Lógica e Design de Programação: Introdução

Capítulo 6

Arrays

Objetivos

- Entendendo arrays e como eles ocupam a memória do computador
- Manipulando um array para substituir decisões embutidas
- Usando uma constante nomeada para referir-se à dimensão de um array
- Declaração e inicialização de arrays
- Arrays variáveis e constantes

Objetivos (cont.)

- Procurando em um array por um casamento exato
- Usando arrays paralelos
- Buscando em um array por um casamento de faixas
- Permanecendo dentro dos limites dos arrays
- Usando um loop `for` para processar arrays

Entendendo arrays e como ocupam a memória do computador

- **Array**:
  - Uma série ou lista de variáveis na memória do computador
  - Todas elas possuem o mesmo nome e tipo de dado
  - Estes são diferenciadas por números especiais chamados subscritos
- **Subscrito** (**índice**):
  - Um número que indica a posição de um item específico dentro de um array
  - Subscritos são sempre uma sequência de inteiros

Como arrays ocupam a memória do computador

- Cada variável dentro de um array tem o mesmo nome e o mesmo tipo de dado
- **Elemento**: cada variável separada do array
- Cada elemento do array ocupa uma área próxima ou vizinha aos demais elementos do array na memória
- **Dimensão do array**: número de elementos que um array conterá

Como arrays ocupam a memória do computador (cont.)

