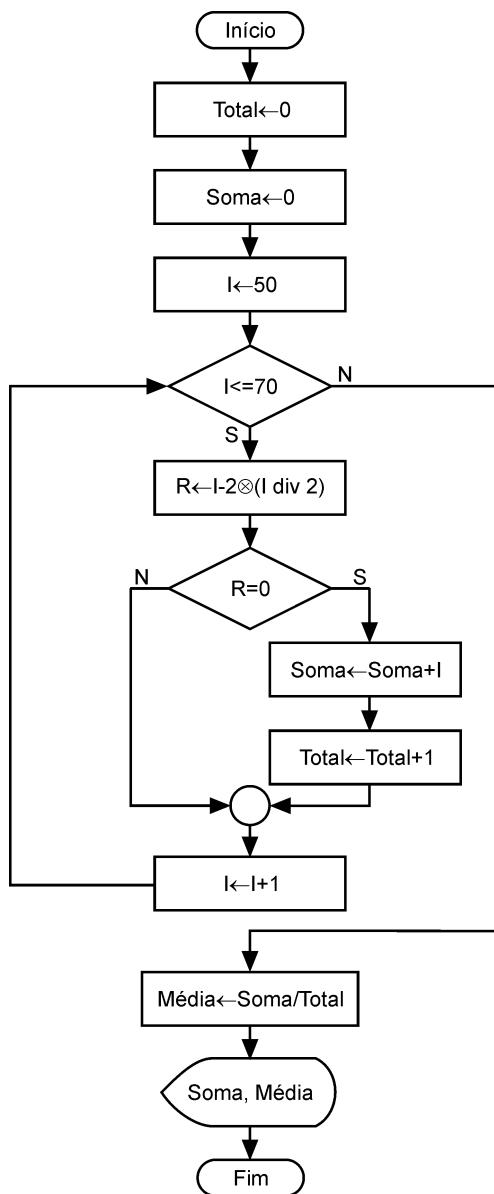


Português estruturado

```

programa Cap05_Ex1o_Pg132
var
  FAT, R, I, J : inteiro
início
  I ← 1
  enquanto (I ≤ 10) faça
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      FAT ← 1
      J ← 1
      enquanto (J ≤ I) faça
        FAT ← FAT * J
        J ← J + 1
      fim_enquanto
      escreva FAT
    fim_se
    I ← I + 1
  fim_enquanto
fim
  
```

p) Diagrama de bloco

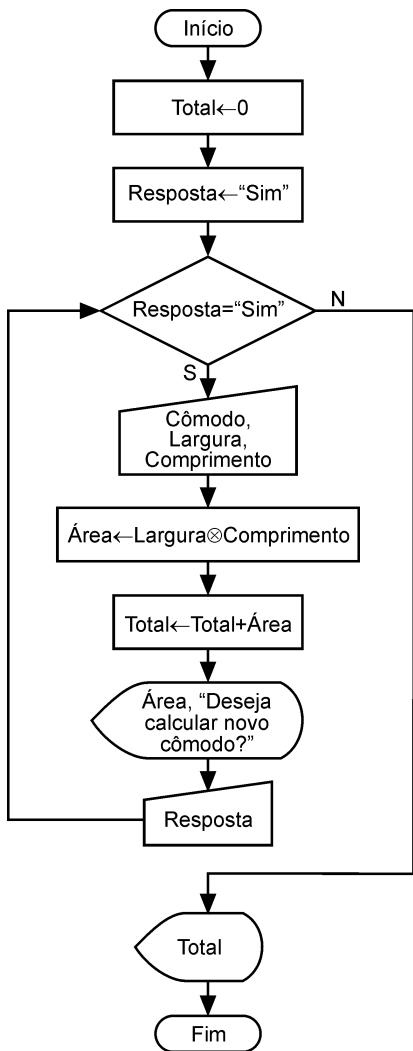


Português estruturado

```

programa Cap05_Ex1p_Pg132
var
  TOTAL, SOMA, R, I : inteiro
  MÉDIA : real
início
  TOTAL ← 0
  SOMA ← 0
  I ← 50
  enquanto (I ≤ 70) faça
    R ← I - 2 * (I div 2)
    se (R = 0) então
      SOMA ← SOMA + I
      TOTAL ← TOTAL + 1
    fim_se
    I ← I + 1
  fim_enquanto
  MÉDIA ← SOMA / TOTAL
  escreva SOMA, MÉDIA
fim
  
```

q) Diagrama de bloco

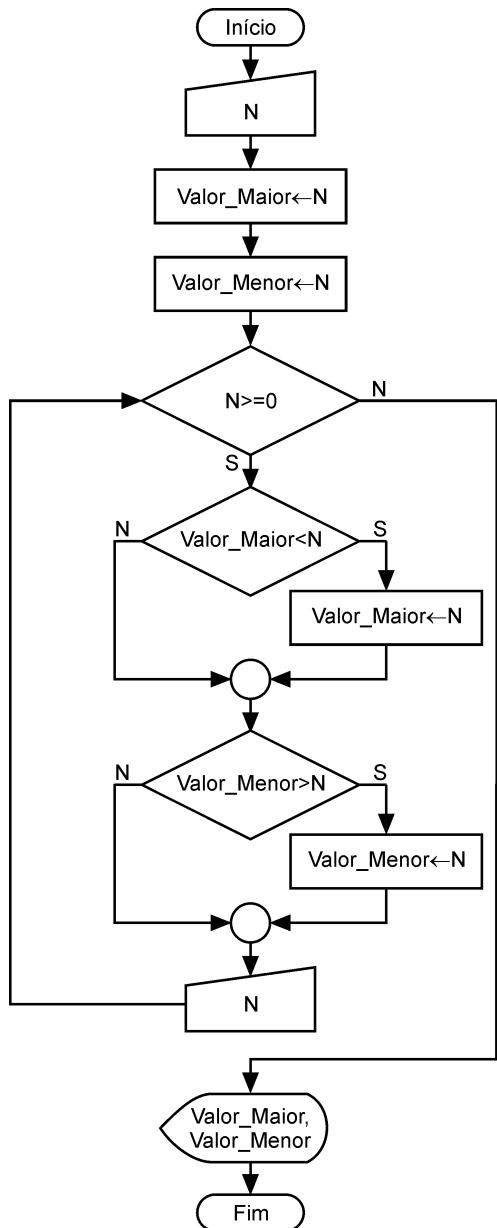


Português estruturado

```

programa Cap05_Ex1q_Pg132
var
  CÔMODO, RESPOSTA : caractere
  TOTAL, ÁREA, LARGURA, COMPRIMENTO : real
início
  TOTAL ← 0
  RESPOSTA ← "SIM"
  enquanto (RESPOSTA = "SIM") faça
    leia CÔMODO, LARGURA, COMPRIMENTO
    ÁREA ← LARGURA * COMPRIMENTO
    TOTAL ← TOTAL + ÁREA
    escreva ÁREA, "Deseja calcular novo cômodo? "
    leia RESPOSTA
  fim_enquanto
  escreva TOTAL
fim
  
```

r) Diagrama de bloco

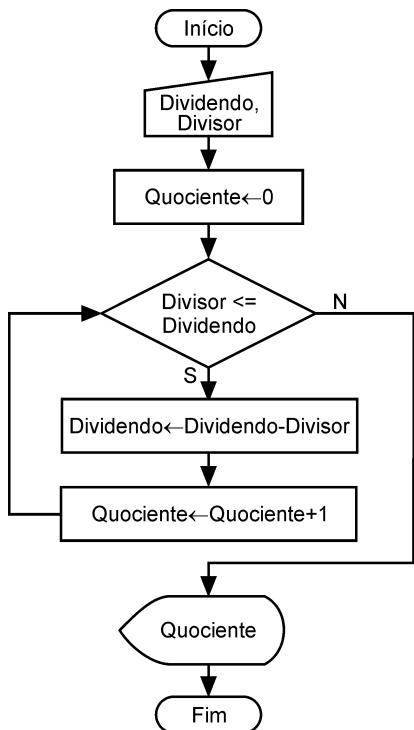


Português estruturado

```

programa Cap05_Ex1r_Pg132
var
    N, VALOR_MAIOR, VALOR_MENOR : inteiro
início
    leia N
    VALOR_MAIOR ← N
    VALOR_MENOR ← N
    enquanto (N ≥ 0) faça
        se (VALOR_MAIOR < N) então
            VALOR_MAIOR ← N
        fim_se
        se (VALOR_MENOR > N) então
            VALOR_MENOR ← N
        fim_se
        leia N
    fim_enquanto
    escreva VALOR_MAIOR, VALOR_MENOR
fim
    
```

s) Diagrama de bloco



Português estruturado

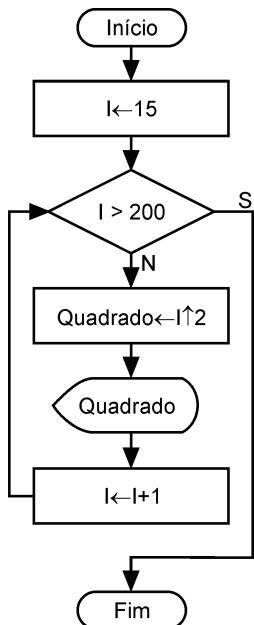
```

programa Cap05_Ex1s_Pg132
var
  QUOCIENTE, DIVIDENDO, DIVISOR : inteiro
início
  leia DIVIDENDO, DIVISOR
  QUOCIENTE ← 0
  enquanto (DIVISOR ≤ DIVIDENDO) faça
    DIVIDENDO ← DIVIDENDO - DIVISOR
    QUOCIENTE ← QUOCIENTE + 1
  fim_enquanto
  escreva QUOCIENTE
fim
  
```

5.2 - Laço de repetição pré-teste - controle condicional falso

Tópico 5.9 - Exercício 2 - Página 131

a) Diagrama de bloco

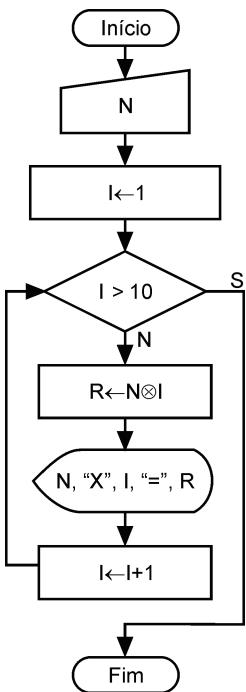


Português estruturado

```

programa Cap05_Ex2a_Pg131
var
  QUADRADO, I : inteiro
início
  I ← 15
  até_seja (I > 200) efetue
    QUADRADO ← I ↑ 2
    escreva QUADRADO
    I ← I + 1
  fim_até_seja
fim
  
```

b) Diagrama de bloco

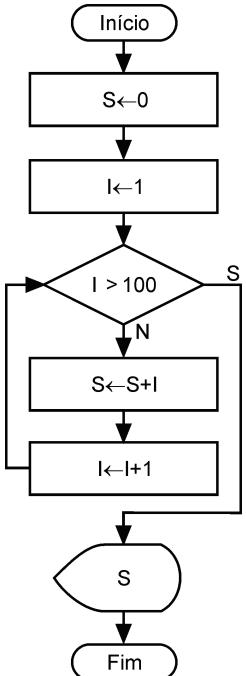


Português estruturado

```

programa Cap05_Ex2b_Pg131
var
  N, I, R : inteiro
início
  leia N
  I ← 1
  até_seja (I > 10) efetue
    R ← N * I
    escreva N, " X ", I, " = ", R
    I ← I + 1
  fim_até_seja
fim
  
```

c) Diagrama de bloco

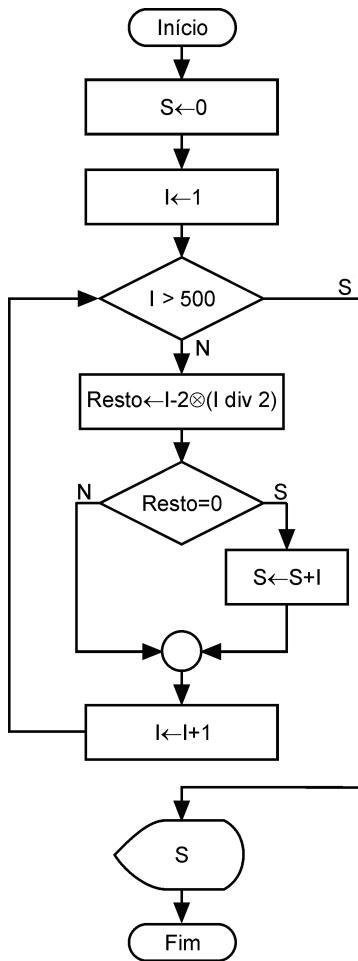


Português estruturado

```

programa Cap05_Ex2c_Pg131
var
  S, I : inteiro
início
  S ← 0
  I ← 1
  até_seja (I > 100) efetue
    S ← S + I
    I ← I + 1
  fim_até_seja
  escreva S
fim
  
```

d) Diagrama de bloco

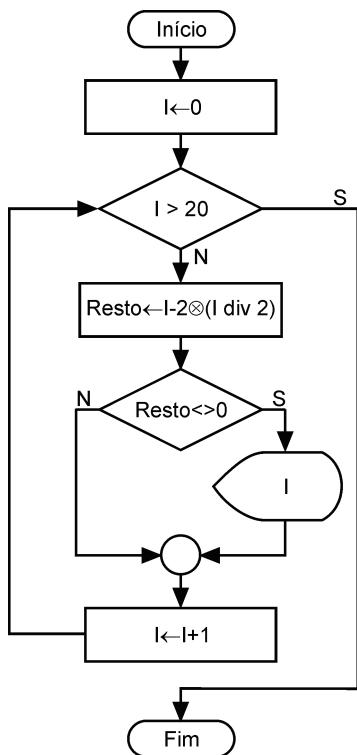


Português estruturado

```

programa Cap05_Ex2d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  I ← 1
  até_seja (I > 500) efetue
    RESTO ← I - 2 * (I div 2)
    se (RESTO = 0) então
      S ← S + I
    fim_se
    I ← I + 1
  fim_até_seja
  escreva S
fim
  
```

e) Diagrama de bloco

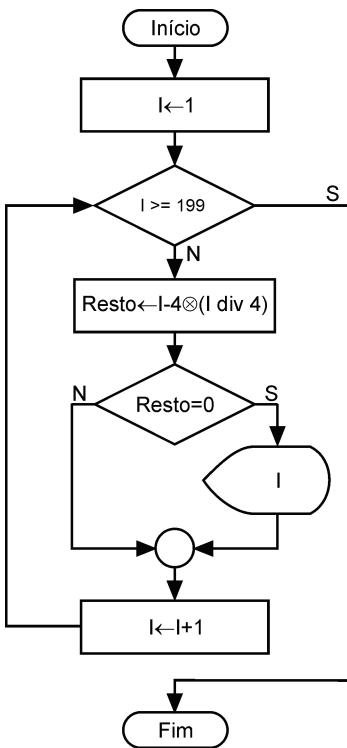


Português estruturado

```

programa Cap05_Ex2e_Pg131
var
  I, RESTO : inteiro
início
  I ← 0
  até_seja (I > 20) efetue
    RESTO ← I - 2 * (I div 2)
    se (RESTO <> 0) então
      escreva I
    fim_se
    I ← I + 1
  fim_até_seja
fim
  
```

f) Diagrama de bloco

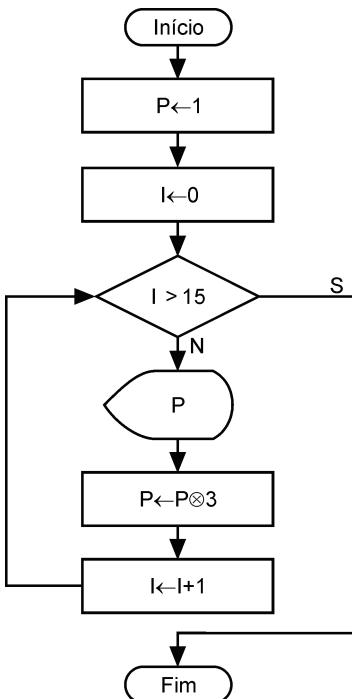


Português estruturado

```

programa Cap05_Ex2f_Pg131
var
  I, RESTO : inteiro
início
  I ← 1
  até_seja (I ≥ 199) efetue
    RESTO ← I - 4 * (I div 4)
    se (RESTO = 0) então
      escreva I
    fim_se
    I ← I + 1
  fim_até_seja
fim
  
```

g) Diagrama de bloco

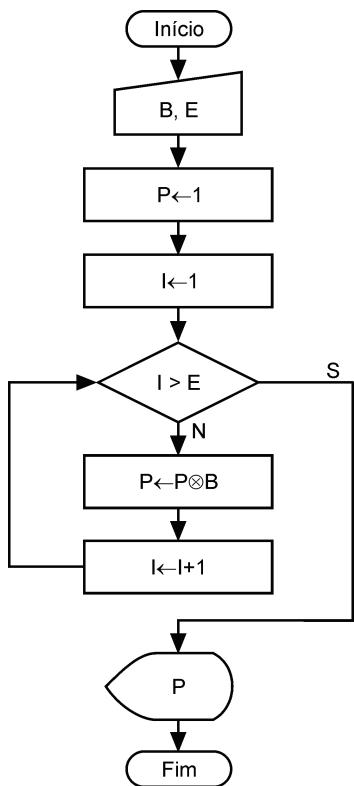


Português estruturado

```

programa Cap05_Ex2g_Pg131
var
  P, I : inteiro
início
  P ← 1
  I ← 0
  até_seja (I > 15) efetue
    escreva P
    P ← P * 3
    I ← I + 1
  fim_até_seja
fim
  
```

h) Diagrama de bloco

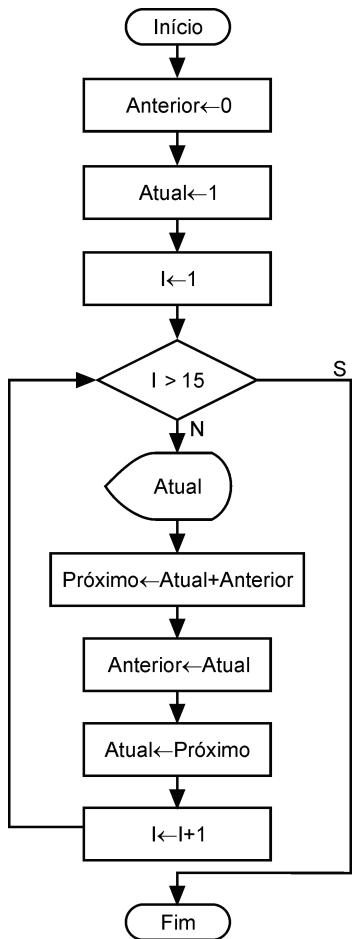


Português estruturado

```

programa Cap05_Ex2h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  I ← 1
  até_seja (I > E) efetue
    P ← P * B
    I ← I + 1
  fim_até_seja
  escreva P
fim
  
```

i) Diagrama de bloco

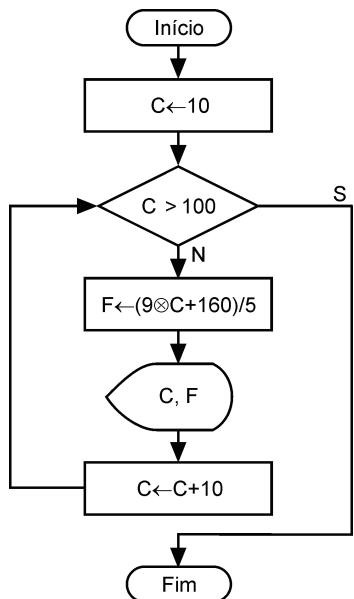


Português estruturado

```

programa Cap05_Ex2i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  I ← 1
  até_seja (I > 15) efetue
    escreva ATUAL
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
    I ← I + 1
  fim_até_seja
fim
  
```

j) Diagrama de bloco

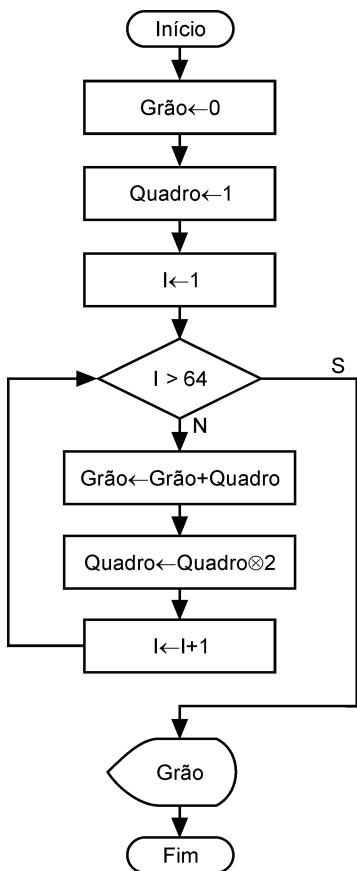


Português estruturado

```

programa Cap05_Ex2j_Pg132
var
  C, F : real
início
  C ← 10
  até_seja (C > 100) efetue
    F ← (9 * C + 160) / 5
    escreva C, F
    C ← C + 10
  fim_até_seja
fim
  
```

k) Diagrama de bloco

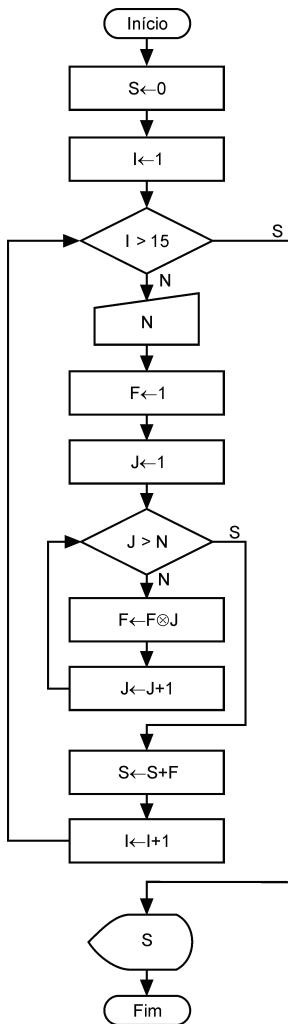


Português estruturado

```

programa Cap05_Ex2k_Pg132
var
  I, GRÃO, QUADRO : inteiro
início
  GRÃO ← 0
  QUADRO ← 1
  I ← 1
  até_seja (I > 64) efetue
    GRÃO ← GRÃO + QUADRO
    QUADRO ← QUADRO * 2
    I ← I + 1
  fim_até_seja
  escreva GRÃO
fim
  
```

I) Diagrama de bloco

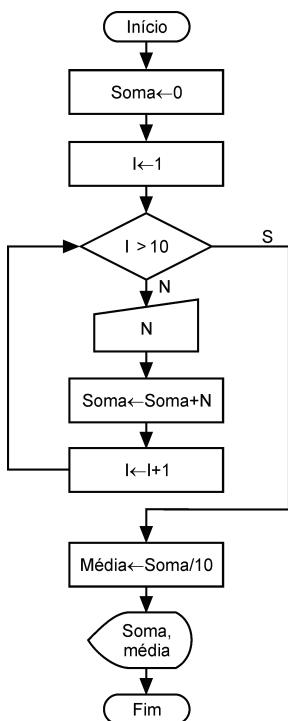


Português estruturado

```

programa Cap05_Ex21_Pg132
var
  I, J, N, S, F : inteiro
início
  S ← 0
  I ← 1
  até_seja (I > 15) efetue
    leia N
    F ← 1
    J ← 1
    até_seja (J > N) efetue
      F ← F * J
      J ← J + 1
    fim_até_seja
    S ← S + F
    I ← I + 1
  fim_até_seja
  escreva S
fim
  
```

m) Diagrama de bloco

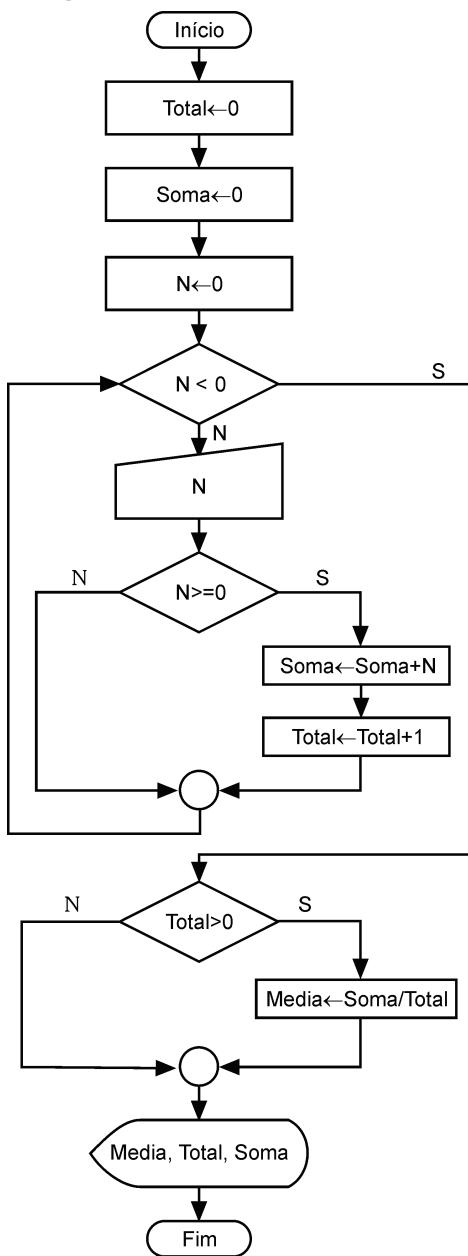


Português estruturado

```

programa Cap05_Ex2m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  I ← 1
  até_seja (I > 10) efetue
    leia N
    SOMA ← SOMA + N
    I ← I + 1
  fim_até_seja
  MÉDIA ← SOMA / 10
  escreva SOMA, MÉDIA
fim
  
```

n) Diagrama de bloco

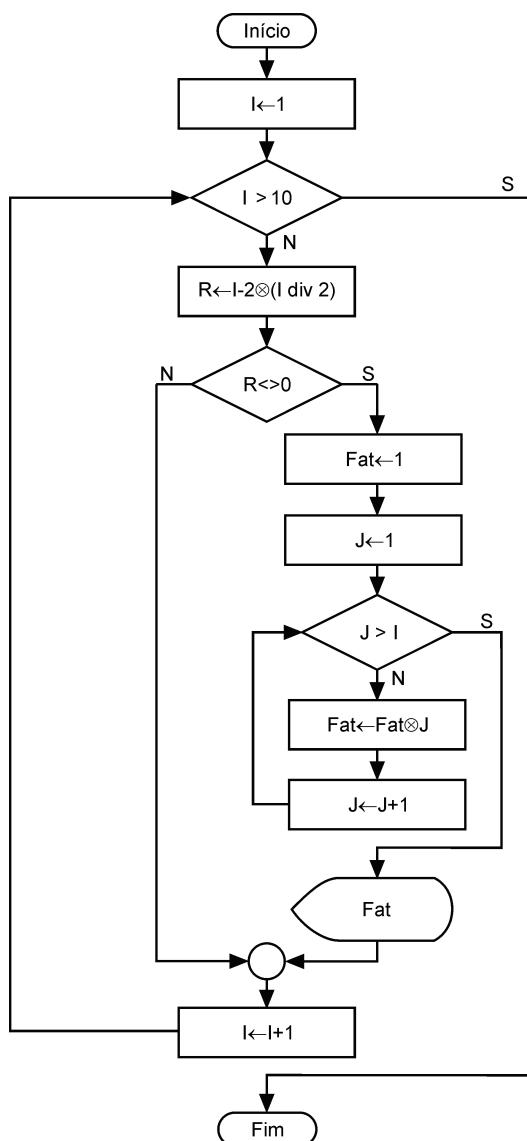


Português estruturado

```

programa Cap05_Ex2n_Pg132
var
    TOTAL : inteiro
    SOMA, MÉDIA, N : real
início
    TOTAL ← 0
    SOMA ← 0
    N ← 0
    até_seja (N < 0) efetue
        leia N
        se (N ≥ 0) então
            SOMA ← SOMA + N
            TOTAL ← TOTAL + 1
        fim_se
    fim_até_seja
    se (TOTAL > 0) então
        MÉDIA ← SOMA / TOTAL
    fim_se
    escreva MÉDIA, TOTAL, SOMA
fim
    
```

o) Diagrama de bloco

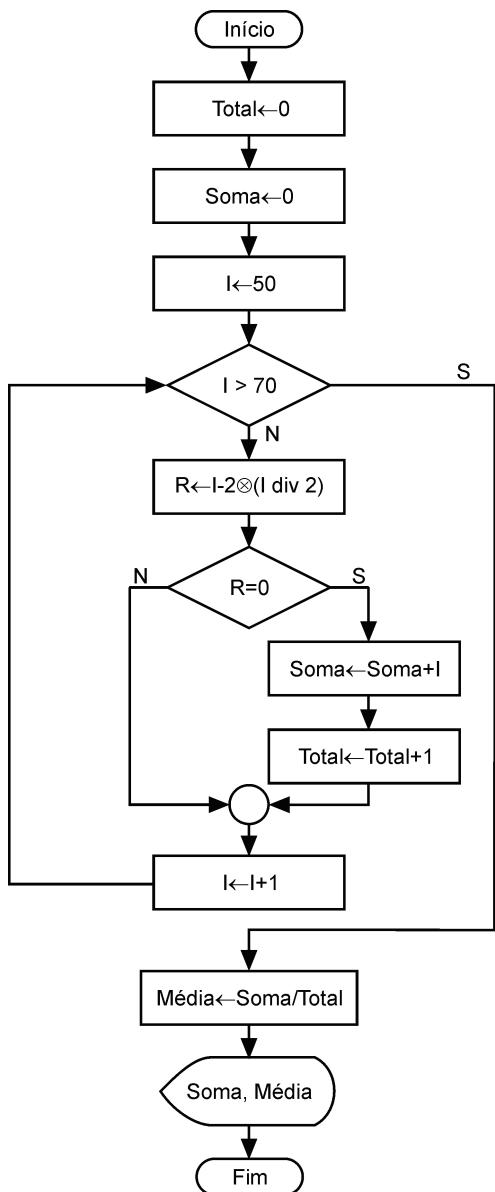


Português estruturado

```

programa Cap05_Ex2o_Pg132
var
  FAT, R, I, J : inteiro
início
  I ← 1
  até_seja (I > 10) efetue
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      FAT ← 1
      J ← 1
      até_seja (J > I) efetue
        FAT ← FAT * J
        J ← J + 1
      fim_até_seja
      escreva FAT
    fim_se
    I ← I + 1
  fim_até_seja
fim
  
```

p) Diagrama de bloco



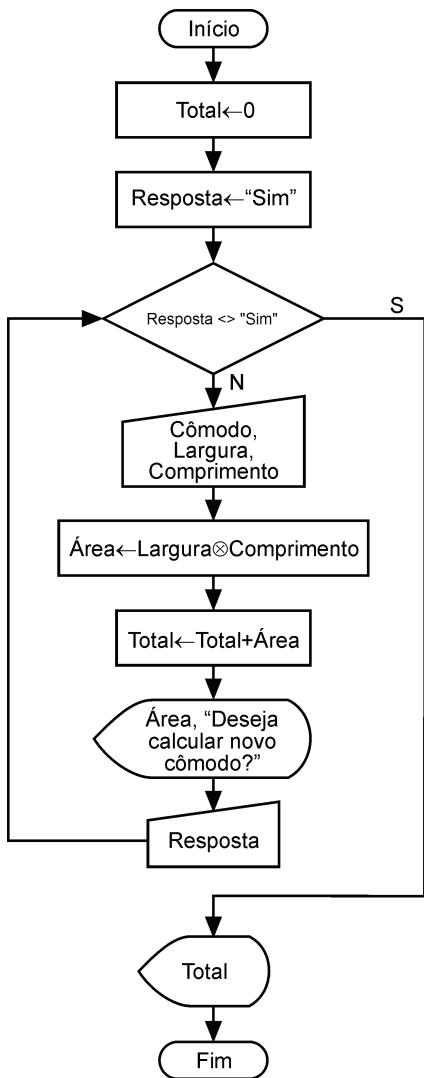
Português estruturado

```

programa Cap05_Ex2p_Pg132
var
    TOTAL, SOMA, R, I : inteiro
    MÉDIA : real
início
    TOTAL ← 0
    SOMA ← 0
    I ← 50
    até_seja (I > 70) efetue
        R ← I - 2 * (I div 2)
        se (R = 0) então
            SOMA ← SOMA + I
            TOTAL ← TOTAL + 1
        fim_se
        I ← I + 1
    fim_até_seja
    MÉDIA ← SOMA / TOTAL
    escreva SOMA, MÉDIA
fim

```

q) Diagrama de bloco

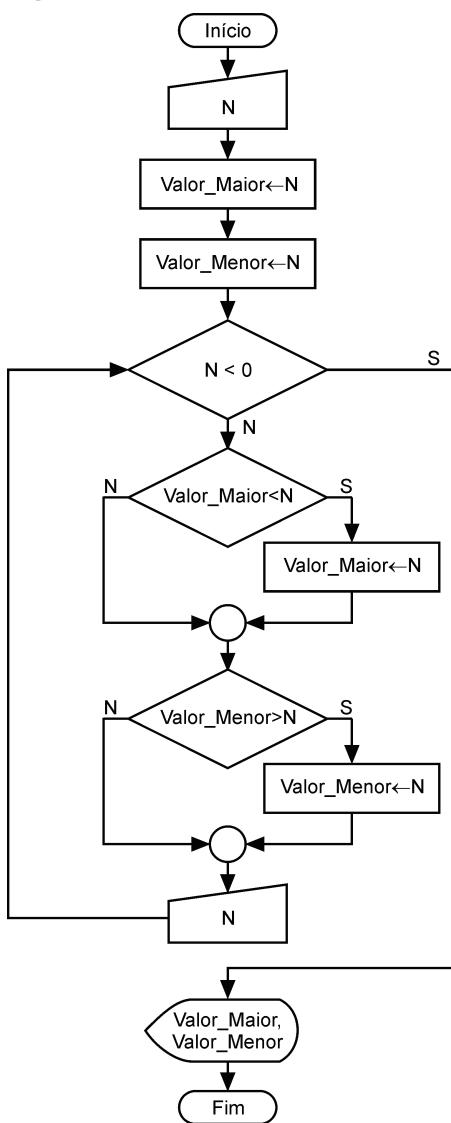


Português estruturado

```

programa Cap05_Ex2q_Pg132
var
  CÔMODO, RESPOSTA : cadeia
  TOTAL, ÁREA, LARGURA, COMPRIMENTO : real
início
  TOTAL ← 0
  RESPOSTA ← "SIM"
  até_seja (RESPOSTA <> "SIM") efetue
    leia CÔMODO, LARGURA, COMPRIMENTO
    ÁREA ← LARGURA * COMPRIMENTO
    TOTAL ← TOTAL + ÁREA
    escreva ÁREA, "Deseja calcular novo cômodo? "
    leia RESPOSTA
  fim_até_seja
  escreva TOTAL
fim
  
```

r) Diagrama de bloco

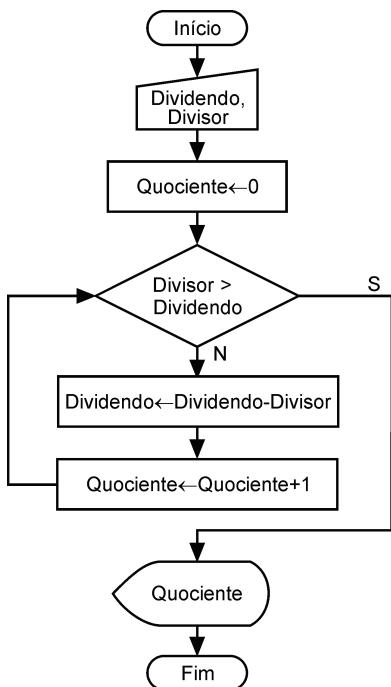


Português estruturado

```

programa Cap05_Ex2r_Pg132
var
    N, VALOR_MAIOR, VALOR_MENOR : inteiro
início
    leia N
    VALOR_MAIOR ← N
    VALOR_MENOR ← N
    até_seja (N < 0) efetue
        se (VALOR_MAIOR < N) então
            VALOR_MAIOR ← N
        fim_se
        se (VALOR_MENOR > N) então
            VALOR_MENOR ← N
        fim_se
    leia N
    fim_até_seja
    escreva VALOR_MAIOR, VALOR_MENOR
fim
    
```

s) Diagrama de bloco



Português estruturado

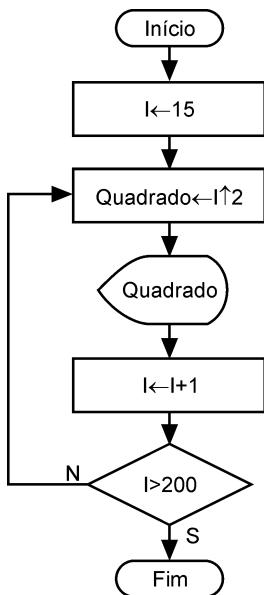
```

programa Cap05_Ex2s_Pg132
var
    QUOCIENTE, DIVIDENDO, DIVISOR : inteiro
início
    leia DIVIDENDO, DIVISOR
    QUOCIENTE ← 0
    até_seja (DIVISOR > DIVIDENDO) efetue
        DIVIDENDO ← DIVIDENDO - DIVISOR
        QUOCIENTE ← QUOCIENTE + 1
    fim_até_seja
    escreva QUOCIENTE
fim
    
```

5.3 - Laço de repetição pós-teste - controle condicional falso

Tópico 5.9 - Exercício 3 - Página 131

a) Diagrama de blocos

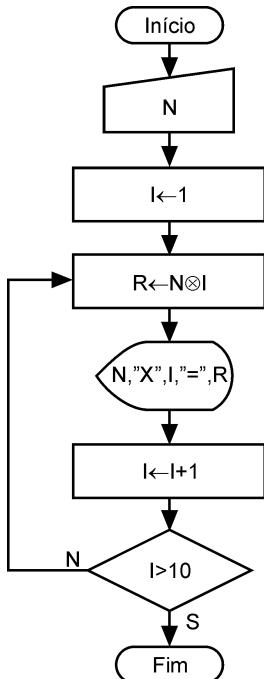


Português estruturado

```

programa Cap05_Ex3a_Pg131
var
  QUADRADO, I : inteiro
início
  I ← 15
  repita
    QUADRADO ← I ↑ 2
    escreva QUADRADO
    I ← I + 1
  até_que (I > 200)
fim
  
```

b) Diagrama de blocos

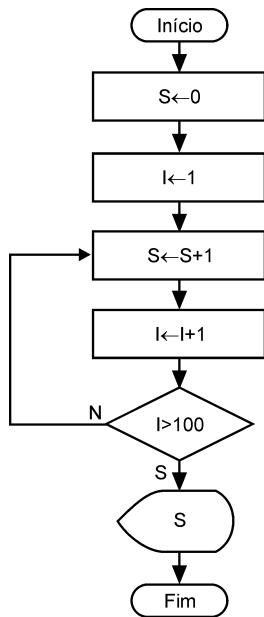


Português Estruturado

```

programa Cap05_Ex3b_Pg131
var
  N, I, R : inteiro
início
  leia N
  I ← 1
  repita
    R ← N * I
    escreva N, " X ", I, " = ", R
    I ← I + 1
  até_que (I > 10)
fim
  
```

c) Diagrama de blocos

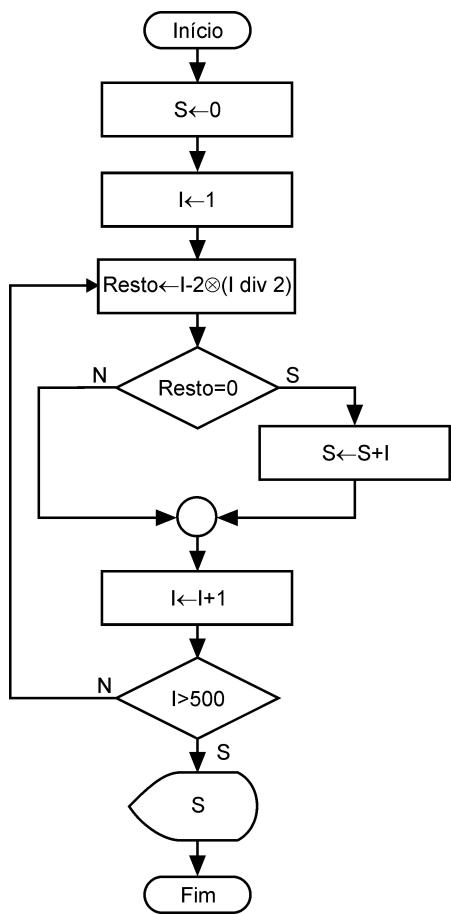


Português Estruturado

```

programa Cap05_Ex3c_Pg131
var
  S, I : inteiro
início
  S ← 0
  I ← 1
repita
  S ← S + I
  I ← I + 1
até_que (I > 100)
escreva S
fim
  
```

d) Diagrama de blocos

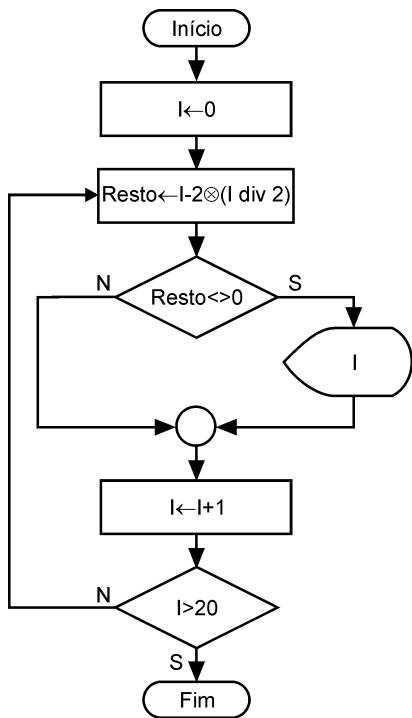


Português Estruturado

```

programa Cap05_Ex3d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  I ← 1
repita
  RESTO ← I - 2 * (I div 2)
  se (RESTO = 0) então
    S ← S + I
  fim_se
  I ← I + 1
até_que (I > 500)
escreva S
fim
  
```

e) Diagrama de blocos

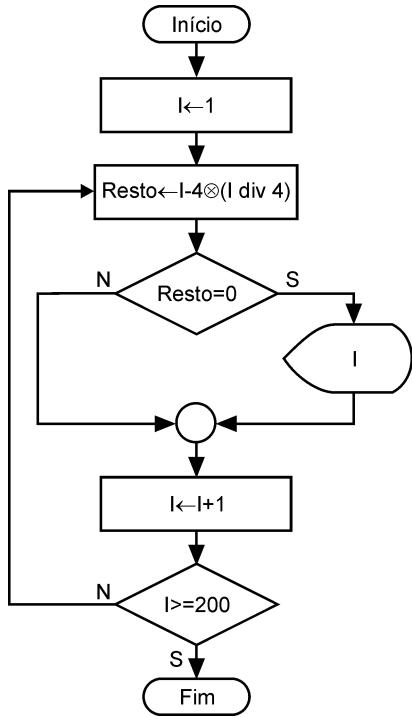


Português Estruturado

```

programa Cap05_Ex3e_Pg131
var
  I, RESTO : inteiro
início
  I ← 0
repita
  RESTO ← I - 2 * (I div 2)
  se (RESTO <> 0) então
    escreva I
  fim_se
  I ← I + 1
até_que (I > 20)
fim
  
```

f) Diagrama de blocos

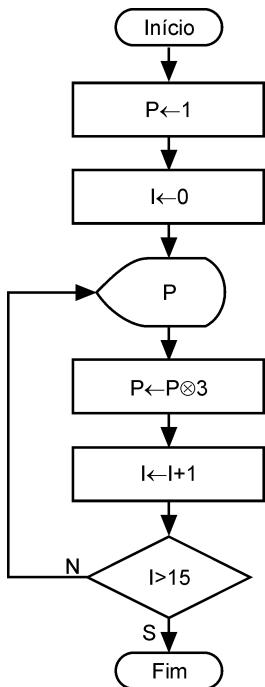


Português Estruturado

```

programa Cap05_Ex3f_Pg131
var
  I, RESTO : inteiro
início
  I ← 1
repita
  RESTO ← I - 4 * (I div 4)
  se (RESTO = 0) então
    escreva I
  fim_se
  I ← I + 1
até_que (I >= 200)
fim
  
```

g) Diagrama de blocos

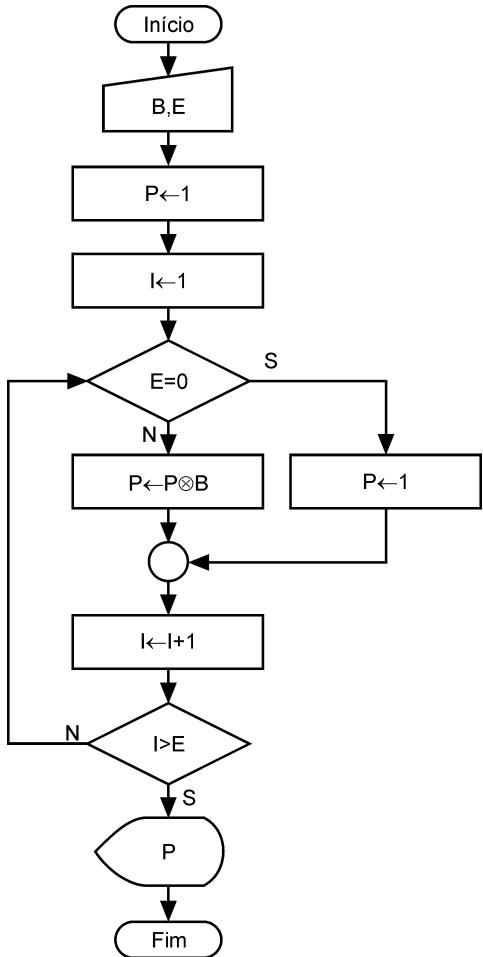


Português Estruturado

```

programa Cap05_Ex3g_Pg131
var
  P, I : inteiro
início
  P ← 1
  I ← 0
repita
  escreva P
  P ← P * 3
  I ← I + 1
até que (I > 15)
fim
  
```

h) Diagrama de blocos



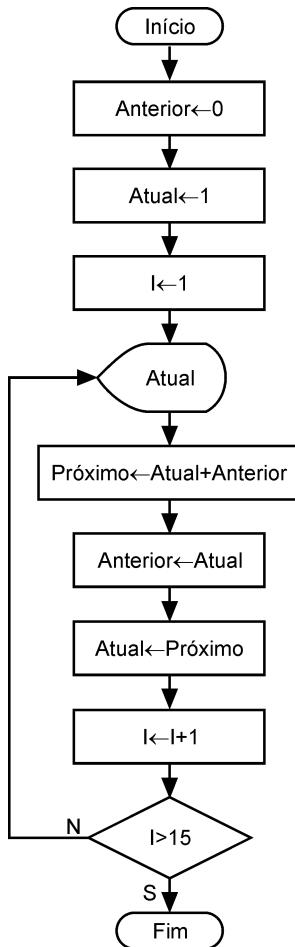
Português Estruturado

```

programa Cap05_Ex3h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  I ← 1
repita
  se (E = 0) então
    P ← 1
  senão
    P ← P * B
  fim-se
  I ← I + 1
até que (I > E)
escreva P
fim
  
```

Tópico 5.9 - Exercício 3 - Página 132

i) Diagrama de blocos

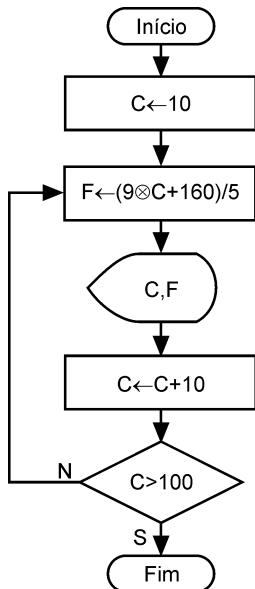


Português Estruturado

```

programa Cap05_Ex3i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  I ← 1
repita
  escreva ATUAL
  PRÓXIMO ← ATUAL + ANTERIOR
  ANTERIOR ← ATUAL
  ATUAL ← PRÓXIMO
  I ← I + 1
até_que (I > 15)
fim
  
```

j) Diagrama de blocos

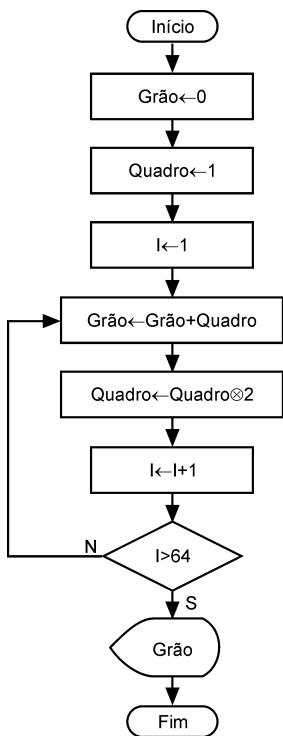


Português estruturado

```

programa Cap05_Ex3j_Pg132
var
  C, F : real
início
  C ← 10
repita
  F ← (9 * C + 160) / 5
  escreva C, F
  C ← C + 10
até_que (C > 100)
fim
  
```

k) Diagrama de blocos

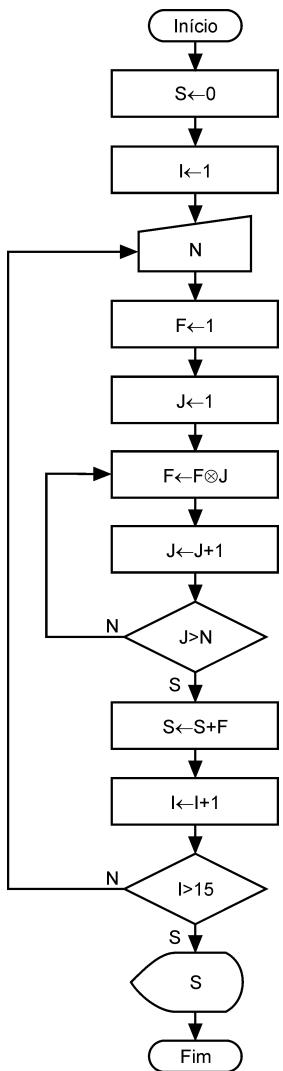


Português estruturado

```

programa Cap05_Ex3k_Pg132
var
  I, GRÃO, QUADRO : inteiro
início
  GRÃO ← 0
  QUADRO ← 1
  I ← 1
repita
  GRÃO ← GRÃO + QUADRO
  QUADRO ← QUADRO * 2
  I ← I + 1
até_que (I > 64)
escreva GRÃO
fim
  
```

l) Diagrama de blocos

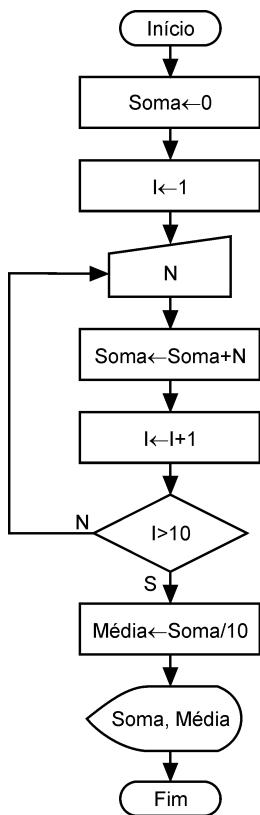


Português estruturado

```

programa Cap05_Ex3l_Pg132
var
  I, J, N, S, F : inteiro
início
  S ← 0
  I ← 1
repita
  leia N
  F ← 1
  J ← 1
repita
  F ← F * J
  J ← J + 1
até_que (J > N)
  S ← S + F
  I ← I + 1
até_que (I > 15)
escreva S
fim
  
```

m) Diagrama de blocos

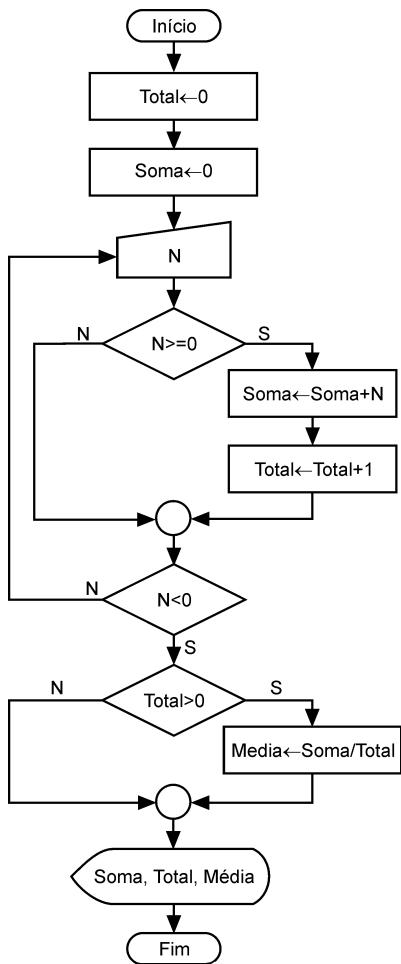


Português estruturado

```

programa Cap05_Ex3m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  I ← 1
repita
  leia N
  SOMA ← SOMA + N
  I ← I + 1
até que (I > 10)
MÉDIA ← SOMA / 10
escreva SOMA, MÉDIA
fim
  
```

n) Diagrama de blocos

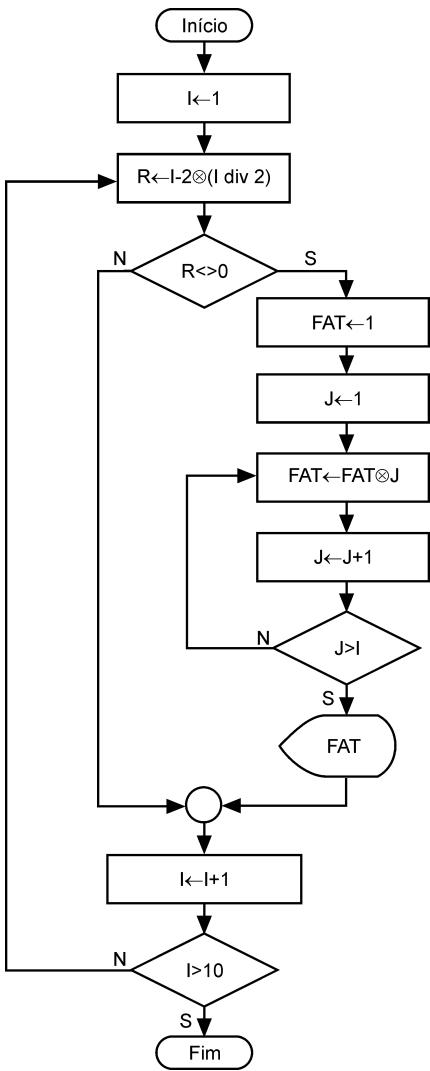


Português estruturado

```

programa Cap05_Ex3n_Pg132
var
  TOTAL : inteiro
  SOMA, MÉDIA, N : real
início
  TOTAL ← 0
  SOMA ← 0
repita
  leia N
  se (N >= 0) então
    SOMA ← SOMA + N
    TOTAL ← TOTAL + 1
  fim_se
até que (N < 0)
se (TOTAL > 0) então
  MÉDIA ← SOMA / TOTAL
fim_se
escreva SOMA, TOTAL, MÉDIA
fim
  
```

o) Diagrama de blocos

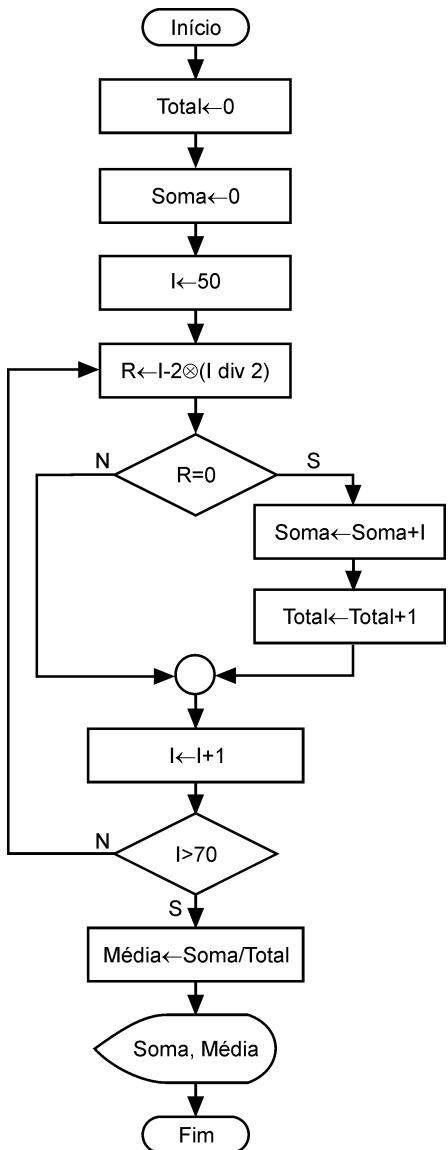


Português estruturado

```

programa Cap05_Ex3o_Pg132
var
  FAT, R, I, J : inteiro
início
  I ← 1
  repita
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      FAT ← 1
      J ← 1
      repita
        FAT ← FAT * J
        J ← J + 1
      até que (J > I)
      escreva FAT
    fim_se
    I ← I + 1
  até que (I > 10)
fim
  
```

p) Diagrama de blocos

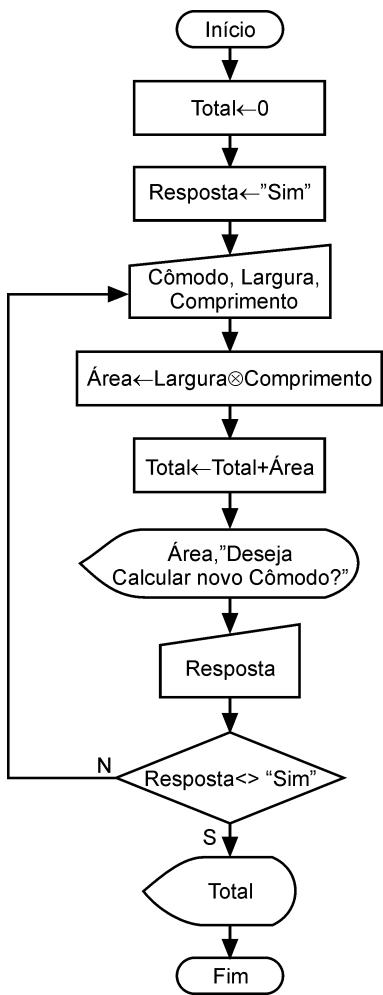


Português estruturado

```

programa Cap05_Ex3p_Pg132
var
  TOTAL, SOMA, R, I : inteiro
  MÉDIA : real
início
  TOTAL ← 0
  SOMA ← 0
  I ← 50
repita
  R ← I - 2 * (I div 2)
  se (R = 0) então
    SOMA ← SOMA + I
    TOTAL ← TOTAL + 1
  fim_se
  I ← I + 1
até_que (I > 70)
MÉDIA ← SOMA / TOTAL
escreva SOMA, MÉDIA
fim
  
```

q) Diagrama de blocos

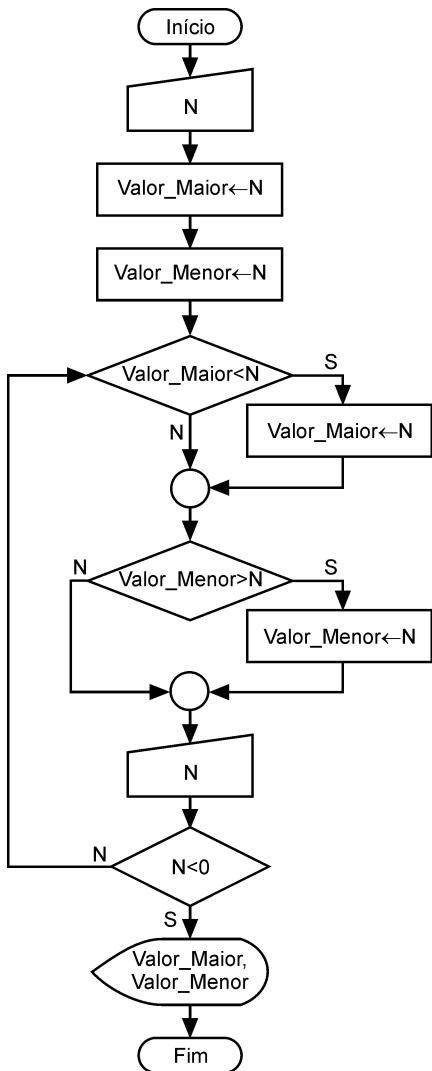


Português estruturado

```

programa Cap05_Ex3q_Pg132
var
    CÔMODO, RESPOSTA : cadeia
    TOTAL, ÁREA, LARGURA, COMPRIMENTO : real
início
    TOTAL ← 0
    RESPOSTA ← "SIM"
repita
    leia CÔMODO, LARGURA, COMPRIMENTO
    ÁREA ← LARGURA * COMPRIMENTO
    TOTAL ← TOTAL + ÁREA
    escreva ÁREA, "Deseja calcular novo cômodo? "
    leia RESPOSTA
até_QUE (RESPOSTA <> "SIM")
escreva TOTAL
fim
    
```

r) Diagrama de blocos

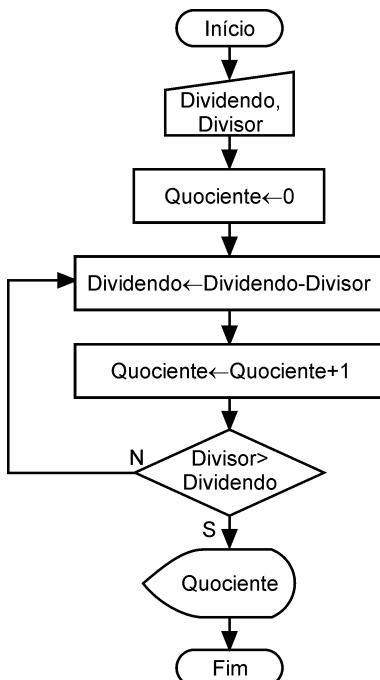


Português estruturado

```

programa Cap05_Ex3r_Pg132
var
    N, VALOR_MAIOR, VALOR_MENOR : inteiro
início
    leia N
    VALOR_MAIOR ← N
    VALOR_MENOR ← N
    repita
        se (VALOR_MAIOR < N) então
            VALOR_MAIOR ← N
        fim_se
        se (VALOR_MENOR > N) então
            VALOR_MENOR ← N
        fim_se
        leia N
        até_que (N < 0)
        escreva VALOR_MAIOR, VALOR_MENOR
fim
    
```

s) Diagrama de blocos



Português estruturado

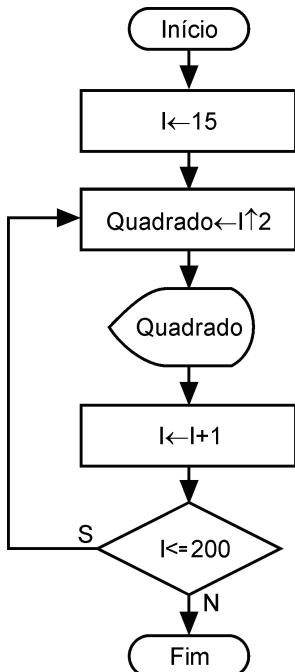
```

programa Cap05_Ex3s_Pg132
var
    QUOCIENTE, DIVIDENDO, DIVISOR : inteiro
início
    leia DIVIDENDO, DIVISOR
    QUOCIENTE ← 0
    repita
        DIVIDENDO ← DIVIDENDO - DIVISOR
        QUOCIENTE ← QUOCIENTE + 1
        até_que (DIVISOR > DIVIDENDO)
        escreva QUOCIENTE
fim
    
```

5.4 - Laço de repetição pós-teste - controle condicional verdadeiro

Tópico 5.9 - Exercício 4 - Página 131

a) Diagrama de blocos

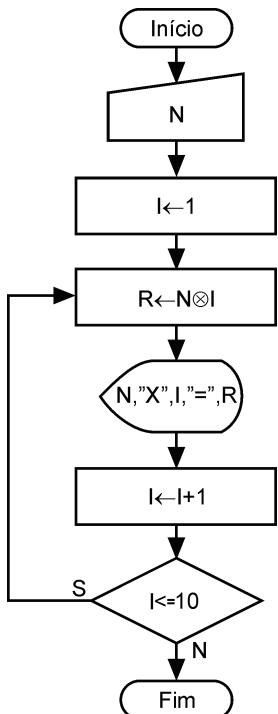


Português estruturado

```

programa Cap05_Ex4a_Pg131
var
    QUADRADO, I : inteiro
início
    I ← 15
    continua
        QUADRADO ← I ↑ 2
        escreva QUADRADO
        I ← I + 1
    enquanto_for (I <= 200)
fim
    
```

b) Diagrama de blocos

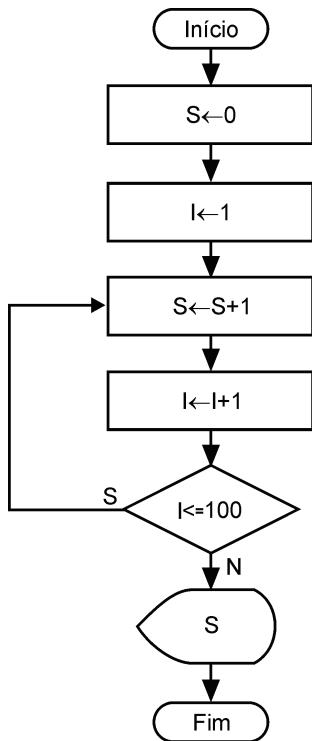


Português estruturado

```

programa Cap05_Ex4b_Pg131
var
    N, I, R : inteiro
início
    leia N
    I ← 1
    continua
        R ← N * I
        escreva N, " X ", I, " = ", R
        I ← I + 1
    enquanto_for (I <= 10)
fim
    
```

c) Diagrama de blocos

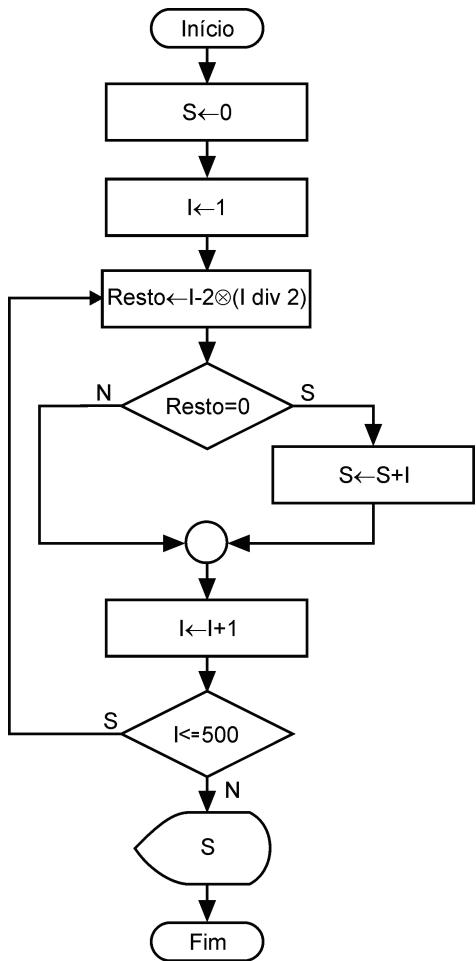


Português estruturado

```

programa Cap05_Ex4c_Pg131
var
  S, I : inteiro
início
  S ← 0
  I ← 1
  continua
    S ← S + I
    I ← I + 1
  enquanto_for (I ≤ 100)
    escreva S
  fim
  
```

d) Diagrama de blocos

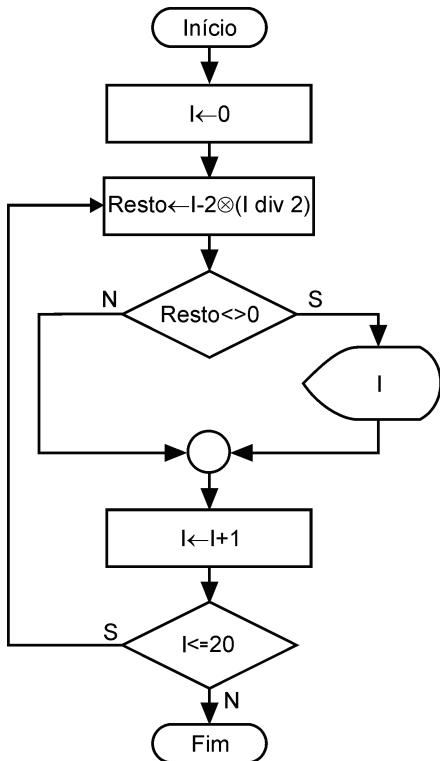


Português estruturado

```

programa Cap05_Ex4d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  I ← 1
  continua
    RESTO ← I - 2 * (I div 2)
    se (RESTO = 0) então
      S ← S + I
    fim_se
    I ← I + 1
  enquanto_for (I ≤ 500)
  escreva S
fim
  
```

e) Diagrama de blocos

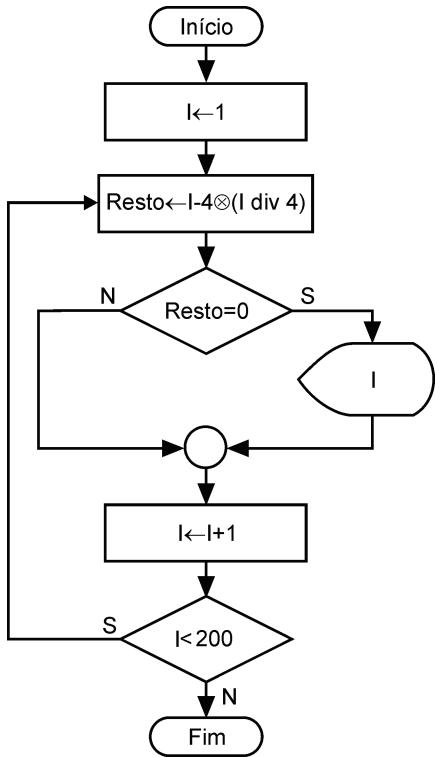


Português estruturado

```

programa Cap05_Ex4e_Pg131
var
  I, RESTO : inteiro
início
  I ← 0
  continua
    RESTO ← I - 2 * (I div 2)
    se (RESTO <> 0) então
      escreva I
    fim_se
    I ← I + 1
  enquanto_for (I <= 20)
fim
  
```

f) Diagrama de blocos

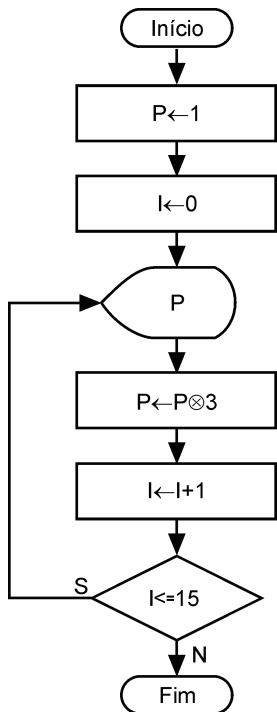


Português estruturado

```

programa Cap05_Ex4f_Pg131
var
  I, RESTO : inteiro
início
  I ← 1
  continua
    RESTO ← I - 4 * (I div 4)
    se (RESTO = 0) então
      escreva I
    fim_se
    I ← I + 1
  enquanto_for (I < 200)
fim
  
```

g) Diagrama de blocos

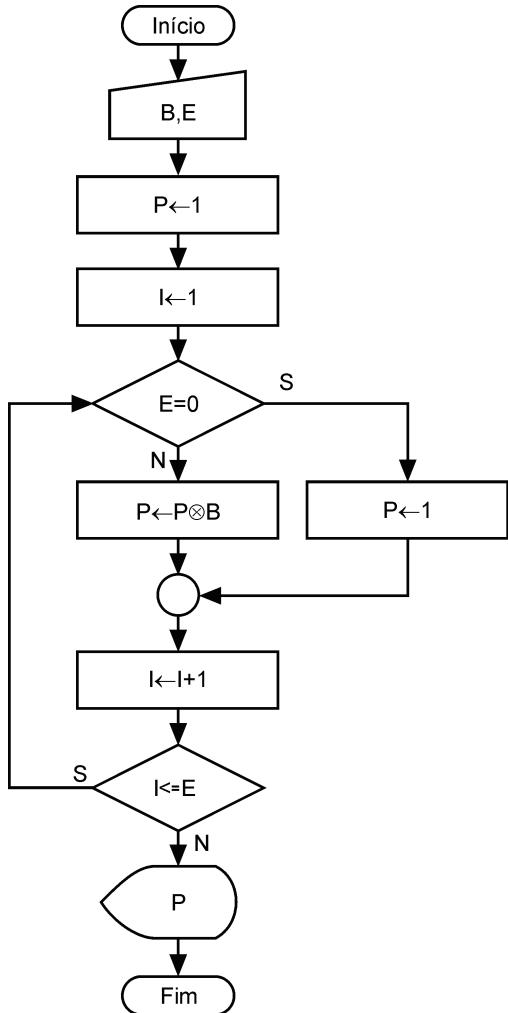


Português estruturado

```

programa Cap05_Ex4g_Pg131
var
  P, I : inteiro
início
  P ← 1
  I ← 0
  continua
    escreva P
    P ← P * 3
    I ← I + 1
  enquanto_for (I ≤ 15)
fim
  
```

h) Diagrama de blocos

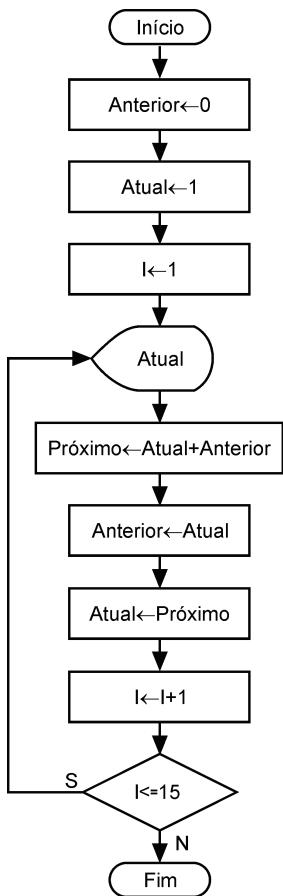


Português estruturado

```

programa Cap05_Ex4h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  I ← 1
  continua
    se (E = 0) então
      P ← 1
    senão
      P ← P * B
    fim-se
    I ← I + 1
  enquanto_for (I ≤ E)
  escreva P
fim
  
```

i) Diagrama de blocos

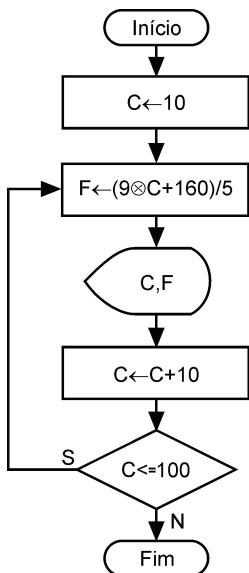


Português estruturado

```

programa Cap05_Ex4i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  I ← 1
  continua
    escreva ATUAL
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
    I ← I + 1
  enquanto_for (I <= 15)
fim
  
```

j) Diagrama de blocos

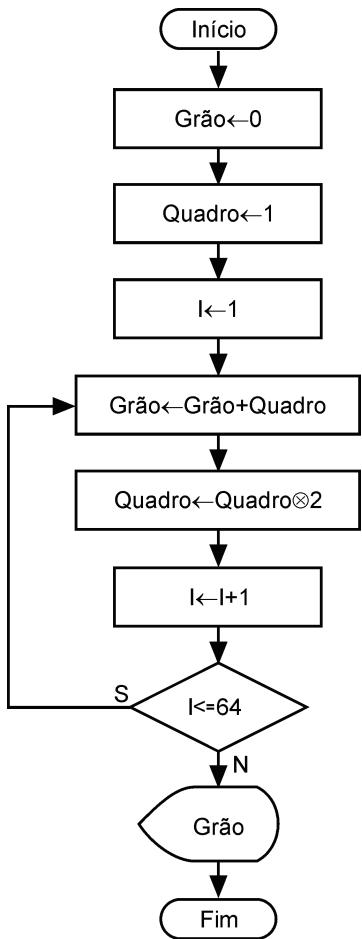


Português estruturado

```

programa Cap05_Ex4j_Pg132
var
  C, F : real
início
  C ← 10
  continua
    F ← (9 * C + 160) / 5
    escreva C, F
    C ← C + 10
  enquanto_for (C <= 100)
fim
  
```

k) Diagrama de blocos

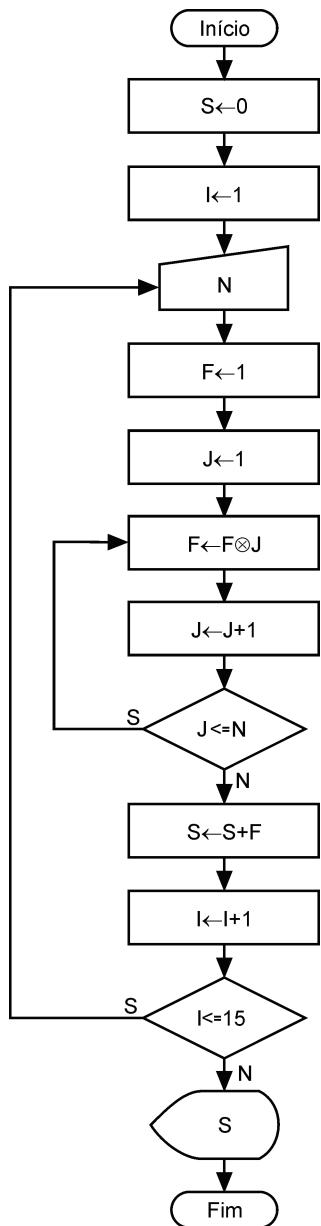


Português estruturado

```

programa Cap05_Ex4k_Pg132
var
  I, GRÃO, QUADRO : inteiro
início
  GRÃO ← 0
  QUADRO ← 1
  I ← 1
  continua
    GRÃO ← GRÃO + QUADRO
    QUADRO ← QUADRO * 2
    I ← I + 1
  enquanto_for (I <= 64)
    escreva GRÃO
fim
  
```

I) Diagrama de blocos

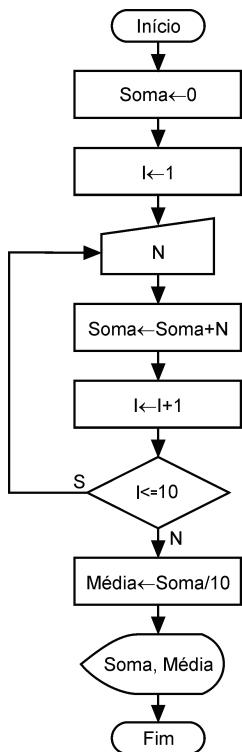


Português estruturado

```

programa Cap05_Ex41_Pg132
var
  I, J, N, S, F : inteiro
início
  S ← 0
  I ← 1
  continua
    leia N
    F ← 1
    J ← 1
    continua
      F ← F * J
      J ← J + 1
    enquanto_for (J <= N)
      S ← S + F
      I ← I + 1
    enquanto_for (I <= 15)
    escreva S
fim
  
```

m) Diagrama de blocos

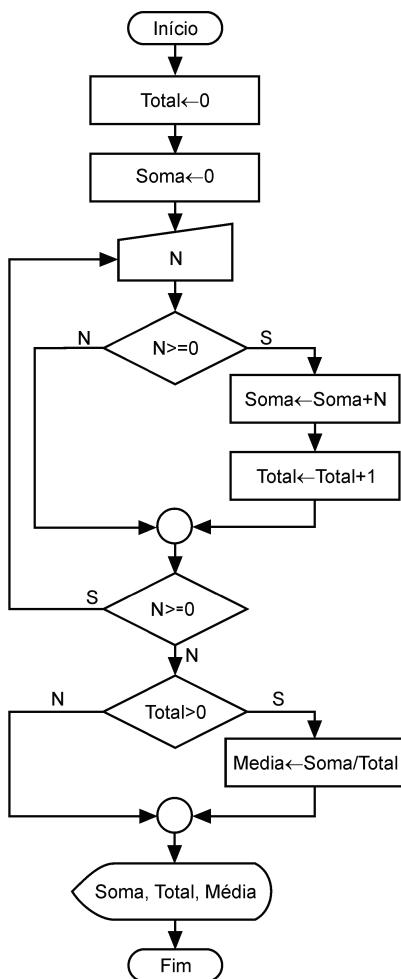


Português estruturado

```

programa Cap05_Ex4m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  I ← 1
  continua
  leia N
  SOMA ← SOMA + N
  I ← I + 1
enquanto_for (I ≤ 10)
  MÉDIA ← SOMA / 10
  escreva SOMA, MÉDIA
fim
  
```

n) Diagrama de blocos

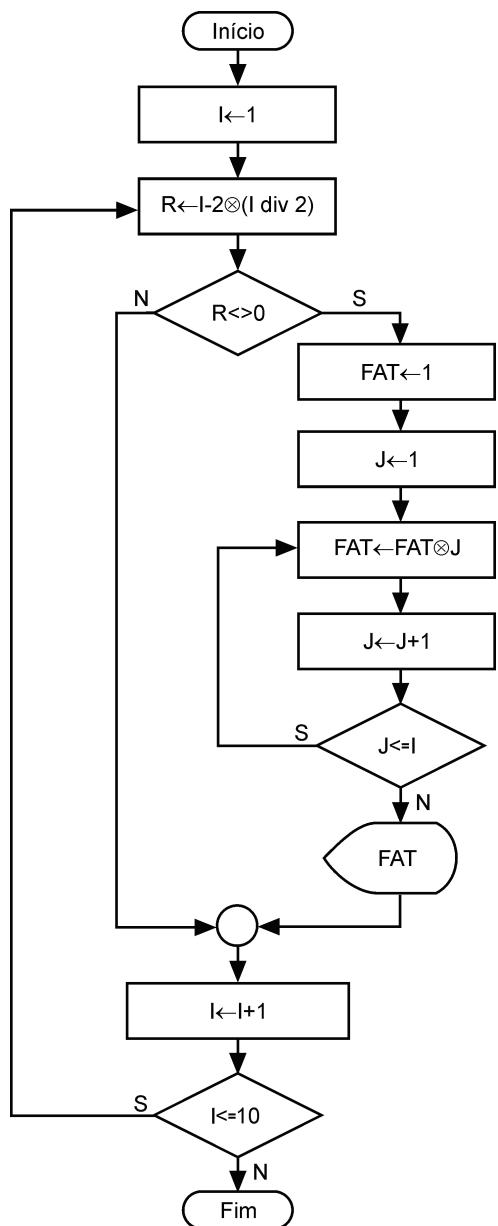


Português estruturado

```

programa Cap05_Ex4n_Pg132
var
  TOTAL : inteiro
  SOMA, MÉDIA, N : real
início
  TOTAL ← 0
  SOMA ← 0
  continua
  leia N
  se (N ≤ 0) então
    SOMA ← SOMA + N
    TOTAL ← TOTAL + 1
  fim_se
enquanto_for (N ≥ 0)
  se (TOTAL > 0) então
    MÉDIA ← SOMA / TOTAL
  fim_se
  escreva SOMA, TOTAL, MÉDIA
fim
  
```

o) Digrama de blocos

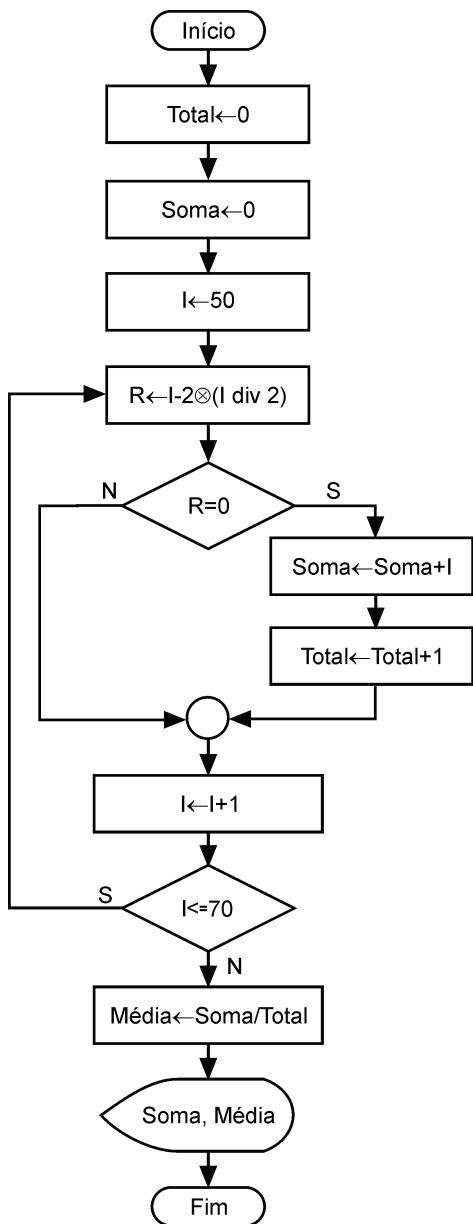


Português estruturado

```

programa Cap05_Ex4o_Pg132
var
    FAT, R, I, J : inteiro
início
    I ← 1
    continua
    R ← I - 2 * (I div 2)
    se (R <> 0) então
        FAT ← 1
        J ← 1
        continua
        FAT ← FAT * J
        J ← J + 1
        enquanto_for (J <= I)
            escreva FAT
        fim_se
        I ← I + 1
    enquanto_for (I <= 10)
fim
    
```

p) Digrama de blocos

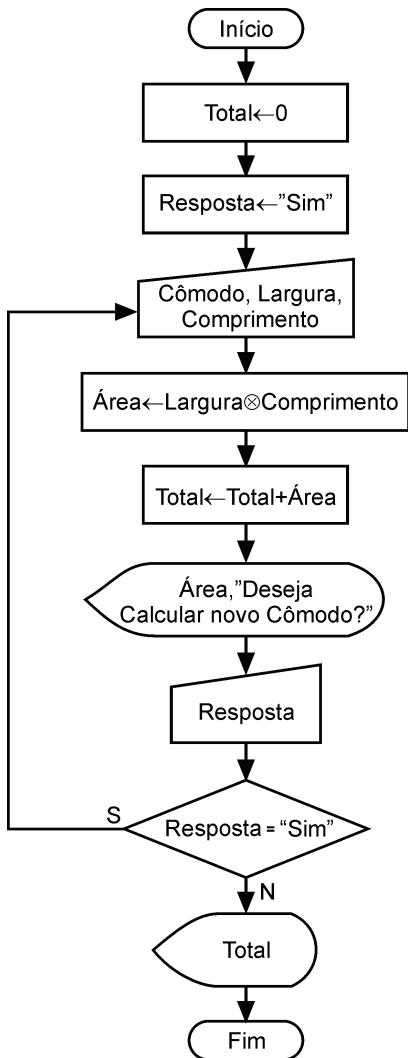


Português estruturado

```

programa Cap05_Ex4p_Pg132
var
  TOTAL, SOMA, R, I : inteiro
  MÉDIA : real
início
  TOTAL ← 0
  SOMA ← 0
  I ← 50
  continua
  R ← I - 2 * (I div 2)
  se (R = 0) então
    SOMA ← SOMA + I
    TOTAL ← TOTAL + 1
  fim_se
  I ← I + 1
enquanto_for (I <= 70)
  MÉDIA ← SOMA / TOTAL
  escreva SOMA, MÉDIA
fim
  
```

q) Digrama de blocos

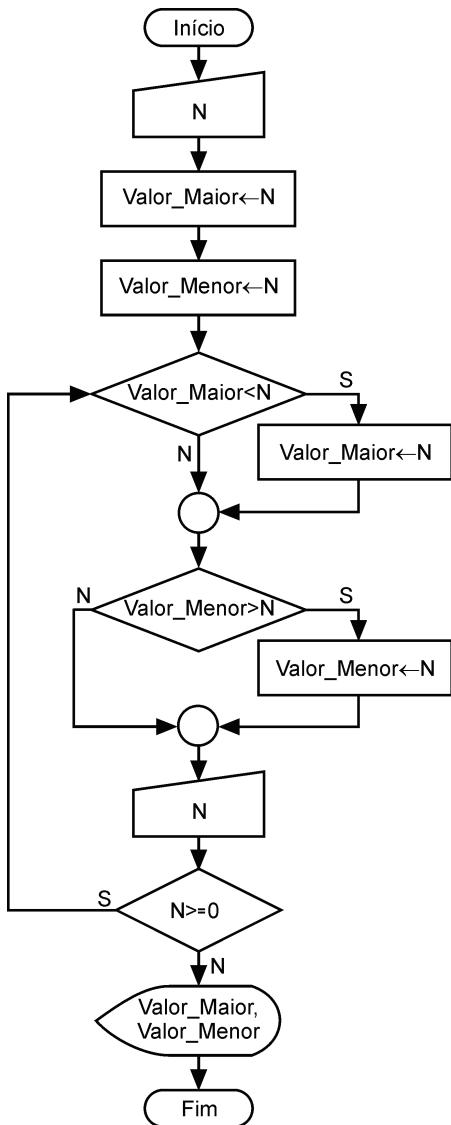


Português estruturado

```

programa Cap05_Ex4q_Pg132
var
  CÔMODO, RESPOSTA : cadeia
  TOTAL, ÁREA, LARGURA, COMPRIMENTO : real
início
  TOTAL ← 0
  RESPOSTA ← "SIM"
  continua
    leia CÔMODO, LARGURA, COMPRIMENTO
    ÁREA ← LARGURA * COMPRIMENTO
    TOTAL ← TOTAL + ÁREA
    escreva ÁREA, "Deseja calcular novo cômodo? "
    leia RESPOSTA
  enquanto_for (RESPOSTA = "SIM")
    escreva TOTAL
  fim
  
```

r) Diagrama de blocos

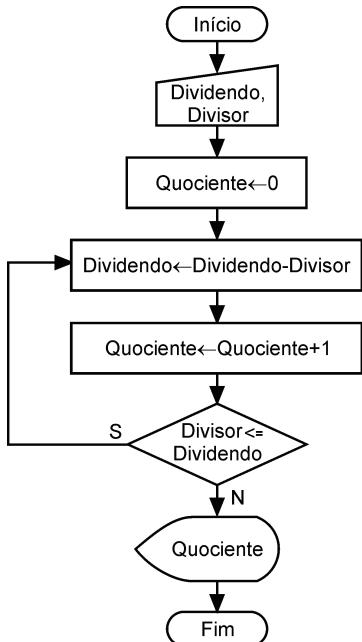


Português estruturado

```

programa Cap05_Ex4r_Pg132
var
  N, VALOR_MAIOR, VALOR_MENOR : inteiro
início
  leia N
  VALOR_MAIOR ← N
  VALOR_MENOR ← N
  continua
    se (VALOR_MAIOR < N) então
      VALOR_MAIOR ← N
    fim_se
    se (VALOR_MENOR > N) então
      VALOR_MENOR ← N
    fim_se
    leia N
  enquanto_for (N ≥ 0)
    escreva VALOR_MAIOR, VALOR_MENOR
fim
  
```

s) Diagrama de blocos



Português estruturado

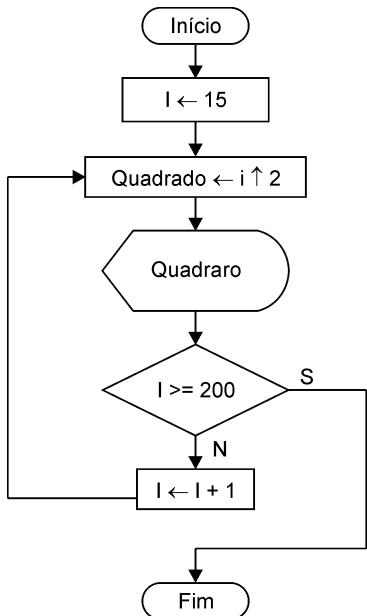
```

programa Cap05_Ex4s_Pg132
var
  QUOCIENTE, DIVIDENDO, DIVISOR : inteiro
início
  leia DIVIDENDO, DIVISOR
  QUOCIENTE ← 0
  continua
    DIVIDENDO ← DIVIDENDO - DIVISOR
    QUOCIENTE ← QUOCIENTE + 1
  enquanto_for (DIVISOR ≤ DIVIDENDO)
    escreva QUOCIENTE
fim
  
```

5.5 - Laço de Repetição Condicional Seletivo

Tópico 5.9 - Exercício 5 - Página 131

a) Diagrama de blocos

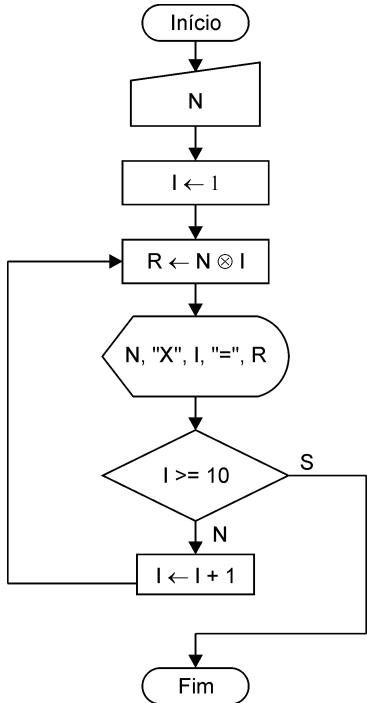


Português estruturado

```

programa Cap05_Ex5a_Pg131
var
  QUADRADO, I : inteiro
início
  I ← 15
  laço
    QUADRADO ← I ↑ 2
    escreva QUADRADO
    saia_caso (I >= 200)
    I ← I + 1
  fim_laço
fim
  
```

b) Diagrama de blocos

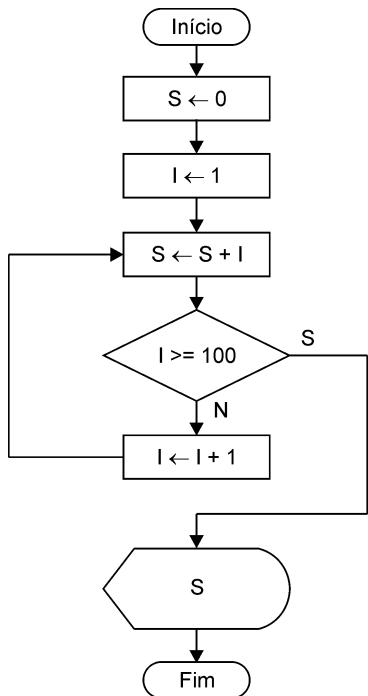


Português estruturado

```

programa Cap05_Ex5b_Pg131
var
  N, I, R : inteiro
início
  leia N
  I ← 1
  laço
    R ← N * I
    escreva N, " X ", I, " = ", R
    saia_caso (I >= 10)
    I ← I + 1
  fim_laço
fim
  
```

c) Diagrama de blocos

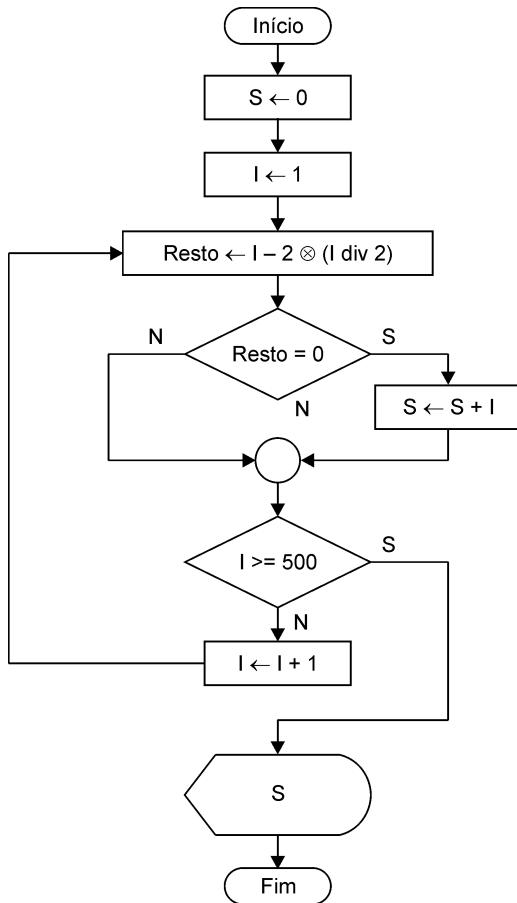


Português estruturado

```

programa Cap05_Ex5c_Pg131
var
  S, I : inteiro
início
  S ← 0
  I ← 1
laço
  S ← S + I
  saia_caso (I >= 100)
  I ← I + 1
fim_laço
escreva S
fim
  
```

d) Diagrama de blocos

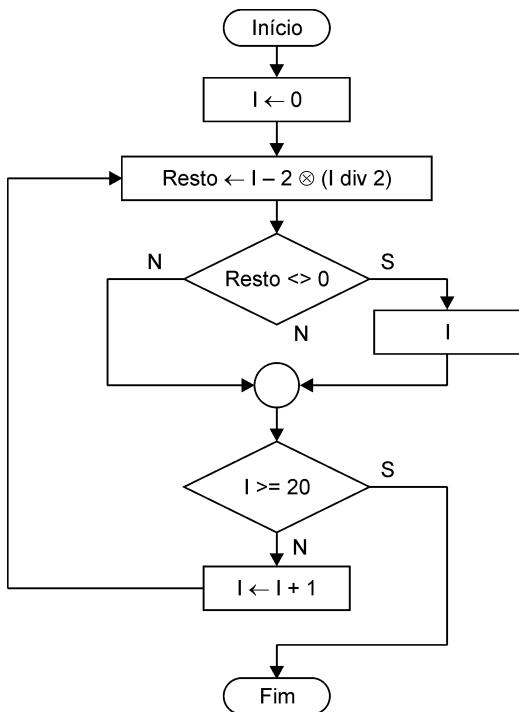


Português estruturado

```

programa Cap05_Ex5d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  I ← 1
laço
  RESTO ← I - 2 * (I div 2)
  se (RESTO = 0) então
    S ← S + I
  fim_se
  saia_caso (I >= 500)
  I ← I + 1
fim_laço
escreva S
fim
  
```

e) Diagrama de blocos

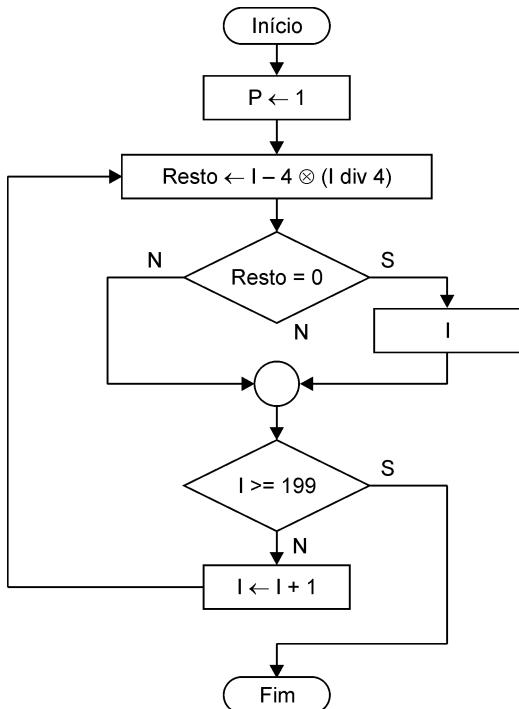


Português estruturado

```

programa Cap05_Ex5e_Pg131
var
  I, RESTO : inteiro
início
  I ← 0
  laço
    RESTO ← I - 2 * (I div 2)
    se (RESTO <> 0) então
      escreva I
    fim_se
    saia_caso (I ≥ 20)
    I ← I + 1
  fim_laço
fim
  
```

f) Diagrama de blocos

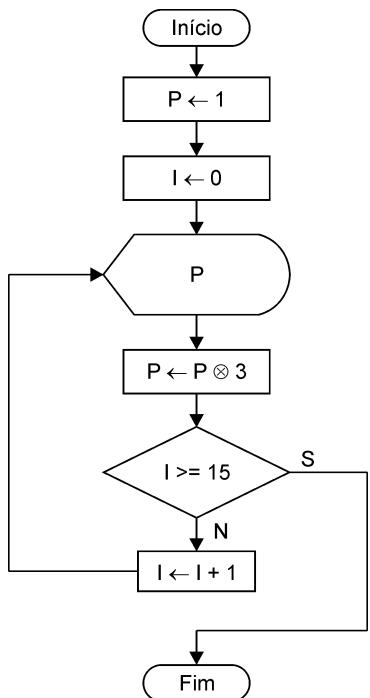


Português estruturado

```

programa Cap05_Ex5f_Pg131
var
  I, RESTO : inteiro
início
  I ← 1
  laço
    RESTO ← I - 4 * (I div 4)
    se (RESTO = 0) então
      escreva I
    fim_se
    saia_caso (I ≥ 199)
    I ← I + 1
  fim_laço
fim
  
```

g) Diagrama de blocos

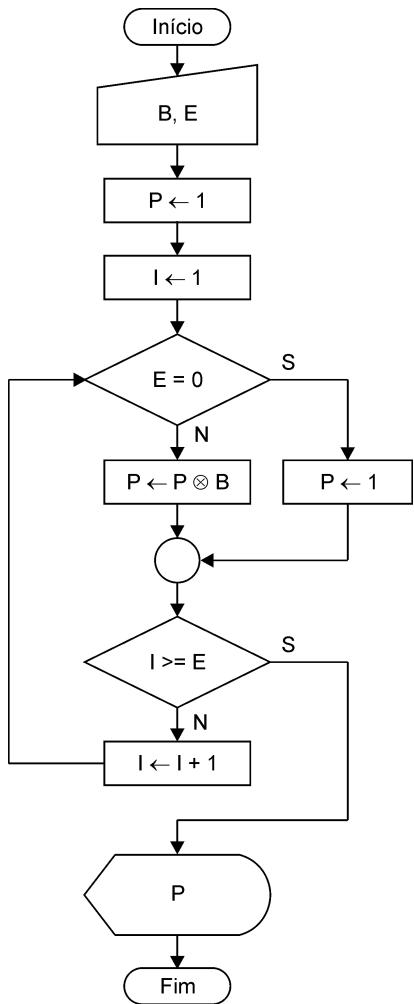


Português estruturado

```

programa Cap05_Ex5g_Pg131
var
  P, I : inteiro
início
  P ← 1
  I ← 0
  laço
    escreva P
    P ← P * 3
    saia_caso (I ≥ 15)
    I ← I + 1
  fim_laço
fim
  
```

h) Diagrama de blocos

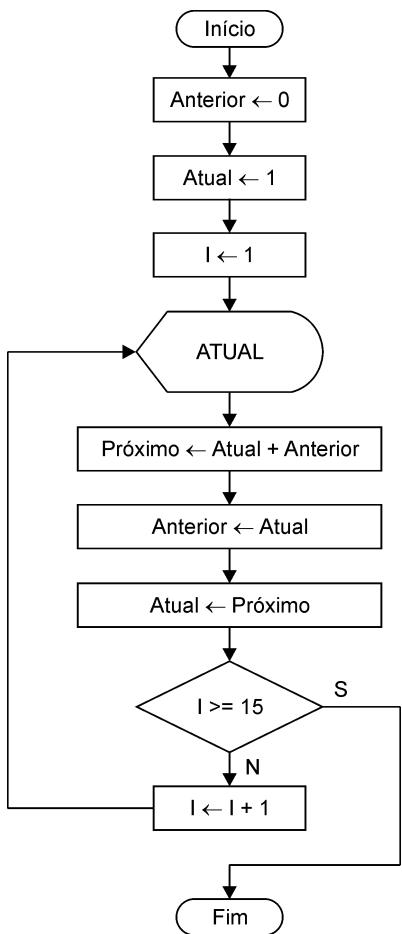


Português estruturado

```

programa Cap05_Ex5h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  I ← 1
  laço
    se (E = 0) então
      P ← 1
    senão
      P ← P * B
    fim-se
    saia_caso (I ≥ E)
    I ← I + 1
  fim_laço
  escreva P
fim
  
```

i) Diagrama de blocos

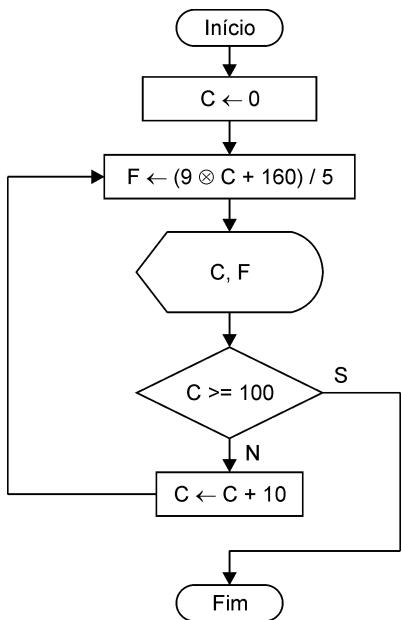


Português estruturado

```

programa Cap05_Ex5i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  I ← 1
  laço
    escreva ATUAL
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
    saia_caso (I >= 15)
    I ← I + 1
  fim_laço
fim
  
```

j) Diagrama de blocos

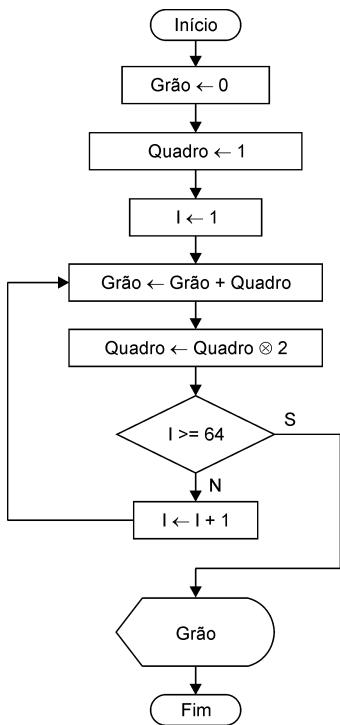


Português estruturado

```

programa Cap05_Ex5j_Pg132
var
  C, F : real
início
  C ← 10
  laço
    F ← (9 * C + 160) / 5
    escreva C, F
    saia_caso (C >= 100)
    C ← C + 10
  fim_laço
fim
  
```

k) Diagrama de blocos

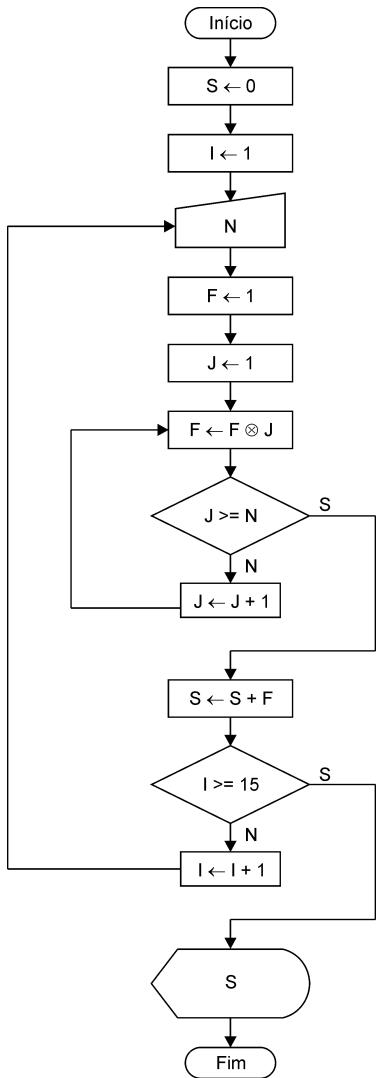


Português estruturado

```

programa Cap05_Ex5k_Pg132
var
    I, GRÃO, QUADRO : inteiro
início
    GRÃO ← 0
    QUADRO ← 1
    I ← 1
    laço
        GRÃO ← GRÃO + QUADRO
        QUADRO ← QUADRO * 2
        saia_caso (I >= 64)
        I ← I + 1
    fim_laço
    escreva GRÃO
fim
    
```

l) Diagrama de blocos

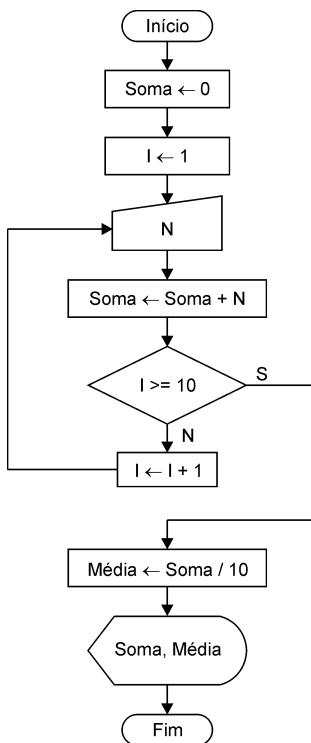


Português estruturado

```

programa Cap05_Ex5l_Pg132
var
    I, J, N, S, F : inteiro
início
    S ← 0
    I ← 1
    laço
        leia N
        F ← 1
        J ← 1
        laço
            F ← F * J
            saia_caso (J >= N)
            J ← J + 1
        fim_laço
        S ← S + F
        saia_caso (I >= 15)
        I ← I + 1
    fim_laço
    escreva S
fim
    
```

m) Diagrama de blocos

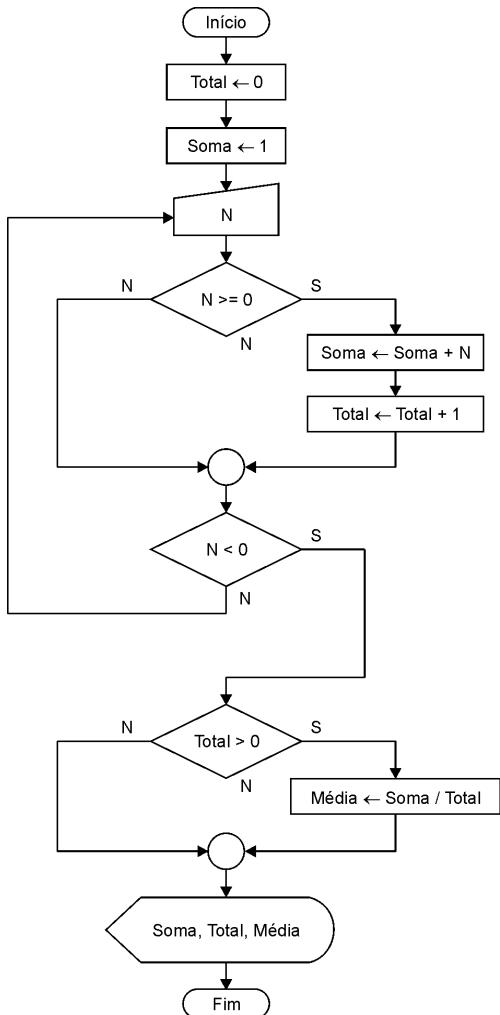


Português estruturado

```

programa Cap05_Ex5m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  I ← 1
  laço
    leia N
    SOMA ← SOMA + N
    saia_caso (I ≥ 10)
    I ← I + 1
  fim_laço
  MÉDIA ← SOMA / 10
  escreva SOMA, MÉDIA
fim
  
```

n) Diagrama de blocos

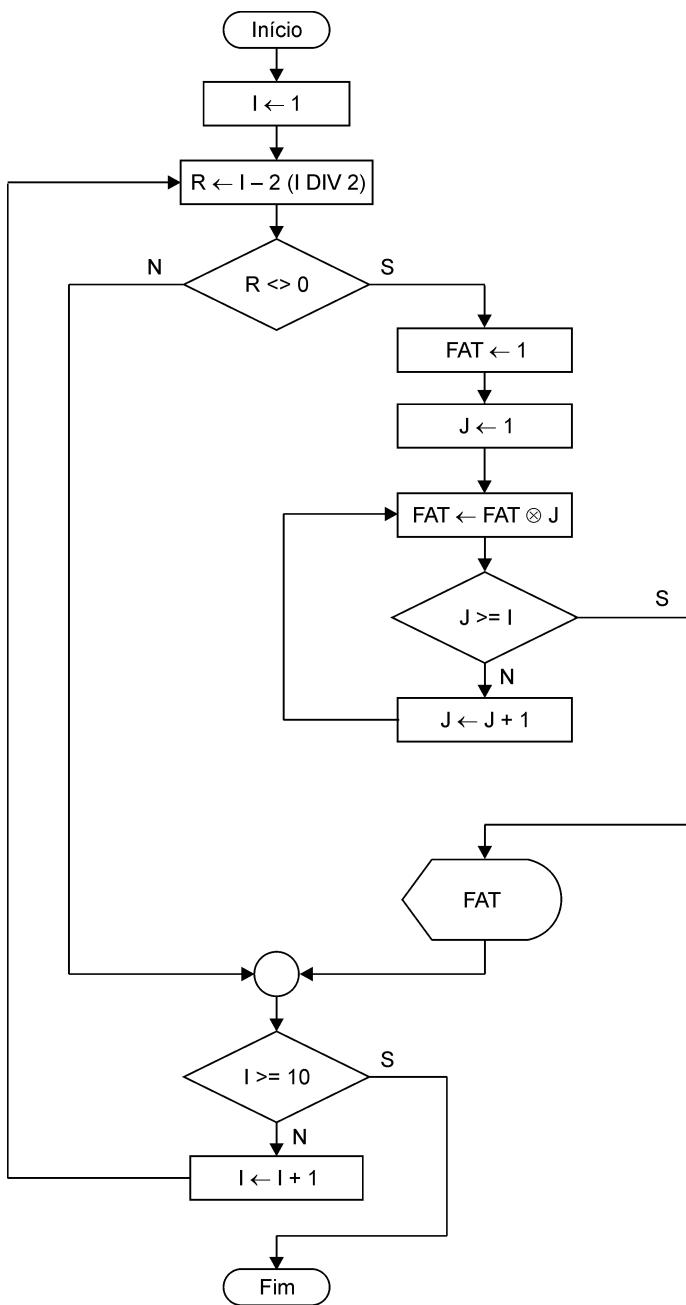


Português estruturado

```

programa Cap05_Ex5n_Pg132
var
  TOTAL : inteiro
  SOMA, MÉDIA, N : real
início
  TOTAL ← 0
  SOMA ← 0
  laço
    leia N
    se (N ≥ 0) então
      SOMA ← SOMA + N
      TOTAL ← TOTAL + 1
    fim_se
    saia_caso (N < 0)
  fim_laço
  se (TOTAL > 0) então
    MÉDIA ← SOMA / TOTAL
  fim_se
  escreva SOMA, TOTAL, MÉDIA
fim
  
```

o) Digrama de blocos

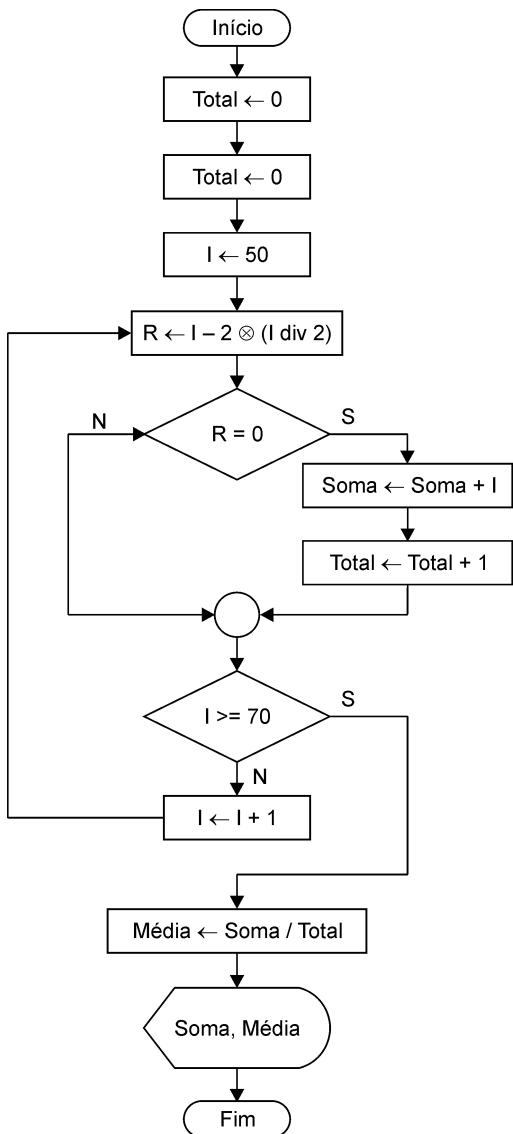


Português estruturado

```

programa Cap05_Ex5o_Pg132
var
  FAT, R, I, J : inteiro
início
  I ← 1
  laço
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      FAT ← 1
      J ← 1
      laço
        FAT ← FAT * J
        saia_caso (J >= I)
        J ← J + 1
      fim_laço
      escreva FAT
    fim_se
    saia_caso (I >= 10)
    I ← I + 1
  fim_laço
fim
  