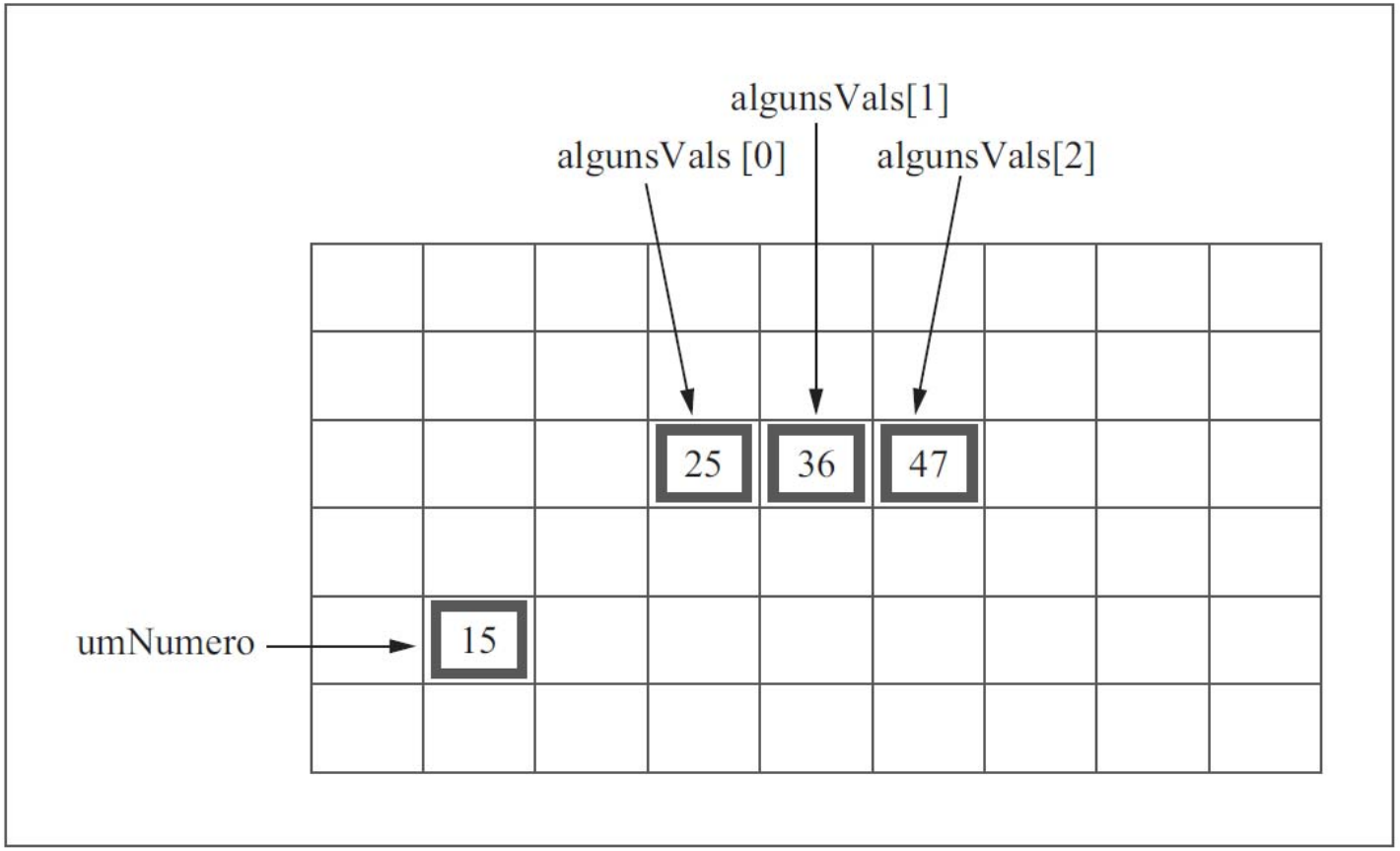


Figura 6-1 Aparência de um array de três elementos e de uma variável na memória do computador

Como arrays ocupam a memória do computador (cont.)

- Todos os elementos do array têm o mesmo nome de grupo
  - Cada elemento individualmente tem também um subscrito único
  - Indica o quanto ele dista do primeiro elemento
  - Qualquer subscrito de arrays é sempre uma sequência de inteiros
- Dependendo das regras da sintaxe da linguagem de programação usada, coloca-se o subscrito dentro de parênteses ou de colchetes depois do nome do grupo

Manipulando um array para substituir decisões embutidas

- Exemplo: Departamento de Recursos Humanos – estatísticas
  - Lista o número de empregados que declararam 0, 1, 2, 3, 4 ou 5 dependentes
    - Assuma que nenhum empregado tenha mais do que cinco dependentes
- Aplicação que produz as contas para as seis categorias dependentes
  - Usando uma série de decisões
- Aplicação não aumenta a escala para mais dependentes