```

p) Diagrama de blocos

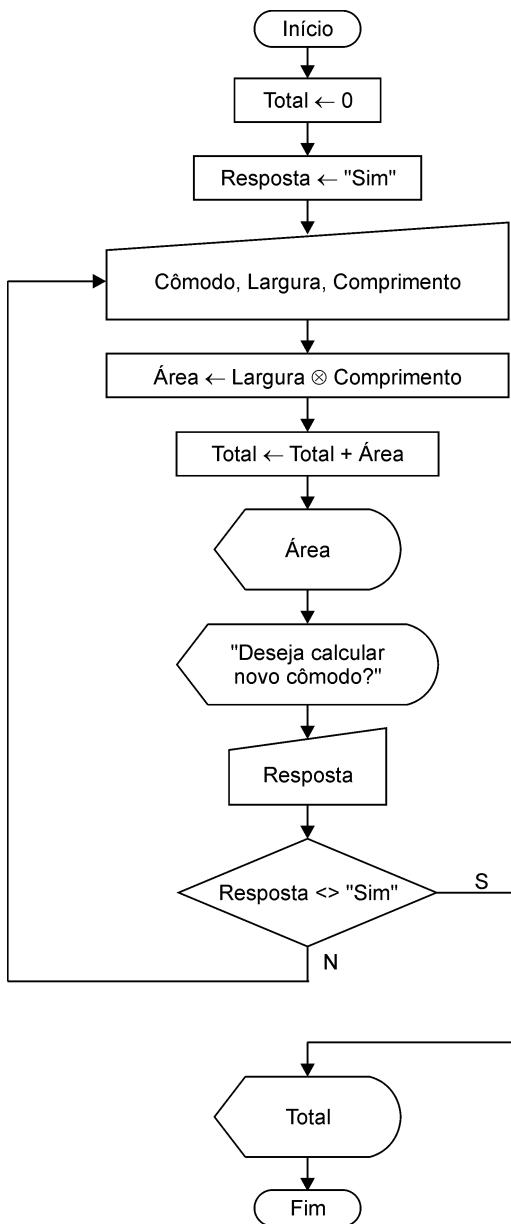


Português estruturado

```

programa Cap05_Ex5p_Pg132
var
  TOTAL, SOMA, R, I : inteiro
  MÉDIA : real
início
  TOTAL ← 0
  SOMA ← 0
  I ← 50
laço
  R ← I - 2 * (I div 2)
  se (R = 0) então
    SOMA ← SOMA + I
    TOTAL ← TOTAL + 1
  fim_se
  saia_caso (I ≥ 70)
  I ← I + 1
fim_laço
MÉDIA ← SOMA / TOTAL
escreva SOMA, MÉDIA
fim
  
```

q) Digrama de blocos

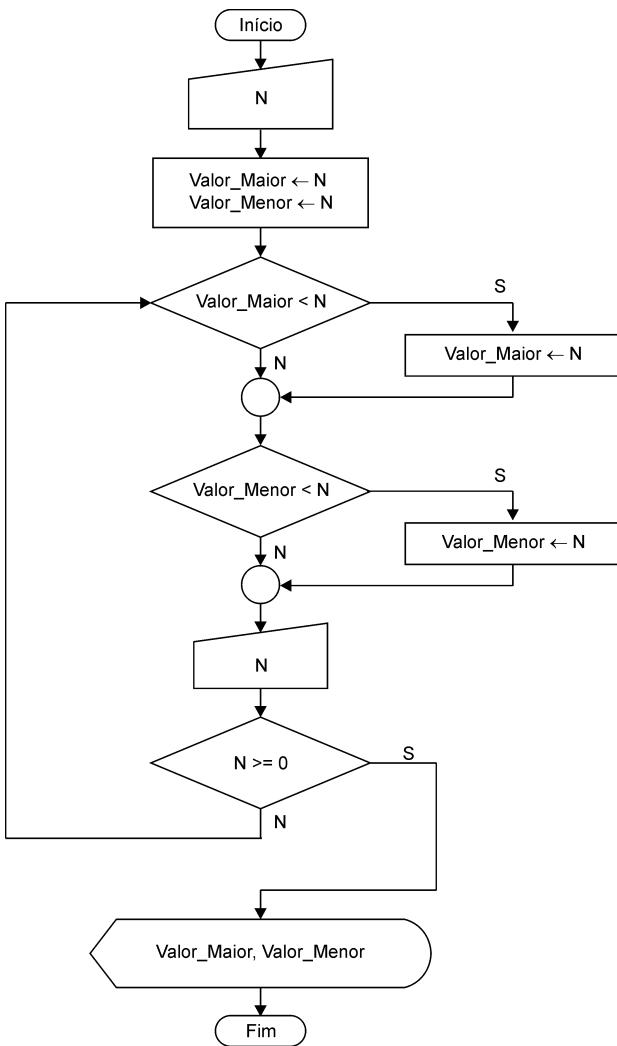


Português estruturado

```

programa Cap05_Ex5q_Pg132
var
    CÔMODO, RESPOSTA : cadeia
    TOTAL, ÁREA, LARGURA, COMPRIMENTO : real
início
    TOTAL ← 0
    RESPOSTA ← "SIM"
    laço
        leia CÔMODO, LARGURA, COMPRIMENTO
        ÁREA ← LARGURA * COMPRIMENTO
        TOTAL ← TOTAL + ÁREA
        escreva ÁREA, "Deseja calcular novo cômodo? "
        leia RESPOSTA
        saia_caso (RESPOSTA <> "SIM")
    fim_laço
    escreva TOTAL
fim
    
```

r) Diagrama de blocos

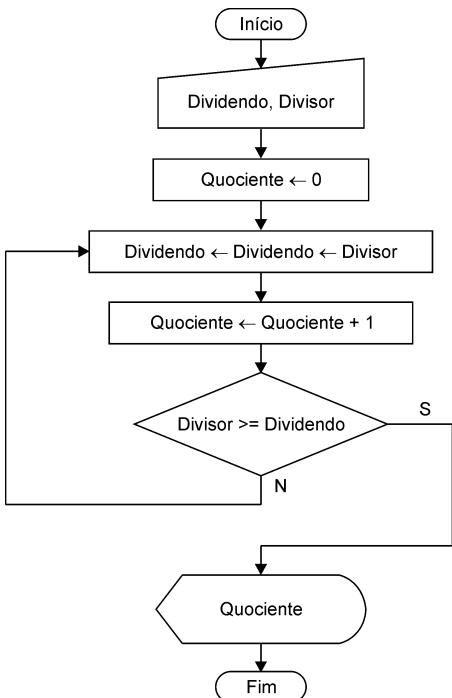


Português estruturado

```

programa Cap05_Ex5r_Pg132
var
    N, VALOR_MAIOR, VALOR_MENOR : inteiro
início
    leia N
    VALOR_MAIOR ← N
    VALOR_MENOR ← N
    laço
        se (VALOR_MAIOR < N) então
            VALOR_MAIOR ← N
        fim_se
        se (VALOR_MENOR > N) então
            VALOR_MENOR ← N
        fim_se
        leia N
    fim_laço (N >= 0)
    escreva VALOR_MAIOR, VALOR_MENOR
fim
    
```

s) Diagrama de blocos



Português estruturado

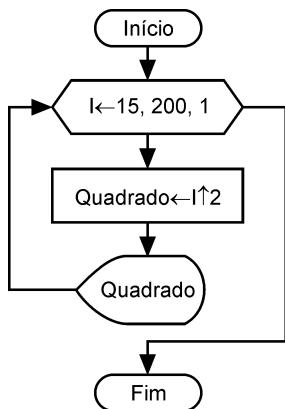
```

programa Cap05_Ex5s_Pg132
var
    QUOCIENTE, DIVIDENDO, DIVISOR : inteiro
início
    leia DIVIDENDO, DIVISOR
    QUOCIENTE ← 0
    laço
        DIVIDENDO ← DIVIDENDO - DIVISOR
        QUOCIENTE ← QUOCIENTE + 1
        saia_caso (DIVISOR >= DIVIDENDO)
    fim_laço
    escreva QUOCIENTE
fim
    
```

5.6 - Laço de Repetição Incondicional

Tópico 5.9 - Exercício 6 - Página 131

a) Diagrama de blocos

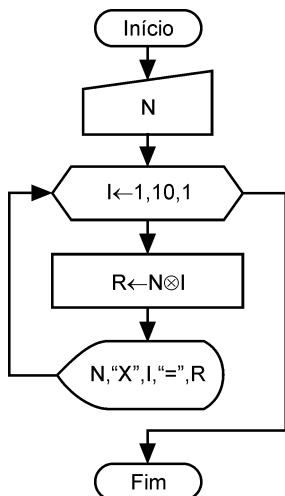


Português estruturado

```

programa Cap05_Ex6a_Pg131
var
  QUADRADO, I : inteiro
início
  para I de 15 até 200 passo 1 faça
    QUADRADO ← I ↑ 2
    escreva QUADRADO
  fim_para
fim
  
```

b) Diagrama de blocos

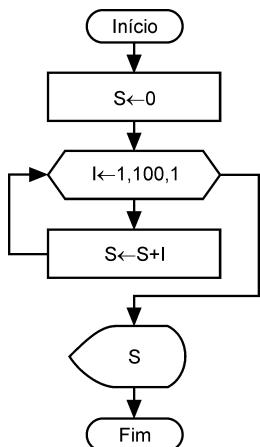


Português estruturado

```

programa Cap05_Ex6b_Pg131
var
  N, I, R : inteiro
início
  leia N
  para I de 1 até 10 passo 1 faça
    R ← N * I
    escreva N, " X ", I, " = ", R
  fim_para
fim
  
```

c) Diagrama de blocos

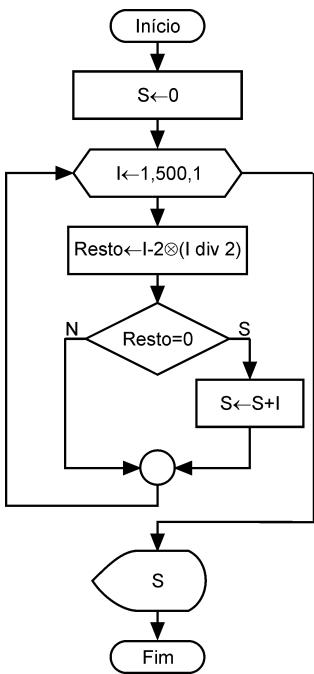


Português estruturado

```

programa Cap05_Ex6c_Pg131
var
  S, I : inteiro
início
  S ← 0
  para I de 1 até 100 passo 1 faça
    S ← S + I
  fim_para
  escreva S
fim
  
```

d) Diagrama de blocos

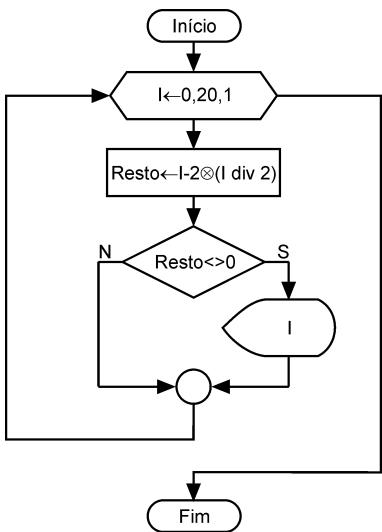


Português estruturado

```

programa Cap05_Ex6d_Pg131
var
  S, I, RESTO : inteiro
início
  S ← 0
  para I de 1 até 500 passo 1 faça
    RESTO ← I - 2 * (I div 2)
    se (RESTO = 0) então
      S ← S + I
    fim_se
  fim_para
  escreva S
fim
  
```

e) Diagrama de blocos

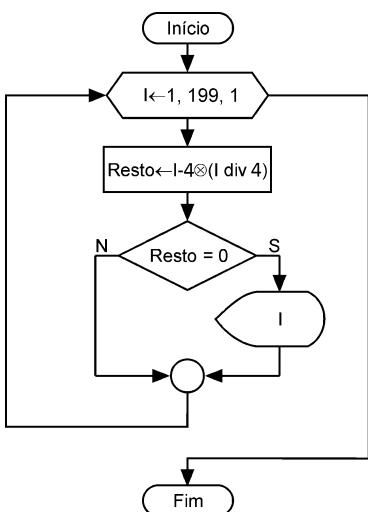


Português estruturado

```

programa Cap05_Ex6e_Pg131
var
  I, RESTO : inteiro
início
  para I de 0 até 20 passo 1 faça
    RESTO ← I - 2 * (I div 2)
    se (RESTO <> 0) então
      escreva I
    fim_se
  fim_para
fim
  
```

f) Diagrama de blocos

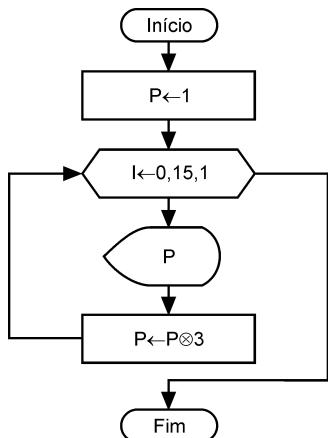


Português estruturado

```

programa Cap05_Ex6f_Pg131
var
  I, RESTO : inteiro
início
  para I de 1 até 199 passo 1 faça
    RESTO ← I - 4 * (I div 4)
    se (RESTO = 0) então
      escreva I
    fim_se
  fim_para
fim
  
```

g) Diagrama de blocos

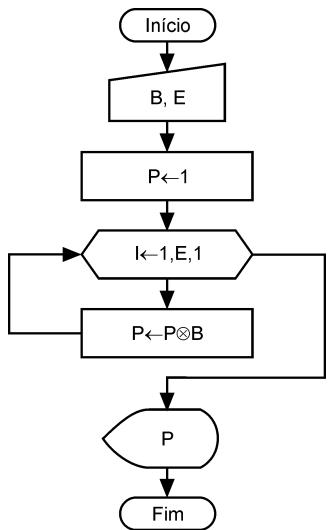


Português estruturado

```

programa Cap05_Ex6g_Pg131
var
  P, I : inteiro
início
  P ← 1
  para I de 0 até 15 passo 1 faça
    escreva P
    P ← P * 3
  fim_para
fim
  
```

h) Diagrama de blocos

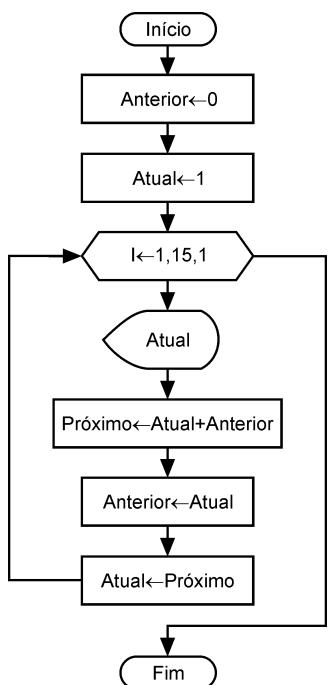


Português estruturado

```

programa Cap05_Ex6h_Pg131
var
  P, I, B, E : inteiro
início
  leia B, E
  P ← 1
  para I de 1 até E passo 1 faça
    P ← P * B
  fim_para
  escreva P
fim
  
```

i) Diagrama de blocos

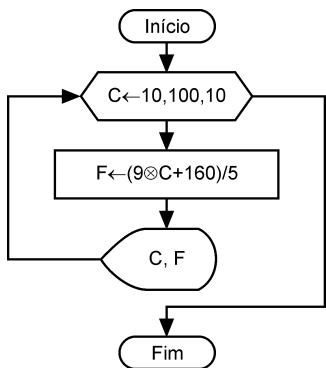


Português estruturado

```

programa Cap05_Ex6i_Pg132
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  para I de 1 até 15 passo 1 faça
    escreva ATUAL
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
  fim_para
fim
  
```

j) Diagrama de blocos

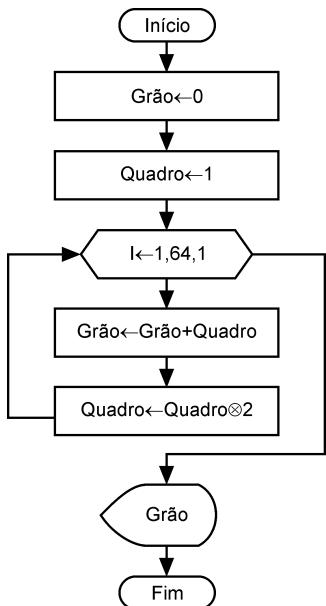


Português estruturado

```

programa Cap05_Ex6j_Pg132
var
  C, F : real
início
  para C de 10 até 100 passo 10 faça
    F ← (9 * C + 160) / 5
    escreva C, F
  fim_para
fim
  
```

k) Diagrama de blocos

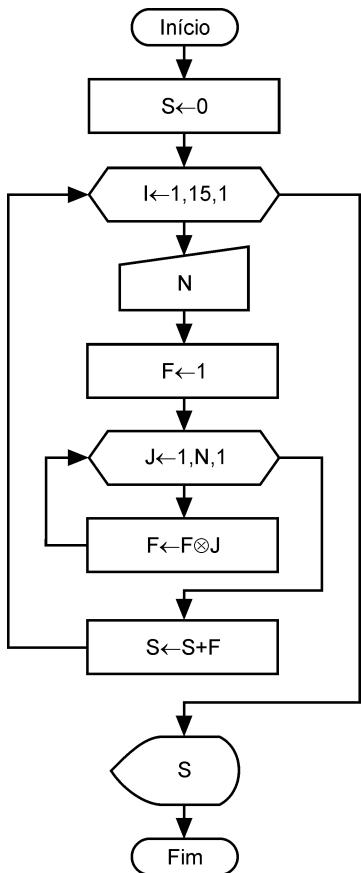


Português estruturado

```

programa Cap05_Ex6k_Pg132
var
  I, GRÃO, QUADRO : inteiro
início
  GRÃO ← 0
  QUADRO ← 1
  para I de 1 até 64 passo 1 faça
    GRÃO ← GRÃO + QUADRO
    QUADRO ← QUADRO * 2
  fim_para
  escreva GRÃO
fim
  
```

l) Diagrama de blocos

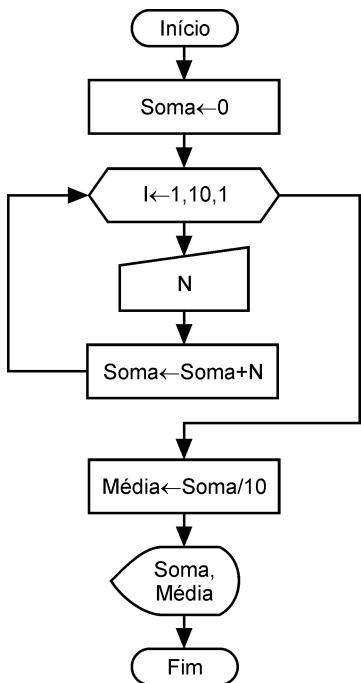


Português estruturado

```

programa Cap05_Ex6l_Pg132
var
  I, J, N, S, F : inteiro
início
  S ← 0
  para I de 1 até 15 passo 1 faça
    leia N
    F ← 1
    para J de 1 até N passo 1 faça
      F ← F * J
    fim_para
    S ← S + F
  fim_para
  escreva S
fim
  
```

m) Diagrama de blocos



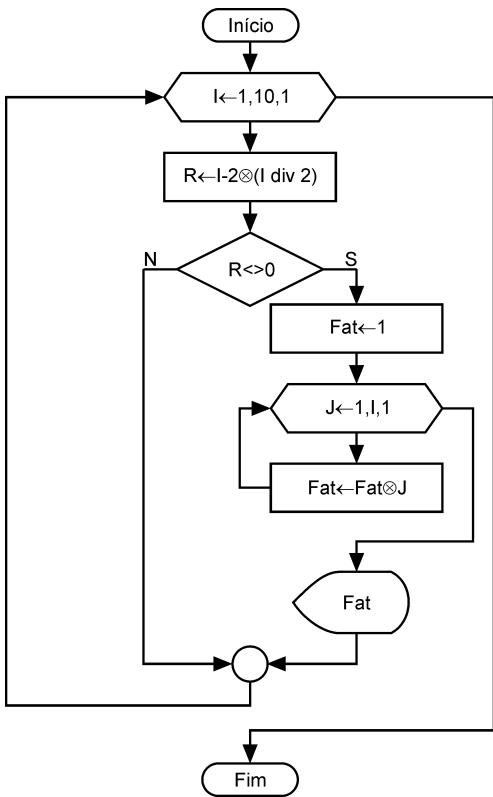
Português estruturado

```

programa Cap05_Ex6m_Pg132
var
  I, N, SOMA : inteiro
  MÉDIA : real
início
  SOMA ← 0
  para I de 1 até 10 passo 1 faça
    leia N
    SOMA ← SOMA + N
  fim_para
  MÉDIA ← SOMA / 10
  escreva SOMA, MÉDIA
fim
  
```

n) Não é possível resolver este exercício com este tipo de laço de repetição.

o) Diagrama de blocos

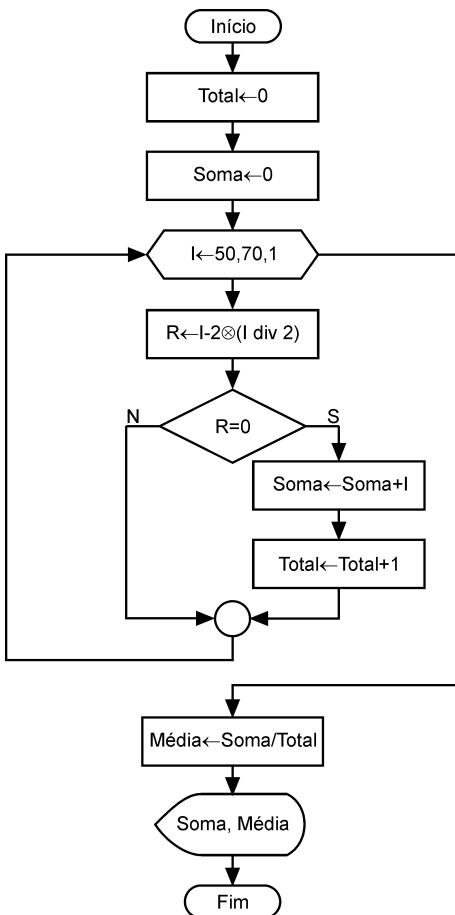


Português estruturado

```

programa Cap05_Ex6o_Pg132
var
  FAT, R, I, J : inteiro
início
  para I de 1 até 10 passo 1 faça
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      FAT ← 1
      para J de 1 até I passo 1 faça
        FAT ← FAT * J
      fim_para
      escreva FAT
    fim_se
  fim_para
fim
  
```

p) Diagrama de blocos



Português estruturado

```

programa Cap05_Ex6p_Pg132
var
  TOTAL, SOMA, R, I : inteiro
  MÉDIA : real
início
  TOTAL ← 0
  SOMA ← 0
  para I de 50 até 70 passo 1 faça
    R ← I - 2 * (I div 2)
    se (R = 0) então
      SOMA ← SOMA + I
      TOTAL ← TOTAL + 1
    fim_se
  fim_para
  MÉDIA ← SOMA / TOTAL
  escreva SOMA, MÉDIA
fim
  
```

q) Não é possível resolver este exercício com esse tipo de laço de repetição.

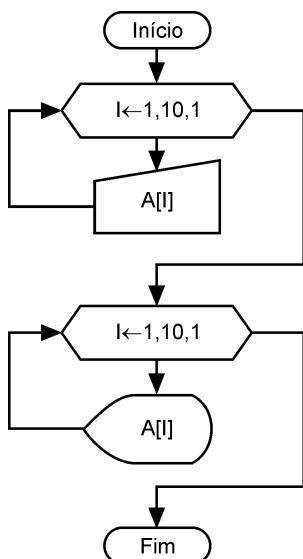
r) Não é possível resolver este exercício com esse tipo de laço de repetição.

s) Não é possível resolver este exercício com esse tipo de laço de repetição.

6 - Exercícios de fixação do capítulo 6

Tópico 6.4 - Exercício 1 - Página 142

a) Diagrama de blocos

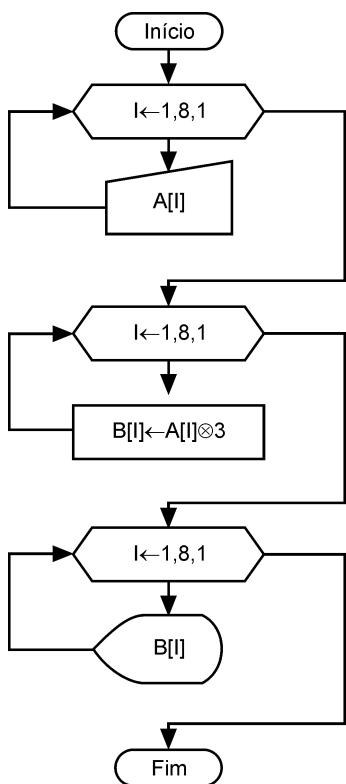


Português estruturado

```

programa Cap06_Ex1a_Pg142
var
    A : conjunto[1..10] de cadeia
    I : inteiro
início
    para I de 1 até 10 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 10 passo 1 faça
        escreva A[I]
    fim_para
fim
    
```

b) Diagrama de blocos

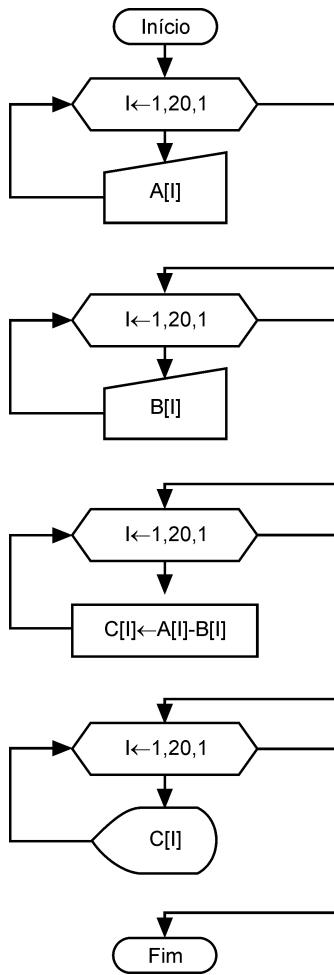


Português estruturado

```

programa Cap06_Ex1b_Pg142
var
    A, B : conjunto[1..8] de inteiro
    I : inteiro
início
    para I de 1 até 8 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 8 passo 1 faça
        B[I] ← A[I] * 3
    fim_para
    para I de 1 até 8 passo 1 faça
        escreva B[I]
    fim_para
fim
    
```

c) Diagrama de blocos

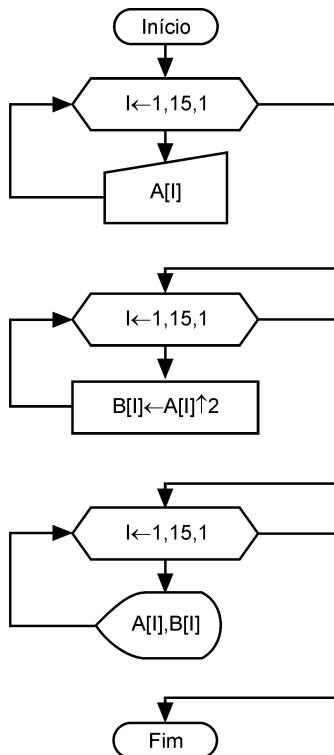


Português estruturado

```

programa Cap06_Ex1c_Pg142
var
  A, B, C : conjunto[1..20] de real
  I : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 20 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 20 passo 1 faça
    C[I] ← A[I] - B[I]
  fim_para
  para I de 1 até 20 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

d) Diagrama de blocos

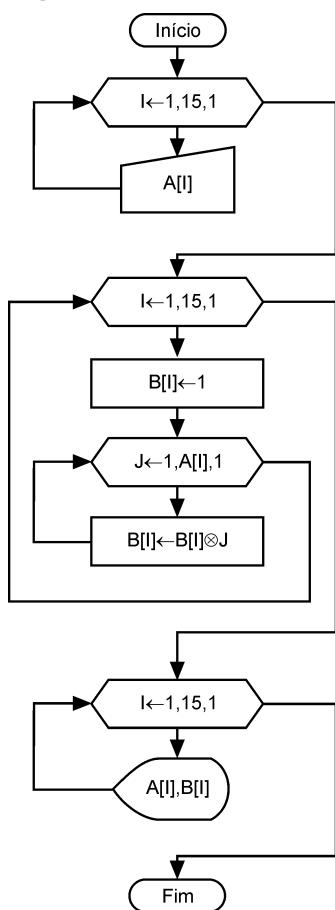


Português estruturado

```

programa Cap06_Ex1d_Pg142
var
  A, B : conjunto[1..15] de inteiro
  I : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    B[I] ← A[I] ↑ 2
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

e) Diagrama de blocos

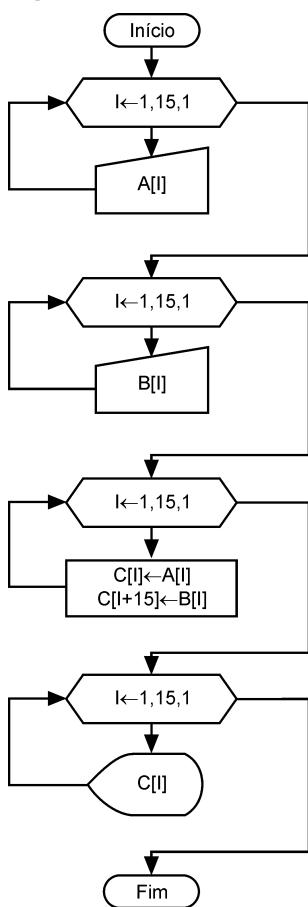


Português estruturado

```

programa Cap06_Exle_Pg142
var
  A, B : conjunto[1..15] de inteiro
  I, J : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    B[I] ← 1
    para J de 1 até A[I] passo 1 faça
      B[I] ← B[I] * J
    fim_para
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

f) Diagrama de blocos

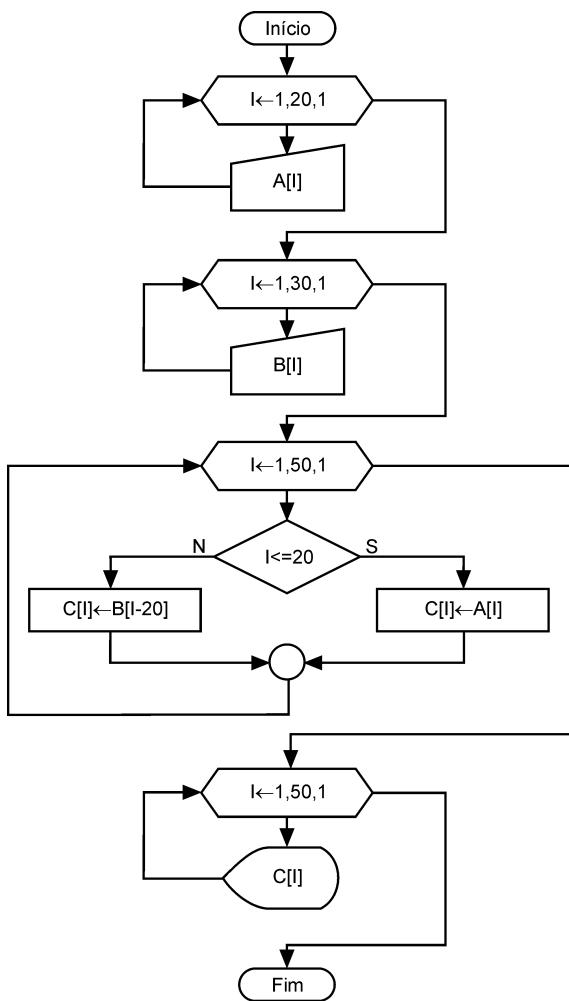


Português estruturado

```

programa Cap06_Exlf_Pg142
var
  A, B : conjunto[1..15] de inteiro
  C : conjunto[1..30] de inteiro
  I : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    C[I] ← A[I]
    C[I + 15] ← B[I]
  fim_para
  para I de 1 até 30 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

g) Diagrama de blocos

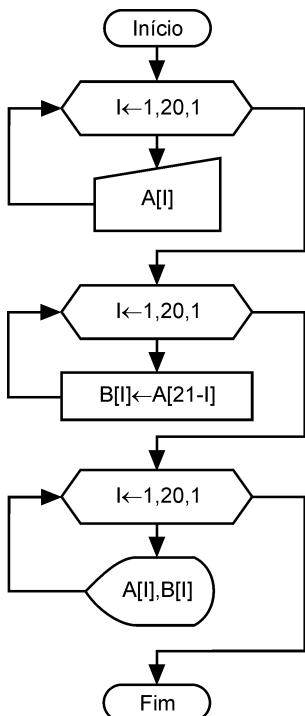


Português estruturado

```

programa Cap06_Ex1g_Pg142
var
  A : conjunto[1..20] de cadeia
  B : conjunto[1..30] de cadeia
  C : conjunto[1..50] de cadeia
  I : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 30 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 50 passo 1 faça
    se (I <= 20) então
      C[I] ← A[I]
    senão
      C[I] ← B[I - 20]
    fim_se
  fim_para
  para I de 1 até 50 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

h) Diagrama de blocos

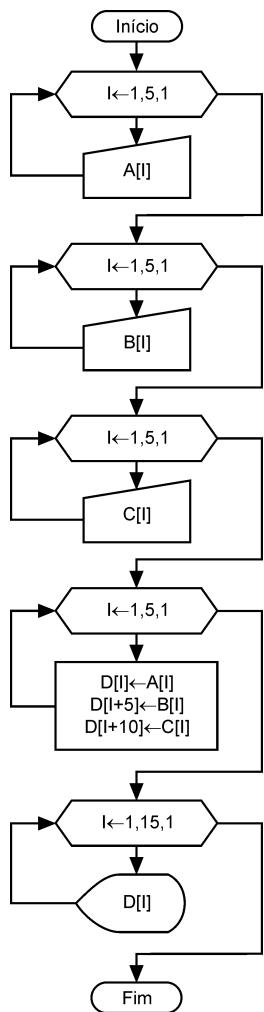


Português estruturado

```

programa Cap06_Ex1h_Pg142
var
  A, B : conjunto[1..20] de real
  I, J : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 20 passo 1 faça
    B[I] ← A[21 - I]
  fim_para
  para I de 1 até 20 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

i) Diagrama de blocos



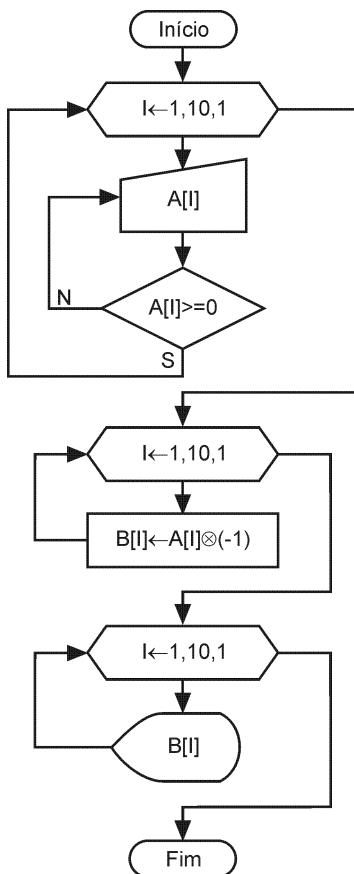
Português estruturado

```

programa Cap06_Ex1i_Pg142
var
  A, B, C : conjunto[1..5] de real
  D : conjunto[1..15] de real
  I : inteiro
início
  para I de 1 até 5 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 5 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 5 passo 1 faça
    leia C[I]
  fim_para
  para I de 1 até 5 passo 1 faça
    D[I] ← A[I]
    D[I + 5] ← B[I]
    D[I + 10] ← C[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva D[I]
  fim_para
fim
  
```

Tópico 6.4 - Exercício 1 - Página 143

k) Diagrama de blocos

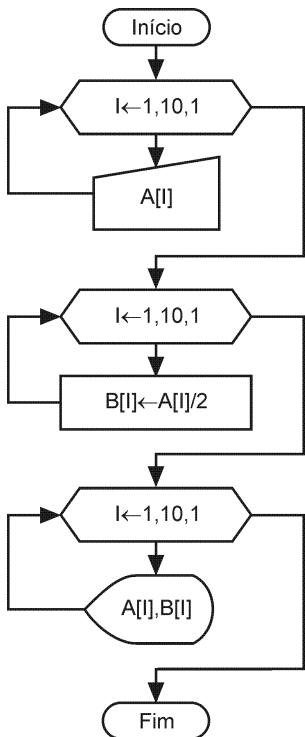


Português estruturado

```

programa Cap06_Ex1k_Pg143
var
  A, B : conjunto[1..10] de inteiro
  I : inteiro
início
  para I de 1 até 10 passo 1 faça
    repita
      leia A[I]
      até_que (A[I] >= 0)
    fim_para
    para I de 1 até 10 passo 1 faça
      B[I] ← A[I] * (-1)
    fim_para
    para I de 1 até 10 passo 1 faça
      escreva B[I]
    fim_para
fim
  
```

l) Diagrama de blocos

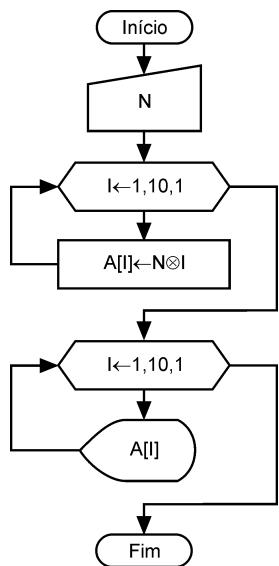


Português estruturado

```

programa Cap06_Ex1l_Pg143
var
  A : conjunto[1..10] de inteiro
  B : conjunto[1..10] de real
  I : inteiro
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 10 passo 1 faça
    B[I] ← A[I] / 2
  fim_para
  para I de 1 até 10 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

m) Diagrama de blocos

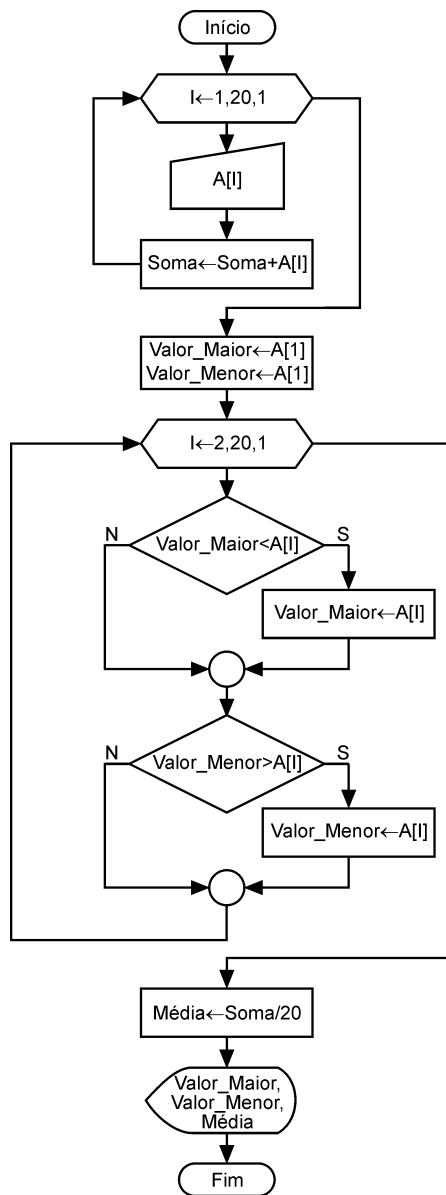


Português estruturado

```

programa Cap06_Ex1m_Pg143
var
  A : conjunto[1..10] de inteiro
  I, N : inteiro
início
  leia N
  para I de 1 até 10 passo 1 faça
    A[I] ← N * I
  fim_para
  para I de 1 até 10 passo 1 faça
    escreva A[I]
  fim_para
fim
  
```

n) Diagrama de blocos

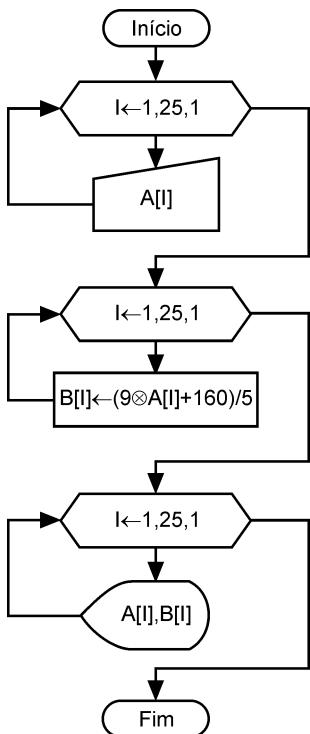


Português estruturado

```

programa Cap06_Ex1n_Pg143
var
  A : conjunto[1..20] de real
  SOMA, MÉDIA, VALOR_MAIOR, VALOR_MENOR : real
  I : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
    SOMA ← SOMA + A[I]
  fim_para
  VALOR_MAIOR ← A[1]
  VALOR_MENOR ← A[1]
  para I de 2 até 20 passo 1 faça
    se (VALOR_MAIOR < A[I]) então
      VALOR_MAIOR ← A[I]
    fim_se
    se (VALOR_MENOR > A[I]) então
      VALOR_MENOR ← A[I]
    fim_se
  fim_para
  MÉDIA ← SOMA / 20
  escreva VALOR_MAIOR, VALOR_MENOR, MÉDIA
fim
  
```

o) Diagrama de blocos

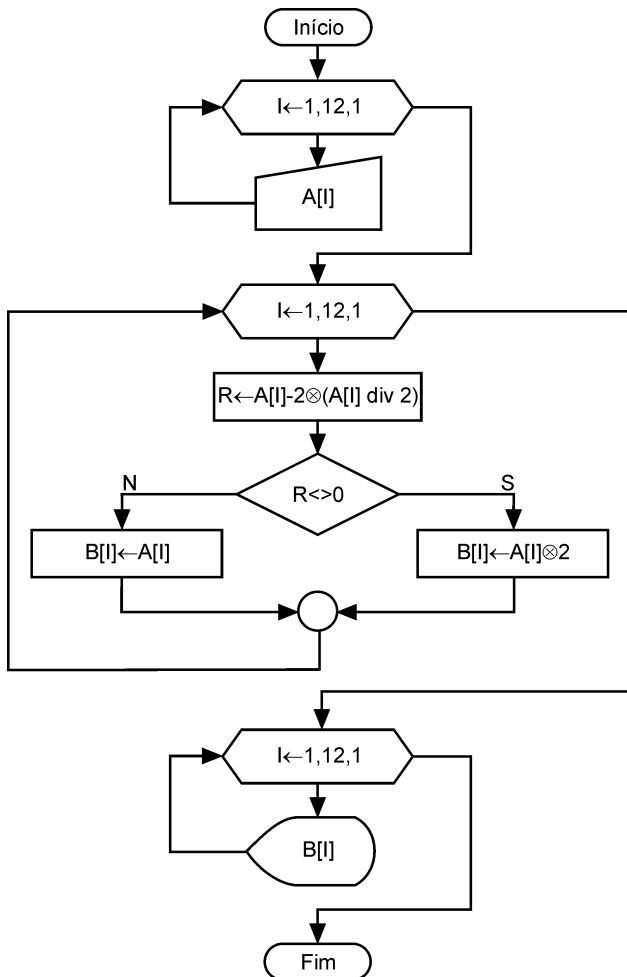


Português estruturado

```

programa Cap06_Ex1o_Pg143
var
    A, B : conjunto[1..25] de real
    I : inteiro
início
    para I de 1 até 25 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 25 passo 1 faça
        B[I] ← (9 * A[I] + 160) / 5
    fim_para
    para I de 1 até 25 passo 1 faça
        escreva A[I], B[I]
    fim_para
fim
  
```

p) Diagrama de blocos

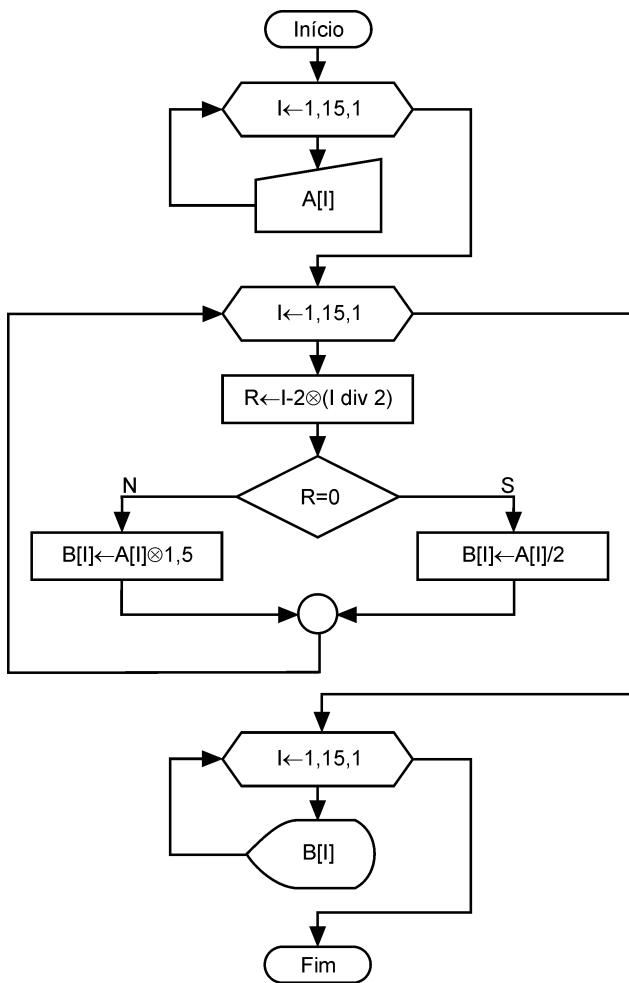


Português estruturado

```

programa Cap06_Ex1p_Pg143
var
    A, B : conjunto[1..12] de inteiro
    I, R : inteiro
início
    para I de 1 até 12 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 12 passo 1 faça
        R ← A[I] - 2 * (A[I] div 2)
        se (R <> 0) então
            B[I] ← A[I] * 2
        senão
            B[I] ← A[I]
        fim_se
    fim_para
    para I de 1 até 12 passo 1 faça
        escreva B[I]
    fim_para
fim
  
```

q) Diagrama de blocos

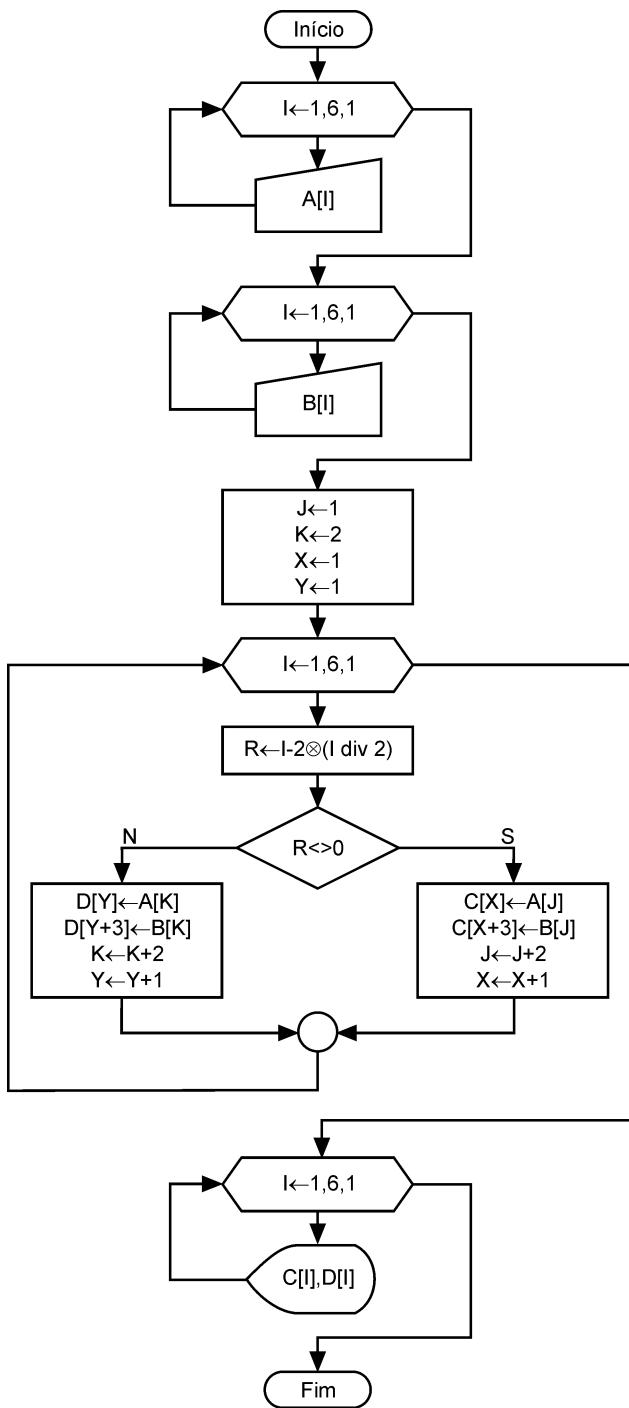


Português estruturado

```

programa Cap06_Ex1q_Pg143
var
  A, B : conjunto[1..15] de real
  I, R : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    R ← I - 2 * (I div 2)
    se (R = 0) então
      B[I] ← A[I] / 2
    senão
      B[I] ← A[I] * 1,5
    fim_se
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva B[I]
  fim_para
fim
  
```

r) Diagrama de blocos

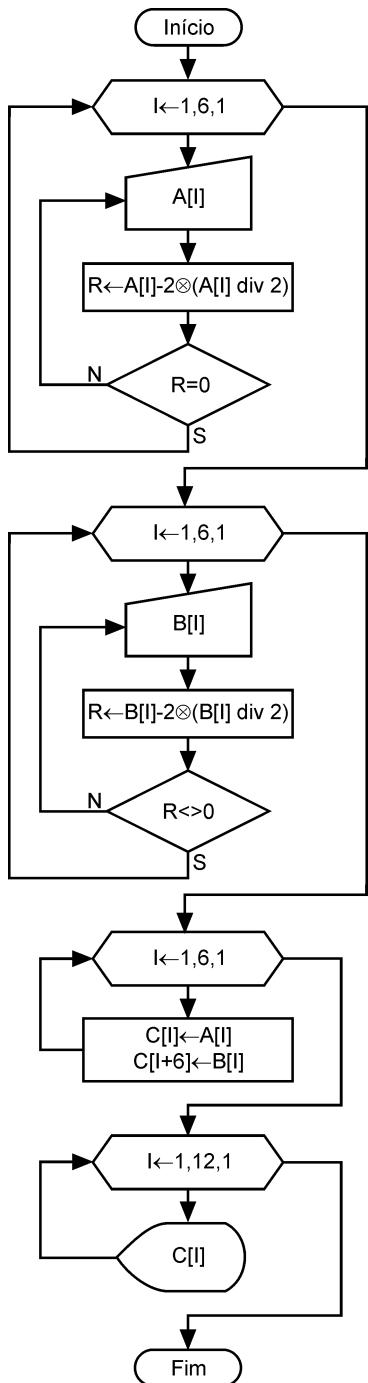


Português estruturado

```

programa Cap06_Ex1r_Pg143
var
  A, B, C, D : conjunto[1..6] de inteiro
  I, J, K, R, X, Y : inteiro
início
  para I de 1 até 6 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 6 passo 1 faça
    leia B[I]
  fim_para
  J ← 1
  K ← 2
  X ← 1
  Y ← 1
  para I de 1 até 6 passo 1 faça
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      C[X] ← A[J]
      C[X + 3] ← B[J]
      J ← J + 2
      X ← X + 1
    senão
      D[Y] ← A[K]
      D[Y + 3] ← B[K]
      K ← K + 2
      Y ← Y + 1
    fim_se
  fim_para
  para I de 1 até 6 passo 1 faça
    escreva C[I], D[I]
  fim_para
fim
  
```

s) Diagrama de blocos

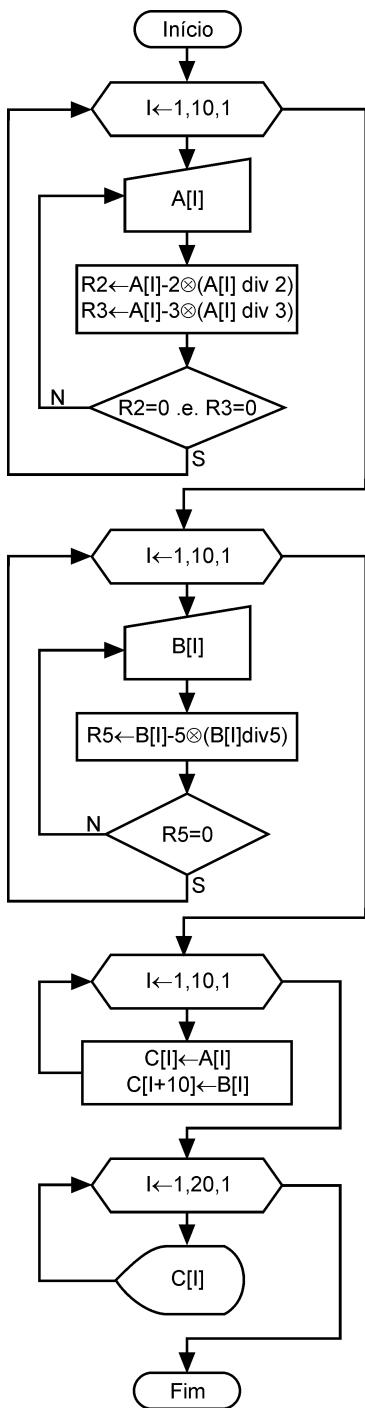


Português estruturado

```

programa Cap06_Ex1s_Pg143
var
  A, B : conjunto[1..6] de inteiro
  C : conjunto[1..12] de inteiro
  I, R : inteiro
início
  para I de 1 até 6 passo 1 faça
    repita
      leia A[I]
      R ← A[I] - 2 * (A[I] div 2)
      até_que (R = 0)
    fim_para
  para I de 1 até 6 passo 1 faça
    repita
      leia B[I]
      R ← B[I] - 2 * (B[I] div 2)
      até_que (R <> 0)
    fim_para
  para I de 1 até 6 passo 1 faça
    C[I] ← A[I]
    C[I + 6] ← B[I]
  fim_para
  para I de 1 até 12 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

t) Diagrama de blocos

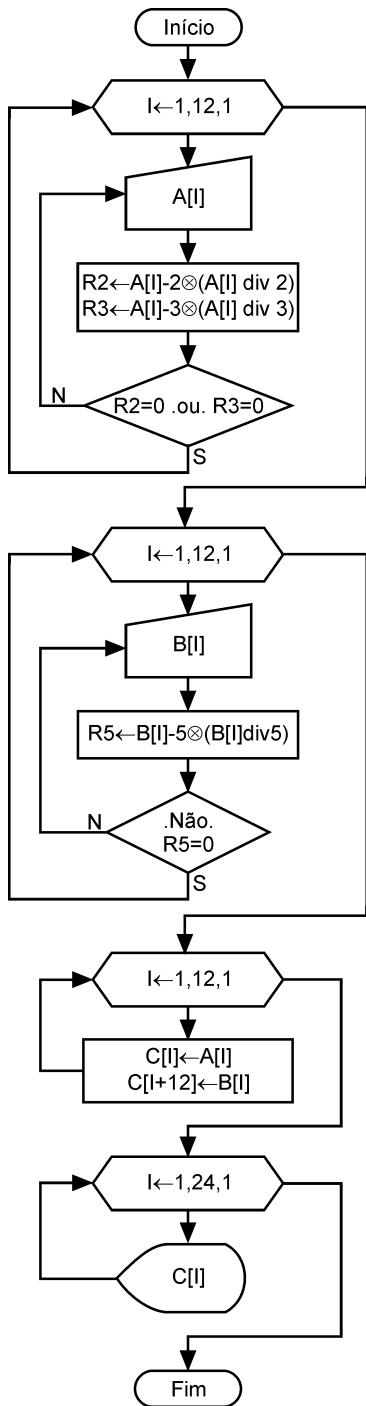


Português estruturado

```

programa Cap06_Ex1t_Pg143
var
  A, B : conjunto[1..10] de inteiro
  C : conjunto[1..20] de inteiro
  I, R2, R3, R5 : inteiro
início
  para I de 1 até 10 passo 1 faça
    repita
      leia A[I]
      R2 ← A[I] - 2 * (A[I] div 2)
      R3 ← A[I] - 3 * (A[I] div 3)
      até_que (R2 = 0) .e. (R3 = 0)
    fim_para
    para I de 1 até 10 passo 1 faça
      repita
        leia B[I]
        R5 ← B[I] - 5 * (B[I] div 5)
        até_que (R5 = 0)
      fim_para
      para I de 1 até 10 passo 1 faça
        C[I] ← A[I]
        C[I + 10] ← B[I]
      fim_para
      para I de 1 até 20 passo 1 faça
        escreva C[I]
      fim_para
    fim
  
```

u) Diagrama de blocos

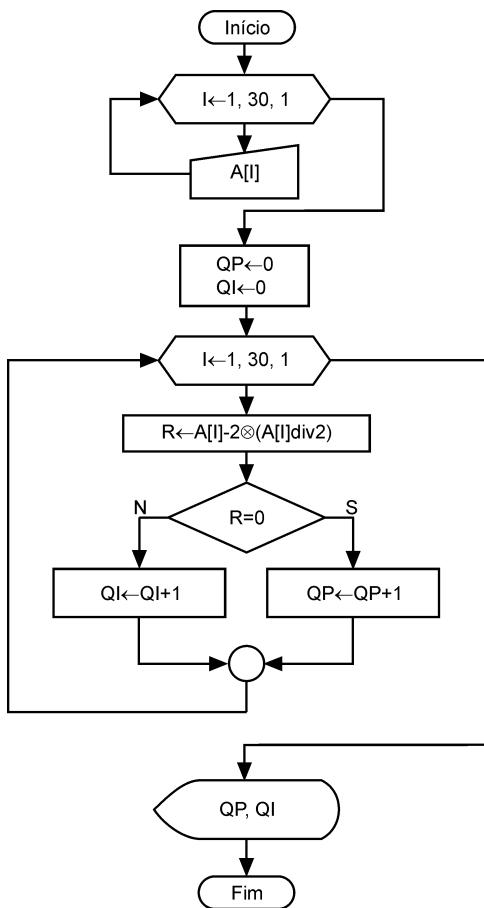


Português estruturado

```

programa Cap06_Exlu_Pg143
var
  A, B : conjunto[1..12] de inteiro
  C : conjunto[1..24] de inteiro
  I, R2, R3, R5 : inteiro
início
  para I de 1 até 12 passo 1 faça
    repita
      leia A[I]
      R2 ← A[I] - 2 * (A[I] div 2)
      R3 ← A[I] - 3 * (A[I] div 3)
      até_que (R2 = 0) .ou. (R3 = 0)
    fim_para
  para I de 1 até 12 passo 1 faça
    repita
      leia B[I]
      R5 ← B[I] - 5 * (B[I] div 5)
      até_que .não. (R5 = 0)
    fim_para
  para I de 1 até 12 passo 1 faça
    C[I] ← A[I]
    C[I + 12] ← B[I]
  fim_para
  para I de 1 até 24 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

v) Diagrama de blocos

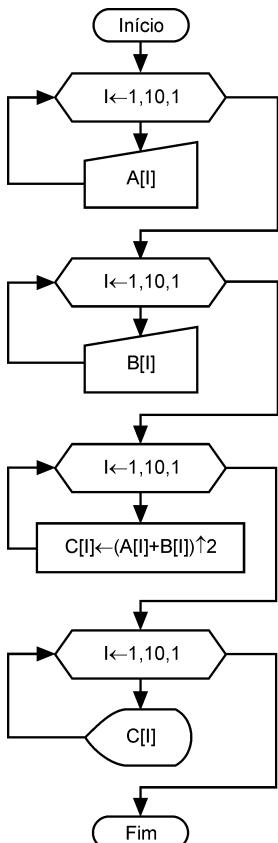


Português estruturado