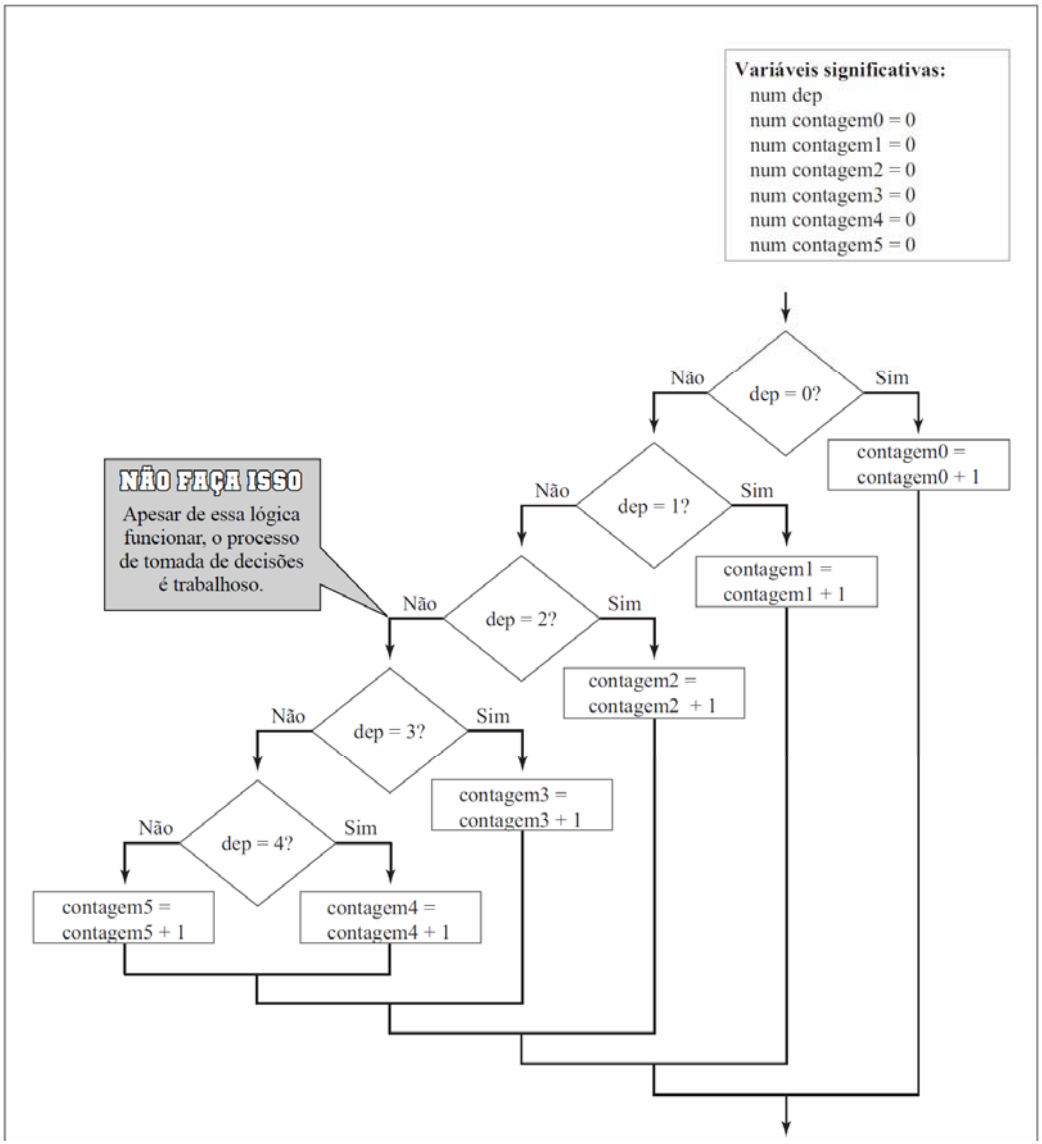


Figura 6-3 Fluxograma e pseudocódigo de processo de tomada de decisões usando uma série de decisões – o jeito difícil

```
se dep = 0 então
    contagem0 = contagem0 + 1
senão
    se dep = 1 então
       contagem1 = contagem1 + 1
    senão
       se dep = 2 então
          contagem2 = contagem2 + 1
       senão
          se dep = 3 então
             contagem3 = contagem 3 + 1
          senão
             se dep = 4 então
                contagem4 = contagem4 + 1
             senão
                contagem5 = contagem5 + 1
             fim-se
          fim-se
       fim-se
    fim-se
fim-se
```

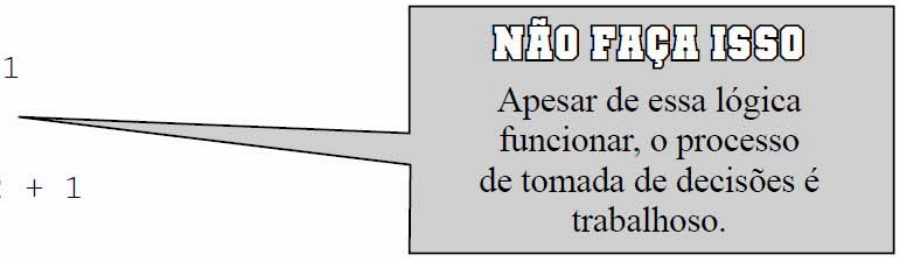


Figura 6-3 Fluxograma e pseudocódigo de processo de tomada de decisões usando uma série de decisões – o jeito difícil (cont.)

Manipulando um array para substituir decisões embutidas (cont.)

- O uso de um array reduz consideravelmente o número de sentenças necessárias
- As seis contas acumuladoras de dependentes podem ser redefinidas em um único array
- Variável como subscrito para o array

Manipulando um array para substituir decisões embutidas (cont.)