```

programa Cap06_Ex1v_Pg144
var
  A : conjunto[1..30] de inteiro
  I, R, QP, QI : inteiro
início
  para I de 1 até 30 passo 1 faça
    leia A[I]
  fim_para
  QP ← 0
  QI ← 0
  para I de 1 até 30 passo 1 faça
    R ← A[I] - 2 * (A[I] div 2)
    se (R = 0) então
      QP ← QP + 1
    senão
      QI ← QI + 1
    fim_se
  fim_para
  escreva QP, QI
fim
  
```

w) Diagrama de blocos

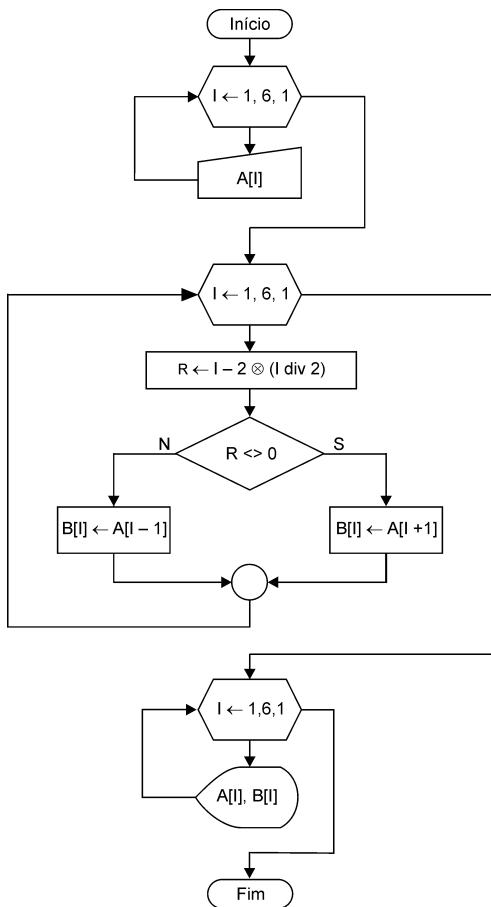


Português estruturado

```

programa Cap06_Ex1w_Pg144
var
  A, B, C : conjunto[1..10] de inteiro
  I : inteiro
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 10 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 10 passo 1 faça
    C[I] ← (A[I] + B[I]) ↑ 2
  fim_para
  para I de 1 até 10 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

x) Diagrama de blocos

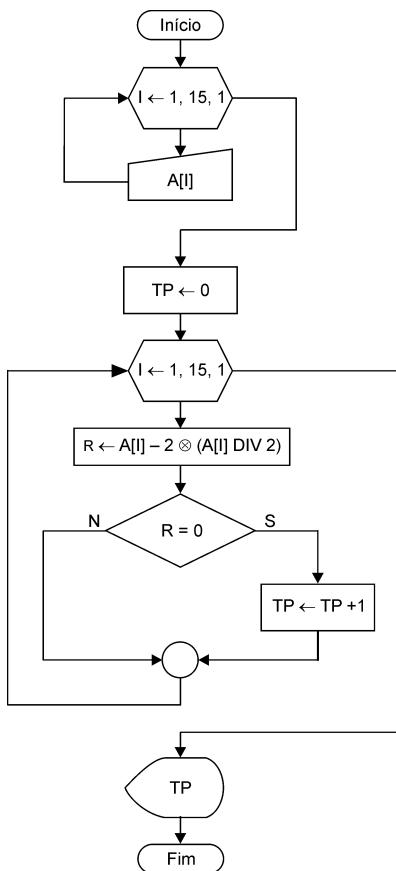


Português estruturado

```

programa Cap06_Ex1x_Pg144
var
  A, B : conjunto[1..6] de real
  I, R : inteiro
início
  para I de 1 até 6 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 6 passo 1 faça
    R ← I - 2 * (I div 2)
    se (R <> 0) então
      B[I] ← A[I + 1]
    senão
      B[I] ← A[I - 1]
    fim_se
  fim_para
  para I de 1 até 6 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

y) Diagrama de blocos

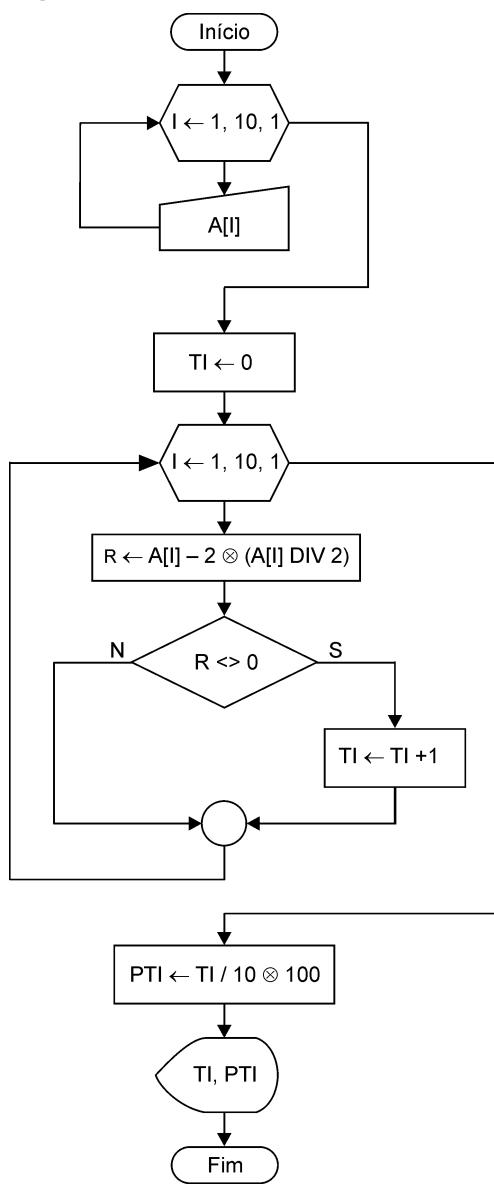


Português estruturado

```

programa Cap06_Ex1y_Pg144
var
  A : conjunto[1..15] de inteiro
  I, TP, R : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  TP ← 0
  para I de 1 até 15 passo 1 faça
    R ← A[I] - 2 * (A[I] DIV 2)
    se (R = 0) então
      TP ← TP + 1
    fim_se
  fim_para
  escreva TP
fim
  
```

z) Diagrama de blocos



Português estruturado

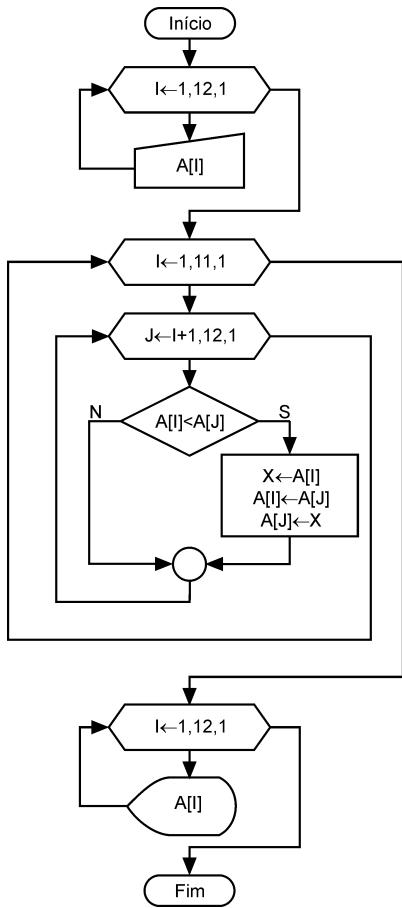
```

programa Cap06_Ex1z_Pg144
var
  A : conjunto[1..10] de inteiro
  I, TI, R : inteiro
  PTI : real
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  TI ← 0
  para I de 1 até 10 passo 1 faça
    R ← A[I] - 2 * (A[I] div 2)
    se (R <> 0) então
      TI ← TI + 1
    fim_se
  fim_para
  PTI ← TI / 10 * 100
  escreva TI, PTI
fim
  
```

7 - Exercícios de fixação do capítulo 7

Tópico 7.5 - Exercício 1 - Página 171

a) Diagrama de blocos

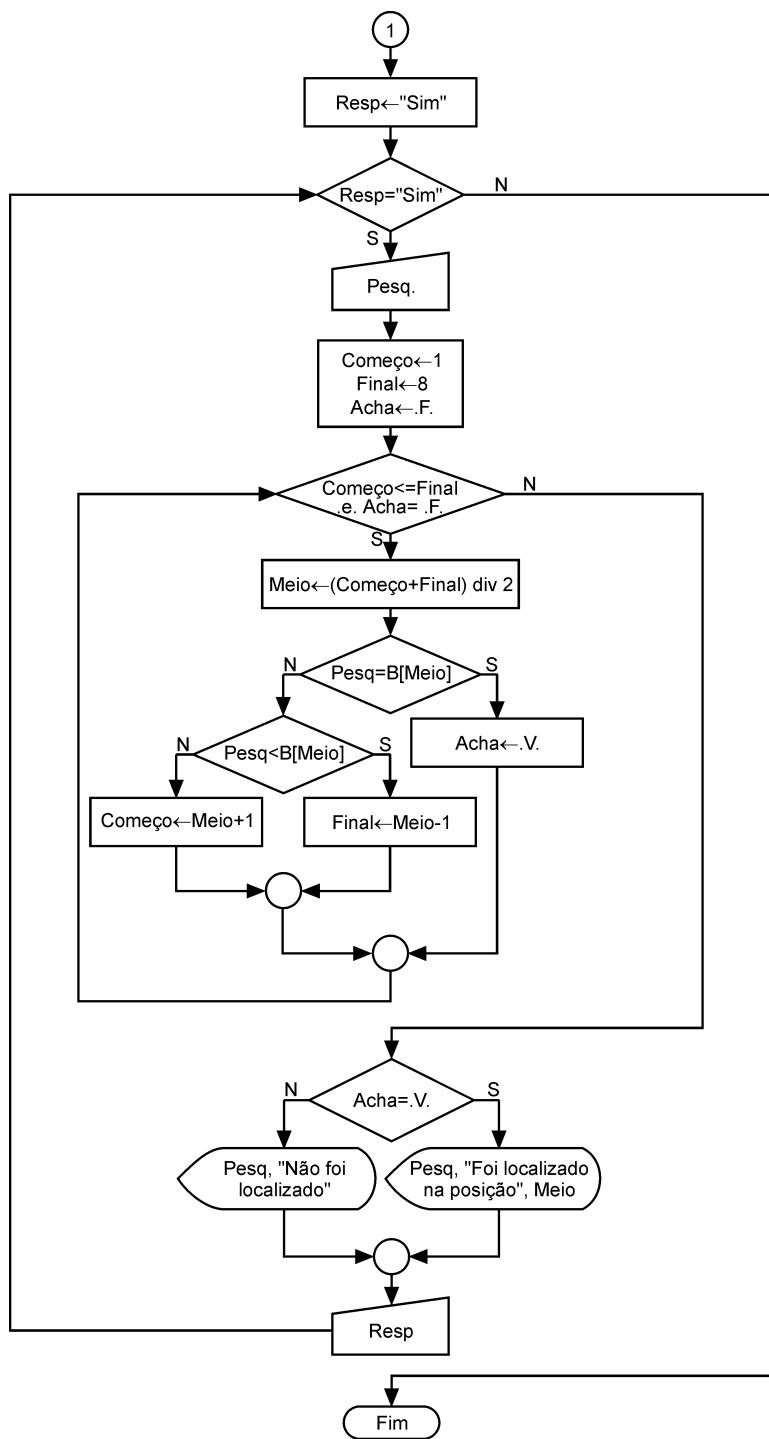
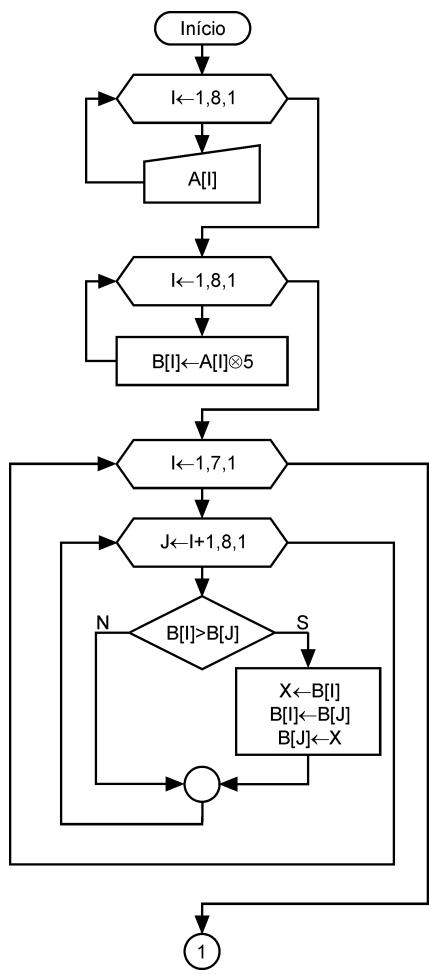


Português estruturado

```

programa Cap07_Ex1a_Pg171
var
  A : conjunto[1..12] de inteiro
  I, J, X : inteiro
início
  para I de 1 até 12 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 11 passo 1 faça
    para J de I + 1 até 12 passo 1 faça
      se (A[I] < A[J]) então
        X ← A[I]
        A[I] ← A[J]
        A[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 12 passo 1 faça
    escreva A[I]
  fim_para
fim
  
```

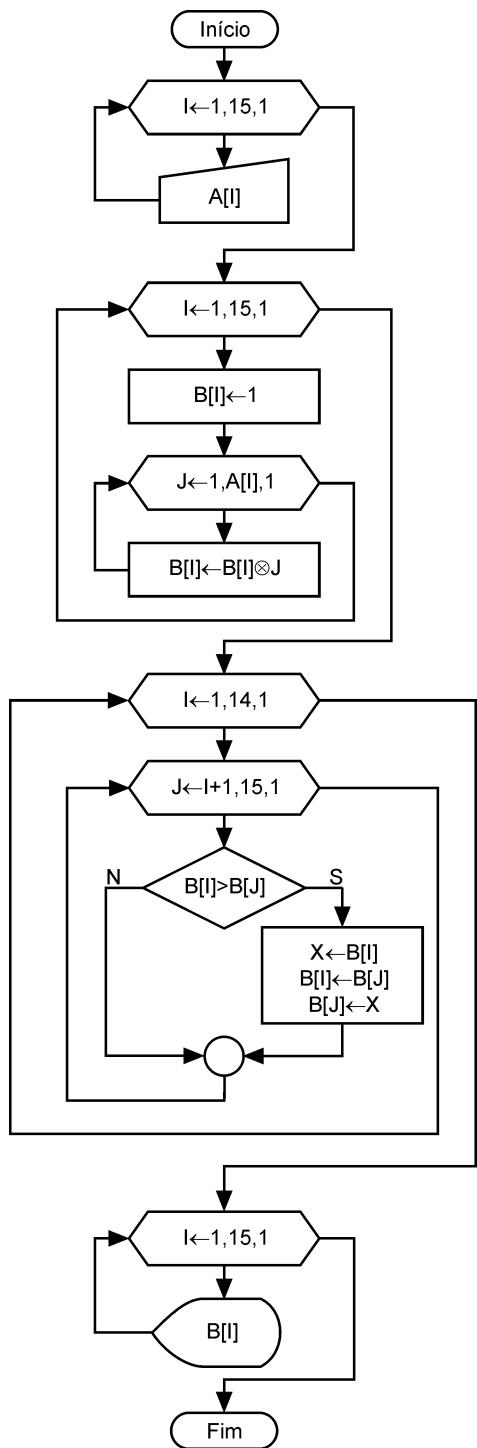
b) Diagrama de blocos



Português Estruturado

```
programa Cap07_Ex1b_Pg171
var
    A, B : conjunto[1..8] de inteiro
    I, J, X, COMEÇO, FINAL, MEIO, PESQ : inteiro
    RESP : cadeia
    ACHA : lógico
início
    para I de 1 até 8 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 8 passo 1 faça
        B[I] ← A[I] * 5
    fim_para
    para I de 1 até 7 passo 1 faça
        para J de I + 1 até 8 passo 1 faça
            se (B[I] > B[J]) então
                X ← B[I]
                B[I] ← B[J]
                B[J] ← X
            fim_se
        fim_para
    fim_para
    RESP ← "SIM"
    enquanto (RESP = "SIM") faça
        leia PESQ
        COMEÇO ← 1
        FINAL ← 8
        ACHA ← .Falso.
        enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso) faça
            MEIO ← (COMEÇO + FINAL) div 2
            se (PESQ = B[MEIO]) então
                ACHA ← .Verdadeiro.
            senão
                se (PESQ < B[MEIO]) então
                    FINAL ← MEIO - 1
                senão
                    COMEÇO ← MEIO + 1
                fim_se
            fim_se
        fim_enquanto
        se (ACHA = .Verdadeiro.) então
            escreva PESQ, " foi localizado na posição ", MEIO
        senão
            escreva PESQ, " não foi localizado"
        fim_se
        leia RESP
    fim_enquanto
fim
```

c) Diagrama de blocos

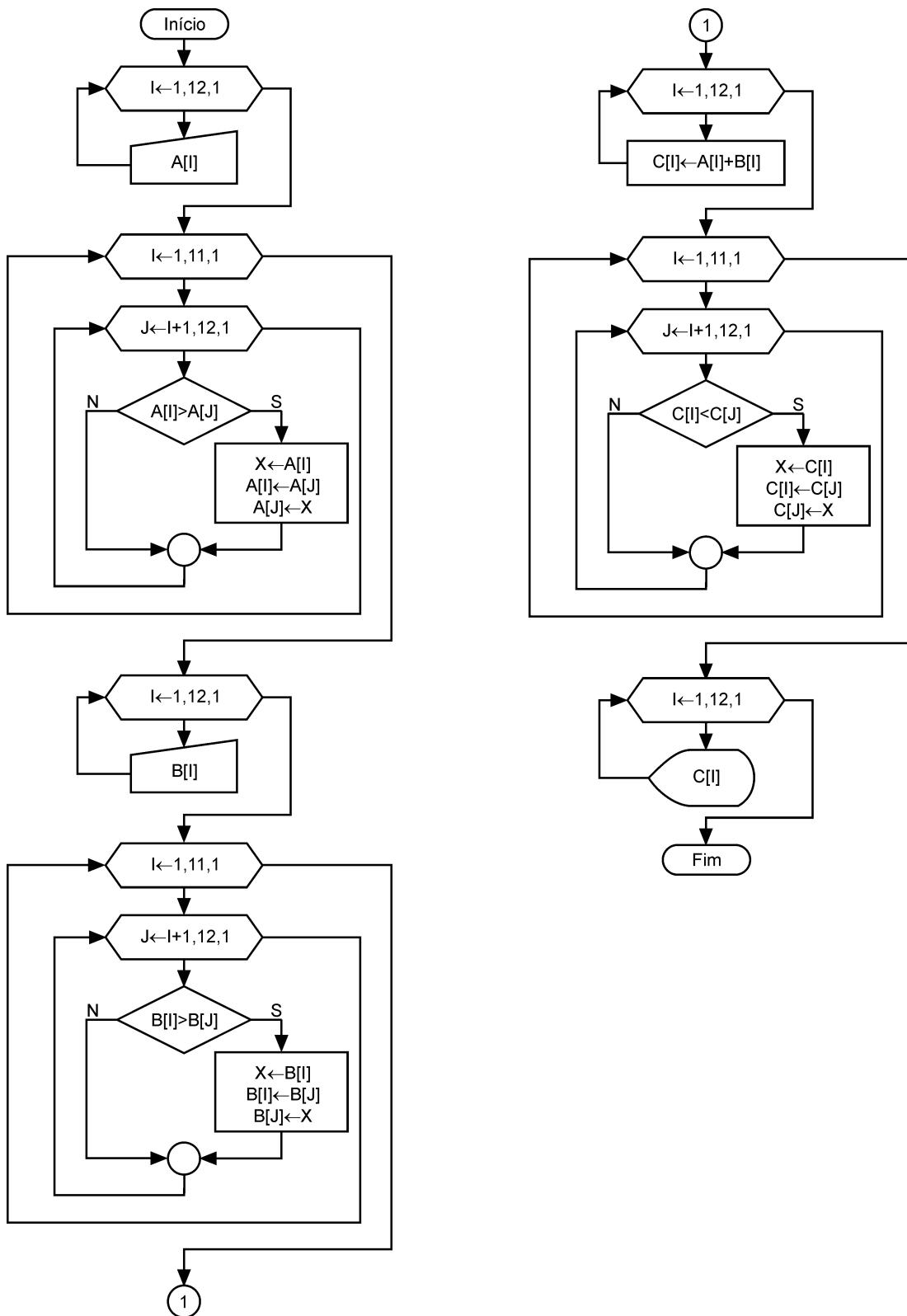


Português estruturado

```

programa Cap07_Ex1c_Pg171
var
  A, B : conjunto[1..15] de inteiro
  I, J, X : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    B[I] ← 1
    para J de 1 até A[I] passo 1 faça
      B[I] ← B[I] * J
    fim_para
  fim_para
  para I de 1 até 14 passo 1 faça
    para J de I + 1 até 15 passo 1 faça
      se (B[I] > B[J]) então
        X ← B[I]
        B[I] ← B[J]
        B[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva B[I]
  fim_para
fim
  
```

d) Diagrama de blocos



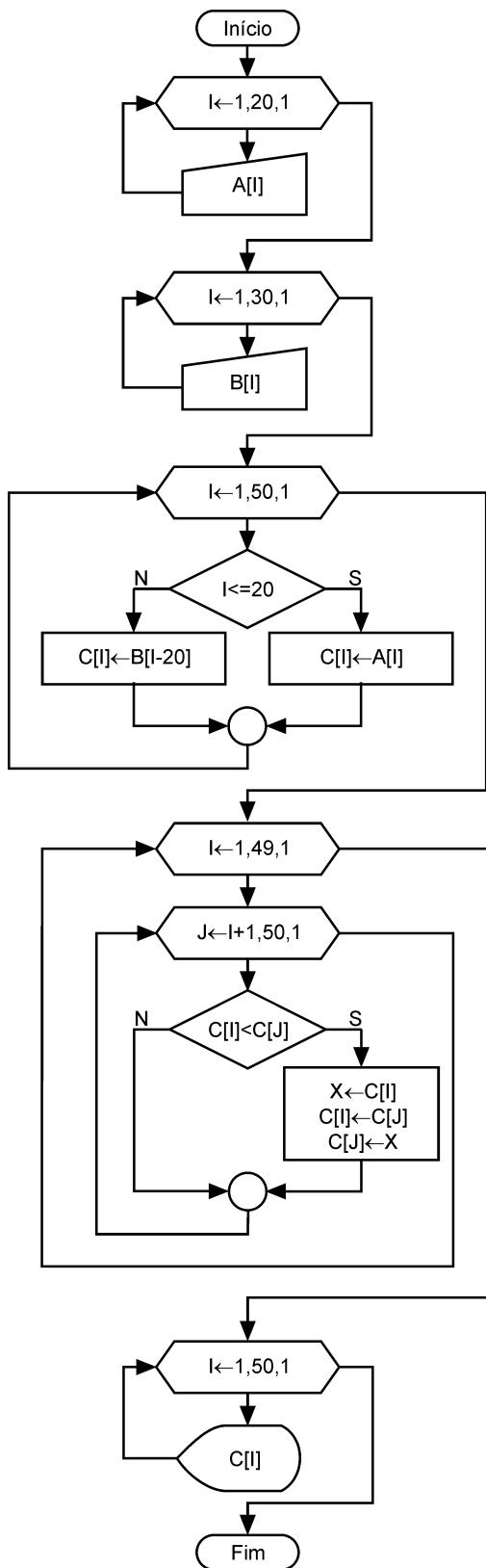
Português Estruturado

```

programa Cap07_Ex1d_Pg171
var
  A, B, C : conjunto[1..12] de real
  I, J, X : inteiro
início
  para I de 1 até 12 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 11 passo 1 faça
    para J de I + 1 até 12 passo 1 faça
      se (A[I] > A[J]) então
        X ← A[I]
        A[I] ← A[J]
        A[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 12 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 11 passo 1 faça
    para J de I + 1 até 12 passo 1 faça
      se (B[I] > B[J]) então
        X ← B[I]
        B[I] ← B[J]
        B[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 12 passo 1 faça
    C[I] ← A[I] + B[I]
  fim_para
  para I de 1 até 11 passo 1 faça
    para J de I + 1 até 12 passo 1 faça
      se (C[I] < C[J]) então
        X ← C[I]
        C[I] ← C[J]
        C[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 12 passo 1 faça
    escreva C[I]
  fim_para
fim

```

e) Diagrama de blocos

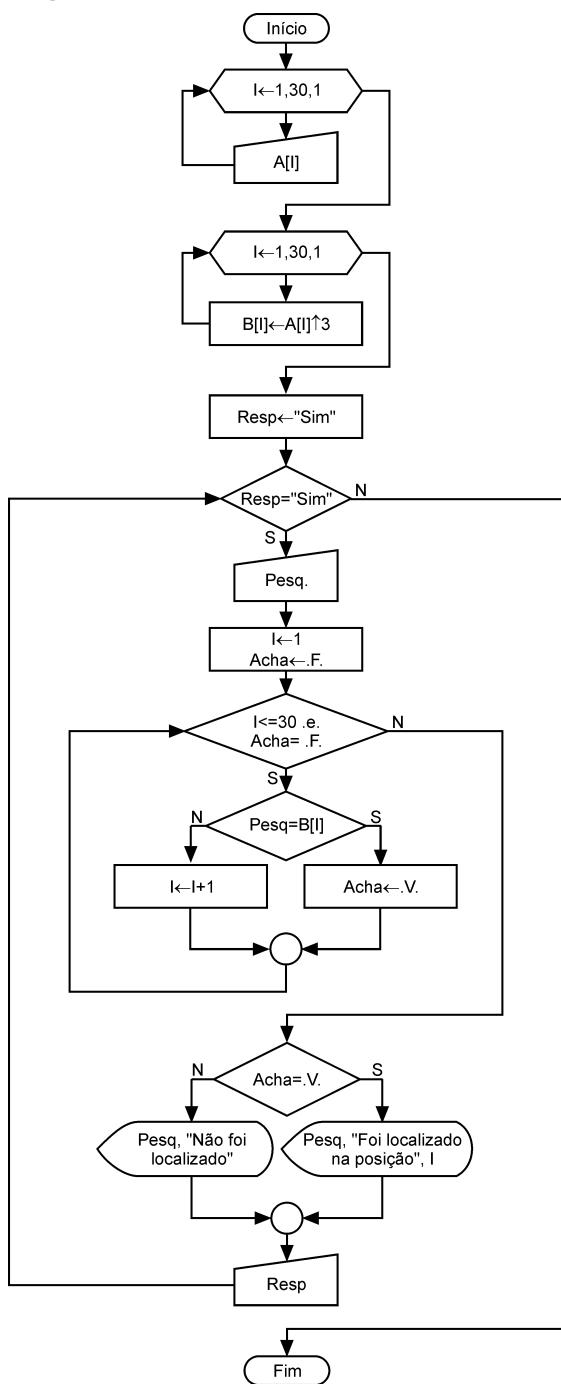


Português estruturado

```

programa Cap07_Exle_Pg171
var
  A : conjunto[1..20] de cadeia
  B : conjunto[1..30] de cadeia
  C : conjunto[1..50] de cadeia
  I, J, X : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 30 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 50 passo 1 faça
    se (I <= 20) então
      C[I] ← A[I]
    senão
      C[I] ← B[I - 20]
    fim_se
  fim_para
  para I de 1 até 49 passo 1 faça
    para J de I + 1 até 50 passo 1 faça
      se (C[I] < C[J]) então
        X ← C[I]
        C[I] ← C[J]
        C[J] ← X
      fim_se
    fim_para
  para I de 1 até 50 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

f) Diagrama de blocos

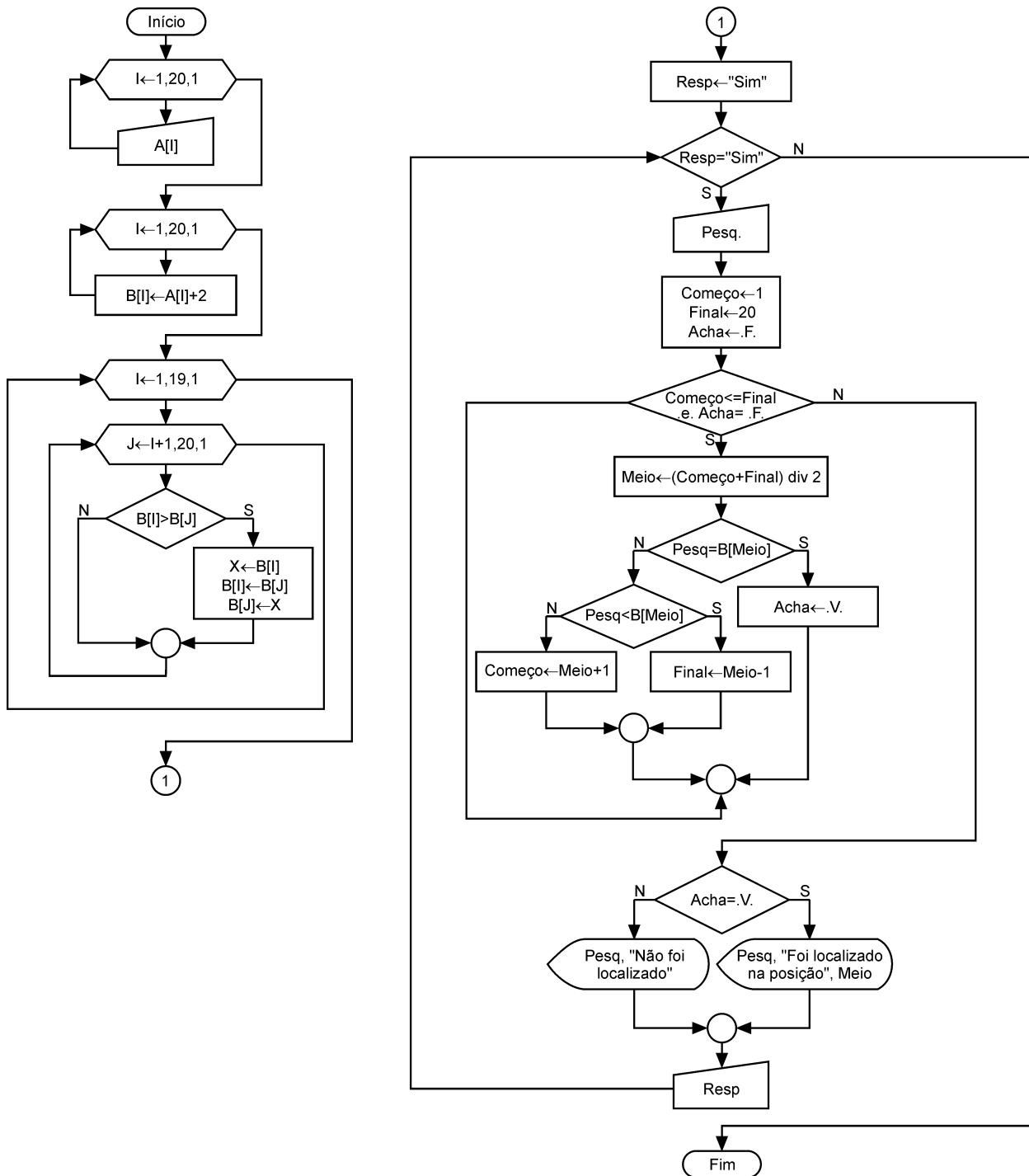


Português estruturado

```

programa Cap07_Ex1f_Pg171
var
  A, B : conjunto[1..30] de inteiro
  I, PESQ : inteiro
  RESP : cadeia
  ACHA : lógico
início
  para I de 1 até 30 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 30 passo 1 faça
    B[I] ← A[I] ↑ 3
  fim_para
  RESP ← "SIM"
  enquanto (RESP = "SIM") faça
    leia PESQ
    I ← 1
    ACHA ← .Falso.
    enquanto (I <= 30) .e. (ACHA = .Falso.) faça
      se (PESQ = B[I]) então
        ACHA ← .Verdadeiro.
      senão
        I ← I + 1
      fim_se
    fim_enquanto
    se (ACHA = .Verdadeiro.) então
      escreva PESQ, " foi localizado na posição ", I
    senão
      escreva PESQ, " não foi localizado"
    fim_se
    leia RESP
  fim_enquanto
fim
  
```

g) Diagrama de blocos



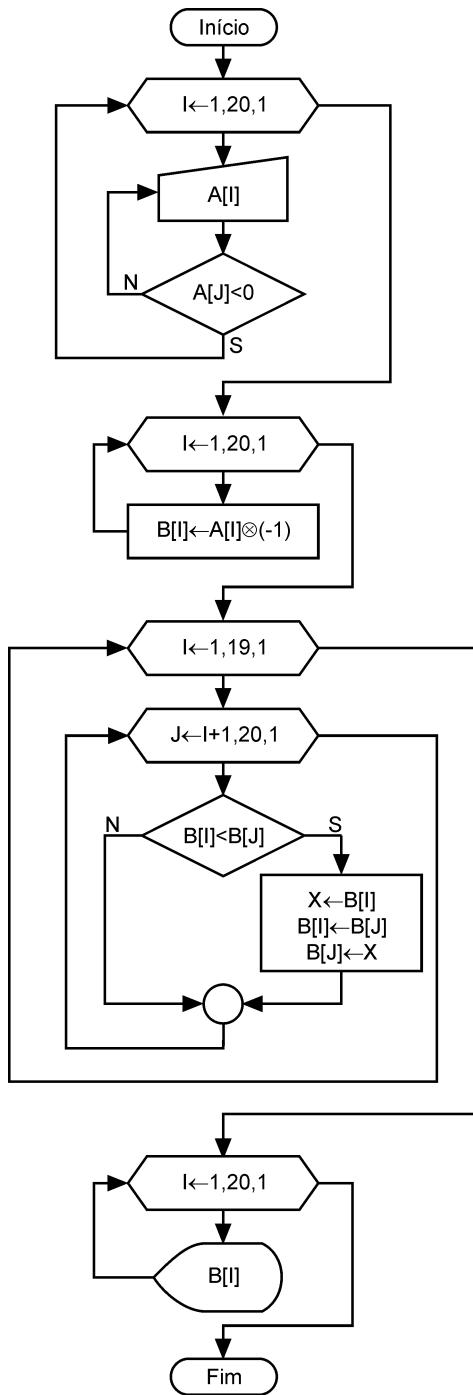
Português Estruturado

```

programa Cap07_Ex1g_Pg171
var
  A, B : conjunto[1..20] de inteiro
  I, J, X, COMEÇO, FINAL, MEIO, PESQ : inteiro
  RESP : cadeia
  ACHA : lógico
início
  para I de 1 até 20 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 20 passo 1 faça
    B[I] ← A[I] + 2
  fim_para
  para I de 1 até 19 passo 1 faça
    para J de I + 1 até 20 passo 1 faça
      se (B[I] > B[J]) então
        X ← B[I]
        B[I] ← B[J]
        B[J] ← X
      fim_se
    fim_para
  fim_para
  RESP ← "SIM"
  enquanto (RESP = "SIM") faça
    leia PESQ
    COMEÇO ← 1
    FINAL ← 20
    ACHA ← .Falso.
    enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso) faça
      MEIO ← (COMEÇO + FINAL) div 2
      se (PESQ = B[MEIO]) então
        ACHA ← .Verdadeiro.
      senão
        se (PESQ < B[MEIO]) então
          FINAL ← MEIO - 1
        senão
          COMEÇO ← MEIO + 1
        fim_se
      fim_se
    fim_enquanto
    se (ACHA = .Verdadeiro.) então
      escreva PESQ, " foi localizado na posição ", MEIO
    senão
      escreva PESQ, " não foi localizado"
    fim_se
    leia RESP
  fim_enquanto
fim

```

h) Diagrama de blocos

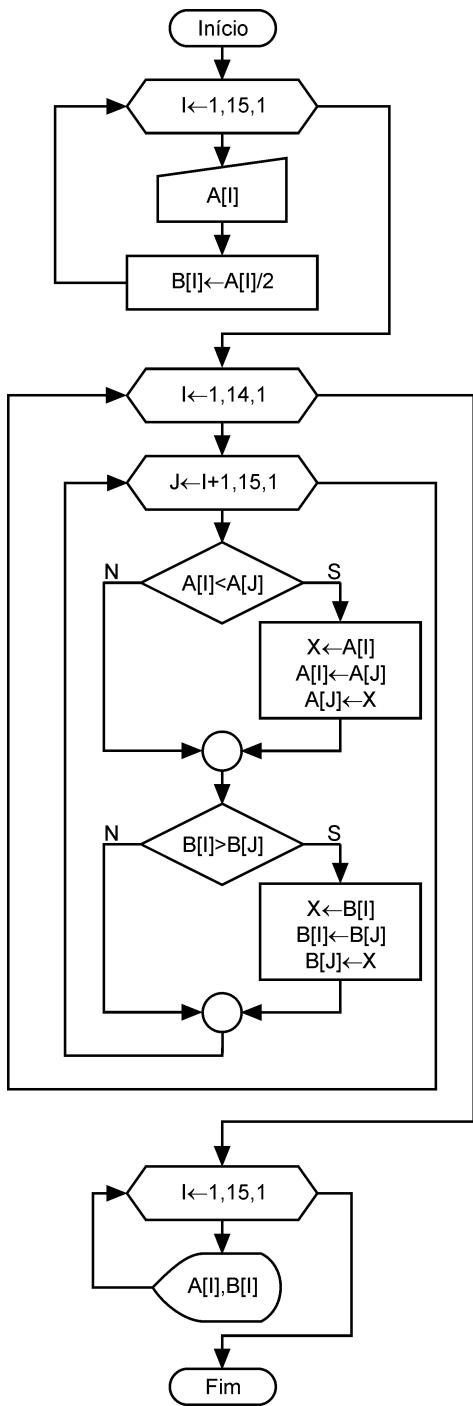


Português estruturado

```

programa Cap07_Ex1h_Pg171
var
  A, B : conjunto[1..20] de inteiro
  I, J, X : inteiro
início
  para I de 1 até 20 passo 1 faça
    repita
      leia A[I]
      até_que (A[I] < 0)
    fim_para
  para I de 1 até 20 passo 1 faça
    B[I] ← A[I] * (-1)
  fim_para
  para I de 1 até 19 passo 1 faça
    para J de I + 1 até 20 passo 1 faça
      se (B[I] < B[J]) então
        X ← B[I]
        B[I] ← B[J]
        B[J] ← X
      fim_se
    fim_para
  para I de 1 até 20 passo 1 faça
    escreva B[I]
  fim_para
fim
  
```

i) Diagrama de blocos

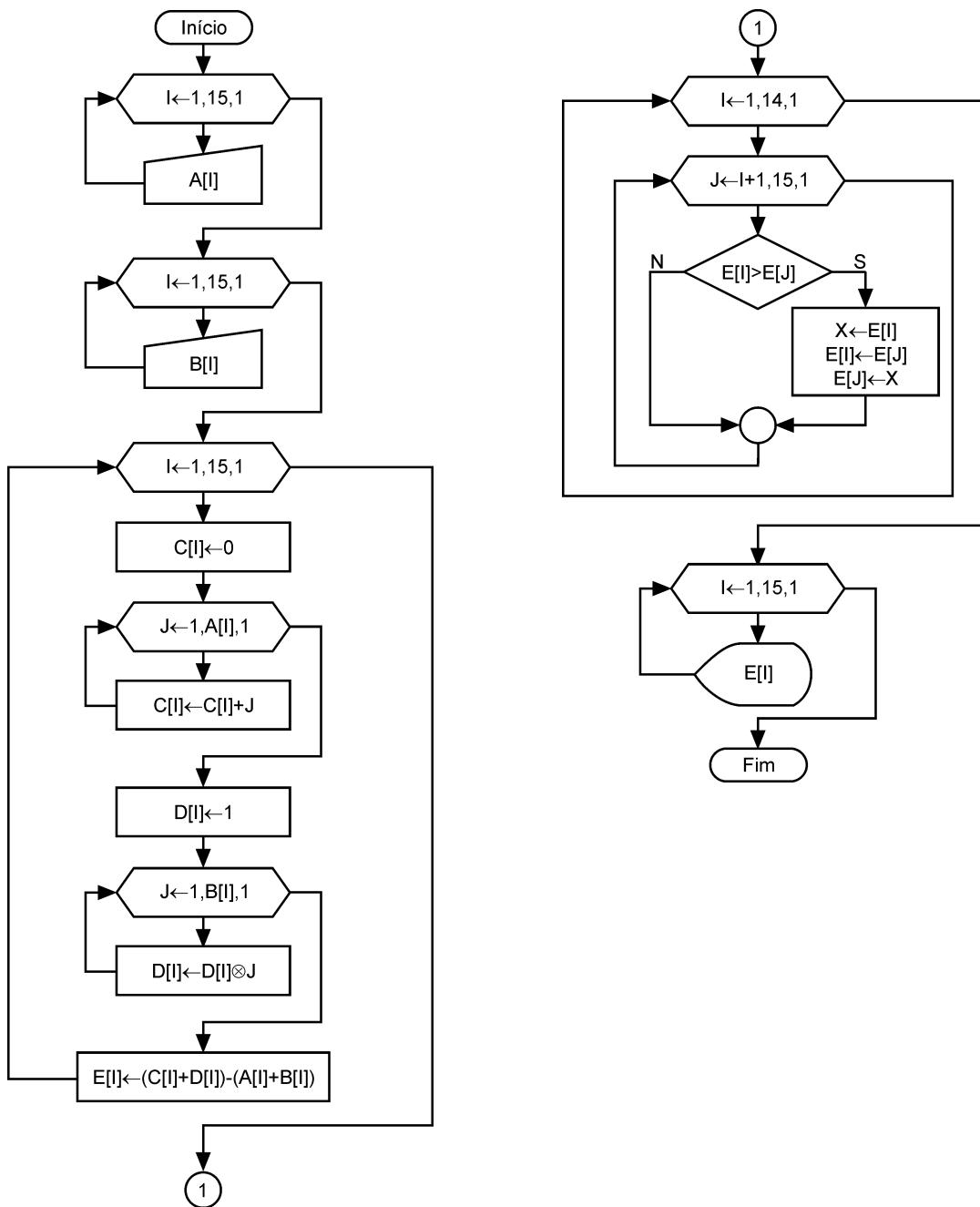


Português estruturado

```

programa Cap07_Ex1i_Pg172
var
  A, B : conjunto[1..15] de real
  I, J, X : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
    B[I] ← A[I] / 2
  fim_para
  para I de 1 até 14 passo 1 faça
    para J de I + 1 até 15 passo 1 faça
      se (A[I] < A[J]) então
        X ← A[I]
        A[I] ← A[J]
        A[J] ← X
      fim_se
      se (B[I] > B[J]) então
        X ← B[I]
        B[I] ← B[J]
        B[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 20 passo 1 faça
    escreva A[I], B[I]
  fim_para
fim
  
```

j) Diagrama de blocos



Português Estruturado

```

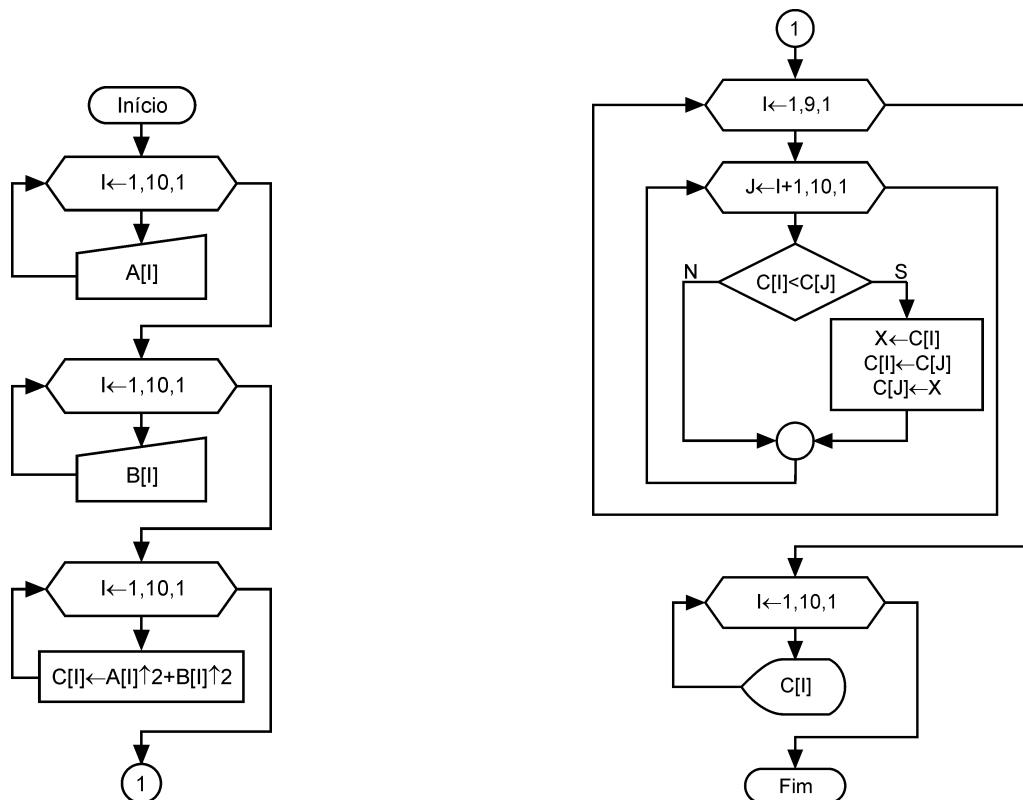
programa Cap07_Ex1j_Pg172
var
  A, B, C, D, E : conjunto[1..15] de inteiro
  I, J, X : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    C[I] ← 0
    para J de 1 até A[I] passo 1 faça
      C[I] ← C[I] + J
    fim_para
    D[I] ← 1
    E[I] ← -(C[I]+D[I]) - (A[I]+B[I])
  fim_para

```

```

para J de 1 até B[I] passo 1 faça
  D[I] ← D[I] * J
fim_para
  E[I] ← (C[I] + D[I]) - (A[I] + B[I])
fim_para
para I de 1 até 14 passo 1 faça
  para J de I + 1 até 15 passo 1 faça
    se (E[I] > E[J]) então
      X ← E[I]
      E[I] ← E[J]
      E[J] ← X
    fim_se
  fim_para
fim_para
para I de 1 até 15 passo 1 faça
  escreva E[I]
fim_para
fim
  
```

k) Diagrama de blocos



Português estruturado

```

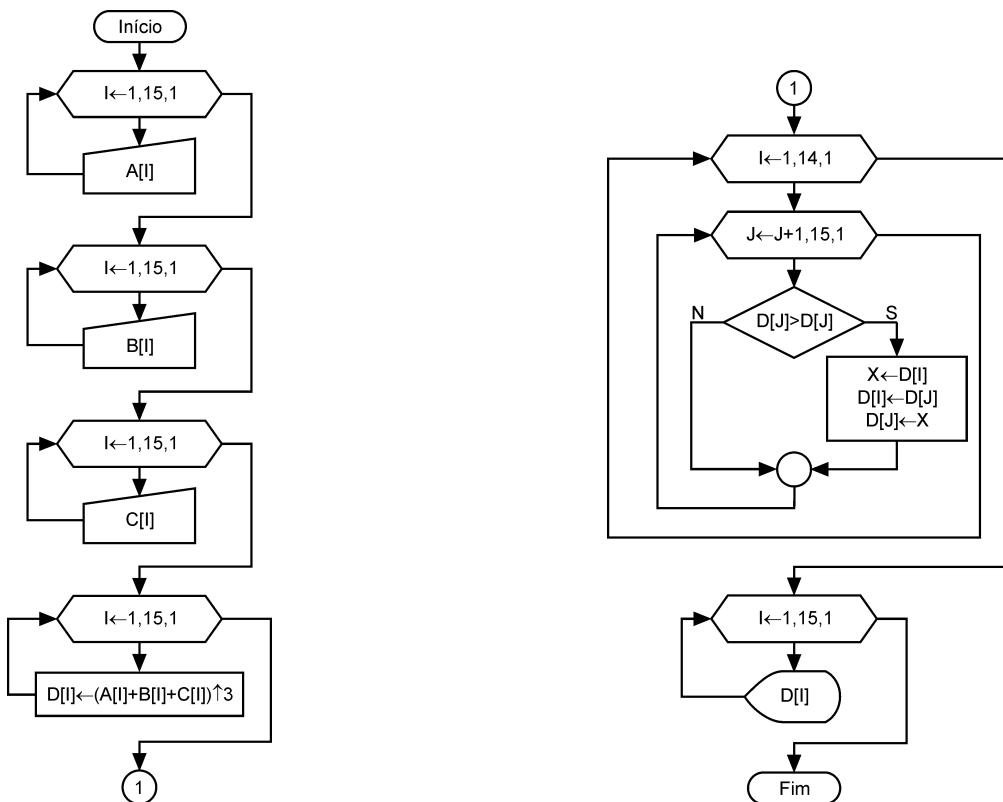
programa Cap07_Ex1k_Pg172
var
  A, B, C : conjunto[1..10] de inteiro
  I, J : inteiro
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 10 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 10 passo 1 faça
    C[I] ← A[I] ↑ 2 + B[I] ↑ 2
  fim_para
  para I de 1 até 9 passo 1 faça
    para J de I + 1 até 10 passo 1 faça
      se (C[I] < C[J]) então
        X ← C[I]
        C[I] ← C[J]
        C[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 10 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

```

C[I] ← C[J]
C[J] ← X
fim_se
fim_para
fim_para
para I de 1 até 10 passo 1 faça
  escreva C[I]
fim_para
fim

```

I) Diagrama de blocos



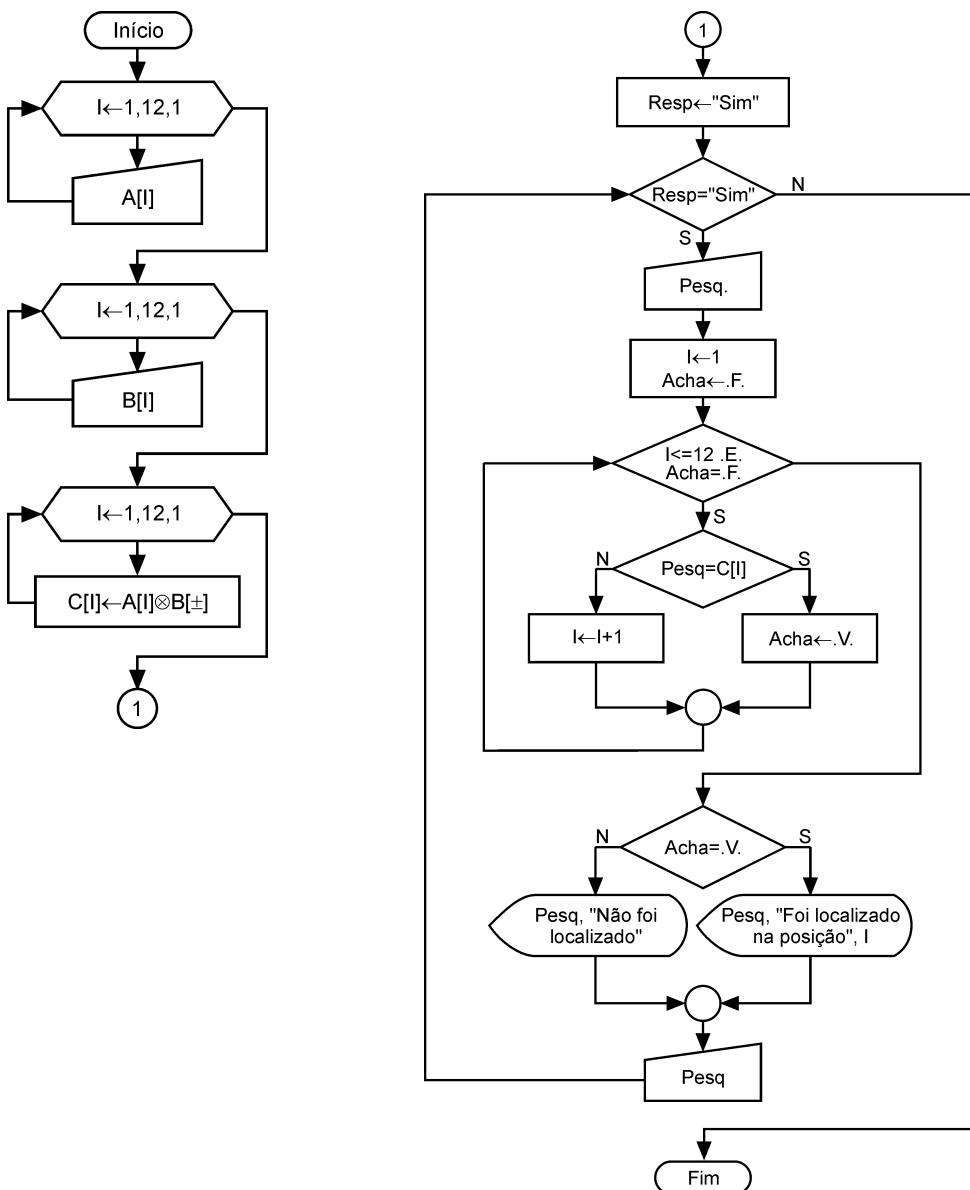
Português Estruturado

```

programa Cap07_Ex11_Pg172
var
  A, B, C, D : conjunto[1..15] de real
  I, J : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    leia C[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    D[I] ← (A[I] + B[I] + C[I]) ↑ 3
  fim_para
  para I de 1 até 14 passo 1 faça
    para J de I + 1 até 15 passo 1 faça
      se (D[I] > D[J]) então
        X ← D[I]
        D[I] ← D[J]
        D[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva D[I]
  fim_para
fim

```

m) Diagrama de blocos



Português Estruturado

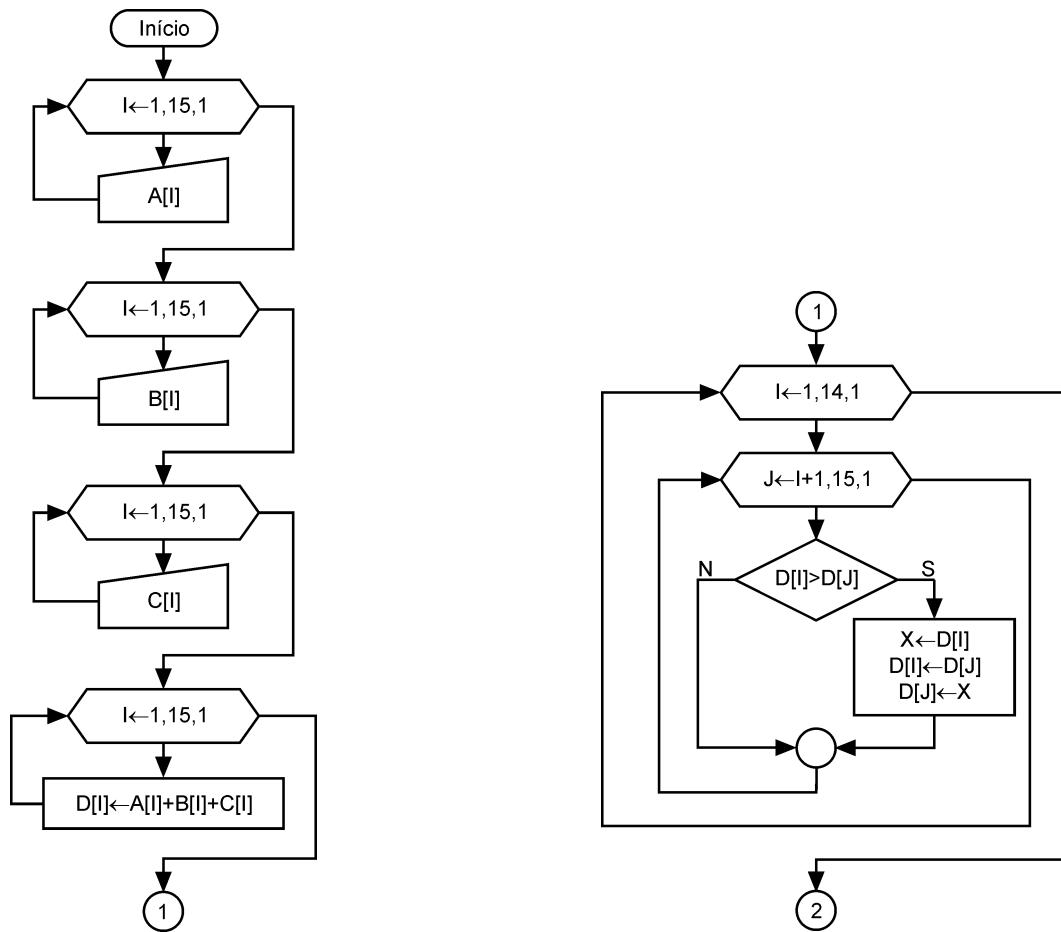
```

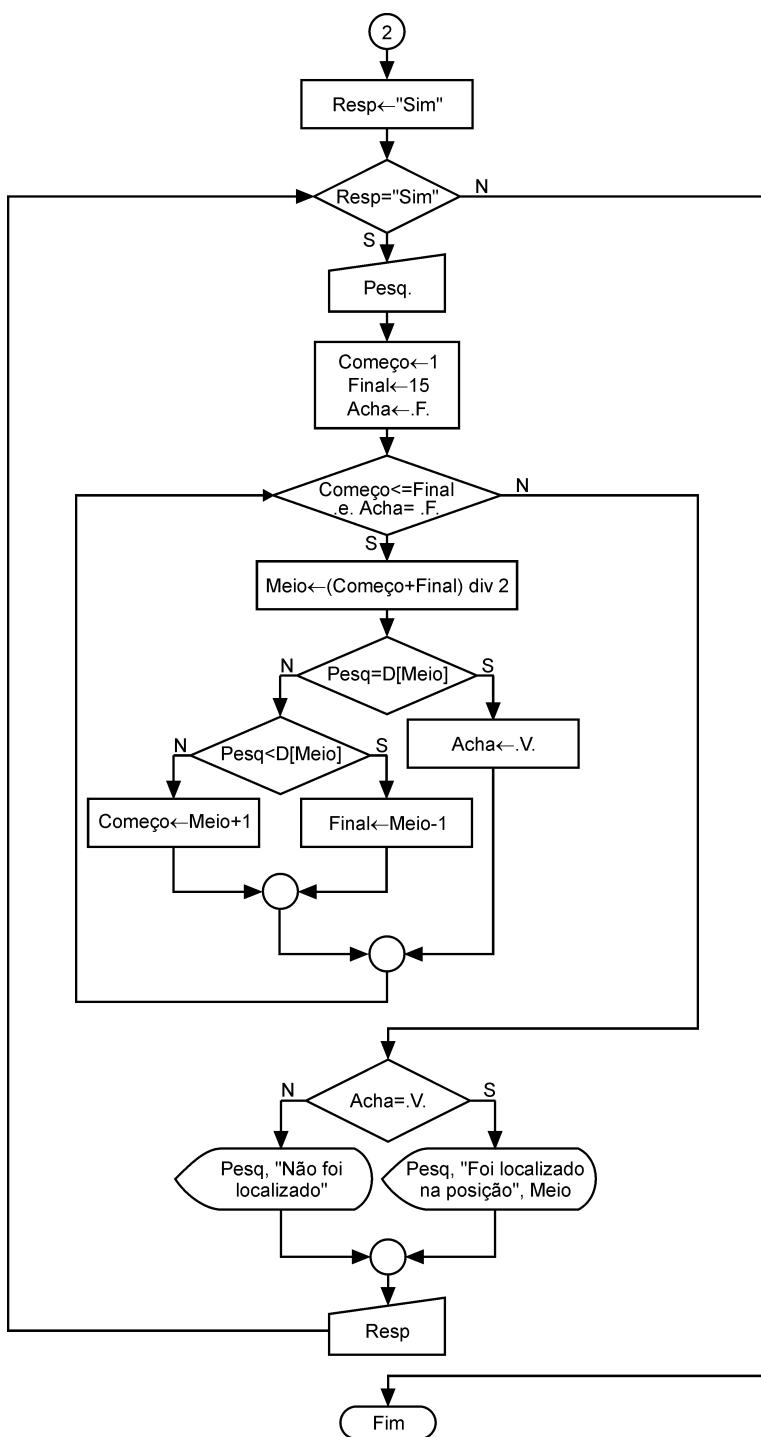
programa Cap07_Ex1m_Pg172
var
    A, B, C : conjunto[1..12] de real
    I : inteiro
    PESQ : real
    RESP : cadeia
    ACHA : lógico
início
    para I de 1 até 12 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 12 passo 1 faça
        leia B[I]
    fim_para
    para I de 1 até 12 passo 1 faça
        C[I] ← A[I] * B[I]
    fim_para
    RESP ← "SIM"
    enquanto (RESP = "SIM") faça
        leia PESQ
        I ← 1
        ACHA ← .Falso.
    
```

```

enquanto (I <= 12) .e. (ACHA = .Falso.) faça
  se (PESQ = C[I]) então
    ACHA ← .Verdadeiro.
  senão
    I ← I + 1
  fim_se
fim_enquanto
se (ACHA = .Verdadeiro.) então
  escreva PESQ, " foi localizado na posição ", I
senão
  escreva PESQ, " não foi localizado"
fim_se
leia RESP
fim_enquanto
fim
  
```

n) Diagrama de blocos

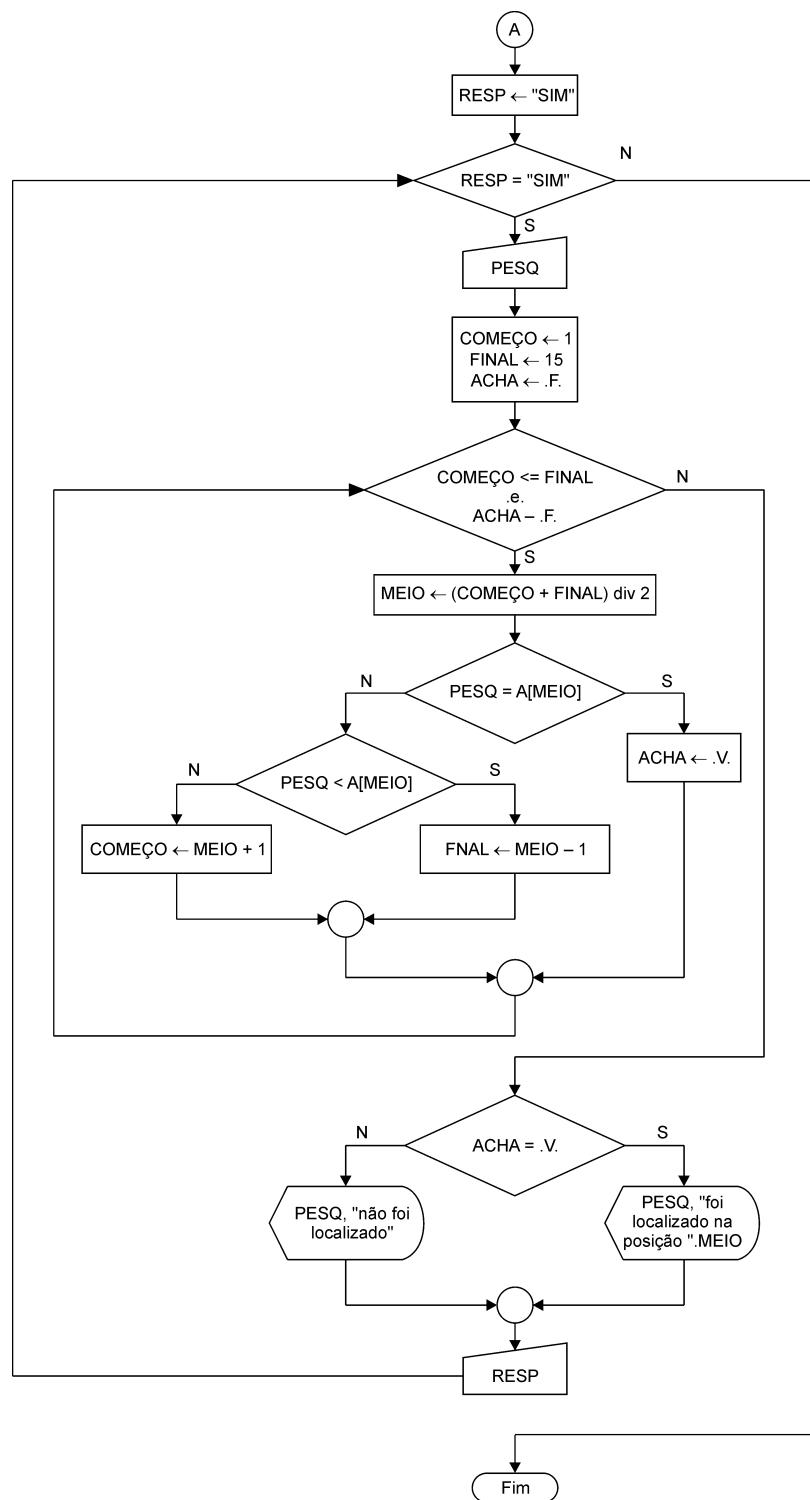
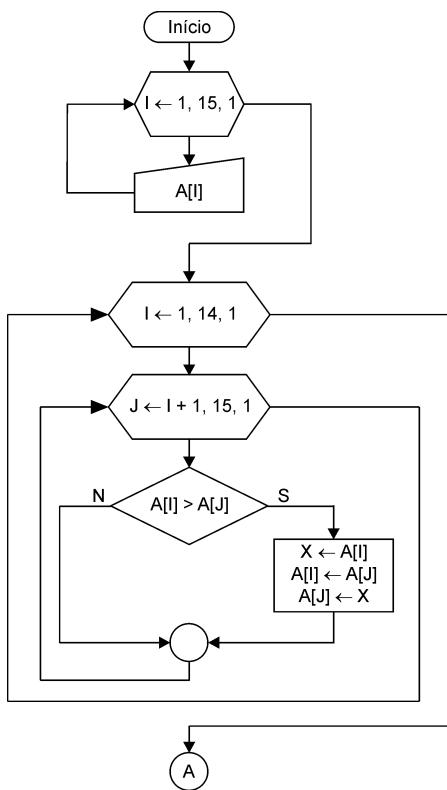




Português Estruturado

```
programa Cap07_Ex1n_Pg172
var
    A, B, C, D : conjunto[1..15] de inteiro
    I, J, X, COMEÇO, FINAL, MEIO, PESQ : inteiro
    RESP : cadeia
    ACHA : lógico
início
    para I de 1 até 15 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 15 passo 1 faça
        leia B[I]
    fim_para
    para I de 1 até 15 passo 1 faça
        leia C[I]
    fim_para
    para I de 1 até 15 passo 1 faça
        D[I] ← A[I] + B[I] + C[I]
    fim_para
    para I de 1 até 14 passo 1 faça
        para J de I + 1 até 15 passo 1 faça
            se (D[I] > D[J]) então
                X ← D[I]
                D[I] ← D[J]
                D[J] ← X
            fim_se
        fim_para
    fim_para
    RESP ← "SIM"
    enquanto (RESP = "SIM") faça
        leia PESQ
        COMEÇO ← 1
        FINAL ← 15
        ACHA ← .Falso.
        enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso.) faça
            MEIO ← (COMEÇO + FINAL) div 2
            se (PESQ = D[MEIO]) então
                ACHA ← .Verdadeiro.
            senão
                se (PESQ < D[MEIO]) então
                    FINAL ← MEIO - 1
                senão
                    COMEÇO ← MEIO + 1
                fim_se
            fim_se
        fim_enquanto
        se (ACHA = .Verdadeiro.) então
            escreva PESQ, " foi localizado na posição ", MEIO
        senão
            escreva PESQ, " não foi localizado"
        fim_se
        leia RESP
    fim_enquanto
fim
```

o) Diagrama de blocos



Português Estruturado

```

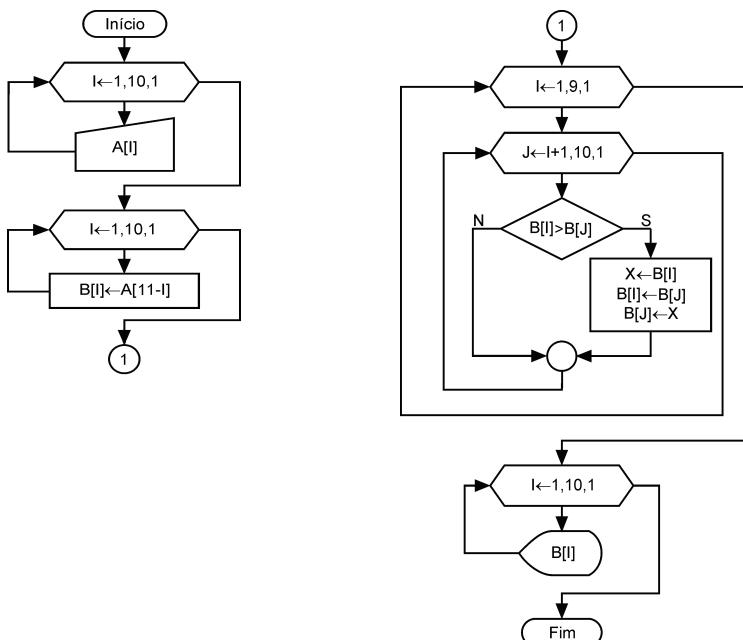
programa Cap07_Ex1o_Pg172
var
  A : conjunto[1..15] de inteiro
  I, J, X, COMEÇO, FINAL, MEIO, PESQ : inteiro
  RESP : cadeia
  ACHA : lógico
início
  para I de 1 até 15 passo 1 faça
    leia A[I]
  fim_para
  
```

```

para I de 1 até 14 passo 1 faça
  para J de I + 1 até 15 passo 1 faça
    se (A[I] > A[J]) então
      X ← A[I]
      A[I] ← A[J]
      A[J] ← X
    fim_se
  fim_para
fim_para
RESP ← "SIM"
enquanto (RESP = "SIM") faça
  leia PESQ
  COMEÇO ← 1
  FINAL ← 15
  ACHA ← .Falso.
  enquanto (COMEÇO <= FINAL) e. (ACHA = .Falso) faça
    MEIO ← (COMEÇO + FINAL) div 2
    se (PESQ = A[MEIO]) então
      ACHA ← .Verdadeiro.
    senão
      se (PESQ < A[MEIO]) então
        FINAL ← MEIO - 1
      senão
        COMEÇO ← MEIO + 1
      fim_se
    fim_se
  fim_enquanto
  se (ACHA = .Verdadeiro.) então
    escreva PESQ, " foi localizado na posição ", MEIO
  senão
    escreva PESQ, " não foi localizado"
  fim_se
  leia RESP
fim_enquanto
fim

```

p) Diagrama de blocos



Português Estruturado

```

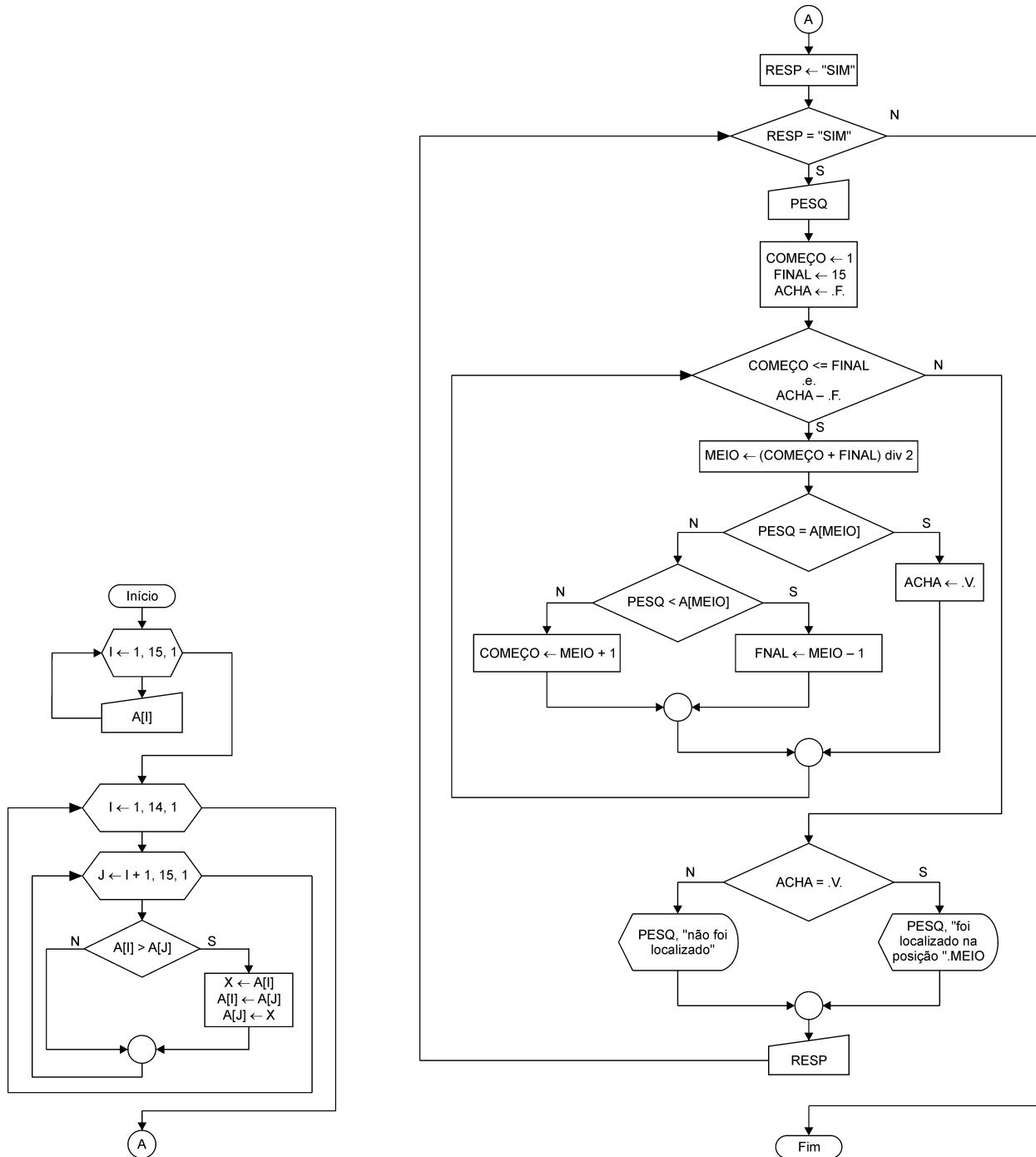
programa Cap07_Ex1p_Pg172
var
  A, B : conjunto[1..10] de cadeia
  I, J : inteiro
  X : cadeia
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para

```

```

para I de 1 até 10 passo 1 faça
  B[I] ← A[11 - I]
fim_para
para I de 1 até 14 passo 1 faça
  para J de I + 1 até 15 passo 1 faça
    se (B[I] > B[J]) então
      X ← B[I]
      B[I] ← B[J]
      B[J] ← X
    fim_se
  fim_para
para I de 1 até 10 passo 1 faça
  escreva B[I]
fim_para
fim
  
```

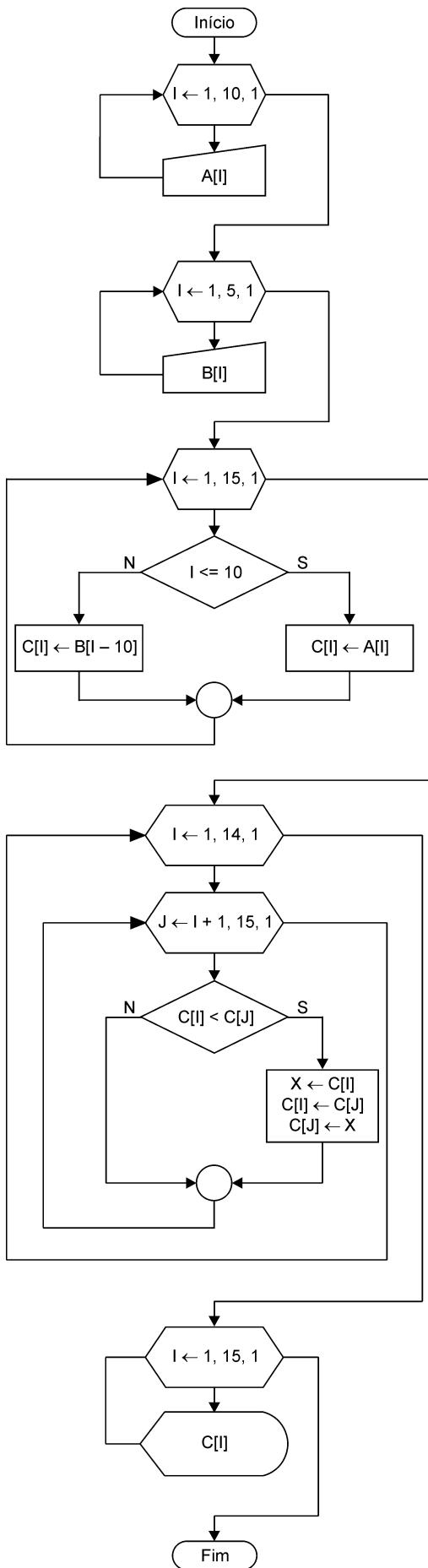
q) Diagramas de blocos



Português estruturado

```
programa Cap07_Ex1q_Pg172
var
    A : conjunto[1..15] de cadeia
    I, J, COMEÇO, FINAL, MEIO, PESQ : inteiro
    RESP : cadeia
    X : cadeia
    ACHA : lógico
início
    para I de 1 até 15 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 14 passo 1 faça
        para J de I + 1 até 15 passo 1 faça
            se (A[I] > A[J]) então
                X ← A[I]
                A[I] ← A[J]
                A[J] ← X
            fim_se
        fim_para
    fim_para
    RESP ← "SIM"
    enquanto (RESP = "SIM") faça
        leia PESQ
        COMEÇO ← 1
        FINAL ← 15
        ACHA ← .Falso.
        enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso) faça
            MEIO ← (COMEÇO + FINAL) div 2
            se (PESQ = A[MEIO]) então
                ACHA ← .Verdadeiro.
            senão
                se (PESQ < A[MEIO]) então
                    FINAL ← MEIO - 1
                senão
                    COMEÇO ← MEIO + 1
                fim_se
            fim_se
        fim_enquanto
        se (ACHA = .Verdadeiro.) então
            escreva PESQ, " foi localizado na posição ", MEIO
        senão
            escreva PESQ, " não foi localizado"
        fim_se
        leia RESP
    fim_enquanto
fim
```

r) Diagrama de blocos

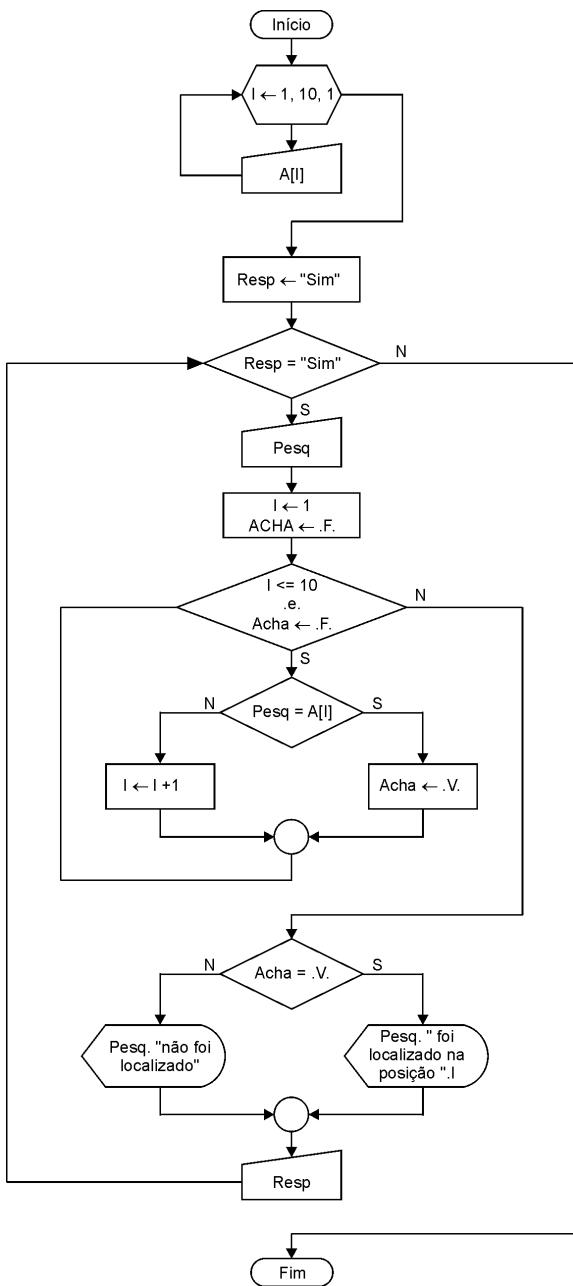


Português estruturado

```

programa Cap07_Ex1r_Pg172
var
  A : conjunto[1..10] de cadeia
  B : conjunto[1..5] de cadeia
  C : conjunto[1..15] de cadeia
  I, J : inteiro
  X : cadeia
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 5 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 15 passo 1 faça
    se (I <= 10) então
      C[I] ← A[I]
    senão
      C[I] ← B[I - 10]
    fim_se
  fim_para
  para I de 1 até 14 passo 1 faça
    para J de I + 1 até 15 passo 1 faça
      se (C[I] < C[J]) então
        X ← C[I]
        C[I] ← C[J]
        C[J] ← X
      fim_se
    fim_para
  fim_para
  para I de 1 até 15 passo 1 faça
    escreva C[I]
  fim_para
fim
  
```

s) Diagrama de blocos



Português estruturado

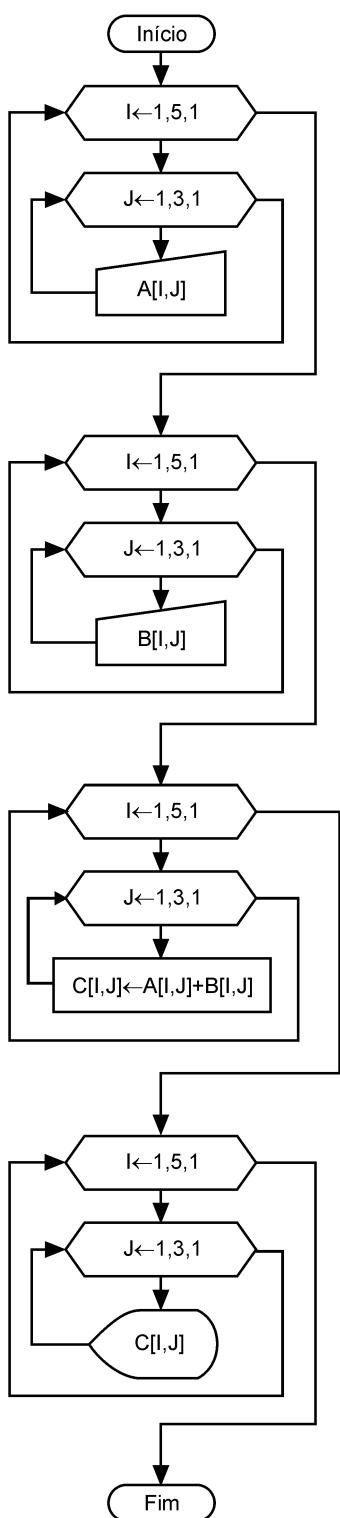
```

programa Cap07_Ex1s_Pg172
var
  A : conjunto[1..10] de real
  I, PESQ : inteiro
  RESP : cadeia
  ACHA : lógico
início
  para I de 1 até 10 passo 1 faça
    leia A[I]
  fim_para
  RESP ← "SIM"
  enquanto (RESP = "SIM") faça
    leia PESQ
    I ← 1
    ACHA ← .Falso.
    enquanto (I <= 10) .e. (ACHA = .Falso.) faça
      se (PESQ = B[I]) então
        ACHA ← .Verdadeiro.
      senão
        I ← I + 1
      fim_se
    fim_enquanto
    se (ACHA = .Verdadeiro.) então
      escreva PESQ, " foi localizado na posição ", I
    senão
      escreva PESQ, " não foi localizado"
    fim_se
    leia RESP
  fim_enquanto
fim
  
```

8 - Exercícios de fixação do capítulo 8

Tópico 8.5 - Exercício 1 a - Página 185

a) Diagrama de blocos



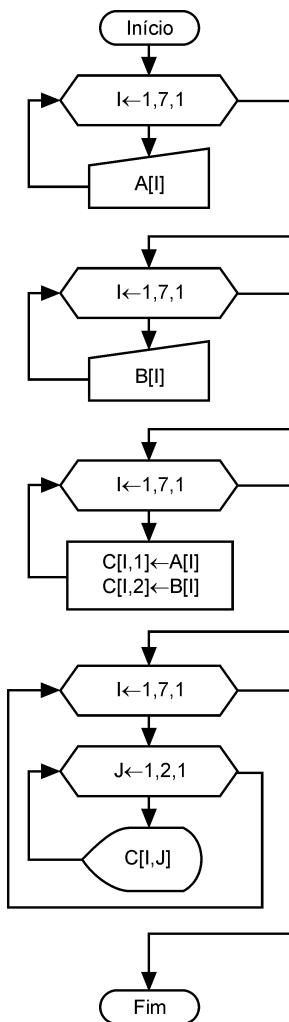
Português estruturado

```

programa Cap08_Ex1a_Pg185
var
    A, B, C : conjunto[1..5,1..3] de inteiro
    I, J : inteiro
início
    para I de 1 até 5 passo 1 faça
        para J de 1 até 3 passo 1 faça
            leia A[I, J]
        fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
        para J de 1 até 3 passo 1 faça
            leia B[I, J]
        fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
        para J de 1 até 3 passo 1 faça
            C[I, J] ← A[I, J] = B[I, J]
        fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
        para J de 1 até 3 passo 1 faça
            escreva C[I, J]
        fim_para
    fim_para
fim
    
```

b) Diagrama de blocos

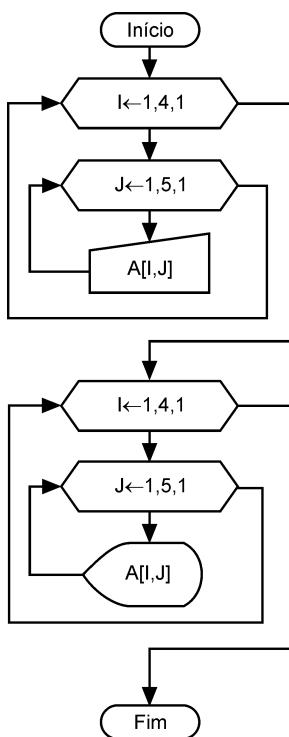
Português estruturado



```

programa Cap08_Ex1b_Pg185
var
  A, B : conjunto[1..7] de inteiro
  C : conjunto[1..7,1..2] de inteiro
  I, J : inteiro
início
  para I de 1 até 7 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 7 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 7 passo 1 faça
    C[I,1] ← A[I]
    C[I,2] ← B[I]
  fim_para
  para I de 1 até 7 passo 1 faça
    para J de 1 até 2 passo 1 faça
      escreva C[I,J]
    fim_para
  fim_para
fim
  
```

c) Diagrama de blocos

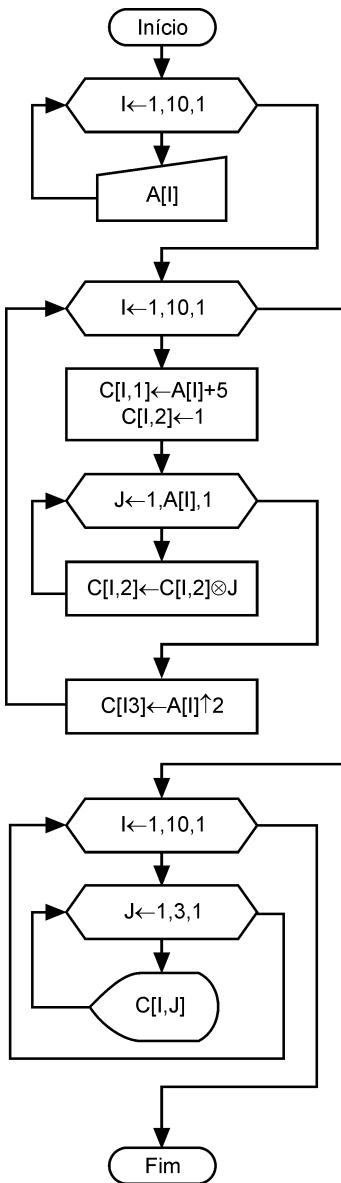


Português estruturado

```

programa Cap08_Ex1c_Pg185
var
  A : conjunto[1..4,1..5] de inteiro
  I, J : inteiro
início
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      escreva A[I,J]
    fim_para
  fim_para
fim
  
```

d) Diagrama de blocos

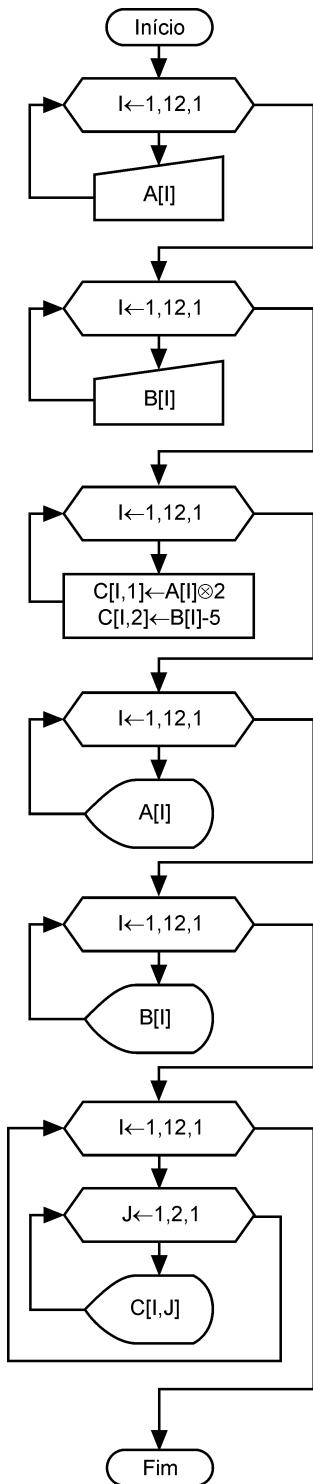


Português estruturado

```

programa Cap08_Ex1d_Pg185
var
    A : conjunto[1..10] de inteiro
    C : conjunto[1..10,1..3] de inteiro
    I, J : inteiro
início
    para I de 1 até 10 passo 1 faça
        leia A[I]
    fim_para
    para I de 1 até 10 passo 1 faça
        C[I,1] ← A[I] + 5
        C[I,2] ← 1
        para J de 1 até A[I] passo 1 faça
            C[I,2] ← C[I,2] * J
        fim_para
        C[I,3] ← A[I] ↑ 2
    fim_para
    para I de 1 até 10 passo 1 faça
        para J de 1 até 3 passo 1 faça
            escreva C[I,J]
        fim_para
    fim_para
fim
    
```

e) Diagrama de blocos

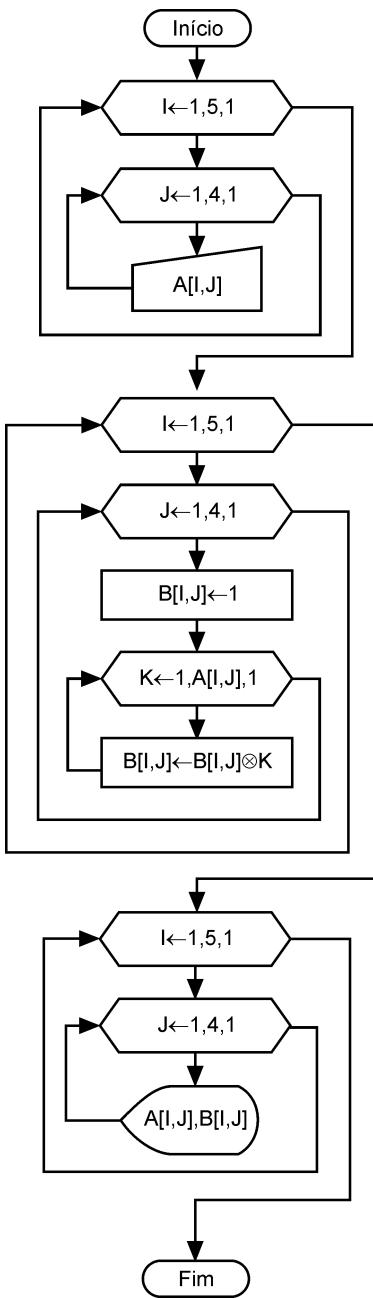


Português estruturado

```

programa Cap08_Exle_Pg186
var
  A, B : conjunto[1..12] de real
  C : conjunto[1..12,1..2] de real
  I, J : inteiro
início
  para I de 1 até 12 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 12 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 12 passo 1 faça
    C[I,1] ← A[I] * 2
    C[I,2] ← B[I] - 5
  fim_para
  para I de 1 até 12 passo 1 faça
    escreva A[I]
  fim_para
  para I de 1 até 12 passo 1 faça
    escreva B[I]
  fim_para
  para I de 1 até 12 passo 1 faça
    para J de 1 até 2 passo 1 faça
      escreva C[I,J]
    fim_para
  fim_para
fim
  
```

f) Diagrama de blocos



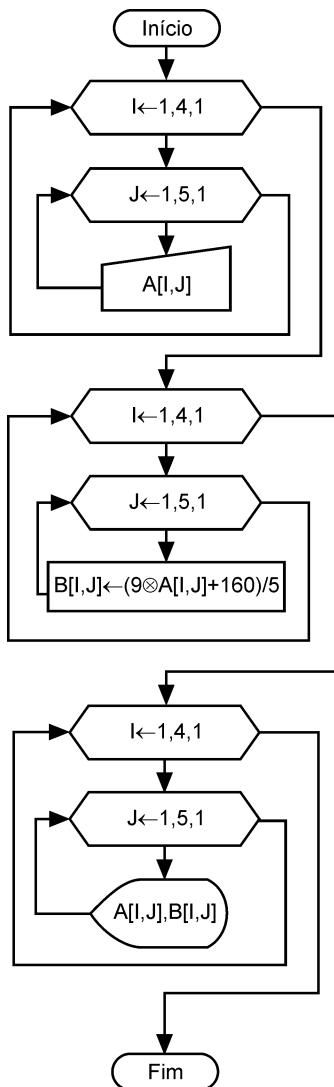
Português estruturado

```

programa Cap08_Ex1f_Pg186
var
    A, B : conjunto[1..5,1..4] de inteiro
    I, J, K : inteiro
início
    para I de 1 até 5 passo 1 faça
        para J de 1 até 4 passo 1 faça
            leia A[I,J]
        fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
        para J de 1 até 4 passo 1 faça
            B[I,J] ← 1
            para K de 1 até A[I,J] passo 1 faça
                B[I,J] ← B[I,J] * K
            fim_para
        fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
        para J de 1 até 4 passo 1 faça
            escreva A[I,J], B[I,J]
        fim_para
    fim_para
fim

```

g) Diagrama de blocos

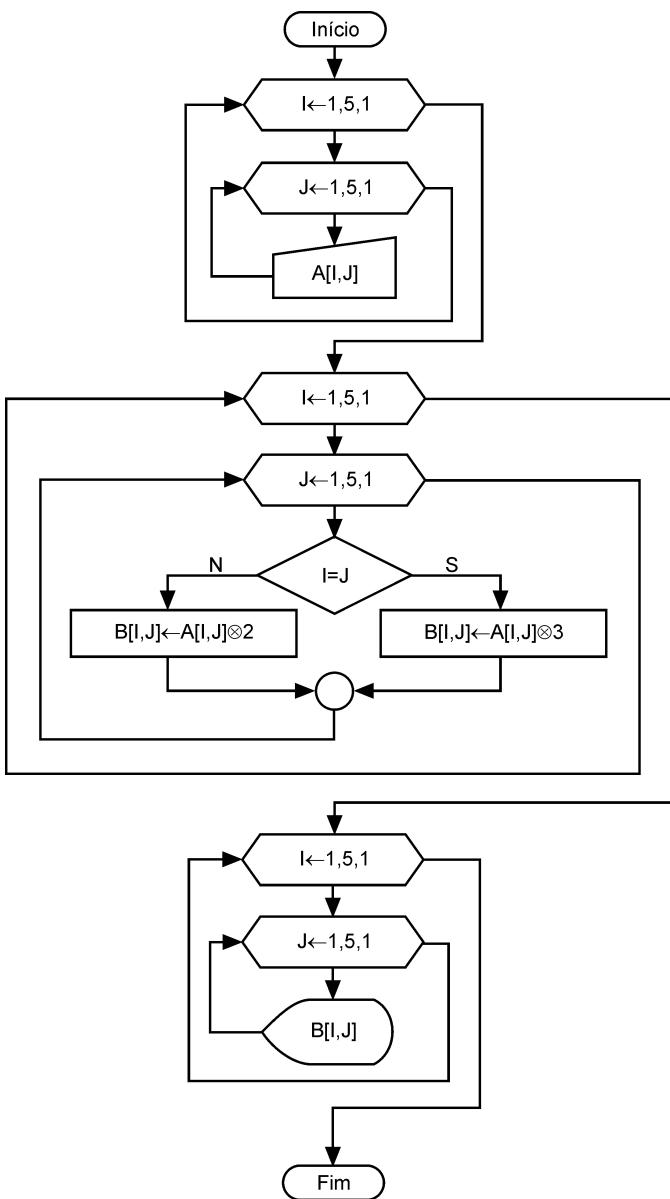


Português estruturado

```

programa Cap08_Ex1g_Pg186
var
  A, B : conjunto[1..4,1..5] de real
  I, J : inteiro
início
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      B[I,J] ← (9 * A[I,J] + 160) / 5
    fim_para
  fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      escreva A[I,J], B[I,J]
    fim_para
  fim_para
fim
  
```

h) Diagrama de blocos

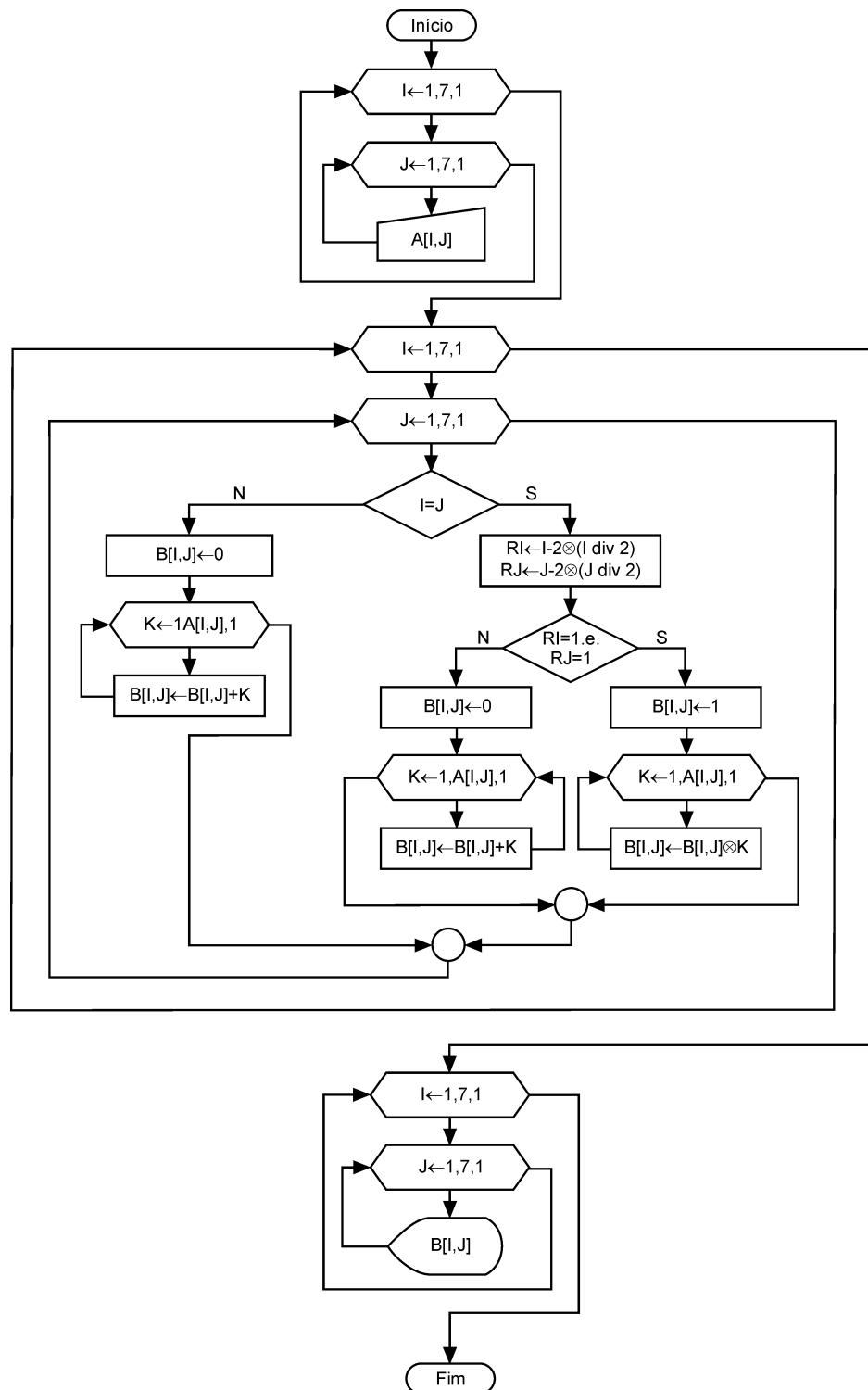


Português estruturado

```

programa Cap08_Ex1h_Pg186
var
  A, B : conjunto[1..5,1..5] de inteiro
  I, J : inteiro
início
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      se (I = J) então
        B[I,J] ← A[I,J] * 3
      senão
        B[I,J] ← A[I,J] * 2
      fim_se
    fim_para
  fim_para
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      escreva B[I,J]
    fim_para
  fim_para
fim
  
```

i) Diagrama de blocos



Português Estruturado

```

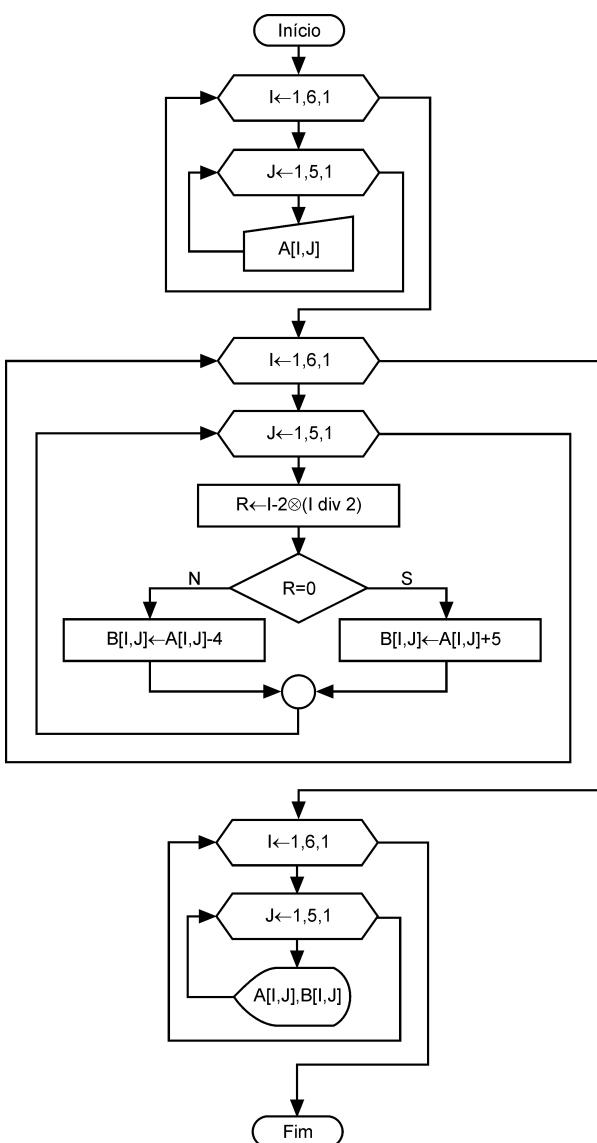
programa Cap08_Ex1i_Pg186
var
  A, B : conjunto[1..7,1..7] de inteiro
  I, J, K, RI, RJ : inteiro
início
  para I de 1 até 7 passo 1 faça
    para J de 1 até 7 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  para I de 1 até 7 passo 1 faça
    para J de 1 até 7 passo 1 faça
  
```

```

se (I = J) então
    RI ← I - 2 * (I div 2)
    RJ ← J - 2 * (J div 2)
    se (RI = 1) .e. (RJ = 1) então
        B[I,J] ← 1
        para K de 1 até A[I,J] passo 1 faça
            B[I,J] ← B[I,J] * K
        fim_para
    senão
        B[I,J] ← 0
        para K de 1 até A[I,J] passo 1 faça
            B[I,J] ← B[I,J] + K
        fim_para
    fim_se
senão
    B[I,J] ← 0
    para K de 1 até A[I,J] passo 1 faça
        B[I,J] ← B[I,J] + K
    fim_para
fim_se
fim_para
para I de 1 até 7 passo 1 faça
    para J de 1 até 7 passo 1 faça
        escreva B[I,J]
    fim_para
fim_para
fim

```

j) Diagrama de blocos



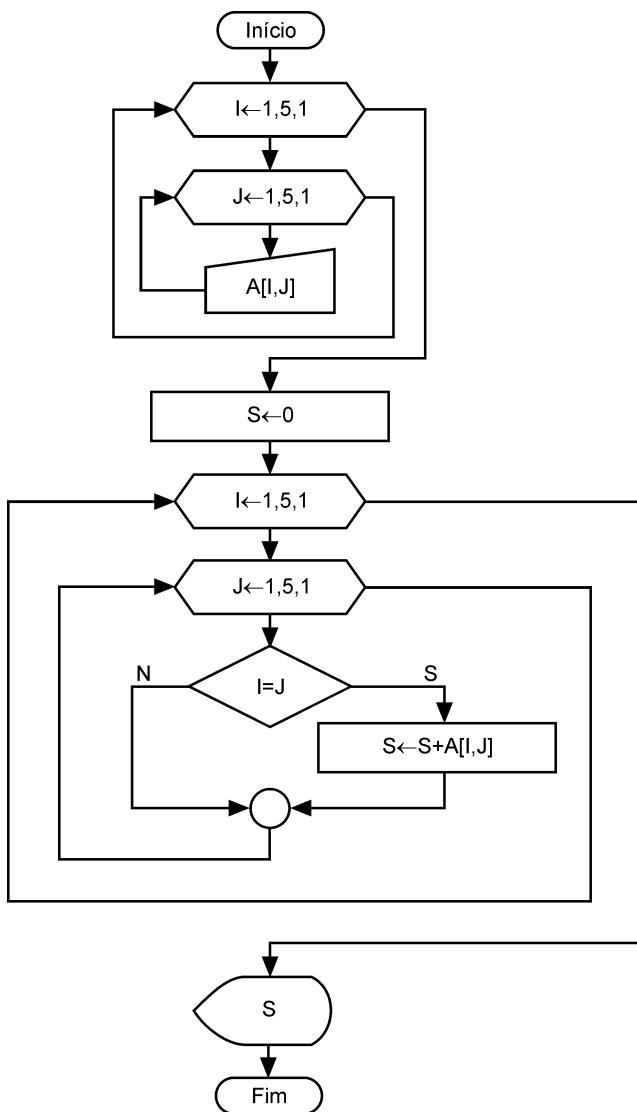
Português estruturado

```

programa Cap08_Ex1j_Pg186
var
    A, B : conjunto[1..6,1..5] de inteiro
    I, J, R : inteiro
início
    para I de 1 até 6 passo 1 faça
        para J de 1 até 5 passo 1 faça
            leia A[I,J]
        fim_para
    fim_para
    para I de 1 até 6 passo 1 faça
        para J de 1 até 5 passo 1 faça
            R ← A[I,J] - 2 * (A[I,J] div 2)
            se (R = 0) então
                B[I,J] ← A[I,J] + 5
            senão
                B[I,J] ← A[I,J] - 4
            fim_se
        fim_para
    fim_para
    para I de 1 até 6 passo 1 faça
        para J de 1 até 5 passo 1 faça
            escreva A[I,J], B[I,J]
        fim_para
    fim_para
fim

```

k) Diagrama de blocos

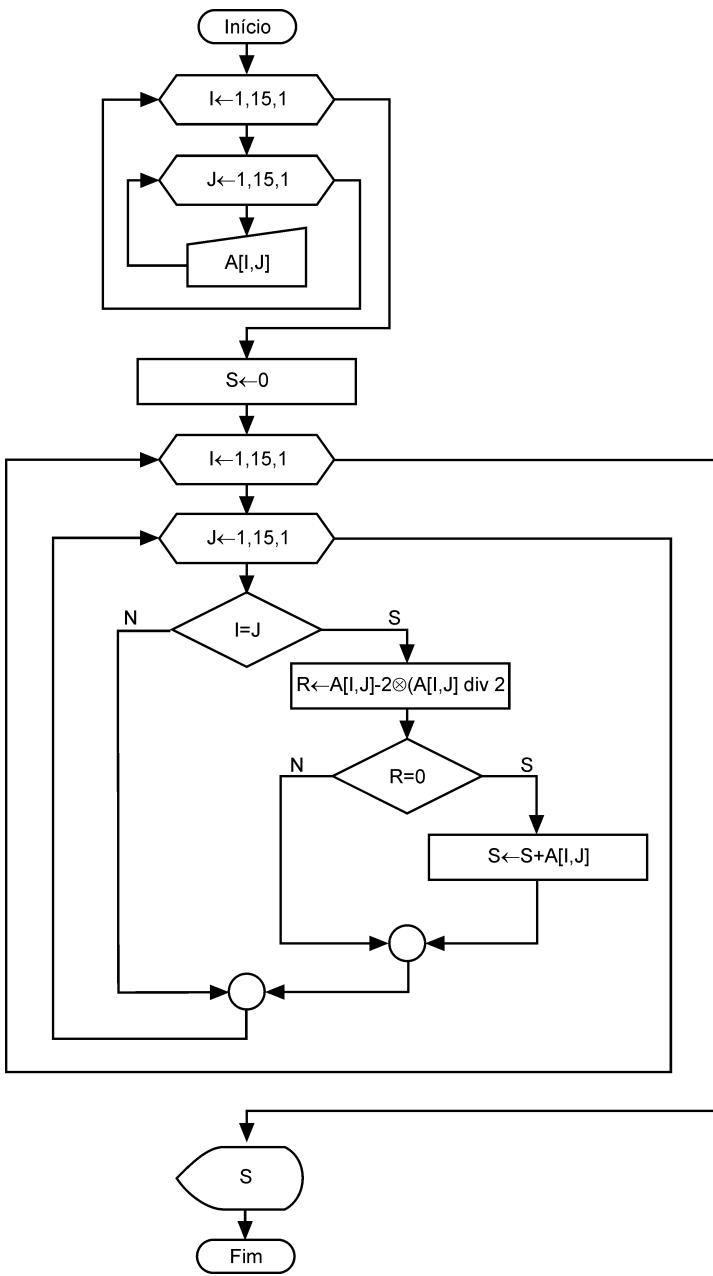


Português estruturado

```

programa Cap08_Ex1k_Pg186
var
  A, B : conjunto[1..5,1..5] de real
  I, J, S : inteiro
início
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  S ← 0
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      se (I = J) então
        S ← S + A[I,J]
      fim_se
    fim_para
  fim_para
  escreva S
fim
  
```

I) Diagrama de blocos

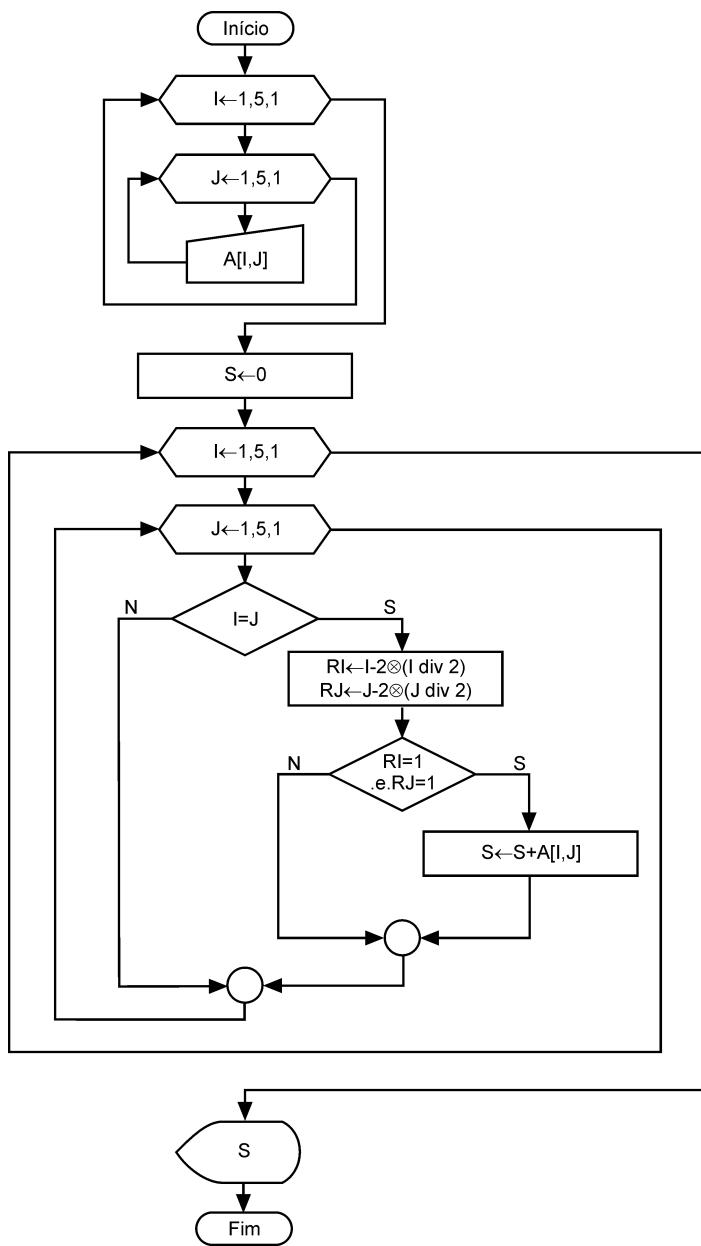


Português estruturado

```

programa Cap08_Ex11_Pg186
var
  A, B : conjunto[1..15,1..15] de
  inteiro
  I, J, S, R : inteiro
início
  para I de 1 até 15 passo 1 faça
    para J de 1 até 15 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  S ← 0
  para I de 1 até 15 passo 1 faça
    para J de 1 até 15 passo 1 faça
      se (I = J) então
        R ← A[I,J] - 2 * (A[I,J] div 2)
        se (R = 0) então
          S ← S + A[I,J]
        fim_se
      fim_se
    fim_para
  fim_para
  escreva S
fim
  
```

m) Diagrama de blocos

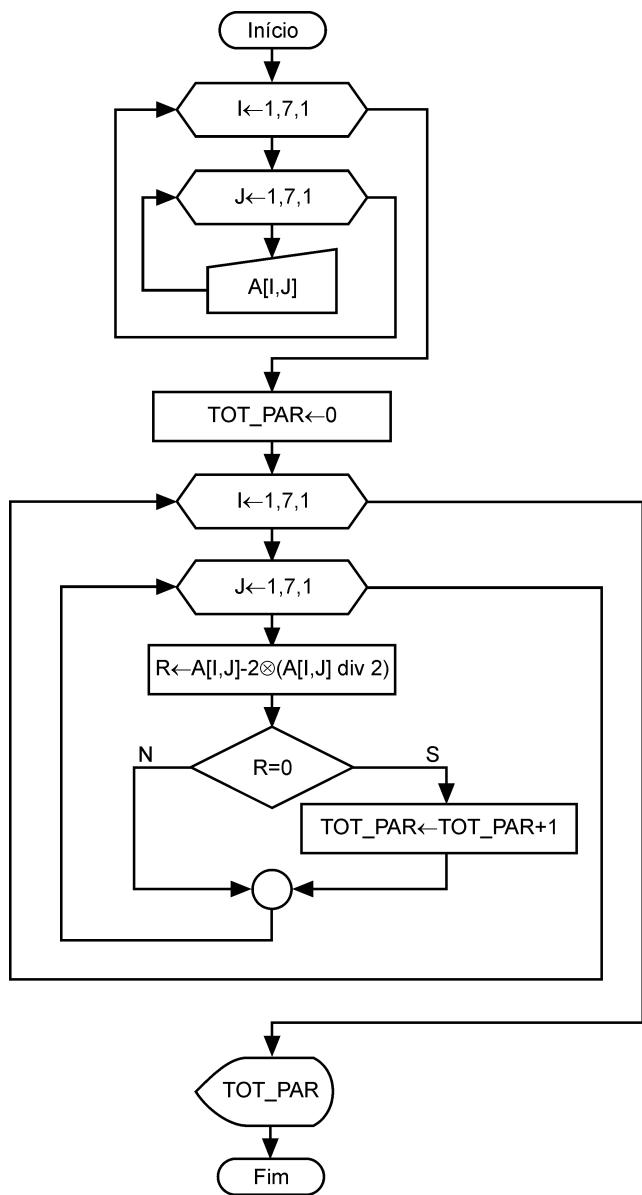


Português estruturado