- Essa variável poderia ser nomeada `dependentes`, `sub`, `índex` ou qualquer outro identificador lícito e que fosse usado como subscrito para o array, contanto que ele fosse:
  - Numérico sem casas decimais
  - Iniciado em 0
  - Incrementado em 1 a cada vez que a lógica passasse pelo loop

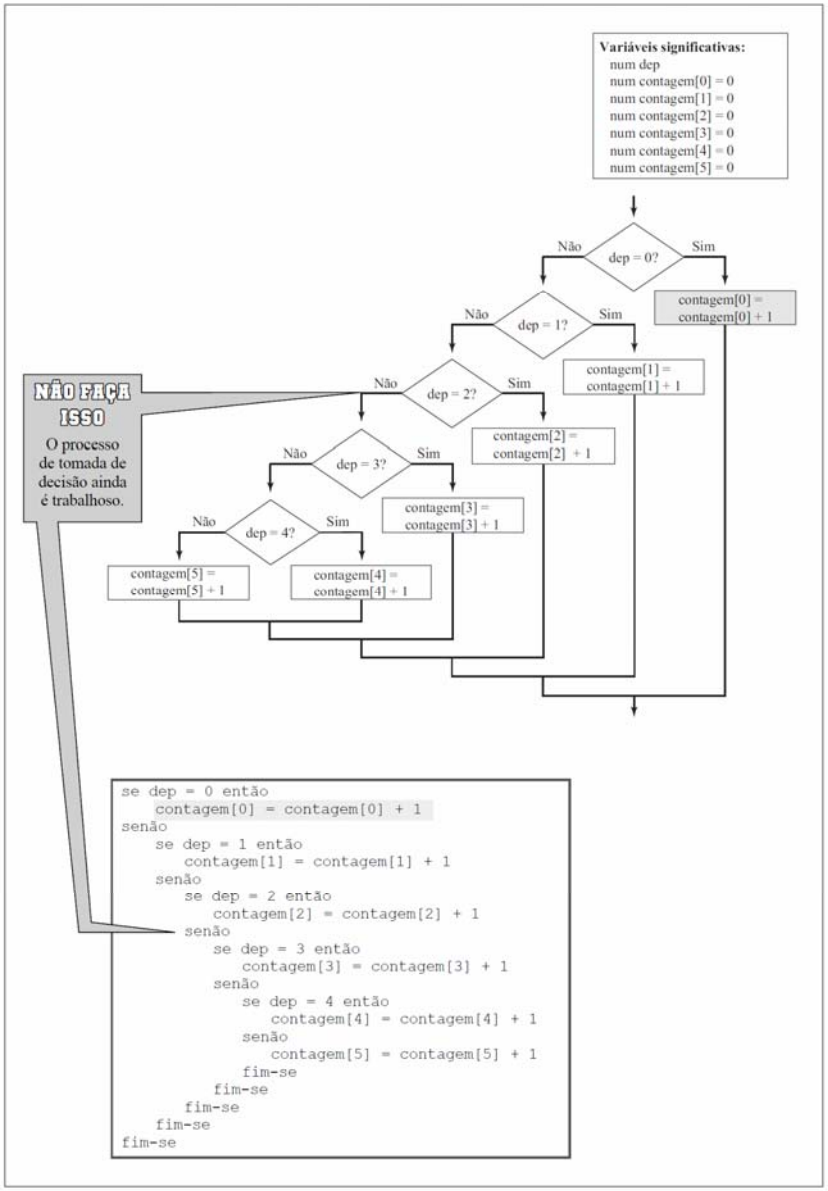


Figura 6-4 Fluxograma e pseudocódigo de processo de tomada de decisão – ainda do jeito difícil