```

programa Cap08_Ex1m_Pg186
var
  A, B : conjunto[1..5,1..5] de real
  I, J, S, RI, RJ : inteiro
início
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  S ← 0
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      se (I = J) então
        RI ← I - 2 * (I div 2)
        RJ ← J - 2 * (J div 2)
        se (RI = 1 .e. (RJ = 1) então
          S ← S + A[I,J]
        fim_se
      fim_se
    fim_para
  fim_para
  escreva S
fim
  
```

n) Diagrama de blocos

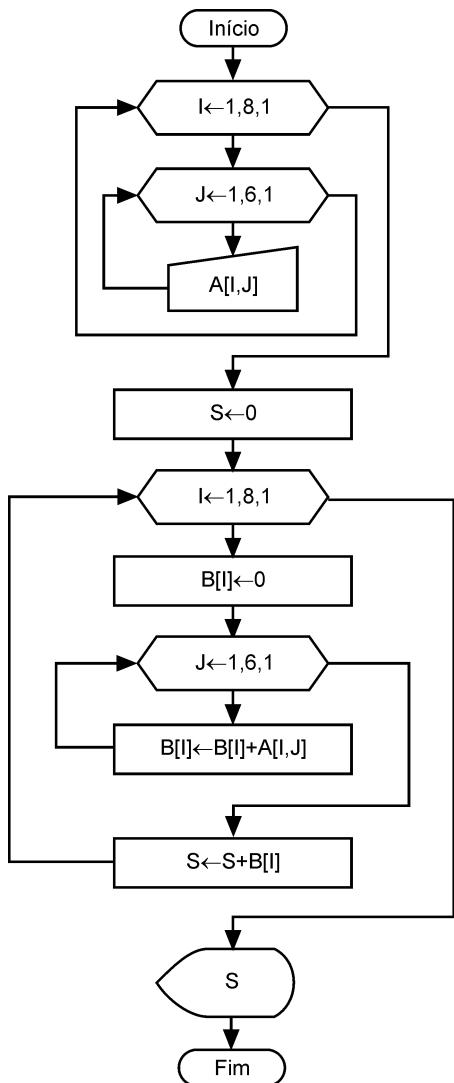


Português estruturado

```

programa Cap08_Ex1n_Pg186
var
  A : conjunto[1..7,1..7] de inteiro
  I, J, TOT_PAR, R : inteiro
início
  para I de 1 até 7 passo 1 faça
    para J de 1 até 7 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  TOT_PAR ← 0
  para I de 1 até 7 passo 1 faça
    para J de 1 até 7 passo 1 faça
      R ← A[I,J] - 2 * (A[I,J] div 2)
      se (R = 0) então
        TOT_PAR ← TOT_PAR + 1
      fim_se
    fim_para
  fim_para
  escreva TOT_PAR
fim
  
```

o) Diagrama de blocos

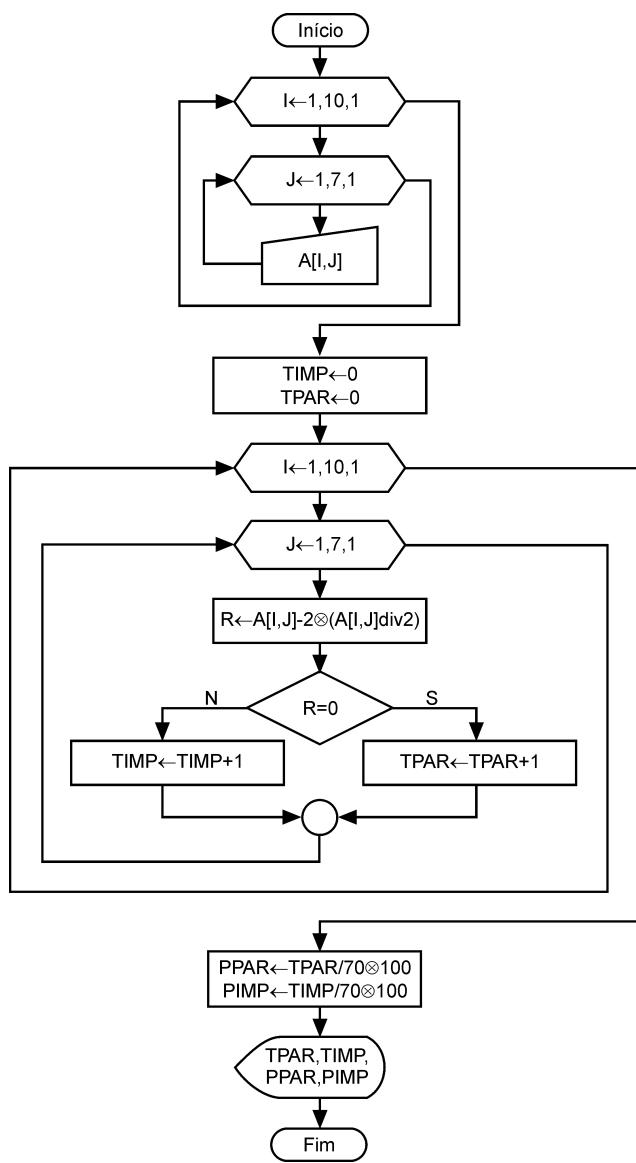


Português estruturado

```

programa Cap08_Ex1o_Pg186
var
  A : conjunto[1..8,1..6] de real
  B : conjunto[1..8] de real
  I, J : inteiro
  S : real
início
  para I de 1 até 8 passo 1 faça
    para J de 1 até 6 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  S ← 0
  para I de 1 até 8 passo 1 faça
    B[I] ← 0
    para J de 1 até 6 passo 1 faça
      B[I] ← B[I] + A[I,J]
    fim_para
    S ← S + B[I]
  fim_para
  escreva S
fim
  
```

p) Diagrama de blocos

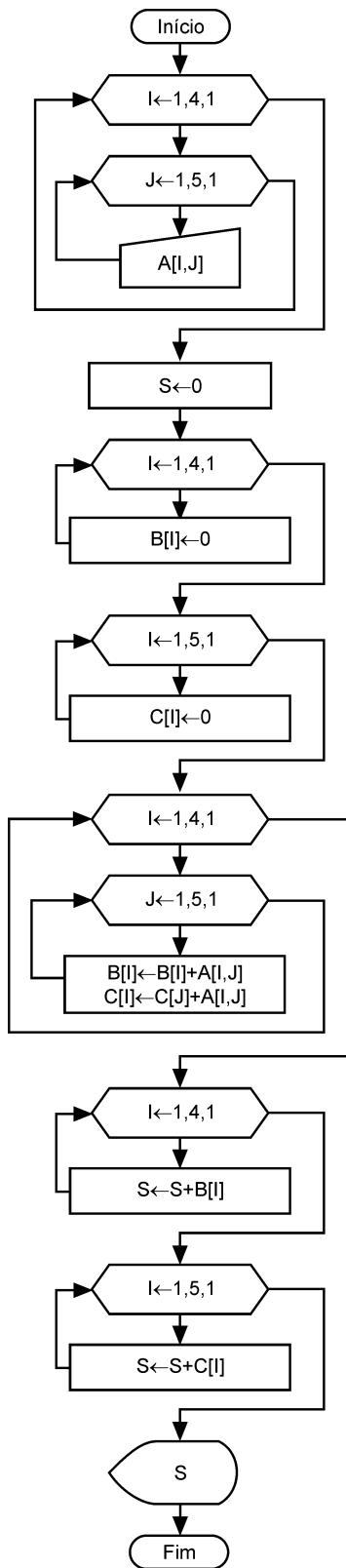


Português estruturado

```

programa Cap08_Ex1p_Pg186
var
  A : conjunto[1..10,1..7] de inteiro
  I, J, TIMP, TPAR, R : inteiro
  PIMP, PPAR : real
início
  para I de 1 até 10 passo 1 faça
    para J de 1 até 7 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  TIMP ← 0
  TPAR ← 0
  para I de 1 até 10 passo 1 faça
    para J de 1 até 7 passo 1 faça
      R ← A[I,J] - 2 * (A[I,J] div 2)
      se (R = 0) então
        TPAR ← TPAR + 1
      senão
        TIMP ← TIMP + 1
      fim_se
    fim_para
  fim_para
  PPAR ← TPAR / 70 * 100
  PIMP ← TIMP / 70 * 100
  escreva TPAR, TIMP, PPAR, PIMP
fim
  
```

q) Diagrama de blocos

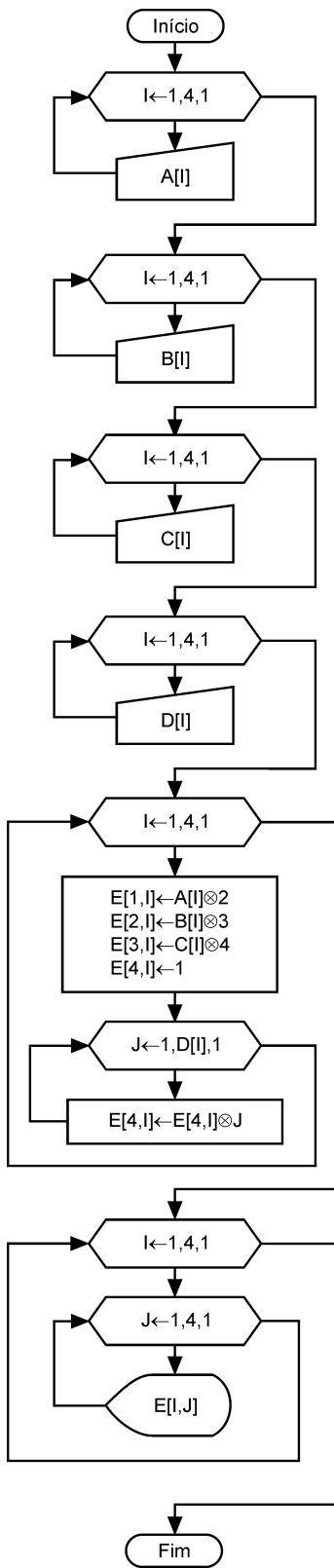


Português estruturado

```

programa Cap08_Ex1q_Pg187
var
  A : conjunto[1..4,1..5] de inteiro
  B : conjunto[1..4] de inteiro
  C : conjunto[1..5] de inteiro
  I, J, S : inteiro
início
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      leia A[I,J]
    fim_para
  fim_para
  S ← 0
  para I de 1 até 4 passo 1 faça
    B[I] ← 0
  fim_para
  para I de 1 até 5 passo 1 faça
    C[I] ← 0
  fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      B[I] ← B[I] + A[I,J]
      C[J] ← C[J] + A[I,J]
    fim_para
  fim_para
  para I de 1 até 4 passo 1 faça
    S ← S + B[I]
  fim_para
  para I de 1 até 5 passo 1 faça
    S ← S + C[I]
  fim_para
  escreva S
fim
  
```

r) Diagrama de blocos

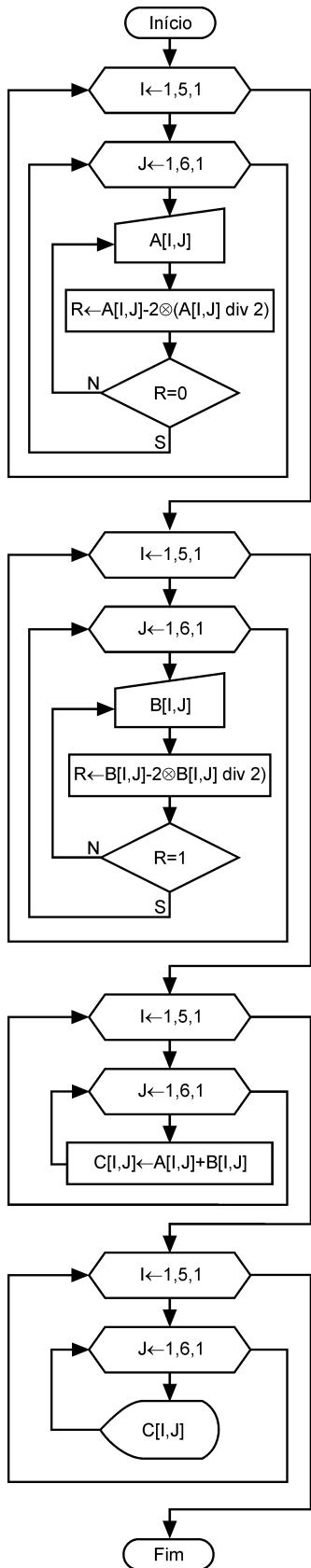


Português estruturado

```

programa Cap08_Ex1r_Pg187
var
  A, B, C, D : conjunto[1..4] de inteiro
  E : conjunto[1..4, 1..4] de inteiro
  I, J : inteiro
início
  para I de 1 até 4 passo 1 faça
    leia A[I]
  fim_para
  para I de 1 até 4 passo 1 faça
    leia B[I]
  fim_para
  para I de 1 até 4 passo 1 faça
    leia C[I]
  fim_para
  para I de 1 até 4 passo 1 faça
    leia D[I]
  fim_para
  para I de 1 até 4 passo 1 faça
    E[1,I] ← A[I] * 2
    E[2,I] ← B[I] * 3
    E[3,I] ← C[I] * 4
    E[4,I] ← 1
    para J de 1 até D[I] passo 1 faça
      E[4,I] ← E[4,I] * J
    fim_para
  fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 4 passo 1 faça
      escreva E[I,J]
    fim_para
  fim_para
fim
  
```

s) Diagrama de blocos

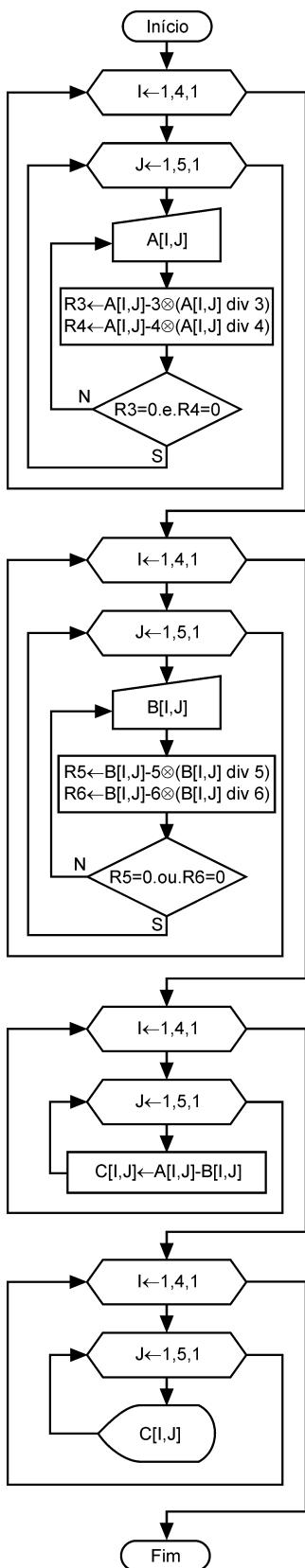


Português estruturado

```

programa Cap08_Exls_Pg187
var
  A, B, C : conjunto[1..5,1..6] de inteiro
  I, J, R : inteiro
início
  para I de 1 até 5 passo 1 faça
    para J de 1 até 6 passo 1 faça
      repita
        leia A[I,J]
        R ← A[I,J] - 2 * (A[I,J] div 2)
        até_que (R = 0)
      fim_para
    fim_para
  para I de 1 até 5 passo 1 faça
    para J de 1 até 6 passo 1 faça
      repita
        leia B[I,J]
        R ← B[I,J] - 2 * (B[I,J] div 2)
        até_que (R = 1)
      fim_para
    fim_para
  para I de 1 até 5 passo 1 faça
    para J de 1 até 6 passo 1 faça
      C[I,J] ← A[I,J] + B[I,J]
    fim_para
  fim_para
  para I de 1 até 5 passo 1 faça
    para J de 1 até 6 passo 1 faça
      escreva C[I,J]
    fim_para
  fim_para
fim
  
```

a) Diagrama de blocos

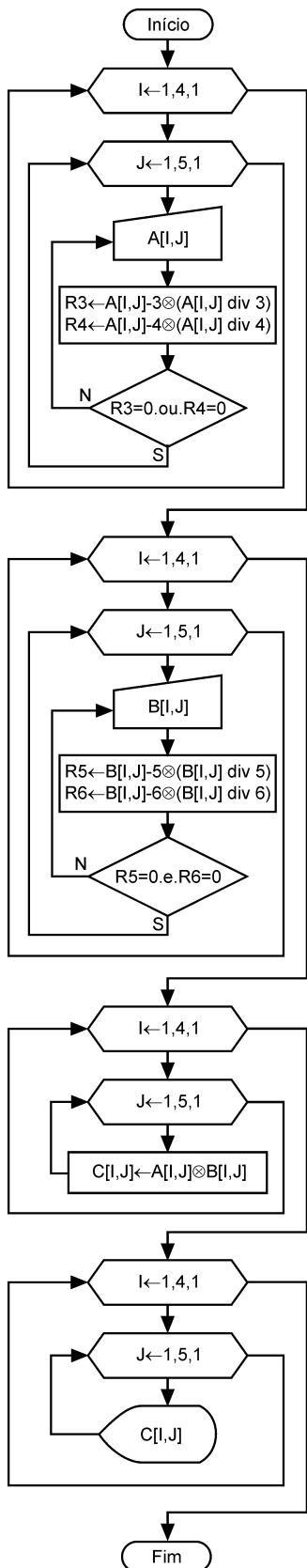


Português estruturado

```

programa Cap08_Ex1t_Pg187
var
  A, B, C : conjunto[1..4,1..5] de inteiro
  I, J, R3, R4, R5, R6 : inteiro
início
  para I de 1 até 4 passo 1 faça
    para J de 1 até 4 passo 1 faça
      repita
        leia A[I,J]
        R3 ← A[I,J] - 3 * (A[I,J] div 3)
        R4 ← A[I,J] - 4 * (A[I,J] div 4)
        até_que (R3 = 0) .e. (R4 = 0)
      fim_para
    fim_para
    para I de 1 até 4 passo 1 faça
      para J de 1 até 5 passo 1 faça
        repita
          leia B[I,J]
          R5 ← B[I,J] - 5 * (B[I,J] div 5)
          R6 ← B[I,J] - 6 * (B[I,J] div 6)
          até_que (R5 = 0) .ou. (R6 = 0)
        fim_para
      fim_para
      para I de 1 até 4 passo 1 faça
        para J de 1 até 5 passo 1 faça
          C[I,J] ← A[I,J] - B[I,J]
        fim_para
      fim_para
    fim
  
```

u) Diagrama de blocos

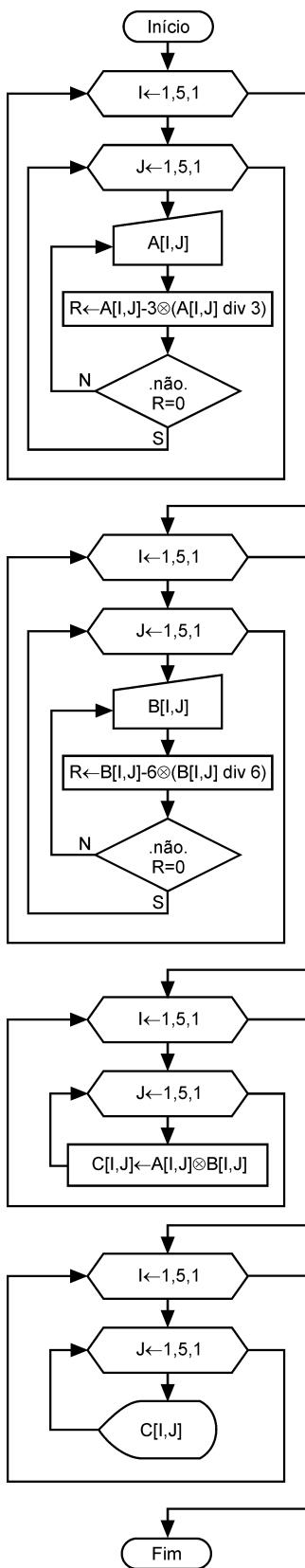


Português estruturado

```

programa Cap08_Exlu_Pg187
var
  A, B, C : conjunto[1..4,1..5] de inteiro
  I, J, R3, R4, R5, R6 : inteiro
início
  para I de 1 até 4 passo 1 faça
    para J de 1 até 4 passo 1 faça
      repita
        leia A[I,J]
        R3 ← A[I,J] - 3 * (A[I,J] div 3)
        R4 ← A[I,J] - 4 * (A[I,J] div 4)
        até_que (R3 = 0) .ou. (R4 = 0)
      fim_para
    fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      repita
        leia B[I,J]
        R5 ← B[I,J] - 5 * (B[I,J] div 5)
        R6 ← B[I,J] - 6 * (B[I,J] div 6)
        até_que (R5 = 0) .e. (R6 = 0)
      fim_para
    fim_para
  para I de 1 até 4 passo 1 faça
    para J de 1 até 5 passo 1 faça
      C[I,J] ← A[I,J] * B[I,J]
    fim_para
  fim_para
fim
  
```

v) Diagrama de blocos



Português estruturado

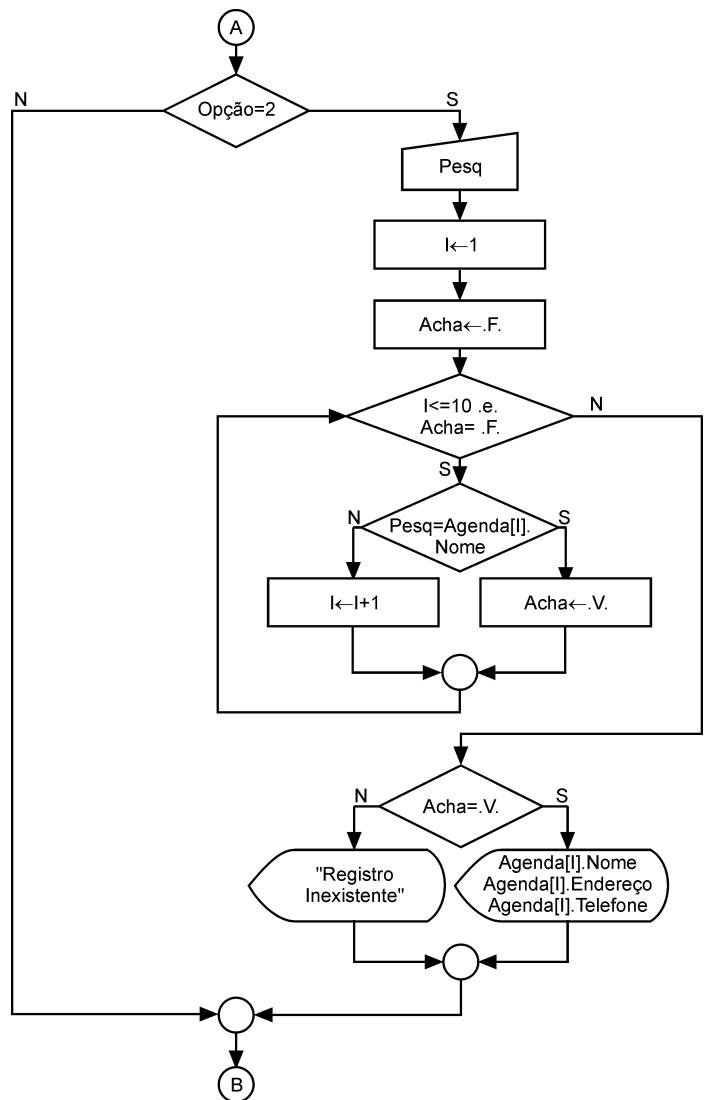
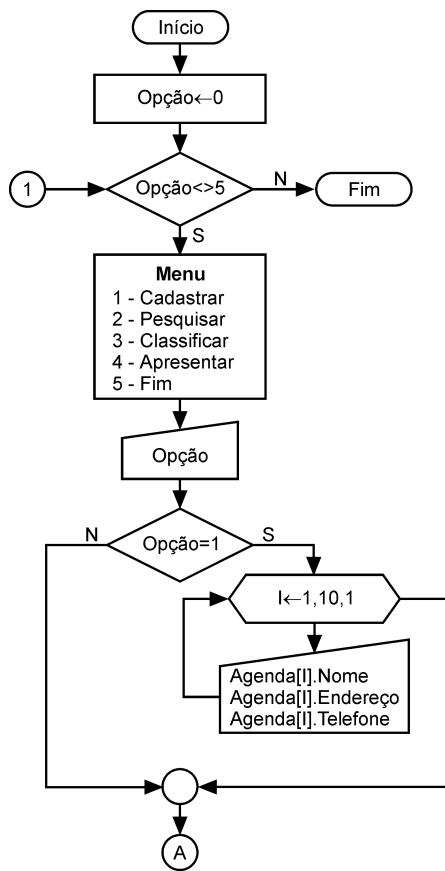
```

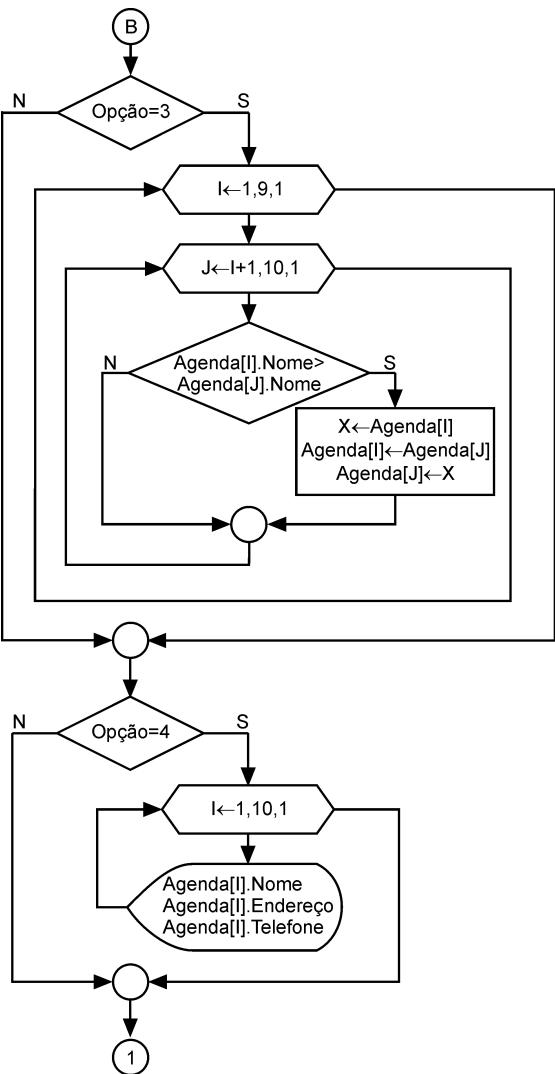
programa Cap08_Ex1v_Pg187
var
  A, B, C : conjunto[1..5,1..5] de inteiro
  I, J, R : inteiro
início
  para I de 1 até 5 passo 1 faça
    para J de 1 até 5 passo 1 faça
      repita
        leia A[I,J]
        R ← A[I,J] - 3 * (A[I,J] div 3)
        até que .não. (R = 0)
      fim_para
    fim_para
    para I de 1 até 5 passo 1 faça
      para J de 1 até 5 passo 1 faça
        repita
          leia B[I,J]
          R ← B[I,J] - 6 * (B[I,J] div 6)
          até que .não. (R = 0)
        fim_para
      fim_para
      para I de 1 até 5 passo 1 faça
        para J de 1 até 5 passo 1 faça
          C[I,J] ← A[I,J] + B[I,J]
        fim_para
      fim_para
      escreva C[I,J]
    fim_para
  fim
  
```

9 - Exercícios de fixação do capítulo 9

Tópico 9.6 - Exercício 1 - Página 207

Diagrama de blocos



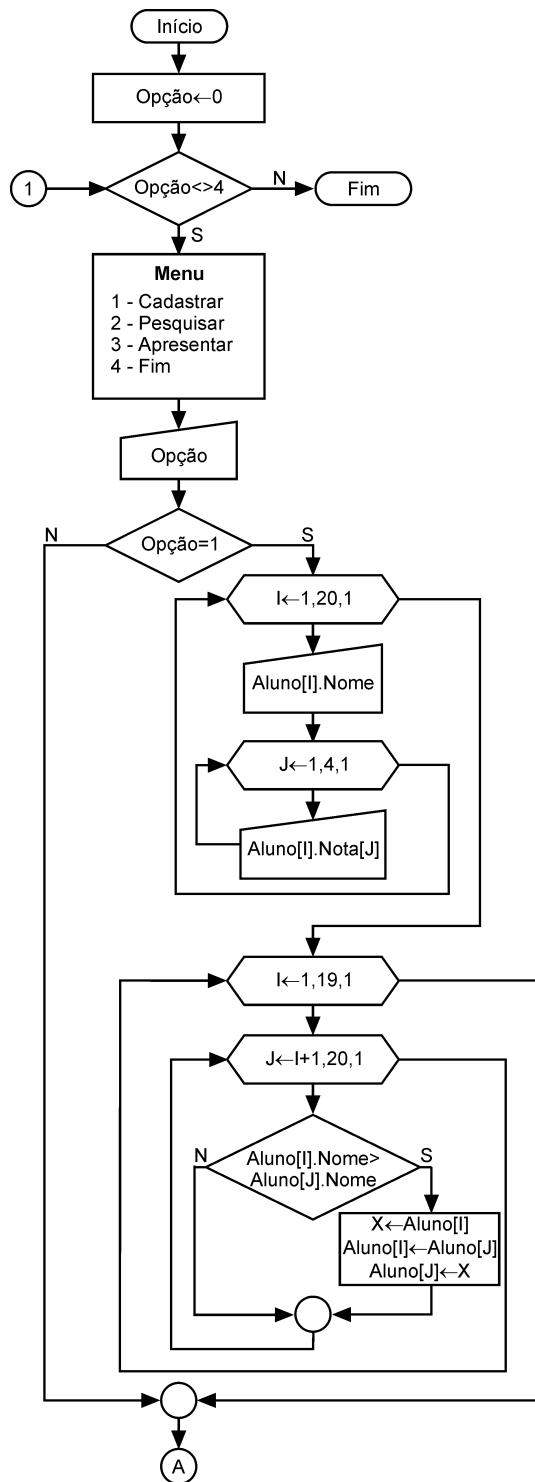


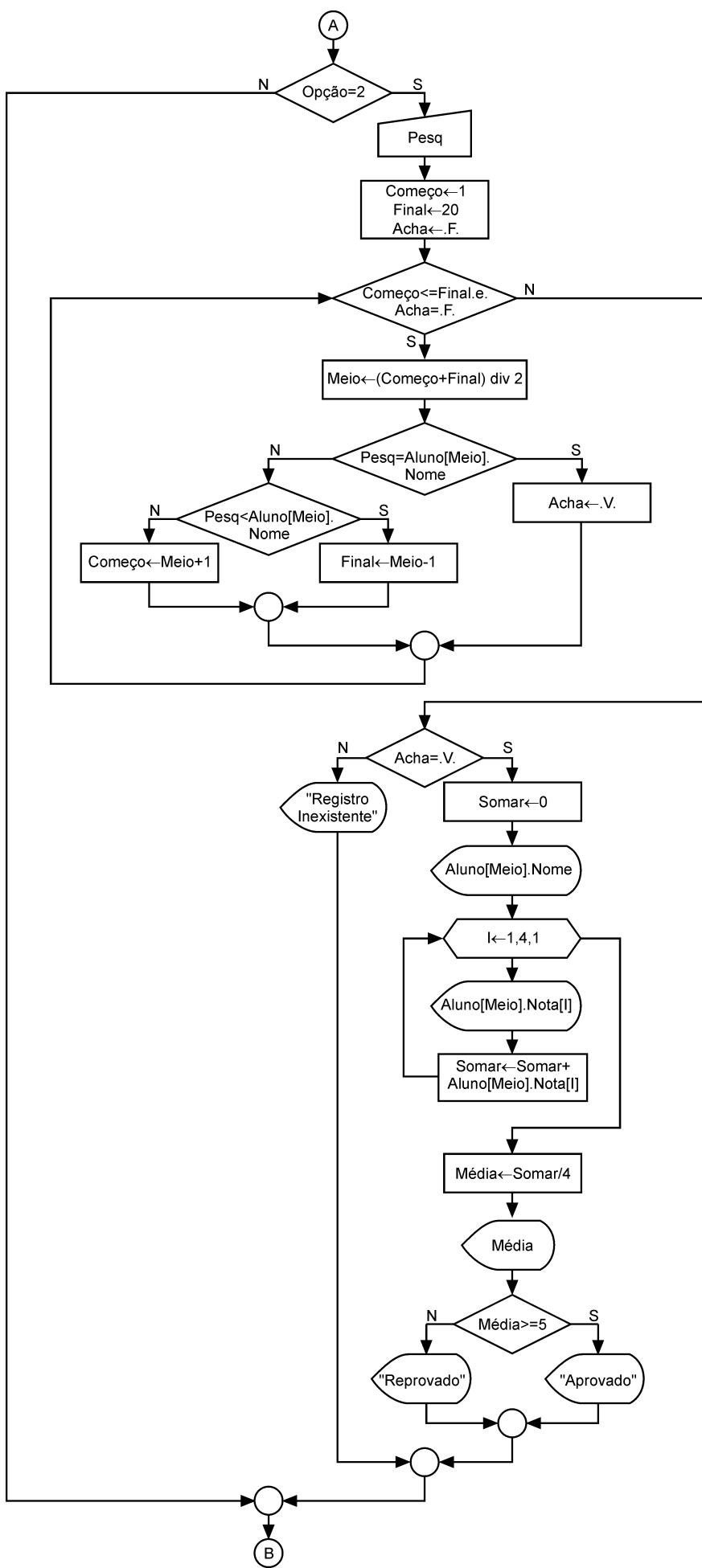
Português estruturado

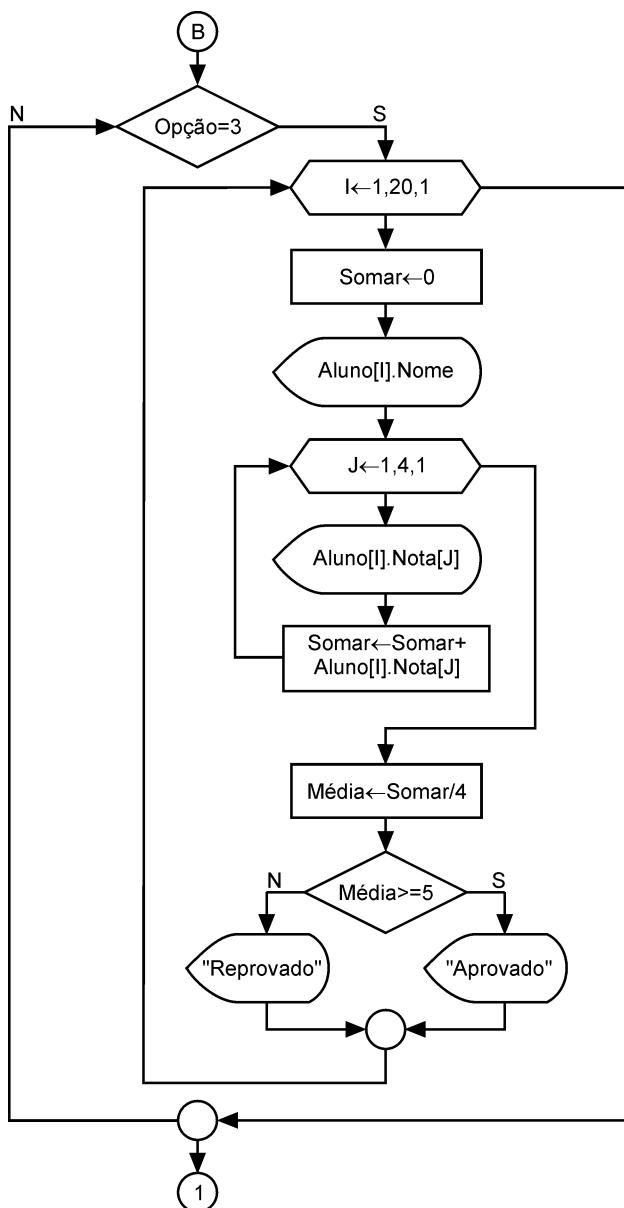
```

programa Cap09_Ex1_Pg207
  tipo
    Dados = registro
      NOME : cadeia
      ENDEREÇO : cadeia
      TELEFONE : cadeia
    fim_registro
  var
    AGENDA : conjunto[1..10] de Dados
    X : Dados
    I, J, OPÇÃO : inteiro
    ACHA : lógico
    PESQ : cadeia
  início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 5) faça
      escreva "1 - Cadastrar"
      escreva "2 - Pesquisar"
      escreva "3 - Classificar"
      escreva "4 - Apresentar"
      escreva "5 - Fim"
    leia OPÇÃO
    se (OPÇÃO = 1) então
      para I de 1 até 10 passo 1 faça
        leia AGENDA[I].NOME
        leia AGENDA[I].ENDEREÇO
        leia AGENDA[I].TELEFONE
      fim_para
    fim_se
    se (OPÇÃO = 2) então
      leia PESQ
      I ← 1
      ACHA ← .Falso.
      enquanto (I <= 10) .e. (ACHA = .Falso.) faça
        se (PESQ = AGENDA[I].NOME) então
          ACHA ← .Verdadeiro.
        senão
          I ← I + 1
        fim_se
      fim_enquanto
      se (ACHA = .Verdadeiro.) então
        escreva AGENDA[I].NOME
        escreva AGENDA[I].ENDEREÇO
        escreva AGENDA[I].TELEFONE
      senão
        escreva "Registro Inexistente"
      fim_se
    fim_se
    se (OPÇÃO = 3) então
      para I de 1 até 9 passo 1 faça
        para J de I+1 até 10 passo 1 faça
          se (AGENDA[I].NOME > AGENDA[J].NOME) então
            X ← AGENDA[I]
            AGENDA[I] ← AGENDA[J]
            AGENDA[J] ← X
          fim_se
        fim_para
      fim_para
    fim_se
    se (OPÇÃO = 4) então
      para I de 1 até 10 passo 1 faça
        escreva AGENDA[I].NOME
        escreva AGENDA[I].ENDEREÇO
        escreva AGENDA[I].TELEFONE
      fim_para
    fim_se
  fim_enquanto
fim
  
```

Tópico 9.6 - Exercício 2 - Página 207







Português Estruturado

```

programa Cap09_Ex2_Pg207
  tipo
    Dados = registro
      NOME : cadeia
      NOTA : conjunto[1..4] de real
    fim_registro
  var
    ALUNO : conjunto[1..20] de Dados
    X : Dados
    I, J, OPÇÃO, COMEÇO, FINAL, MEIO : inteiro
    SOMAR, MÉDIA : real
    ACHA : lógico
    PESQ : cadeia
  início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 4) faça
      escreva "1 - Cadastrar"
      escreva "2 - Pesquisar"
      escreva "3 - Apresentar"
      escreva "4 - fim"
      leia OPÇÃO
      se (OPÇÃO = 1) então
        para I de 1 até 20 passo 1 faça
          leia ALUNO[I].NOME
          para J de 1 até 4 passo 1 faça
            leia ALUNO[I].NOTA[J]
      fim_se
    fim_enquanto
  
```

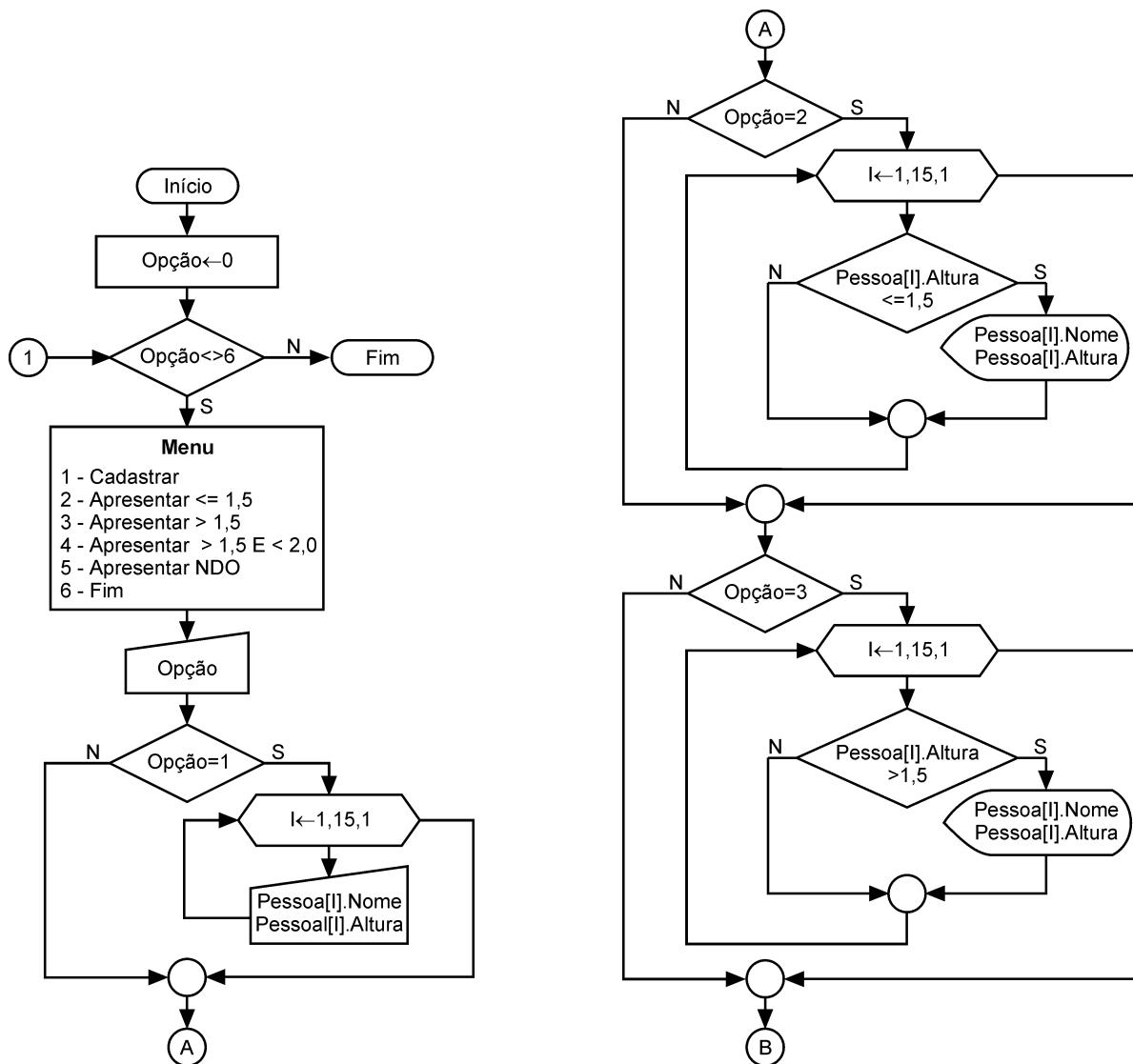
```

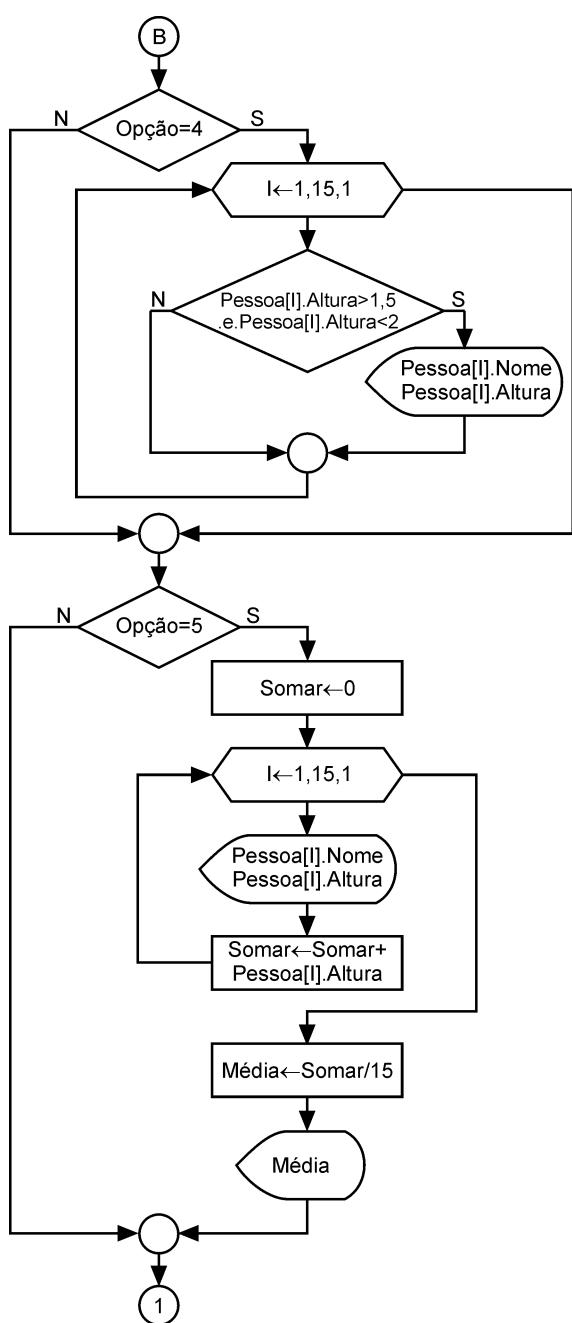
        fim_para
fim_para
para I de 1 até 19 passo 1 faça
    para J de I+1 até 20 passo 1 faça
        se (ALUNO[I].NOME > ALUNO[J].NOME) então
            X ← ALUNO[I]
            ALUNO[I] ← ALUNO[J]
            ALUNO[J] ← X
        fim_se
    fim_para
fim_para
fim_se
se (OPÇÃO = 2) então
    leia PESQ
    COMEÇO ← 1
    FINAL ← 20
    ACHA ← .Falso.
enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso.) faça
    MEIO ← (COMEÇO + FINAL) div 2
    se (PESQ = ALUNO[MEIO].NOME) então
        ACHA ← .Verdadeiro.
    senão
        se (PESQ < ALUNO[MEIO].NOME) então
            FINAL ← MEIO - 1
        senão
            COMECO ← MEIO + 1
        fim_se
    fim_se
fim_enquanto
se (ACHA = .Verdadeiro.) então
    SOMAR ← 0
    escreva ALUNO[MEIO].NOME
    para I de 1 até 4 passo 1 faça
        escreva ALUNO[MEIO].NOTA[I]
        SOMAR ← SOMAR + ALUNO[MEIO].NOTA[I]
    fim_se
    MÉDIA ← SOMAR / 4
    escreva MÉDIA
    se (MÉDIA >= 5) então
        escreva "Aprovado"
    senão
        escreva "Reprovado"
    fim_se
senão
    escreva "Registro Inexistente"
fim_se
fim_se
se (OPÇÃO = 3) então
    para I de 1 até 20 passo 1 faça
        SOMAR ← 0
        escreva ALUNO[I].NOME
        para J de 1 até 4 passo 1 faça
            escreva ALUNO[I].NOTA[J]
            SOMAR ← SOMAR + ALUNO[I].NOTA[J]
        fim_para
        MEDIA ← SOMAR / 4
        escreva MÉDIA
        se (MÉDIA >= 5) então
            escreva "Aprovado"
        senão
            escreva "Reprovado"
        fim_se
    fim_para
fim_se
fim_enquanto
fim

```

Tópico 9.6 - Exercício 3 - Página 207

Diagrama de Blocos





Português estruturado

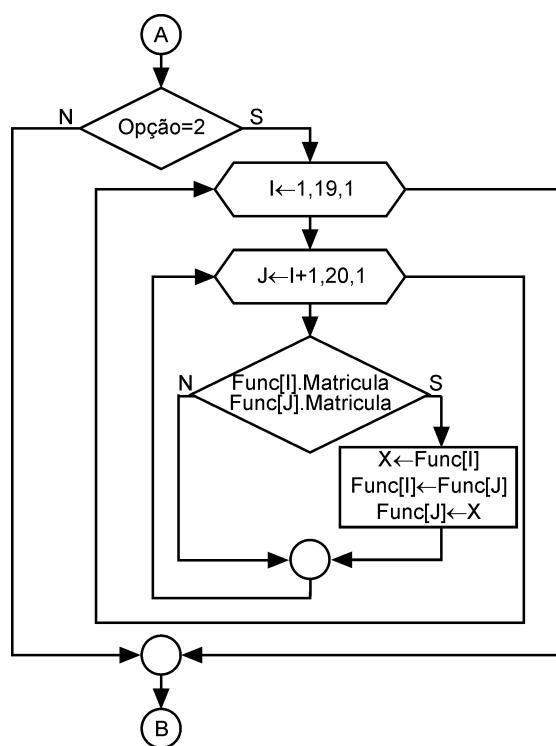
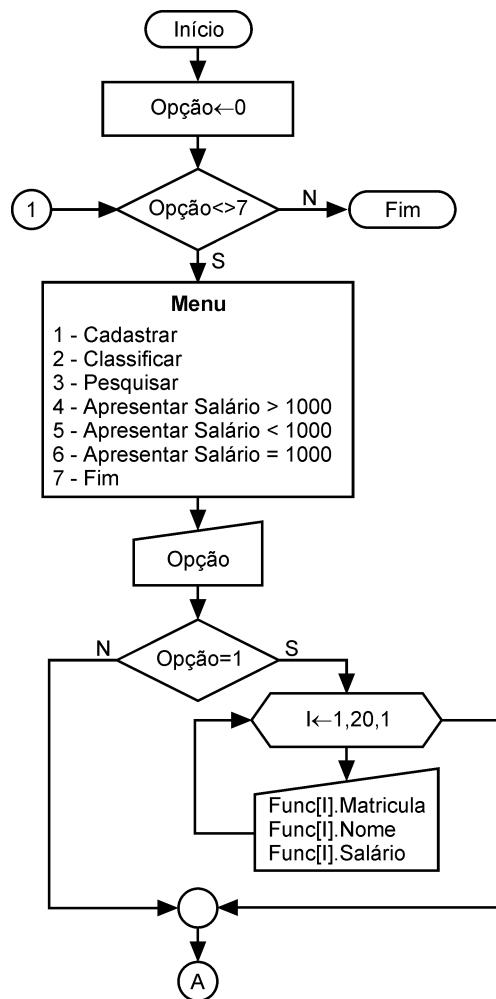
```

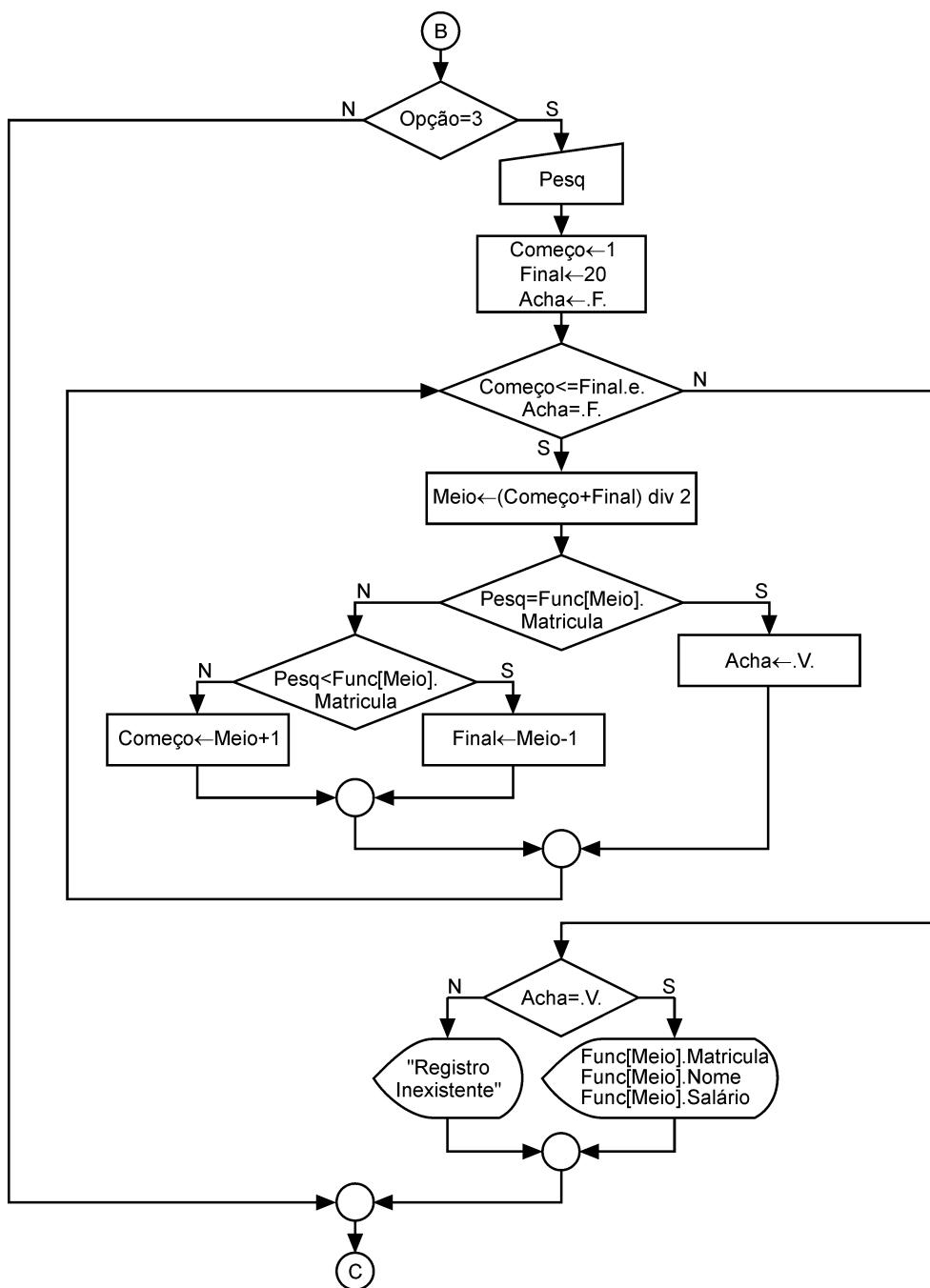
programa Ex3_Pg166
tipo
    Dados = registro
        NOME : cadeia
        ALTURA : real
    fim_registro
var
    PESSOA : conjunto[1..15] de Dados
    I, J, OPÇÃO : inteiro
    SOMAR, MÉDIA : real
início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 6) faça
        escreva "1 - Cadastrar"
        escreva "2 - Apresentar <= 1.5"
        escreva "3 - Apresentar > 1.5"
        escreva "4 - Apresentar > 1.5 e < 2.0"
        escreva "5 - Apresentar tudo"
        escreva "6 - Fim"
        leia OPÇÃO
        se (OPÇÃO = 1) então
            para I de 1 até 15 passo 1 faça
                leia PESSOA[I].NOME
                leia PESSOA[I].ALTURA
            fim_para
        fim_se
        se (OPÇÃO = 2) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA <= 1.5) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 3) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA > 1.5) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 4) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA > 1.5) .e. (PESSOA[I].ALTURA < 2) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 5) então
            Somar ← 0
            para I de 1 até 15 passo 1 faça
                escreva PESSOA[I].NOME
                escreva PESSOA[I].ALTURA
                SOMAR ← SOMAR + PESSOA[I].ALTURA
            fim_para
        MÉDIA ← SOMAR / 15
        escreva MÉDIA
    fim_se
fim_enquanto
fim

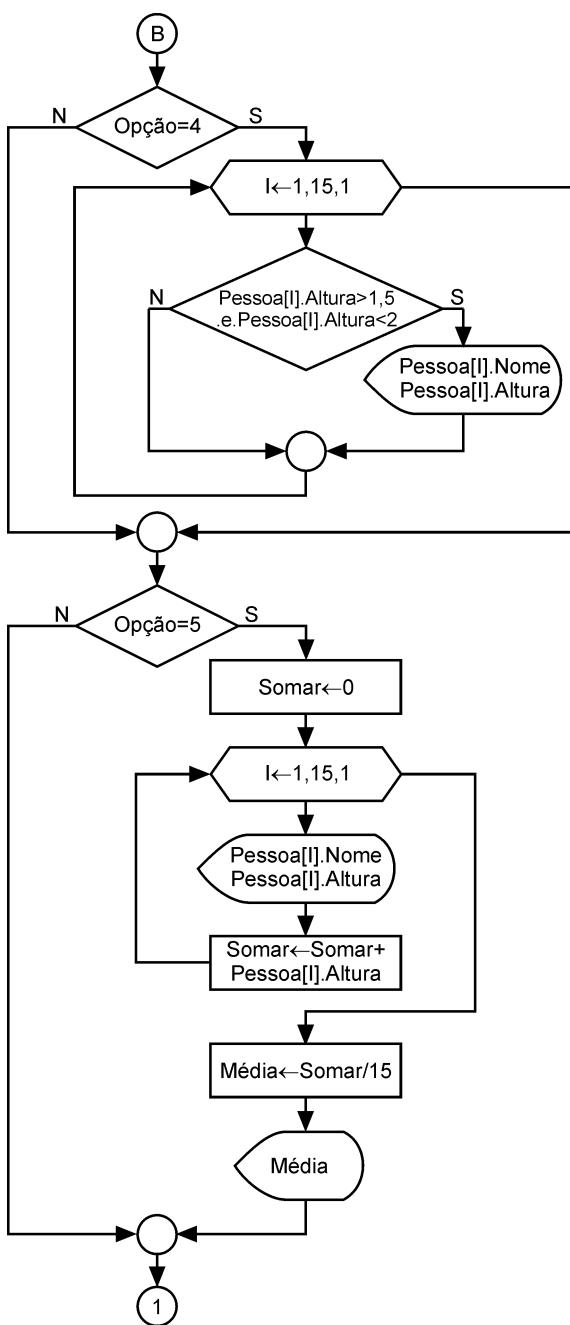
```

Tópico 9.6 - Exercício 4 - Página 207

Diagrama de Blocos





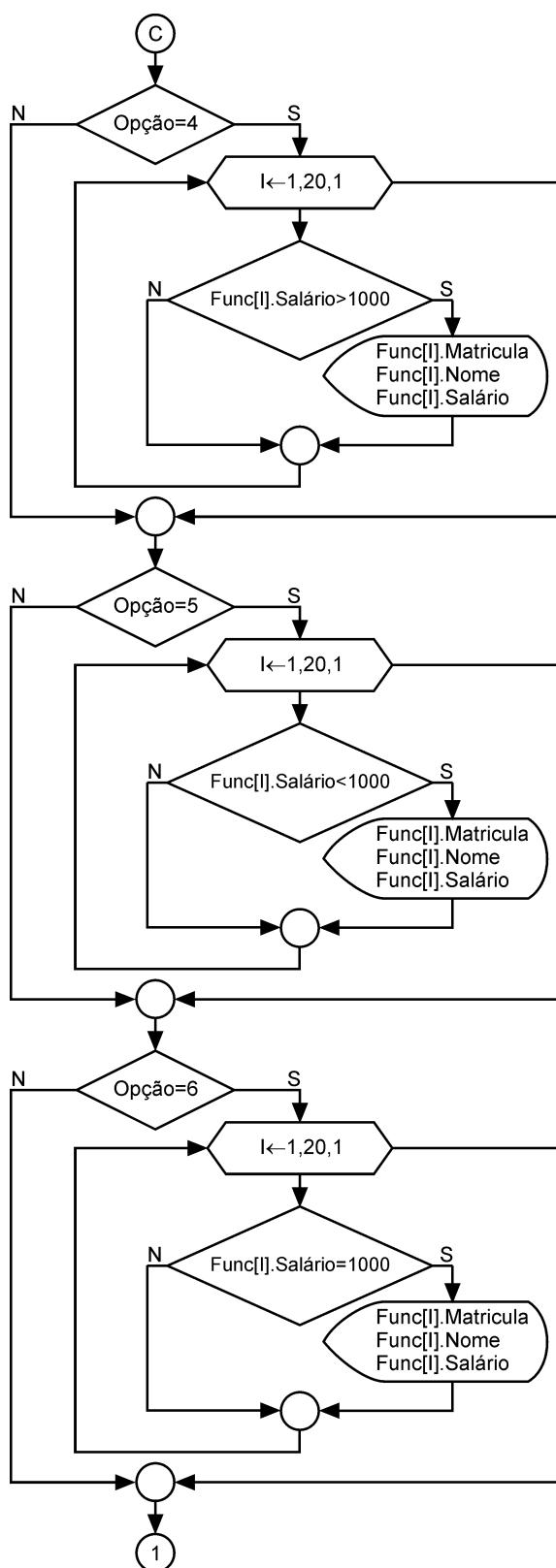


Português estruturado

```

programa Ex3_Pg166
tipo
    Dados = registro
        NOME : cadeia
        ALTURA : real
    fim_registro
var
    PESSOA : conjunto[1..15] de Dados
    I, J, OPÇÃO : inteiro
    SOMAR, MÉDIA : real
início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 6) faça
        escreva "1 - Cadastrar"
        escreva "2 - Apresentar <= 1.5"
        escreva "3 - Apresentar > 1.5"
        escreva "4 - Apresentar > 1.5 e < 2.0"
        escreva "5 - Apresentar tudo"
        escreva "6 - Fim"
        leia OPÇÃO
        se (OPÇÃO = 1) então
            para I de 1 até 15 passo 1 faça
                leia PESSOA[I].NOME
                leia PESSOA[I].ALTURA
            fim_para
        fim_se
        se (OPÇÃO = 2) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA <= 1.5) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 3) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA > 1.5) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 4) então
            para I de 1 até 15 passo 1 faça
                se (PESSOA[I].ALTURA > 1.5) .e. (PESSOA[I].ALTURA < 2) então
                    escreva PESSOA[I].NOME
                    escreva PESSOA[I].ALTURA
                fim_se
            fim_para
        fim_se
        se (OPÇÃO = 5) então
            Somar ← 0
            para I de 1 até 15 passo 1 faça
                escreva PESSOA[I].NOME
                escreva PESSOA[I].ALTURA
                SOMAR ← SOMAR + PESSOA[I].ALTURA
            fim_para
        MÉDIA ← SOMAR / 15
        escreva MÉDIA
    fim_se
fim_enquanto
fim

```



Português Estruturado

```

programa Cap09_Ex4_Pg207
tipo
  Dados = registro
    MATRICULA : inteiro
    NOME : cadeia
    SALÁRIO : real
  fim_registro
var
  FUNC : conjunto [1..20] de Dados

```

```
X : Dados
I, J, OPÇÃO, COMEÇO, FINAL, MEIO : inteiro
ACHA : lógico
PESQ : inteiro
início
  OPÇÃO ← 0
  enquanto (OPÇÃO <> 7) faça
    escreva "1 - Cadastrar"
    escreva "2 - Classificar"
    escreva "3 - Pesquisar"
    escreva "4 - Apresentar salários > 1000"
    escreva "5 - Apresentar salários < 1000"
    escreva "6 - Apresentar salários = 1000"
    escreva "7 - Fim"
    leia OPÇÃO
    se (OPÇÃO = 1) então
      para I de 1 até 20 passo 1 faça
        leia FUNC[I].MATRICULA
        leia FUNC[I].NOME
        leia FUNC[I].SALÁRIO
      fim_para
    fim_se
    se (OPÇÃO = 2) então
      para I de 1 até 19 passo 1 faça
        para J de I+1 até 20 passo 1 faça
          se (FUNC[I].MATRICULA > FUNC[J].MATRICULA) então
            X ← FUNC[I]
            FUNC[I] ← FUNC[J]
            FUNC[J] ← X
          fim_se
        fim_para
      fim_para
    fim_se
    se (OPÇÃO = 3) então
      leia PESQ
      COMEÇO ← 1
      FINAL ← 20
      ACHA ← .Falso.
      enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso..) faça
        MEIO ← (COMEÇO + FINAL) div 2
        se (PESQ = FUNC[MEIO].MATRICULA) então
          ACHA ← .Verdadeiro.
        senão
          se (PESQ < FUNC[MEIO].MATRICULA) então
            FINAL ← MEIO - 1
          senão
            COMECO ← MEIO + 1
          fim_se
        fim_se
      fim_enquanto
      se (ACHA = .Verdadeiro..) então
        escreva FUNC[MEIO].MATRICULA
        escreva FUNC[MEIO].NOME
        escreva FUNC[MEIO].SALÁRIO
      senão
        escreva "Registro Inexistente"
      fim_se
    fim_se
    se (OPCAO = 4) então
      para I de 1 até 20 passo 1 faça
        se (FUNC[I].SALÁRIO > 1000) então
          escreva FUNC[I].MATRICULA
          escreva FUNC[I].NOME
          escreva FUNC[I].SALÁRIO
        fim_se
      fim_para
    fim_se
    se (OPCAO = 5) então
      para I de 1 até 20 passo 1 faça
        se (FUNC[I].SALÁRIO < 1000) então
          escreva FUNC[I].MATRICULA
          escreva FUNC[I].NOME
          escreva FUNC[I].SALÁRIO
        fim_se
      fim_para
    fim_se
    se (OPCAO = 6) então
      para I de 1 até 20 passo 1 faça
        se (FUNC[I].SALÁRIO = 1000) então
```

```

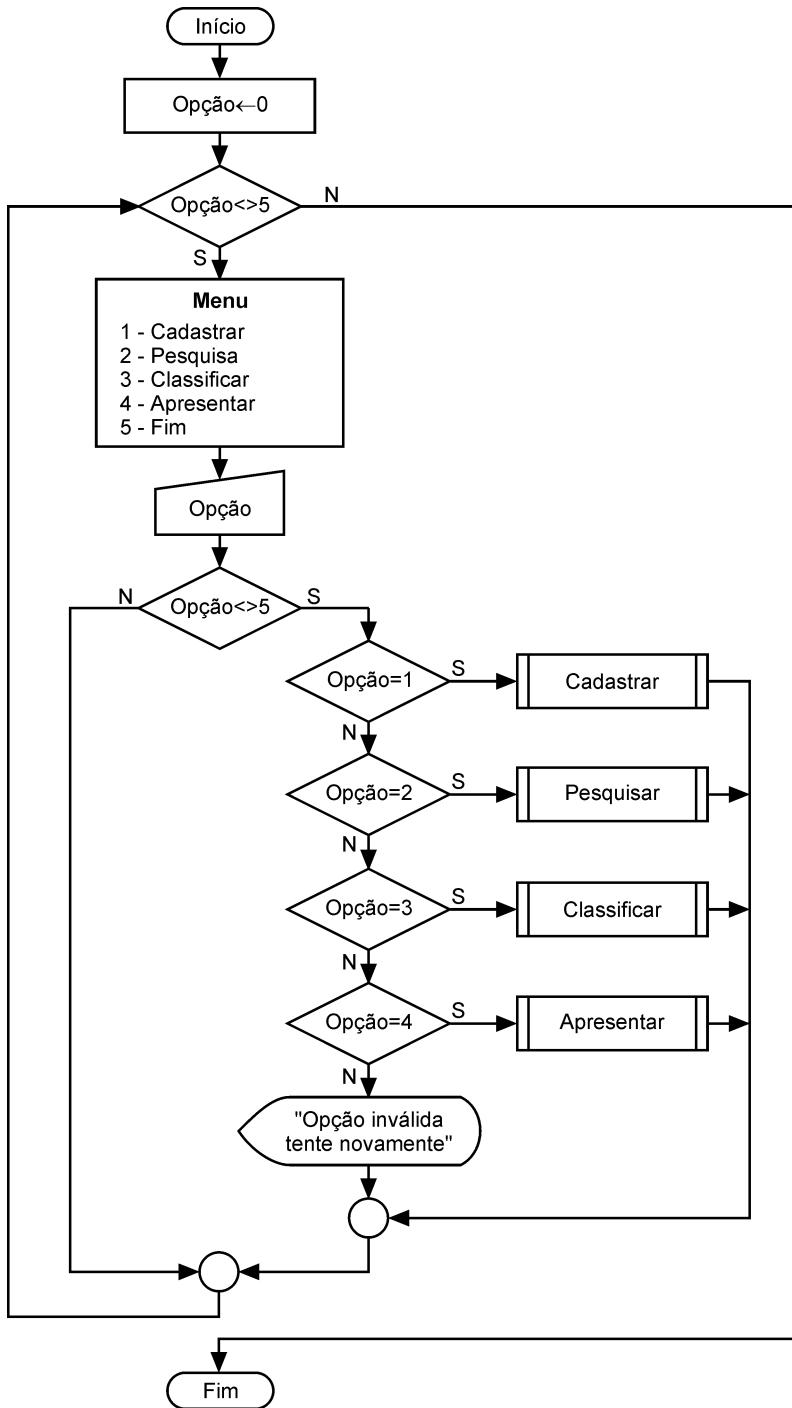
escreva FUNC[I].MATRICULA
escreva FUNC[I].NOME
escreva FUNC[I].SALÁRIO
fim_se
fim_para
fim_se
fim_enquanto
fim

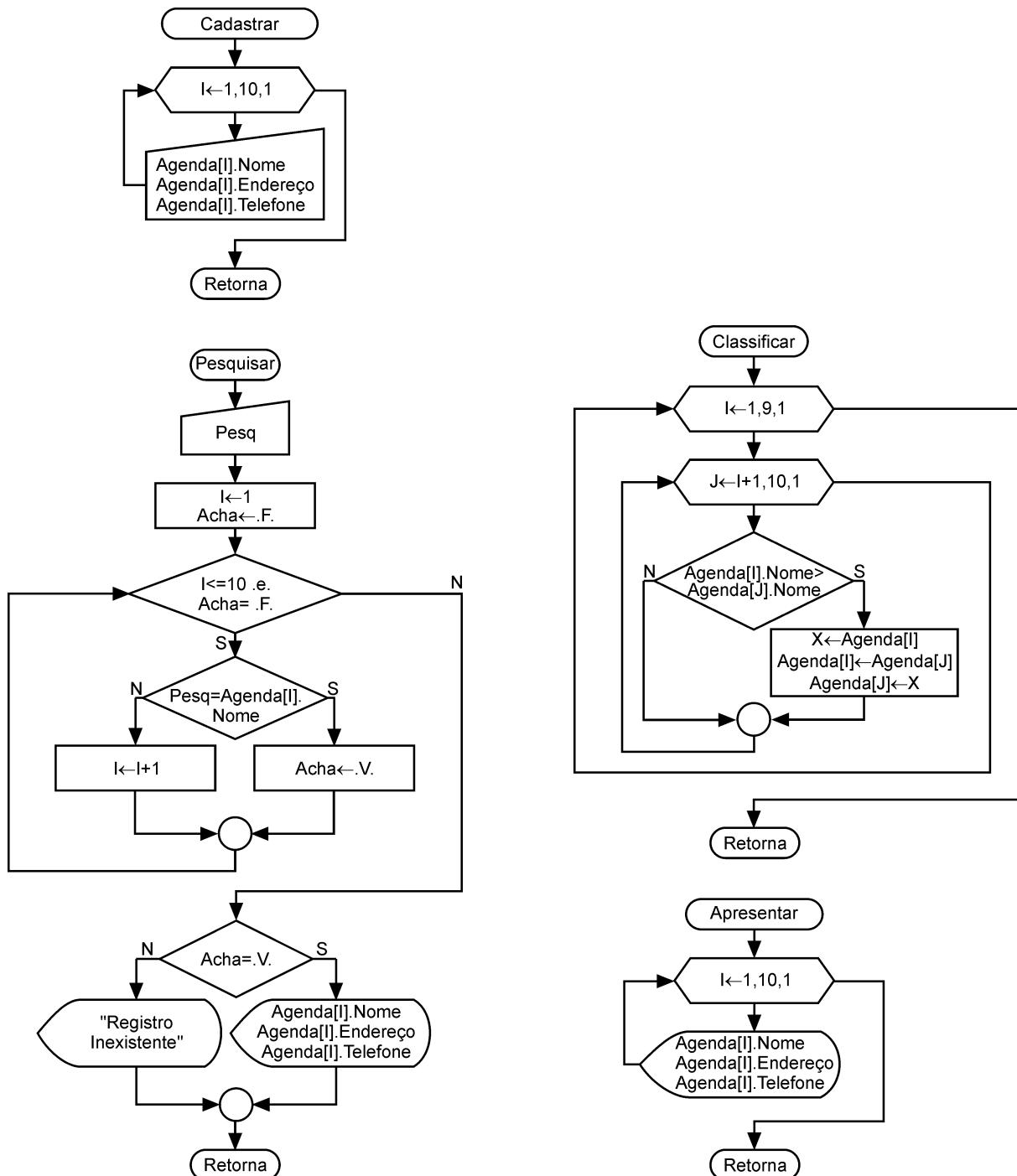
```

10 - Exercícios de fixação do capítulo 10

Tópico 10.9 - Exercício 1 - Página 259

a) Diagramas de Blocos





Português Estruturado

```

programa Cap10_Ex1a_Pg259
tipo
    Dados = registro
        NOME : cadeia
        ENDEREÇO : cadeia
        TELEFONE : cadeia
    fin_registro

var
    AGENDA : conjunto[1..10] de Dados

procedimento Cadastrar
var
    I : inteiro
início
    para I de 1 até 10 passo 1 faça
        leia AGENDA[I].NOME
        leia AGENDA[I].ENDEREÇO
        leia AGENDA[I].TELEFONE
    fim

```

```

    fim_para
fim

procedimento Pesquisar
var
    ACHA : lógico
    PESQ : cadeia
    I, J : inteiro
início
    leia PESQ
    I ← 1
    ACHA ← .Falso.
    enquanto (I <= 10) .e. (ACHA = .Falso.) faça
        se (PESQ = AGENDA[I].NOME) então
            ACHA ← .Verdadeiro.
        senão
            I ← I + 1
        fim_se
    fim_enquanto
    se (ACHA = .Verdadeiro.) então
        escreva AGENDA[I].NOME
        escreva AGENDA[I].ENDERECO
        escreva AGENDA[I].TELEFONE
    senão
        escreva "Registro Inexistente"
    fim_se
fim

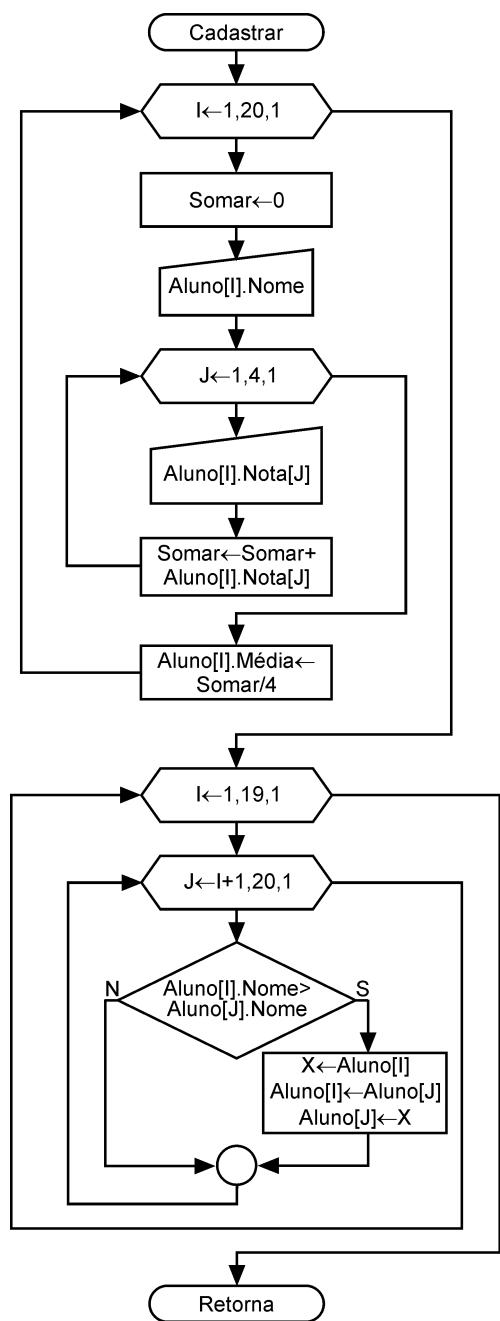
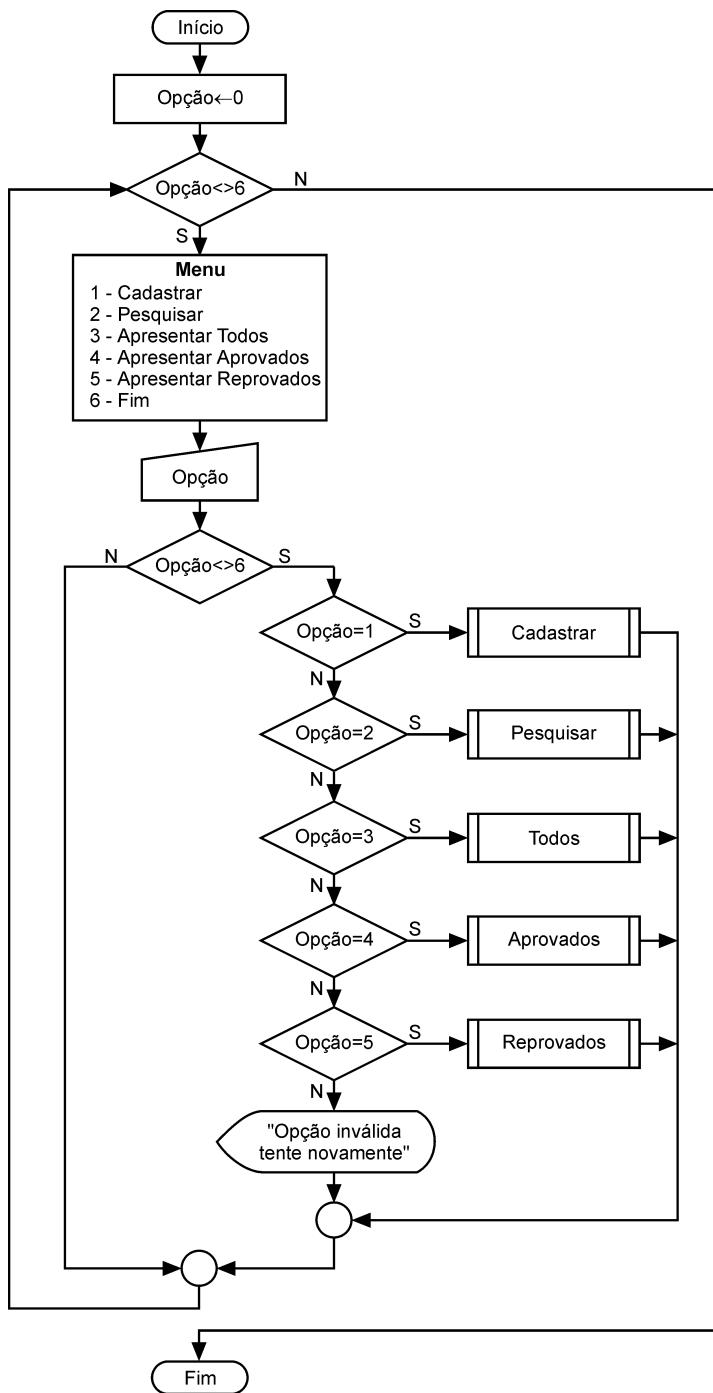
procedimento Classificar
var
    X : Dados
    I, J : inteiro
início
    para I de 1 até 9 passo 1 faça
        para J de I+1 até 10 passo 1 faça
            se (AGENDA[I].NOME > AGENDA[J].NOME) então
                X ← AGENDA[I]
                AGENDA[I] ← AGENDA[J]
                AGENDA[J] ← X
            fim_se
        fim_para
    fim_para
fim

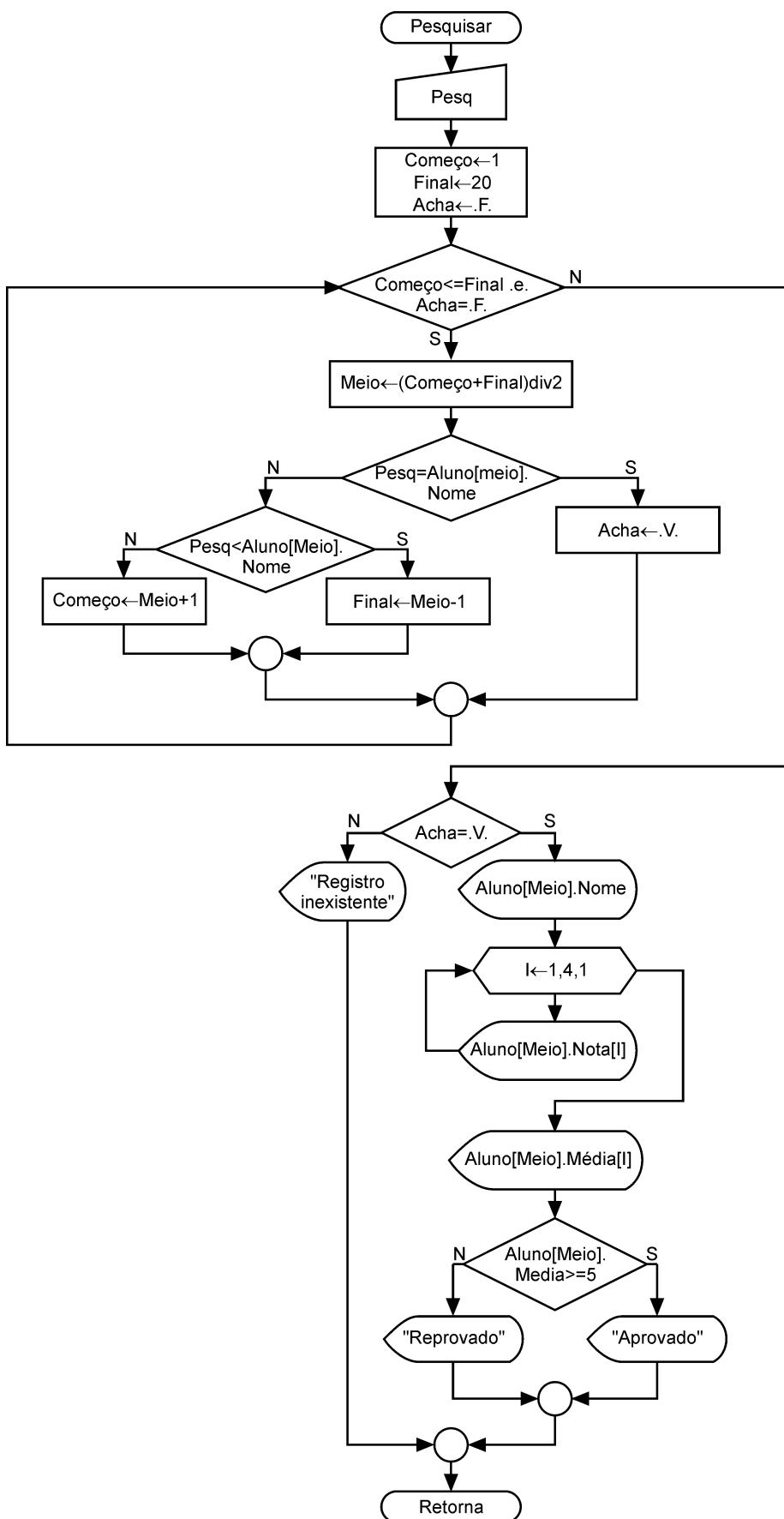
procedimento Apresentar
var
    I : inteiro
início
    para I de 1 até 10 passo 1 faça
        escreva AGENDA[I].NOME
        escreva AGENDA[I].ENDERECO
        escreva AGENDA[I].TELEFONE
    fim_para
fim

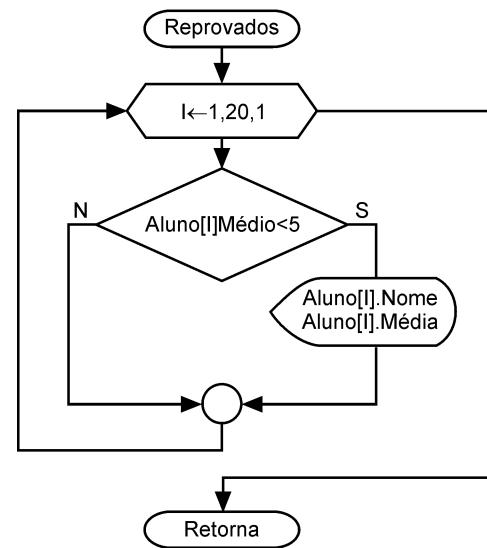
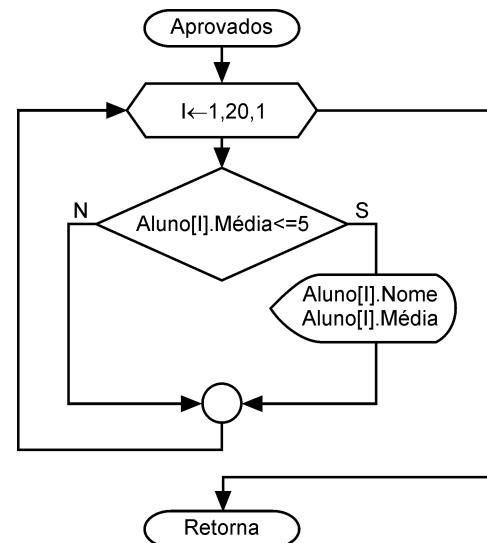
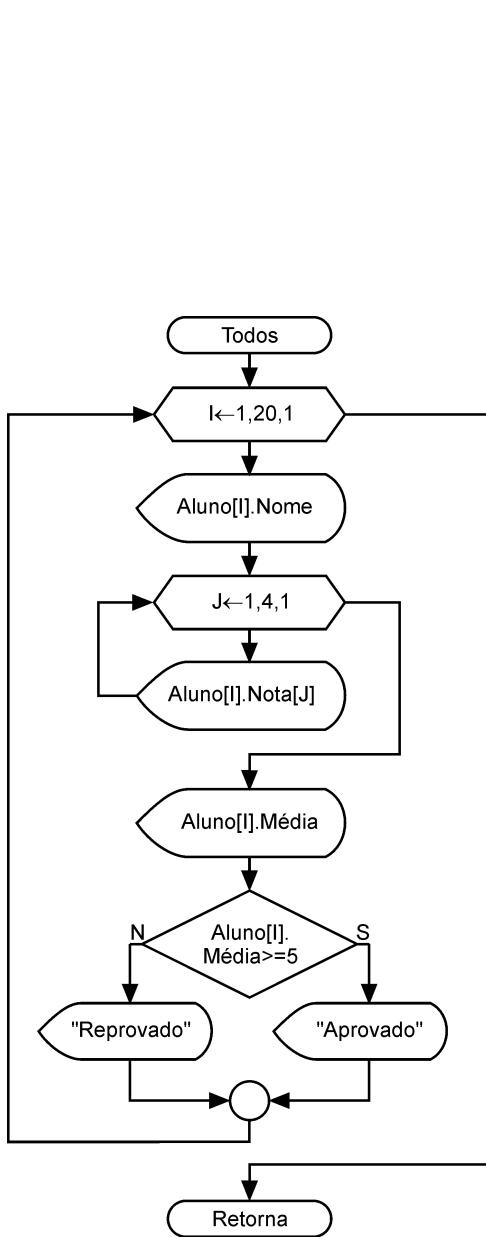
var
    OPÇÃO : inteiro
início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 5) faça
        escreva "1 - Cadastrar"
        escreva "2 - Pesquisar"
        escreva "3 - Classificar"
        escreva "4 - Apresentar"
        escreva "5 - Fim"
    leia OPÇÃO
    se (OPÇÃO <> 5) então
        caso OPÇÃO
            seja 1 faça Cadastrar
            seja 2 faça Pesquisar
            seja 3 faça Classificar
            seja 4 faça Apresentar
        senão
            escreva "Opção inválida - Tente novamente"
        fim_caso
    fim_se
fim_enquanto
fim

```

b) Diagramas de Blocos







Português Estruturado

```

programa Cap10_Ex1b_Pg259
  tipo
    Dados = registro
      NOME : cadeia
      NOTA : conjunto[1..4] de real
      MÉDIA : real
    fim_registro
  var
    ALUNO : conjunto[1..20] de Dados

  procedimento Cadastrar
  var
    I, J : inteiro
    SOMAR : real
    X : Dados
  início
    para I de 1 até 20 passo 1 faça
      SOMAR ← 0
      leia ALUNO[I].NOME
      para J de 1 até 4 passo 1 faça
        leia ALUNO[I].NOTA[J]
        SOMAR ← SOMAR + ALUNO[I].NOTA[J]
      fim_para
      ALUNO[I].MEDIA ← SOMAR / 4
  fim_procedimento
  
```

```

fim_para
para I de 1 até 19 passo 1 faça
  para J de I+1 até 20 passo 1 faça
    se (ALUNO[I].NOME > ALUNO[J].NOME) então
      X ← ALUNO[I]
      ALUNO[I] ← ALUNO[J]
      ALUNO[J] ← X
    fim_se
  fim_para
fim_para
fim

procedimento Pesquisar
var
  I, COMEÇO, FINAL, MEIO : inteiro
  ACHA : lógico
  PESQ: cadeia
início
  leia PESQ
  COMEÇO ← 1
  FINAL ← 20
  ACHA ← .Falso.
  enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso.) faça
    MEIO ← (COMEÇO + FINAL) div 2
    se (PESQ = ALUNO[MEIO].NOME) então
      ACHA ← .Verdadeiro.
    senão
      se (PESQ < ALUNO[MEIO].NOME) então
        FINAL ← MEIO - 1
      senão
        COMEÇO ← MEIO + 1
      fim_se
    fim_se
  fim_enquanto
  se (ACHA = .Verdadeiro.) então
    escreva ALUNO[MEIO].NOME
    para I de 1 até 4 passo 1 faça
      escreva ALUNO[MEIO].NOTA[I]
    fim_se
    escreva ALUNO[MEIO].MÉDIA
    se (ALUNO[MEIO].MÉDIA >= 5) então
      escreva "Aprovado"
    senão
      escreva "Reprovado"
    fim_se
  senão
    escreva "registro Inexistente"
  fim_se
fim
procedimento Todos
var
  I, J : inteiro
início
  para I de 1 até 20 passo 1 faça
    escreva ALUNO[I].NOME
    para J de 1 até 4 passo 1 faça
      escreva ALUNO[I].NOTA[J]
    fim_para
    escreva ALUNO[I].MÉDIA
    se (ALUNO[I].MÉDIA >= 5) então
      escreva "Aprovado"
    senão
      escreva "Reprovado"
    fim_se
  fim_para
fim

procedimento Aprovados
var
  I, J : inteiro
início
  para I de 1 até 20 passo 1 faça
    se (ALUNO[I].MÉDIA >= 5) então
      escreva ALUNO[I].NOME
      escreva ALUNO[I].MÉDIA
    fim_se
  fim_para
fim

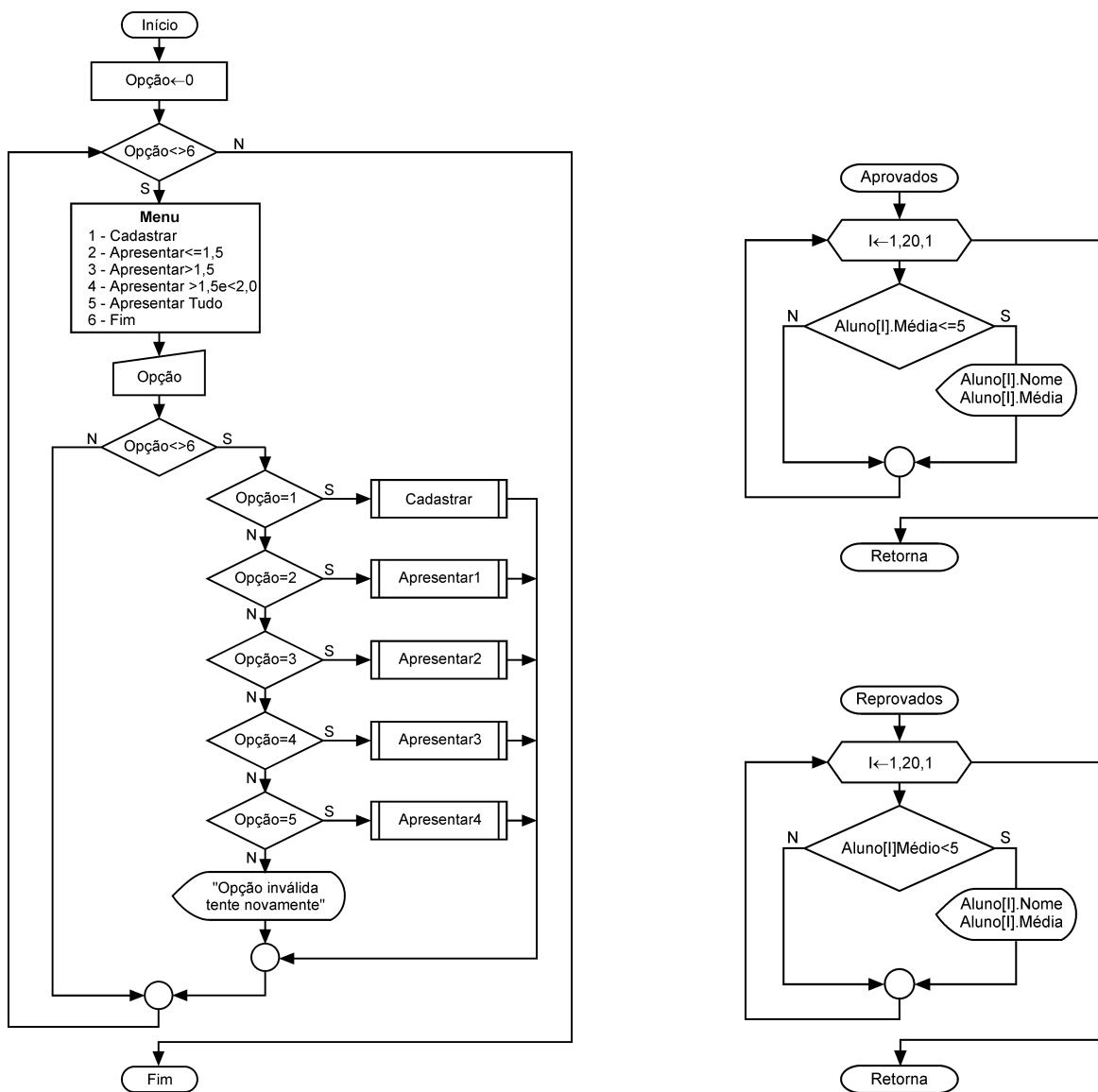
procedimento Reprovados

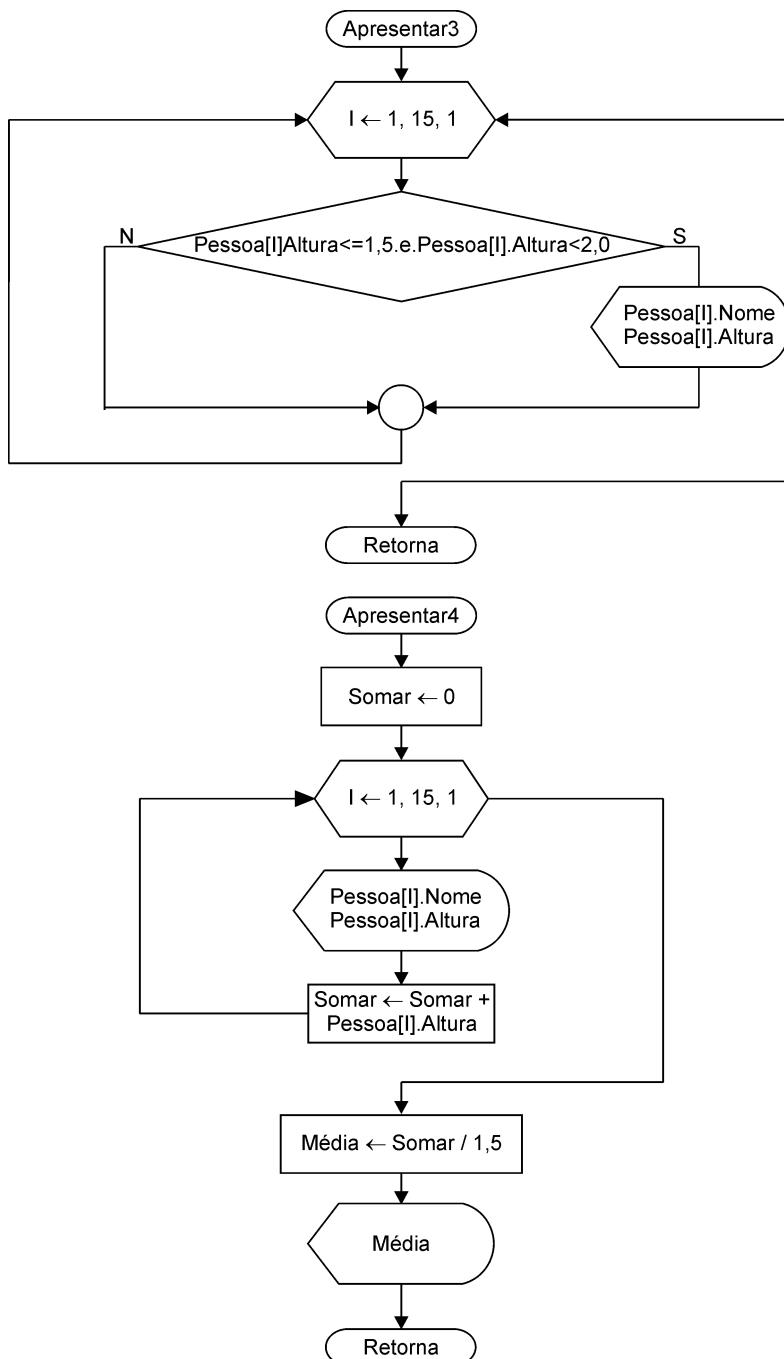
```

```
var
    I, J : inteiro
início
    para I de 1 até 20 passo 1 faça
        se (ALUNO[I].MÉDIA < 5) então
            escreva ALUNO[I].NOME
            escreva ALUNO[I].MÉDIA
        fim_se
    fim_para
fim

var
    OPÇÃO : inteiro
início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 6) faça
        escreva "1 - Cadastrar"
        escreva "2 - Pesquisar"
        escreva "3 - Apresentar Todos"
        escreva "4 - Apresentar Aprovados"
        escreva "5 - Apresentar Reprovados"
        escreva "6 - Fim"
    leia OPÇÃO
    se (OPÇÃO <> 6) então
        caso OPÇÃO
            seja 1 faça Cadastrar
            seja 2 faça Pesquisar
            seja 3 faça Todos
            seja 4 faça Aprovados
            seja 5 faça Reprovados
        senão
            escreva "Opção inválida - Tente novamente"
        fim_caso
    fim_se
fim_enquanto
fim
```

c) Diagramas de Blocos





Português Estruturado

```

programa Cap10_Ex1c_Pg260
tipo
  Dados = registro
    NOME : cadeia
    ALTURA : de real
  fim_registro
var
  PESSOA : conjunto[1..15] de Dados

procedimento Cadastrar
var
  I : inteiro
início
  para I de 1 até 15 passo 1 faça
    leia PESSOA[I].NOME
    leia PESSOA[I].ALTURA
  fim_para
fim

procedimento Apresentar1
var

```

```

I : inteiro
início
  para I de 1 até 15 passo 1 faça
    se (PESSOA[I].ALTURA <= 1.5) então
      escreva PESSOA[I].NOME
      escreva PESSOA[I].ALTURA
    fim_se
  fim_para
fim

procedimento Apresentar2
var
  I : inteiro
início
  para I de 1 até 15 passo 1 faça
    se (PESSOA[I].ALTURA > 1.5) então
      escreva PESSOA[I].NOME
      escreva PESSOA[I].ALTURA
    fim_se
  fim_para
fim

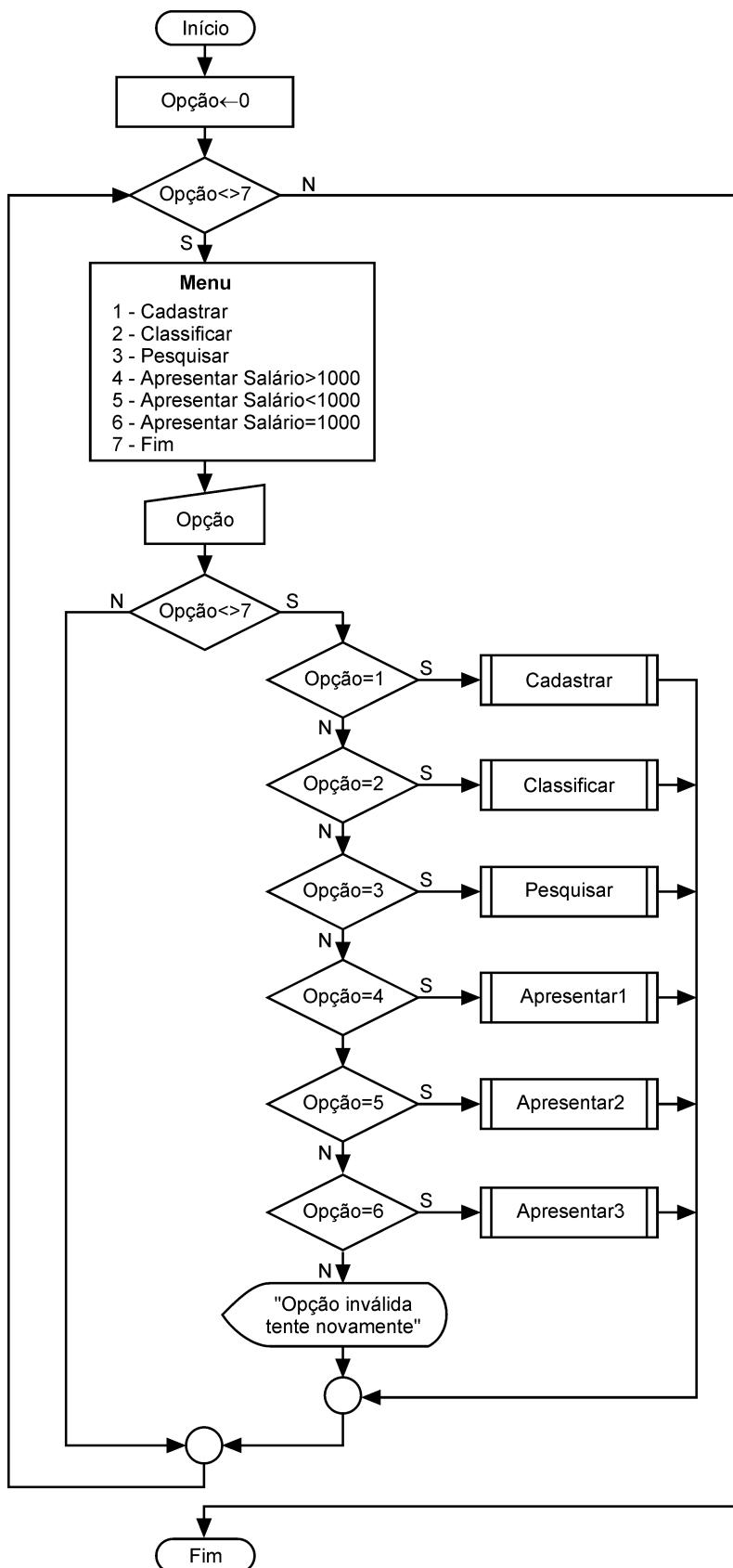
procedimento Apresentar3
var
  I : inteiro
início
  para I de 1 até 15 passo 1 faça
    se (PESSOA[I].ALTURA > 1.5) .e. (PESSOA[I].ALTURA < 2) então
      escreva PESSOA[I].NOME
      escreva PESSOA[I].ALTURA
    fim_se
  fim_para
fim

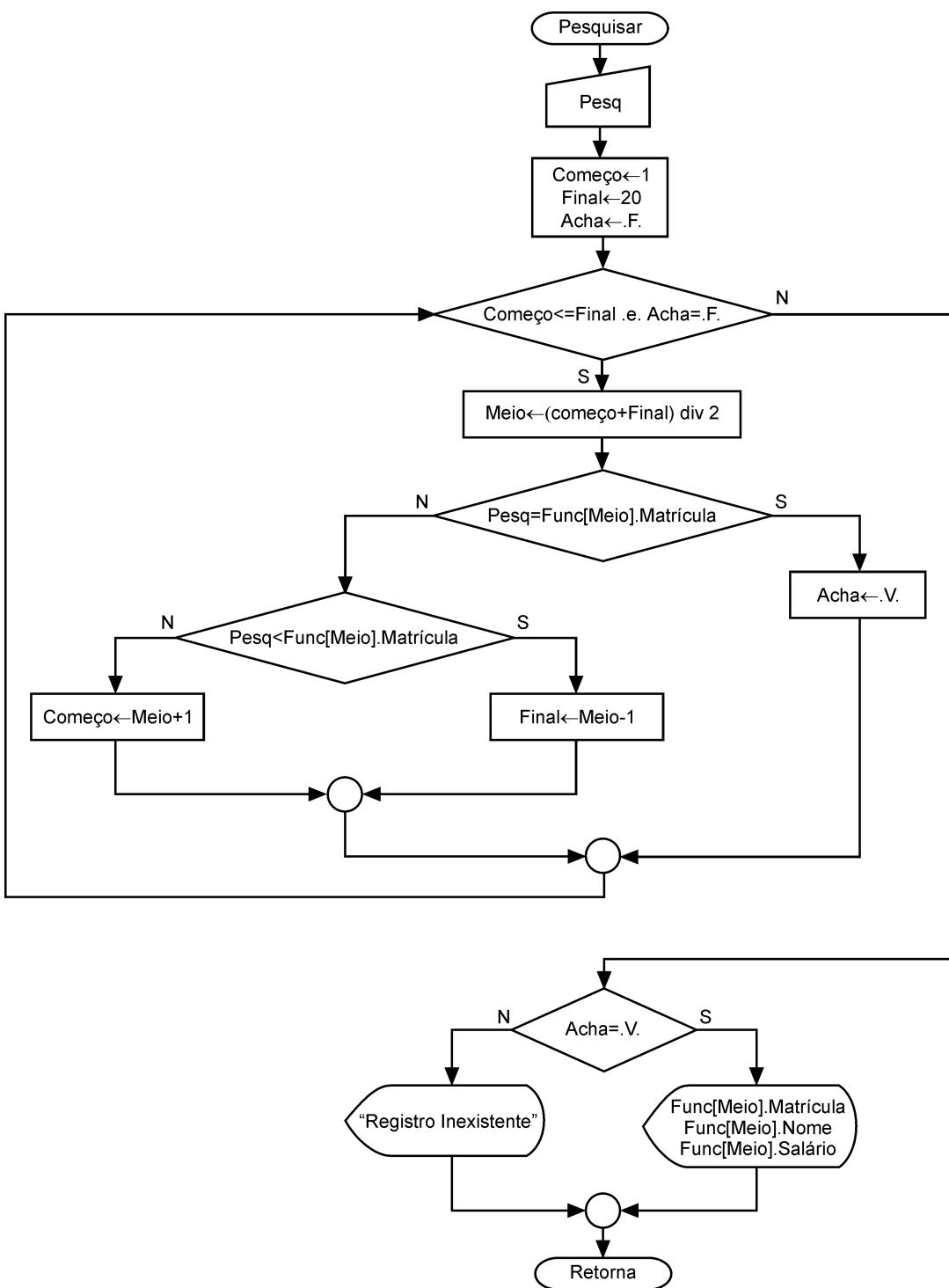
procedimento Apresentar4
var
  I : inteiro
  SOMAR, MÉDIA : real
início
  Somar ← 0
  para I de 1 até 15 passo 1 faça
    escreva PESSOA[I].NOME
    escreva PESSOA[I].ALTURA
    SOMAR ← SOMAR + PESSOA[I].ALTURA
  fim_para
  MÉDIA ← SOMAR / 15
  escreva MÉDIA
fim

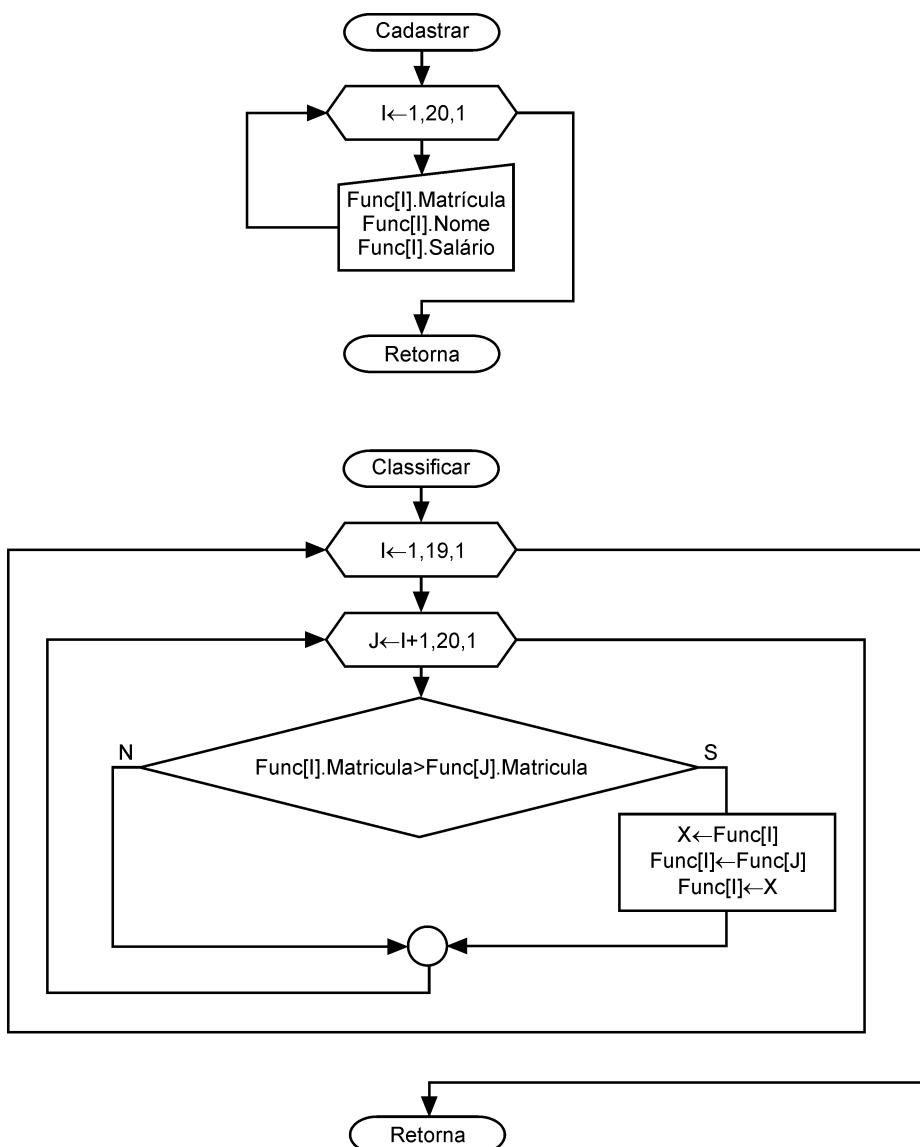
var
  OPÇÃO : inteiro
início
  OPÇÃO ← 0
  enquanto (OPÇÃO <> 6) faça
    escreva "1 - Cadastrar"
    escreva "2 - Apresentar <= 1.5"
    escreva "3 - Apresentar > 1.5"
    escreva "4 - Apresentar > 1.5 e < 2.0"
    escreva "5 - Apresentar tudo"
    escreva "6 - Fim"
  leia OPÇÃO
  se (OPÇÃO <> 6) então
    caso OPÇÃO
      seja 1 faça Cadastrar
      seja 2 faça Apresentar1
      seja 3 faça Apresentar2
      seja 4 faça Apresentar3
      seja 5 faça Apresentar4
    senão
      escreva "Opção inválida - Tente novamente"
    fim_caso
  fim_se
fim_enquanto
fim

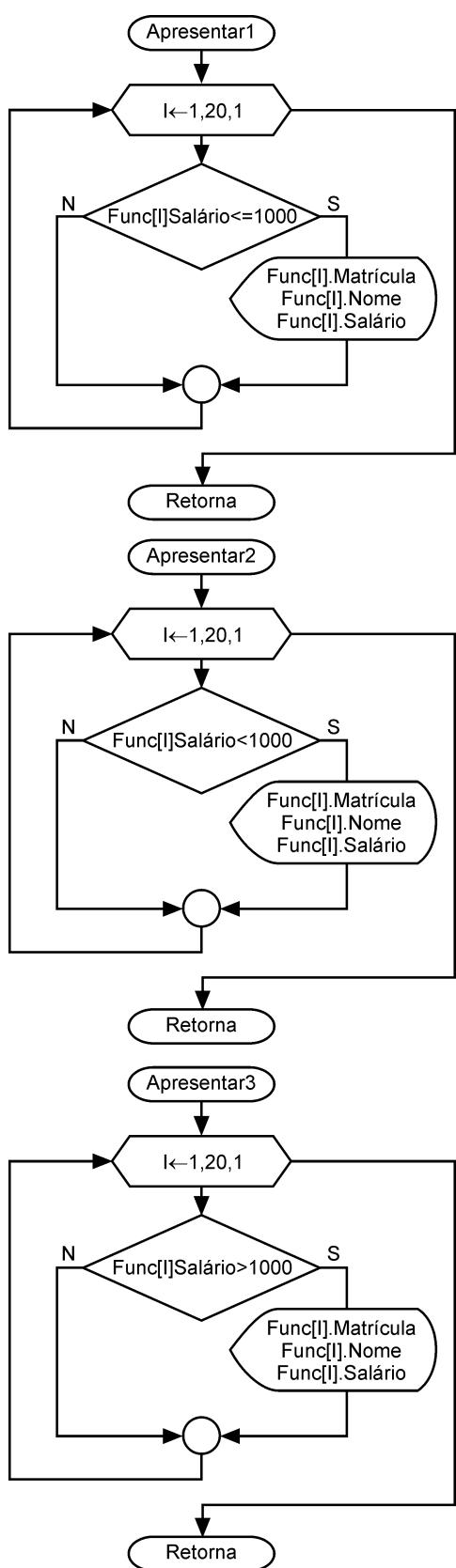
```

d) Diagrama de blocos









Português Estruturado

```

programa Cap10_Ex1d_Pg260
var
tipo
  Dados = registro
    MATRICULA : inteiro
    NOME : cadeia
    SALÁRIO : real
  fim_registro

var
  FUNC : conjunto[1..20] de Dados

procedimento Cadastrar
var
  I : inteiro
início
  para I de 1 até 20 passo 1 faça
    leia FUNC[I].MATRICULA
    leia FUNC[I].NOME
    leia FUNC[I].SALÁRIO
  fim_para
fim

procedimento Classificar
var
  I, J : inteiro
  X : Dados
início
  para I de 1 até 19 passo 1 faça
    para J de I+1 até 20 passo 1 faça
      se (FUNC[I].MATRICULA > FUNC[J].MATRICULA) então
        X ← FUNC[I]
        FUNC[I] ← FUNC[J]
        FUNC[J] ← X
      fim_se
    fim_para
  fim_para
fim

procedimento Pesquisar
var
  COMEÇO, FINAL, MEIO, PESQ : inteiro
  ACHA : lógico
início
  leia PESQ
  COMEÇO ← 1
  FINAL ← 20
  ACHA ← .Falso.
  enquanto (COMEÇO <= FINAL) .e. (ACHA = .Falso.) faça
    MEIO ← (COMEÇO + FINAL) div 2
    se (PESQ = FUNC[MEIO].MATRICULA) então
      ACHA ← .Verdadeiro.
    senão
      se (PESQ < FUNC[MEIO].MATRICULA) então
        FINAL ← MEIO - 1
      senão
        COMEÇO ← MEIO + 1
      fim_se
    fim_enquanto
    se (ACHA = .Verdadeiro.) então
      escreva FUNC[MEIO].MATRICULA
      escreva FUNC[MEIO].NOME
      escreva FUNC[MEIO].SALÁRIO
    senão
      escreva "Registro Inexistente"
    fim_se
fim

procedimento Apresentar1
var
  I : inteiro
início
  para I de 1 até 20 passo 1 faça
    se (FUNC[I].SALÁRIO > 1000) então
      escreva FUNC[I].MATRICULA

```

```

    escreva FUNC[I].NOME
    escreva FUNC[I].SALÁRIO
fim_se
fim_para
fim

procedimento Apresentar2
var
    I : inteiro
início
    para I de 1 até 20 passo 1 faça
        se (FUNC[I].SALÁRIO < 1000) então
            escreva FUNC[I].MATRICULA
            escreva FUNC[I].NOME
            escreva FUNC[I].SALÁRIO
        fim_se
    fim_para
fim

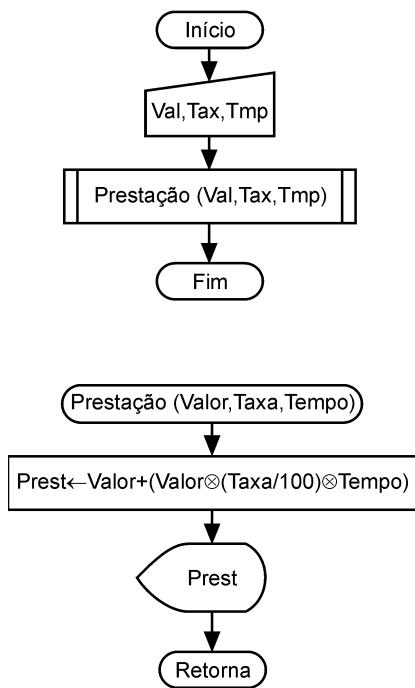
procedimento Apresentar3
var
    I : inteiro
início
    para I de 1 até 20 passo 1 faça
        se (FUNC[I].SALÁRIO = 1000) então
            escreva FUNC[I].MATRICULA
            escreva FUNC[I].NOME
            escreva FUNC[I].SALÁRIO
        fim_se
    fim_para
fim

var
    OPÇÃO : inteiro
início
    OPÇÃO ← 0
    enquanto (OPÇÃO <> 7) faça
        escreva "1 - Cadastrar"
        escreva "2 - Classificar"
        escreva "3 - Pesquisar"
        escreva "4 - Apresentar salários > 1000"
        escreva "5 - Apresentar salários < 1000"
        escreva "6 - Apresentar salários = 1000"
        escreva "7 - Fim"
        leia OPÇÃO
        se (OPÇÃO <> 7) então
            caso OPÇÃO
                seja 1 faça Cadastrar
                seja 2 faça Classificar
                seja 3 faça Pesquisar
                seja 4 faça Apresentar1
                seja 5 faça Apresentar2
                seja 6 faça Apresentar3
            senão
                escreva "Opção inválida - Tente novamente"
            fim_caso
        fim_se
    fim_enquanto
fim

```

Tópico 10.9 - Exercício 2 - Página 260

a) Diagramas de Blocos



Português Estruturado