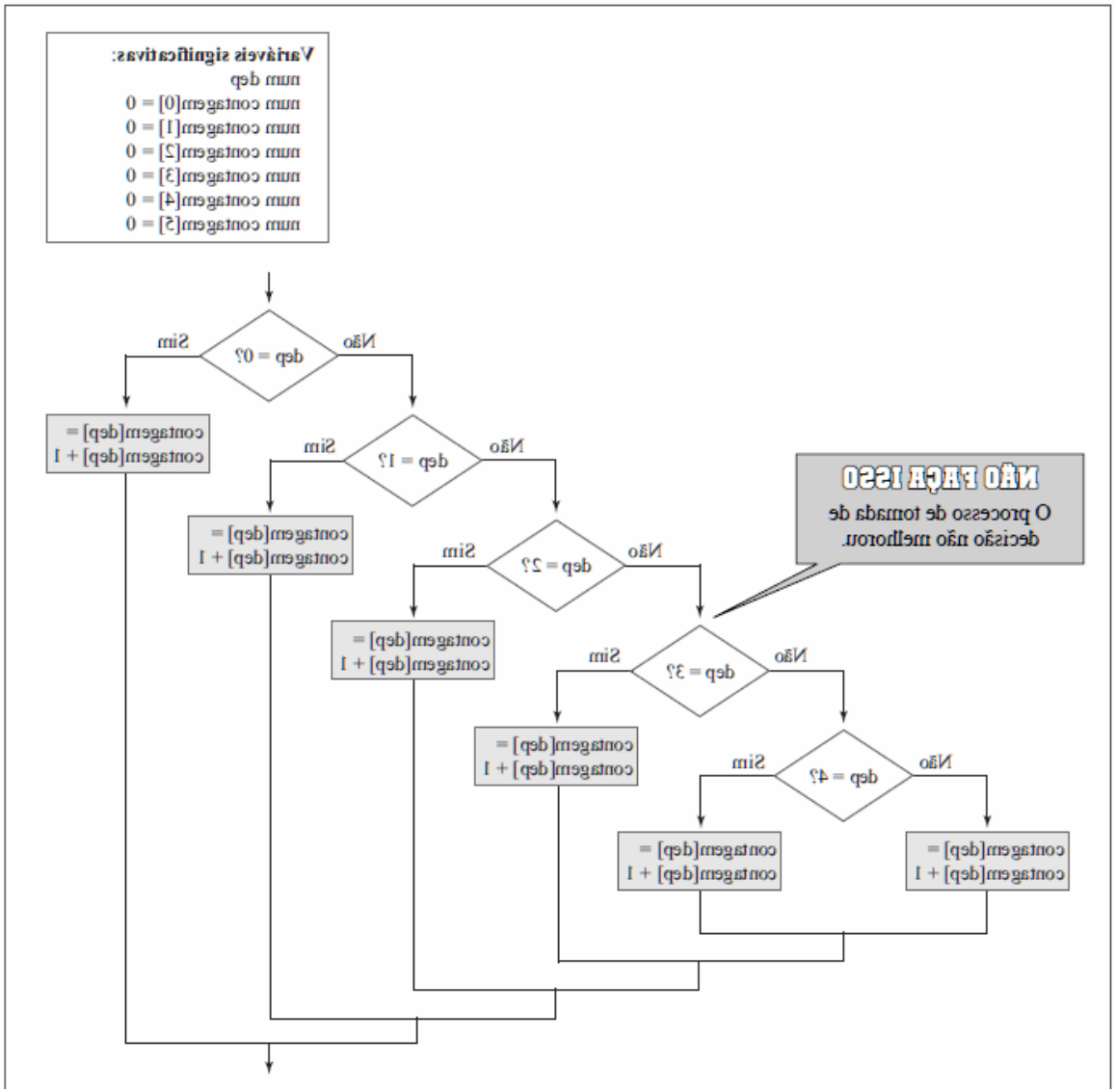


Figura 6-5 Fluxograma e pseudocódigo de processo de tomada de decisão usando um array – ainda de um jeito difícil

Manipulando um array para substituir decisões embutidas (cont.)

```
se dep = 0 então
    contagem[dep] = contagem[dep] + 1
senão
    se dep = 1 então
       contagem[dep] = contagem[dep] + 1
    senão
       se dep = 2 então
          contagem[dep] = contagem[dep] + 1
       senão
          se dep = 3 então
             contagem[dep] = contagem[dep] + 1
          senão
             se dep = 4 então
                contagem[dep] = contagem[dep] + 1
             senão
                contagem[dep] = contagem[dep] + 1
             fim-se
          fim-se
       fim-se
    fim-se
fim-se
```

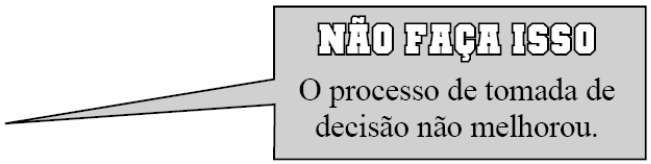


Figura 6-5 Fluxograma e pseudocódigo de processo de tomada de decisão usando um array – ainda de um jeito difícil (cont.)