```

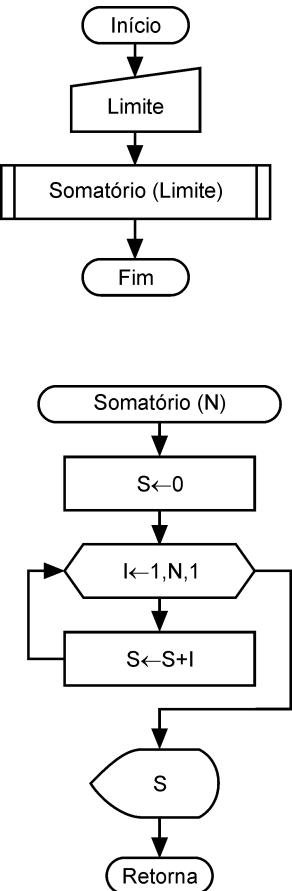
programa Cap10_Ex2a_Pg260
procedimento Prestação(VALOR, TAXA, TEMPO : real)
var
    PREST : real
início
    PREST ← VALOR + (VALOR * (TAXA / 100) * TEMPO)
    escreva PREST
fim

var
    VAL, TAX, TMP : real
início
    leia VAL, TAX, TMP
    Prestação(VAL, TAX, TMP)
fim

```

Português Estruturado

b) Diagramas de Blocos



Português Estruturado

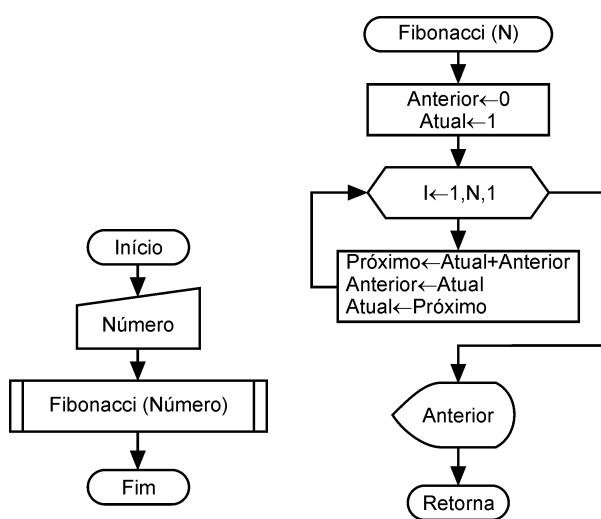
```

programa Cap10_Ex2b_Pg260
procedimento Somatório(N : inteiro)
var
    S, I : inteiro
início
    S ← 0
    para I de 1 até N passo 1 faça
        S ← S + I
    fim_para
    escreva S
fim

var
    LIMITE : inteiro
início
    leia LIMITE
    Somatório(LIMITE)
fim

```

c) Diagramas de Blocos



Português Estruturado

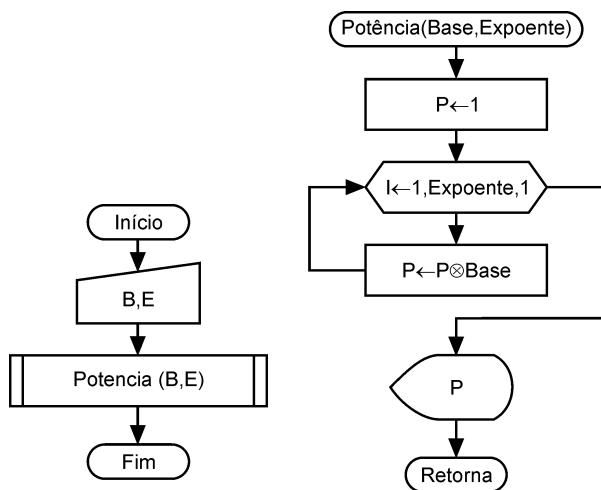
```

programa Cap10_Ex2c_Pg260
procedimento Fibonacci(N : inteiro)
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  para I de 1 até N passo 1 faça
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
  fim_para
  escreva ANTERIOR
  fim

var
  NÚMERO : inteiro
início
  leia NÚMERO
  Fibonacci(NÚMERO)
fim
  
```

Tópico 10.9 - Exercício 2 d - Página 261

d) Diagramas de Blocos



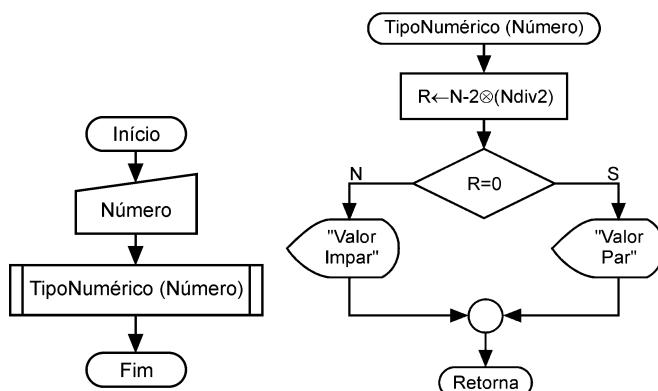
Português Estruturado

```

programa Cap10_Ex2d_Pg261
procedimento Potência(BASE, EXPOENTE : inteiro)
var
  I, P : inteiro
início
  P ← 1
  para I de 1 até EXPOENTE passo 1 faça
    P ← P * BASE
  fim_para
  escreva P
  fim

var
  B, E : inteiro
início
  leia B, E
  Potência(B, E)
fim
  
```

e) Diagramas de Blocos



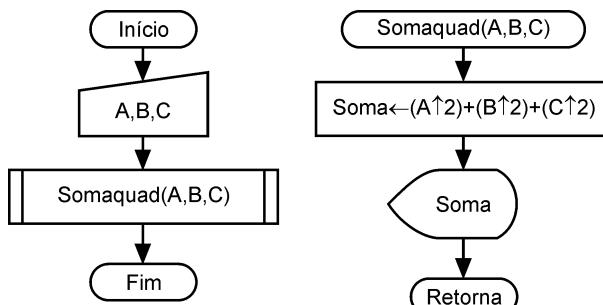
Português Estruturado

```

programa Cap10_Ex2e_Pg261
procedimento TipoNumérico(N : inteiro)
var
  R : inteiro
início
  R ← N - 2 * (N div 2)
  se (R = 0) então
    escreva "Valor Par"
  senão
    escreva "Valor Ímpar"
  fim_se
  fim

var
  NÚMERO : inteiro
início
  leia NÚMERO
  TipoNumérico(NÚMERO)
fim
  
```

f) Diagramas de blocos



Português estruturado

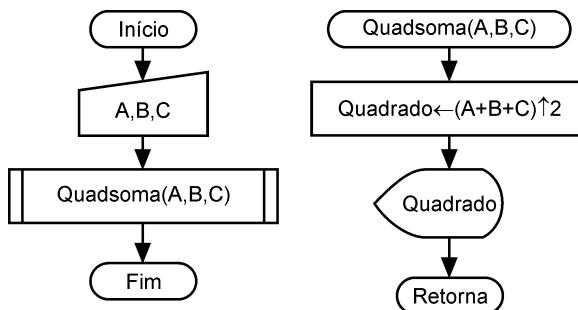
```

programa Cap10_Ex2f_Pg261
procedimento Somaquad(A, B, C : inteiro)
var
  SOMA : inteiro
início
  SOMA ← (A ↑ 2) + (B ↑ 2) + (C ↑ 2)
  escreva SOMA
fim

var
  A, B, C : inteiro
início
  leia A, B, C
  Somaquad(A, B, C)
fim

```

g) Diagramas de blocos



Português estruturado

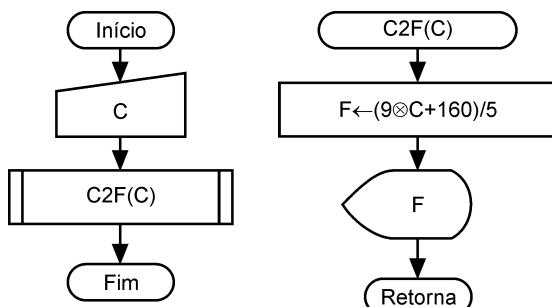
```

programa Cap10_Ex2g_Pg261
procedimento Quadsoma(A, B, C : inteiro)
var
  QUADRADO : inteiro
início
  QUADRADO ← (A + B + C) ↑ 2
  escreva QUADRADO
fim

var
  A, B, C : inteiro
início
  leia A, B, C
  Quadsoma(A, B, C)
fim

```

h) Diagramas de blocos



Português estruturado

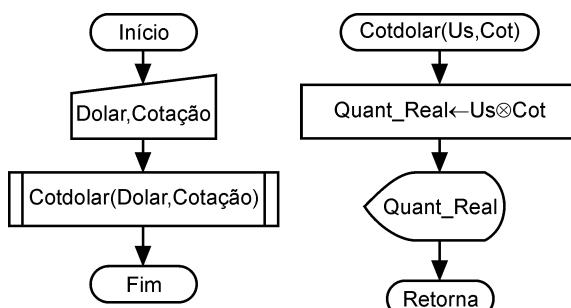
```

programa Cap10_Ex2h_Pg261
procedimento C2F(C : real)
var
  F : real
início
  F ← (9 * C + 160) / 5
  escreva F
fim

var
  C : real
início
  leia C
  C2F(C)
fim

```

i) Diagramas de blocos



Português estruturado

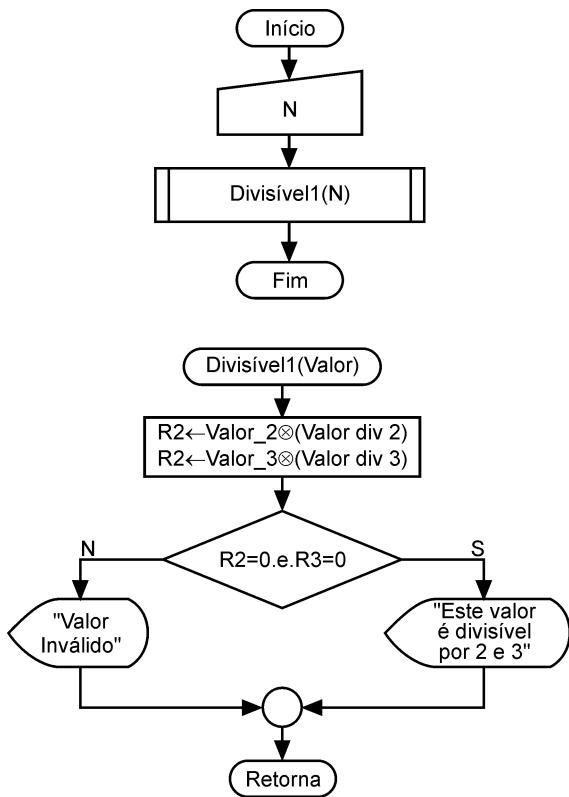
```

programa Cap10_Ex2i_Pg261
procedimento Cotdolar(US, COT : real)
var
  QUANT_REAL : real
início
  QUANT_REAL ← US * COT
  escreva QUANT_REAL
fim

var
  DOLAR, COTAÇÃO : real
início
  leia DOLAR, COTAÇÃO
  Cotdolar(DOLAR, COTAÇÃO)
fim

```

j) Diagramas de blocos



Português estruturado

```

programa Cap10_Ex2j_Pg261
procedimento Divisível1(VALOR : inteiro)
var
    R2, R3 : inteiro
início
    R2 ← VALOR - 2 * (VALOR div 2)
    R3 ← VALOR - 3 * (VALOR div 3)
    se (R2 = 0) .e. (R3 = 0) então
        escreva "Este valor é divisível por 2 e 3"
    senão
        escreva "Valor inválido"
    fim_se
fim

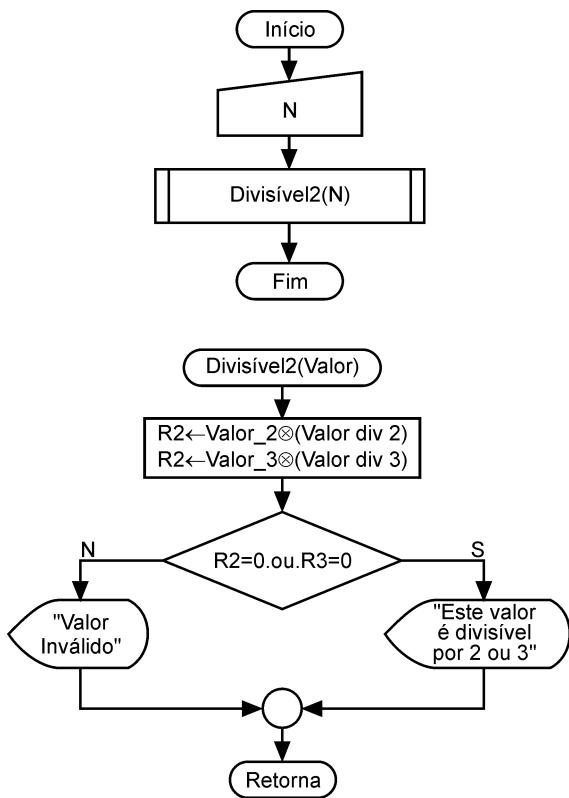
```

```

var
    N : inteiro
início
    leia N
    Divisível1(N)
fim

```

k) Diagramas de blocos



Português estruturado

```

programa Cap10_Ex2k_Pg261
procedimento Divisível2(VALOR : inteiro)
var
    R2, R3 : inteiro
início
    R2 ← VALOR - 2 * (VALOR div 2)
    R3 ← VALOR - 3 * (VALOR div 3)
    se (R2 = 0) .ou. (R3 = 0) então
        escreva "Este valor é divisível por 2 ou 3"
    senão
        escreva "Valor inválido"
    fim_se
fim

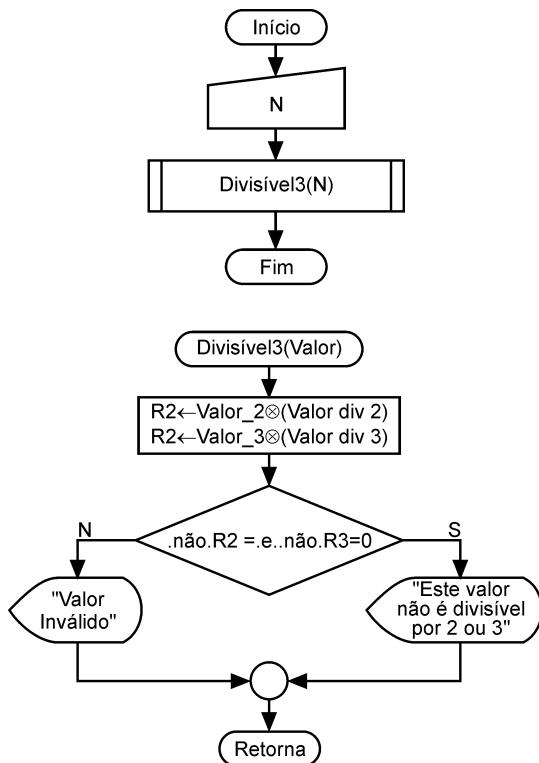
```

```

var
    N : inteiro
início
    leia N
    Divisível2(N)
fim

```

I) Diagramas de blocos



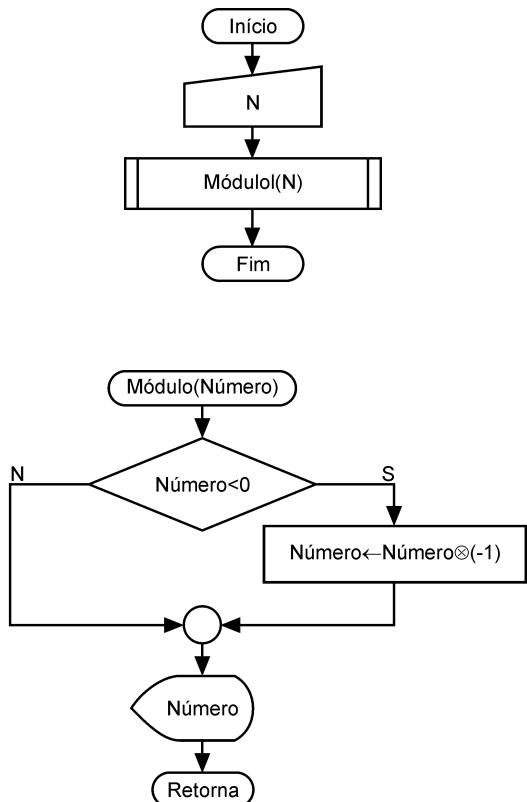
Português estruturado

```

programa Cap10_Ex21_Pg261
procedimento Divisível3(VALOR : inteiro)
var
  R2, R3 : inteiro
início
  R2 ← VALOR - 2 * (VALOR div 2)
  R3 ← VALOR - 3 * (VALOR div 3)
  se .não. (R2 = 0) .e. .não. (R3 = 0) então
    escreva "Este valor não é divisível por 2 e 3"
  senão
    escreva "Valor inválido"
  fim_se
fim

var
  N : inteiro
início
  leia N
  Divisível3(N)
fim
  
```

m) Diagramas de blocos



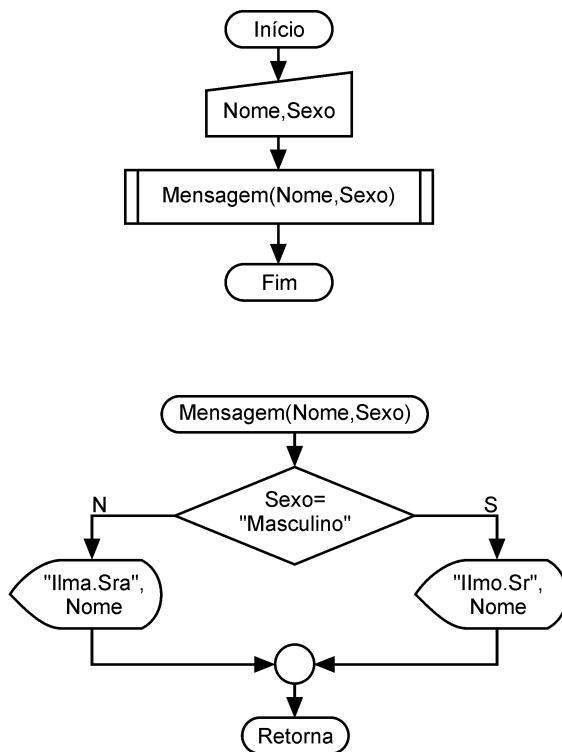
Português Estruturado

```

programa Cap10_Ex2m_Pg261
procedimento Módulo(NÚMERO : inteiro)
início
  se (NÚMERO < 0) então
    NÚMERO ← NÚMERO * (-1)
  fim_se
  escreva NÚMERO
fim

var
  N : inteiro
início
  leia N
  Módulo(N)
fim
  
```

n) Diagramas de blocos



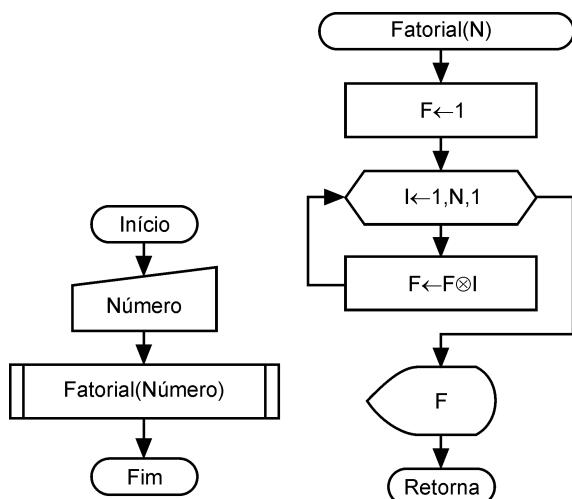
Português estruturado

```

programa Cap10_Ex2n_Pg261
procedimento Mensagem(NOME, SEXO : cadeia)
início
  se (SEXO = "masculino") então
    escreva "Ilmo. Sr. ", NOME
  senão
    escreva "Ilma. Sra. ", NOME
  fim_se
fim

var
  NOME, SEXO : cadeia
início
  leia NOME, SEXO
  Mensagem(NOME, SEXO)
fim
  
```

o) Diagramas de blocos



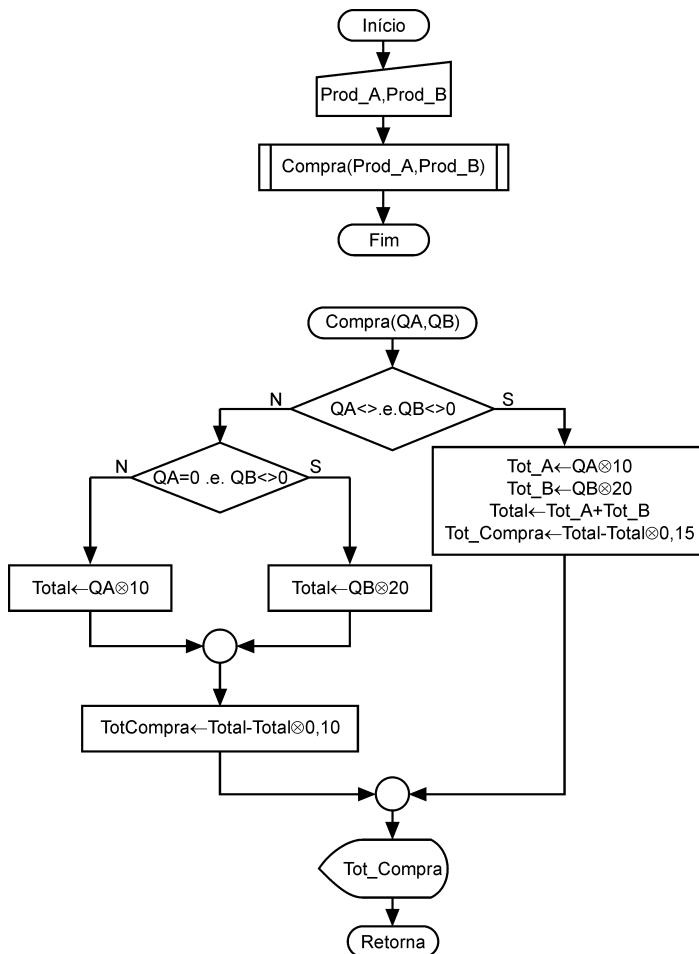
Português estruturado

```

programa Cap10_Ex2o_Pg261
procedimento Fatorial(N : inteiro)
var
  I, F : inteiro
início
  F ← 1
  para I de 1 até N passo 1 faça
    F ← F * I
  fim_para
  escreva F
fim

var
  NÚMERO : inteiro
início
  leia NÚMERO
  Fatorial(NÚMERO)
fim
  
```

p) Diagramas de blocos



Português estruturado

```

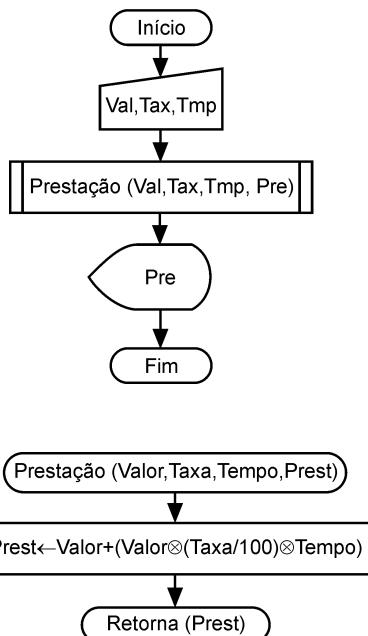
programa Cap10_Ex2p_Pg261

procedimento Compra(QA, QB : inteiro)
var
  TOT_A, TOT_B, TOTAL : inteiro
  TOT_COMPRA : real
início
  se (QA <> 0) .e. (QB <> 0) então
    TOT_A ← QA * 10
    TOT_B ← QB * 20
    TOTAL ← TOT_A + TOT_B
    TOT_COMPRA ← TOTAL - TOTAL * 0.15
  senão
    se (QA = 0) .e. (QB <> 0) então
      TOTAL ← QB * 20
    senão
      TOTAL ← QA * 10
    fim_se
    TOT_COMPRA ← TOTAL - TOTAL * 0.10
  fim_se
  escreva TOT_COMPRA
fim

var
  PROD_A, PROD_B : inteiro
início
  leia PROD_A, PROD_B
  Compra(PROD_A, PROD_B)
fim
  
```

Tópico 10.9 - Exercício 3 - Página 261

a) Diagramas de blocos



Português estruturado

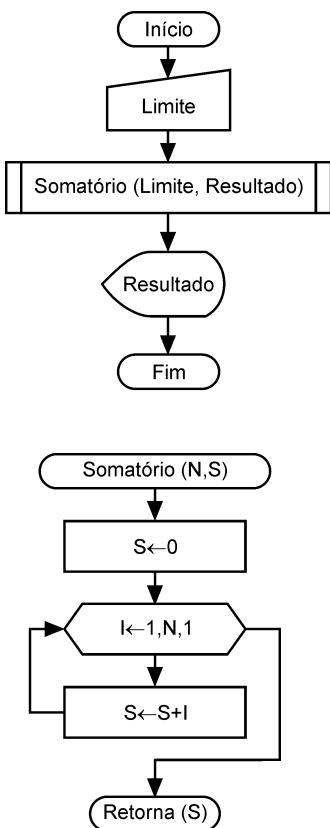
```

programa Cap10_Ex3a_Pg261

procedimento Prestação(VALOR, TAXA, TEMPO : real, var PREST : real)
início
  PREST ← VALOR + (VALOR * (TAXA / 100) * TEMPO)
fim

var
  VAL, TAX, TMP, PRE : real
início
  leia VAL, TAX, TMP
  Prestação(VAL, TAX, TMP, PRE)
  escreva PRE
fim
  
```

b) Diagramas de blocos



Português estruturado

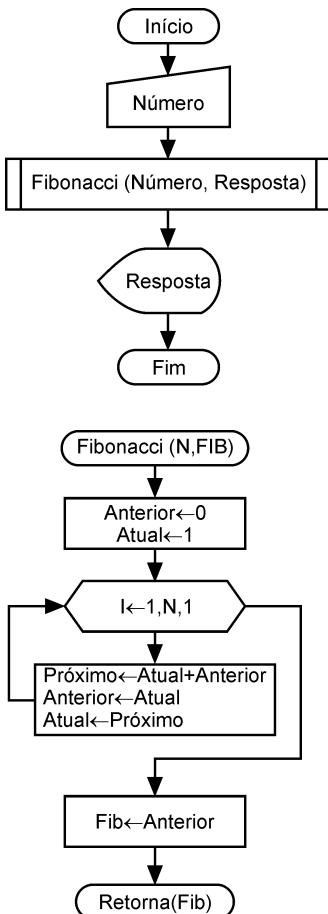
```

programa Cap10_Ex3b_Pg261
procedimento Somatório(N : inteiro, var S : inteiro)
var
    I : inteiro
início
    S ← 0
    para I de 1 até N passo 1 faça
        S ← S + I
    fim_para
fim

var
    LIMITE, RESULTADO : inteiro
início
    leia LIMITE
    Somatório(LIMITE, RESULTADO)
    escreva RESULTADO
fim

```

c) Diagramas de blocos



Português estruturado

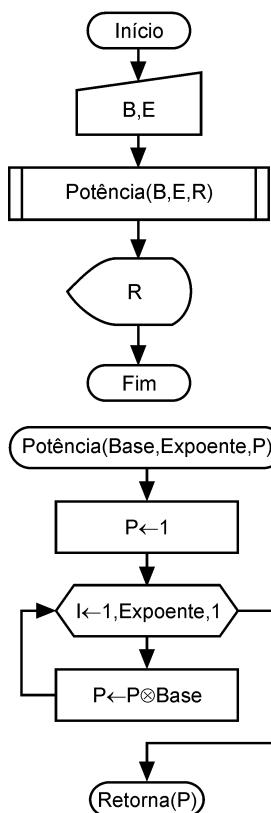
```

programa Cap10_Ex3c_Pg261
procedimento Fibonacci(N : inteiro, var FIB : inteiro)
var
    I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
    ANTERIOR ← 0
    ATUAL ← 1
    para I de 1 até N passo 1 faça
        PRÓXIMO ← ATUAL + ANTERIOR
        ANTERIOR ← ATUAL
        ATUAL ← PRÓXIMO
    fim_para
    FIB ← ANTERIOR
fim

var
    NÚMERO, RESPOSTA : inteiro
início
    leia NÚMERO
    Fibonacci(NÚMERO, RESPOSTA)
    escreva RESPOSTA
fim

```

d) Diagramas de blocos



Português estruturado

```

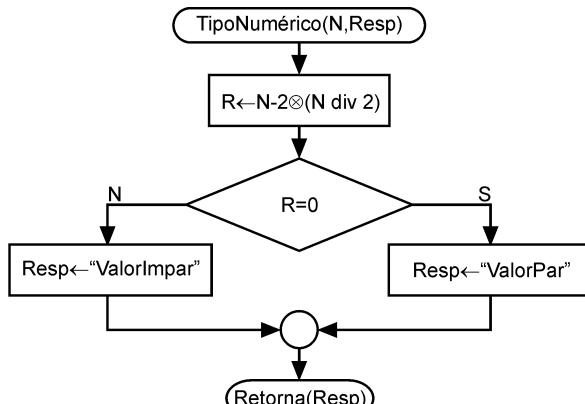
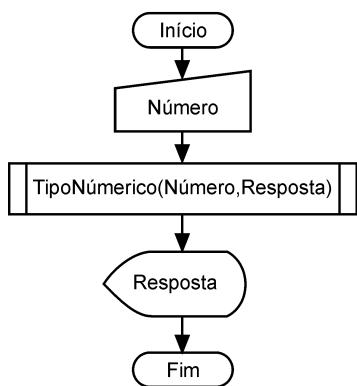
programa Cap10_Ex3d_Pg261

procedimento Potência(BASE, EXPOENTE : inteiro, var P : inteiro)
var
    I : inteiro
início
    P ← 1
    para I de 1 até EXPOENTE passo 1 faça
        P ← P * BASE
    fim_para
fim

var
    B, E, R : inteiro
início
    leia B, E
    Potência(B, E, R)
    escreva R
fim

```

e) Diagramas de blocos



Português Estruturado

```

programa Cap10_Ex3e_Pg261

procedimento TipoNúmero(N : inteiro, var RESP : cadeia)
var
    R : inteiro
início
    R ← N - 2 * (N div 2)
    se (R = 0) então
        RESP ← "Valor par"
    senão
        RESP ← "Valor ímpar"
    fim_se
fim

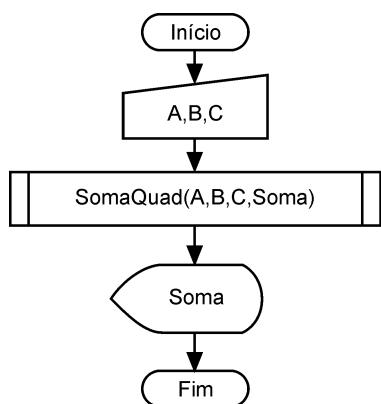
var
    NÚMERO : inteiro
    RESPOSTA : cadeia

```

```

início
    leia NÚMERO
    TipoNumérico(NÚMERO, RESPOSTA)
    escreva RESPOSTA
fim
  
```

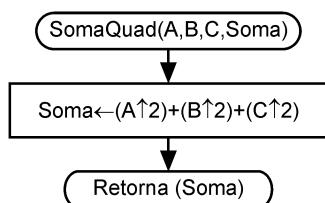
f) Diagramas de blocos



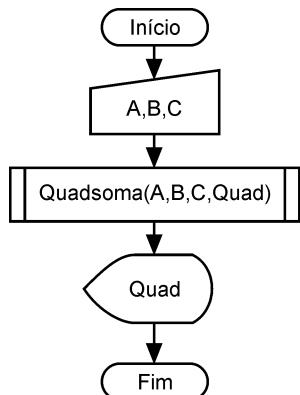
Português estruturado

```

programa Cap10_Ex3f_Pg261
    procedimento Somaquad(A, B, C : inteiro, var SOMA : inteiro)
        início
            SOMA ← (A ↑ 2) + (B ↑ 2) + (C ↑ 2)
        fim
        var
            A, B, C, SOMA : inteiro
        início
            leia A, B, C
            Somaquad(A, B, C, SOMA)
            escreva SOMA
        fim
  
```



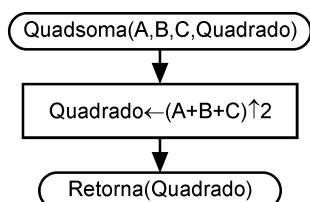
g) Diagramas de blocos



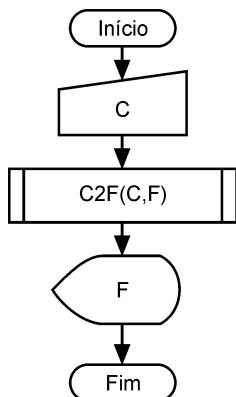
Português estruturado

```

programa Cap10_Ex3g_Pg261
    procedimento Quadsoma(A, B, C : inteiro, var QUADRADO : inteiro)
        início
            QUADRADO ← (A + B + C) ↑ 2
        fim
        var
            A, B, C, QUAD : inteiro
        início
            leia A, B, C
            Quadsoma(A, B, C, QUADRADO)
            escreva QUADRADO
        fim
  
```



h) Diagramas de blocos

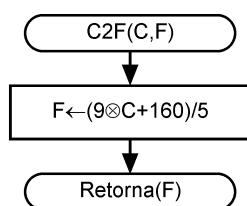


Português estruturado

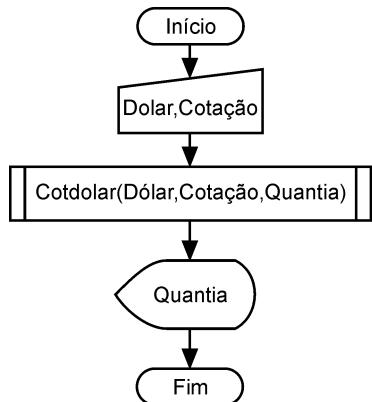
```

programa Cap10_Ex3h_Pg261
  procedimento C2F(C : real, var F : real)
  início
    F ← (9 * C + 160) / 5
  fim

  var
    C, F : real
  início
    leia C
    C2F(C, F)
    escreva F
  fim
  
```



i) Diagramas de blocos

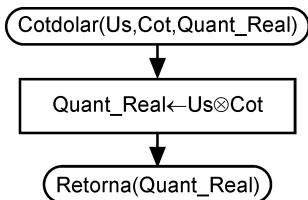


Português estruturado

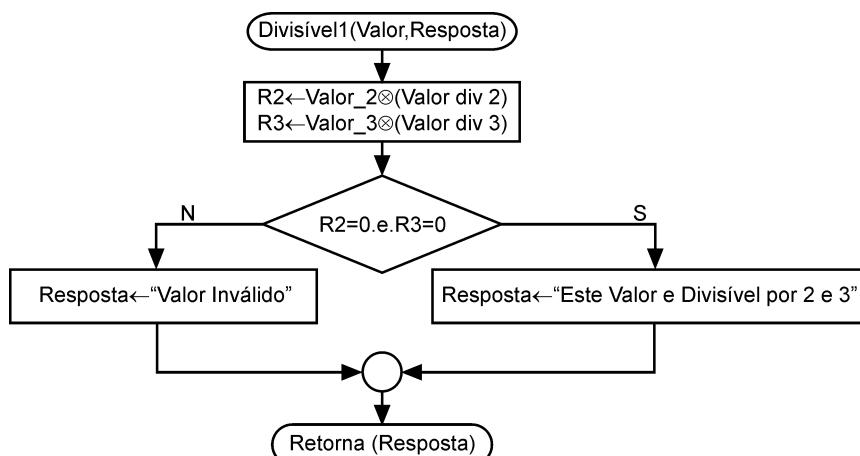
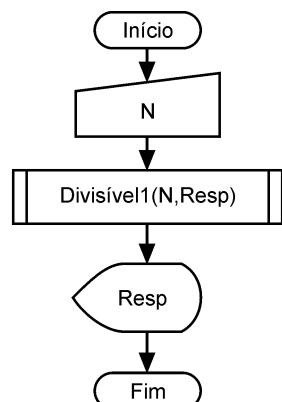
```

programa Cap10_Ex3i_Pg261
  procedimento Cotdolar(US, COT : real, var QUANT_REAL : real)
  início
    QUANT_REAL ← US * COT
  fim

  var
    DOLAR, COTAÇÃO, QUANTIA : real
  início
    leia DOLAR, COTAÇÃO
    Cotdolar(DOLAR, COTAÇÃO, QUANTIA)
    escreva QUANTIA
  fim
  
```



j) Diagramas de Blocos



Português Estruturado

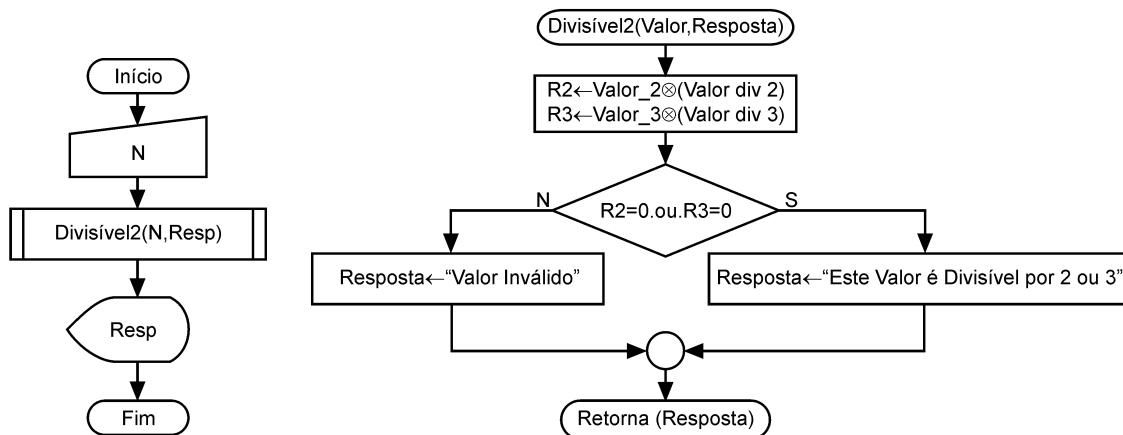
```

programa Cap10_Ex3j_Pg261

procedimento Divisível1(VALOR : inteiro, var RESPOSTA : cadeia)
var
  R2, R3 : inteiro
início
  R2 ← VALOR - 2 * (VALOR div 2)
  R3 ← VALOR - 3 * (VALOR div 3)
  se (R2 = 0) .e. (R3 = 0) então
    RESPOSTA ← "Este valor é divisível por 2 e 3"
  senão
    RESPOSTA ← "Valor inválido"
  fim_se
fim

var
  N : inteiro
  RESP : cadeia
início
  leia N
  Divisível1(N, RESP)
  escreva RESP
fim
  
```

k) Diagramas de Blocos



Português Estruturado

```

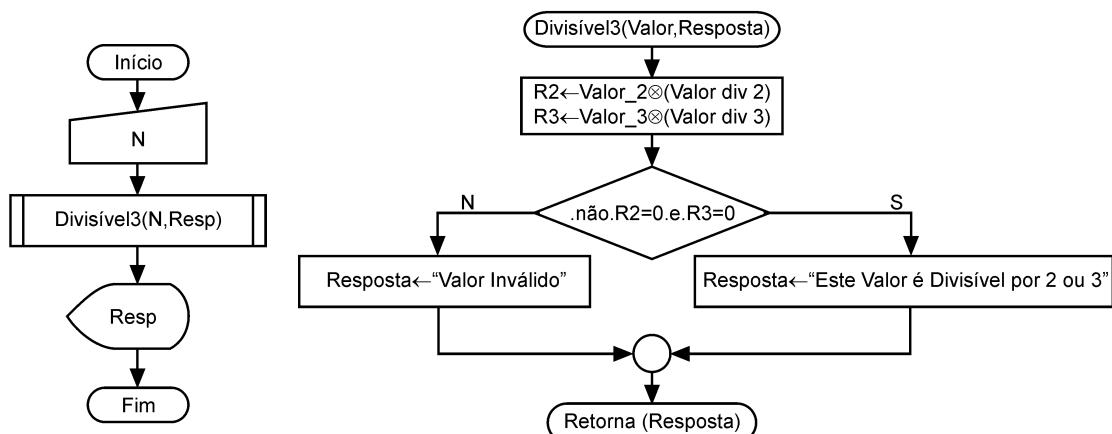
programa Cap10_Ex3k_Pg261

procedimento Divisível2(VALOR : inteiro, var RESPOSTA : cadeia)
var
    R2, R3 : inteiro
início
    R2 ← VALOR - 2 * (VALOR div 2)
    R3 ← VALOR - 3 * (VALOR div 3)
    se (R2 = 0) .ou. (R3 = 0) então
        RESPOSTA ← "Este valor é divisível por 2 ou 3"
    senão
        RESPOSTA ← "Valor inválido"
    fim_se
fim

var
    N : inteiro
    RESP : cadeia
início
    leia N
    Divisível2(N, RESP)
    escreva RESP
fim

```

l) Diagramas de Blocos



Português Estruturado

```

programa Cap10_Ex3l_Pg261

procedimento Divisível3(VALOR : inteiro, var RESPOSTA : cadeia)
var
    R2, R3 : inteiro

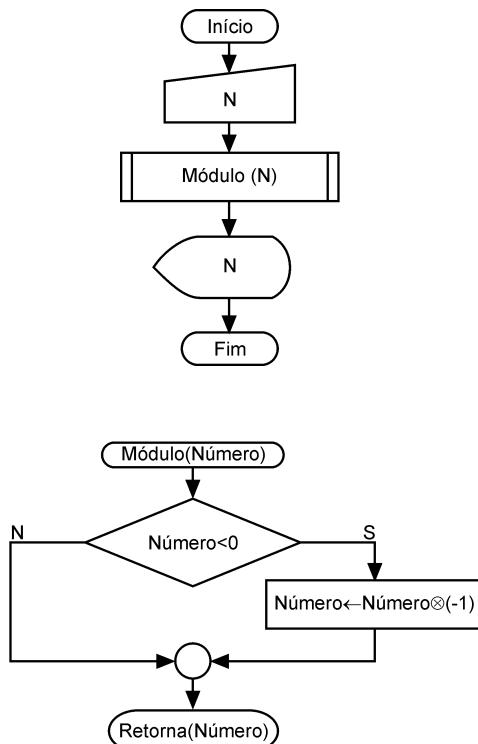
```

```

início
  R2 ← VALOR - 2 * (VALOR div 2)
  R3 ← VALOR - 3 * (VALOR div 3)
  se .não. (R2 = 0) .e. .não. (R3 = 0) então
    RESPOSTA ← "Este valor não é divisível por 2 e 3"
  senão
    RESPOSTA ← "Valor inválido"
  fim_se
fim

var
  N : inteiro
  RESP : cadeia
início
  leia N
  Divisivel3(N, RESP)
  escreva RESP
fim
  
```

m) Diagramas de Blocos



Português Estruturado

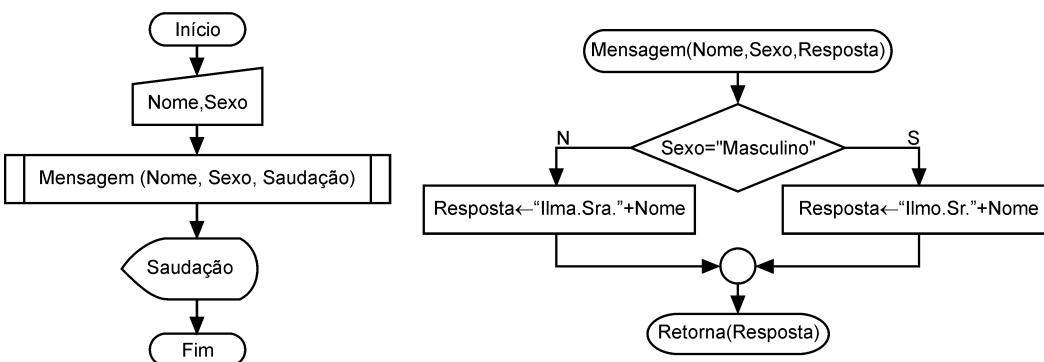
```

programa Cap10_Ex3m_Pg261

procedimento Módulo(var NÚMERO : inteiro)
início
  se (NÚMERO < 0) então
    NÚMERO ← NÚMERO * (-1)
  fim_se
fim

var
  N : inteiro
início
  leia N
  Módulo(N)
  escreva N
fim
  
```

n) Diagramas de Blocos



Português Estruturado

```

programa Cap10_Ex3n_Pg261

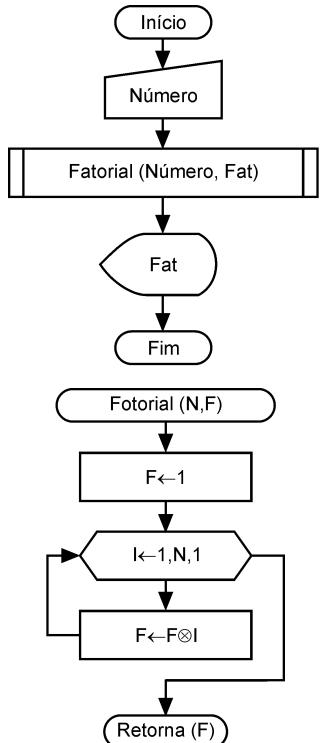
procedimento Mensagem(NOME, SEXO : cadeia, var RESPOSTA : cadeia)
  
```

```

início
  se (SEXO = "masculino") então
    RESPOSTA ← "Ilmo. Sr. " + NOME
  senão
    RESPOSTA ← "Ilma. Sra. " + NOME
  fim_se
fim

var
  NOME, SEXO, SAUDAÇÃO : cadeia
início
  leia NOME, SEXO
  Mensagem(NOME, SEXO, SAUDAÇÃO)
  escreva SAUDAÇÃO
fim
  
```

o) Diagramas de Blocos



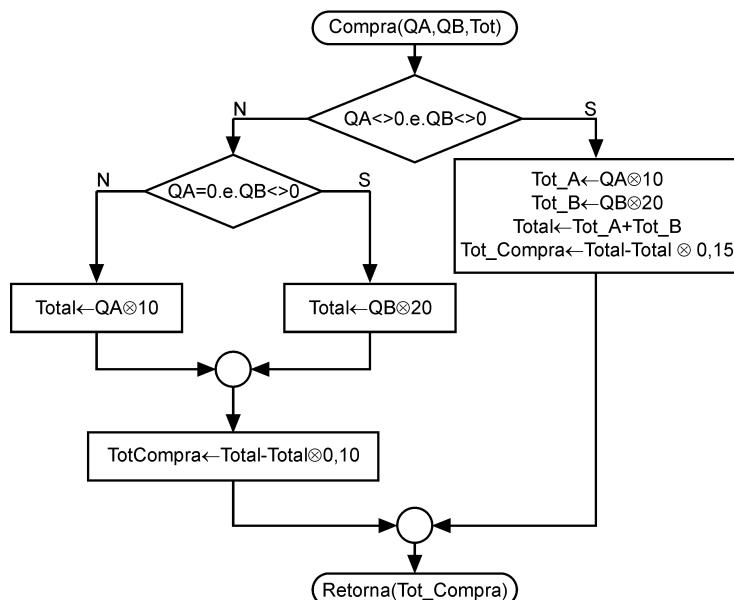
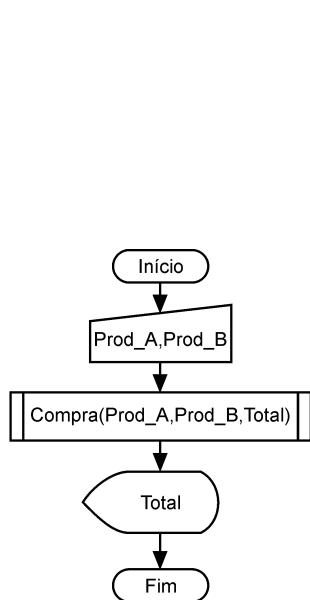
Português Estruturado

```

programa Cap10_Ex3o_Pg261
  procedimento Fatorial(N : inteiro, var F : inteiro)
    var
      I : inteiro
    início
      F ← 1
      para I de 1 até N passo 1 faça
        F ← F * I
      fim_para
    fim

    var
      NÚMERO, FAT : inteiro
    início
      leia NÚMERO
      Fatorial(NÚMERO, FAT)
      escreva FAT
    fim
  
```

p) Diagramas de Blocos



Português Estruturado

```

programa Cap10_Ex3p_Pg261

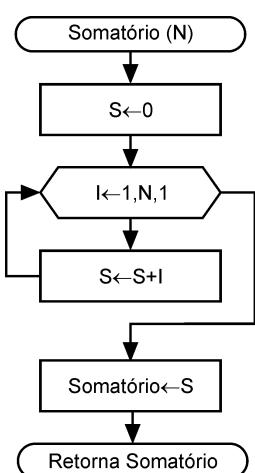
procedimento Compra(QA, QB : inteiro, var TOT : real)
var
    TOT_A, TOT_B, TOTAL : inteiro
    TOT_COMPRA : real
início
    se (QA <> 0) .e. (QB <> 0) então
        TOT_A ← QA * 10
        TOT_B ← QB * 20
        TOTAL ← TOT_A + TOT_B
        TOT_COMPRA ← TOTAL - TOTAL * 0.15
    senão
        se (QA = 0) .e. (QB <> 0) então
            TOTAL ← QB * 20
        senão
            TOTAL ← QA * 10
        fim_se
        TOT_COMPRA ← TOTAL - TOTAL * 0.10
    fim_se
    TOT ← TOT_COMPRA
fim

var
    PROD_A, PROD_B : inteiro
    TOTAL : real
início
    leia PROD_A, PROD_B
    Compra(PROD_A, PROD_B, TOTAL)
    escreva TOTAL
fim

```

Tópico 10.9 - Exercício 4 - Página 262

a) Diagramas de Blocos



Português Estruturado

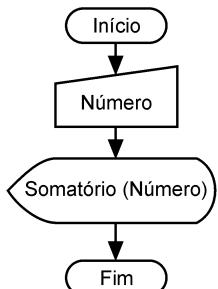
```

programa Cap10_Ex4a_Pg262

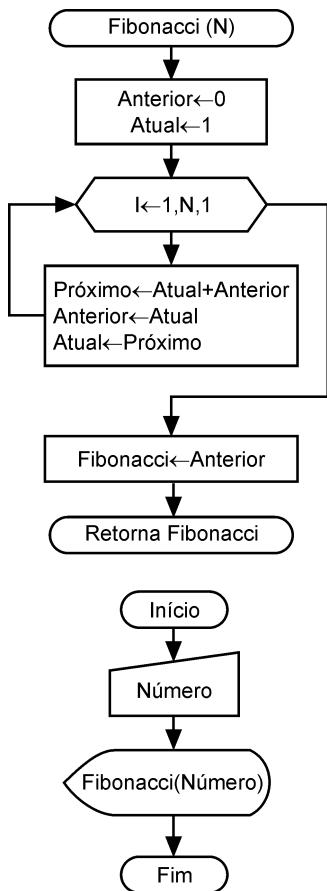
função Somatório(N : inteiro) : inteiro
var
    S, I : inteiro
início
    S ← 0
    para I de 1 até N passo 1 faça
        S ← S + I
    fim_para
    Somatório ← S
fim

var
    NÚMERO : inteiro
início
    leia NÚMERO
    escreva Somatório(NÚMERO)
fim

```



b) Diagramas de Blocos



Português Estruturado

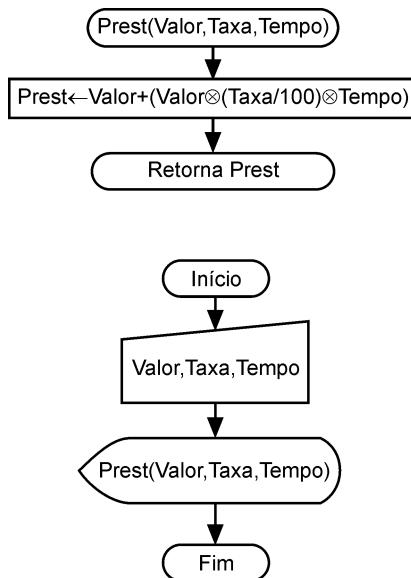
```

programa Cap10_Ex4b_Pg262

função Fibonacci(N : inteiro) : inteiro
var
  I, ATUAL, ANTERIOR, PRÓXIMO : inteiro
início
  ANTERIOR ← 0
  ATUAL ← 1
  para I de 1 até N passo 1 faça
    PRÓXIMO ← ATUAL + ANTERIOR
    ANTERIOR ← ATUAL
    ATUAL ← PRÓXIMO
  fim_para
  Fibonacci ← ANTERIOR
fim

var
  NÚMERO : inteiro
início
  leia NÚMERO
  escreva Fibonacci(NÚMERO)
fim
  
```

c) Diagramas de Bloco



Português Estruturado

```

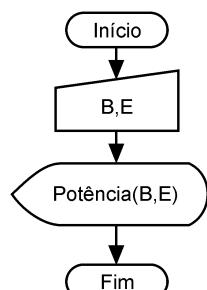
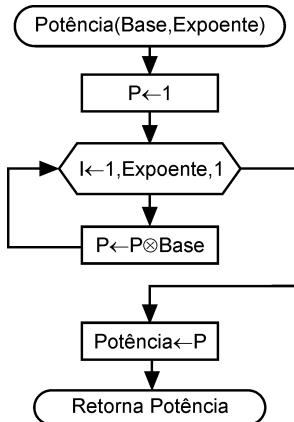
programa Cap10_Ex4c_Pg262

função Prest(VALOR, TAXA, TEMPO : real) : real
início
  Prest ← VALOR + (VALOR * (TAXA / 100) * TEMPO)
fim

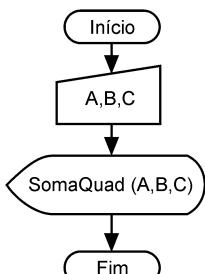
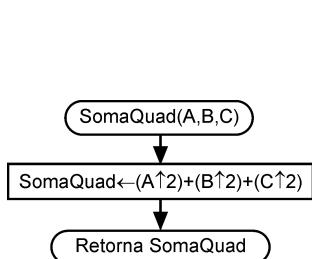
var
  VALOR, TAXA, TEMPO : real
início
  leia VALOR, TAXA, TEMPO
  escreva Prest(VALOR, TAXA, TEMPO)
fim
  
```

d) Diagramas de Blocos

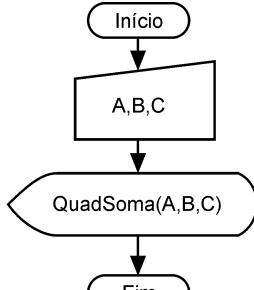
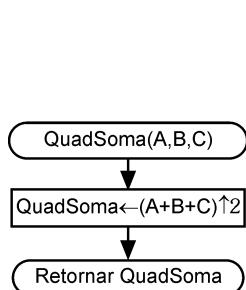
Português Estruturado



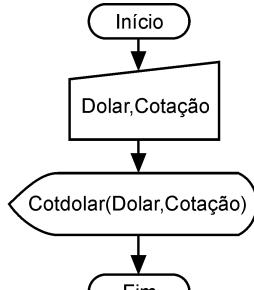
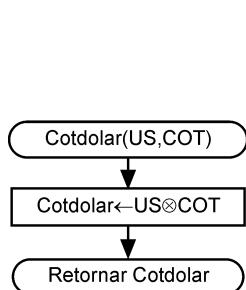
e) Diagramas de Bloco



f) Diagramas de Blocos



g) Diagramas de Blocos



```
programa Cap10_Ex4d_Pg262
```

```

    função Potência(BASE, EXPOENTE : inteiro) : inteiro
    var
        I, P : inteiro
    início
        P ← 1
        para I de 1 até EXPOENTE passo 1 faça
            P ← P * BASE
        fim_para
        Potência ← P
    fim

    var
        B, E : inteiro
    início
        leia B, E
        escreva Potência(B, E)
    fim

```

Português Estruturado

```
programa Cap10_Ex4e_Pg262
```

```

    função SomaQuad(A, B, C : inteiro) : inteiro
    início
        SomaQuad ← (A ↑ 2) + (B ↑ 2) + (C ↑ 2)
    fim

    var
        A, B, C : inteiro
    início
        leia A, B, C
        escreva SomaQuad(A, B, C)
    fim

```

Português Estruturado

```
programa Cap10_Ex4f_Pg262
```

```

    função Quadsoma(A, B, C : inteiro) : inteiro
    início
        Quadsoma ← (A + B + C) ↑ 2
    fim

    var
        A, B, C : inteiro
    início
        leia A, B, C
        escreva Quadsoma(A, B, C)
    fim

```

Português Estruturado

```
programa Cap10_Ex4g_Pg262
```

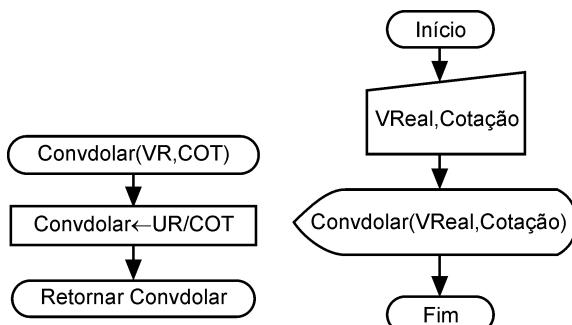
```

    função Cotdolar(US, COT : real) : real
    início
        Cotdolar ← US * COT
    fim

    var
        DOLAR, COTAÇÃO : real
    início
        leia DOLAR, COTAÇÃO
        escreva Cotdolar(DOLAR, COTAÇÃO)
    fim

```

h) Diagramas de Bloco



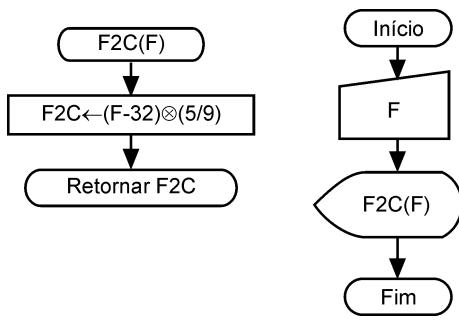
Português Estruturado

```

programa Cap10_Ex4h_Pg262
  função Convdolar(VR, COT : real) : real
  início
    Convdolar ← VR / COT
  fim

  var
    VREAL, COTAÇÃO : real
  início
    leia VREAL, COTAÇÃO
    escreva Convdolar(VREAL, COTAÇÃO)
  fim
  
```

i) Diagramas de Bloco



Português Estruturado

```

programa Cap10_Ex4i_Pg262
  função F2C(F : real) : real
  início
    F2C ← (F - 32) * (5 / 9)
  fim

  var
    F : real
  início
    leia F
    escreva F2C(F)
  fim
  
```