Manipulando um array para substituir decisões embutidas (cont.)

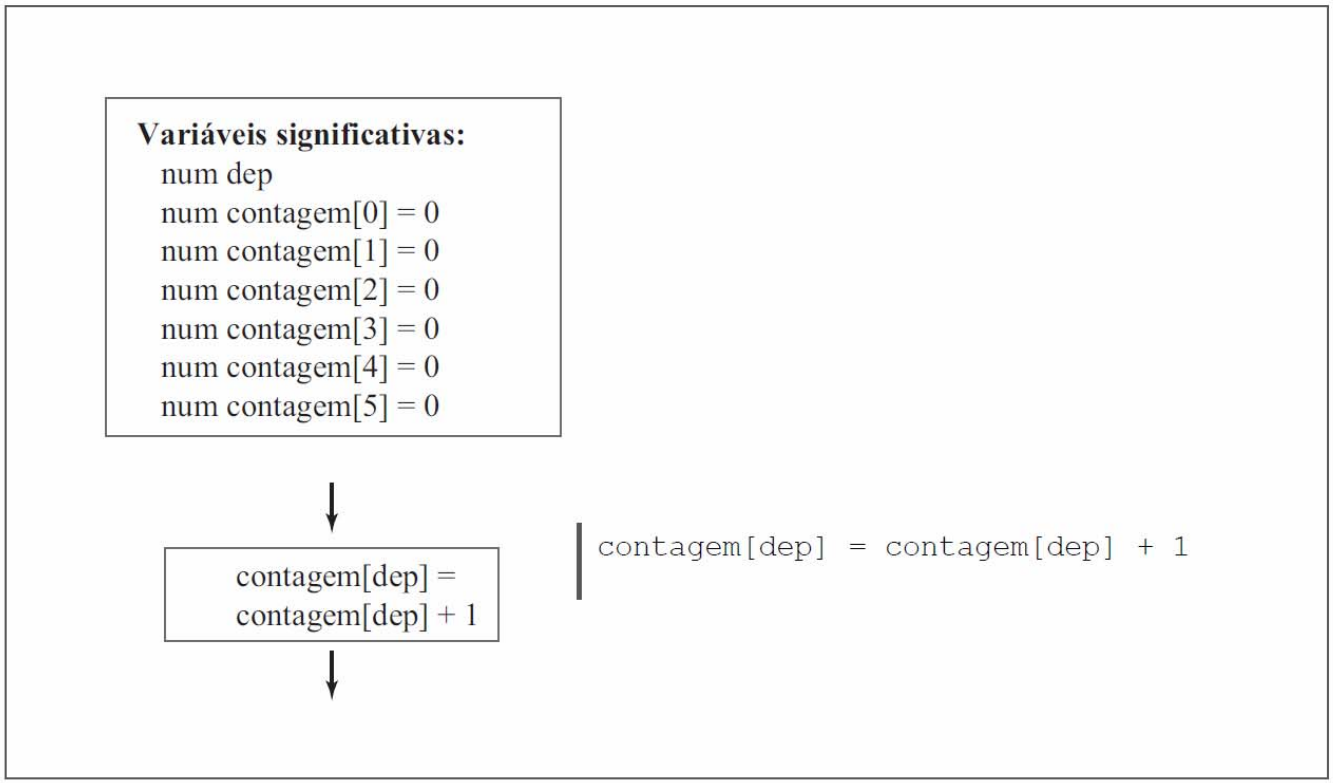


Figura 6-6 Fluxograma e pseudocódigo de processo eficiente de tomada de decisão usando um array

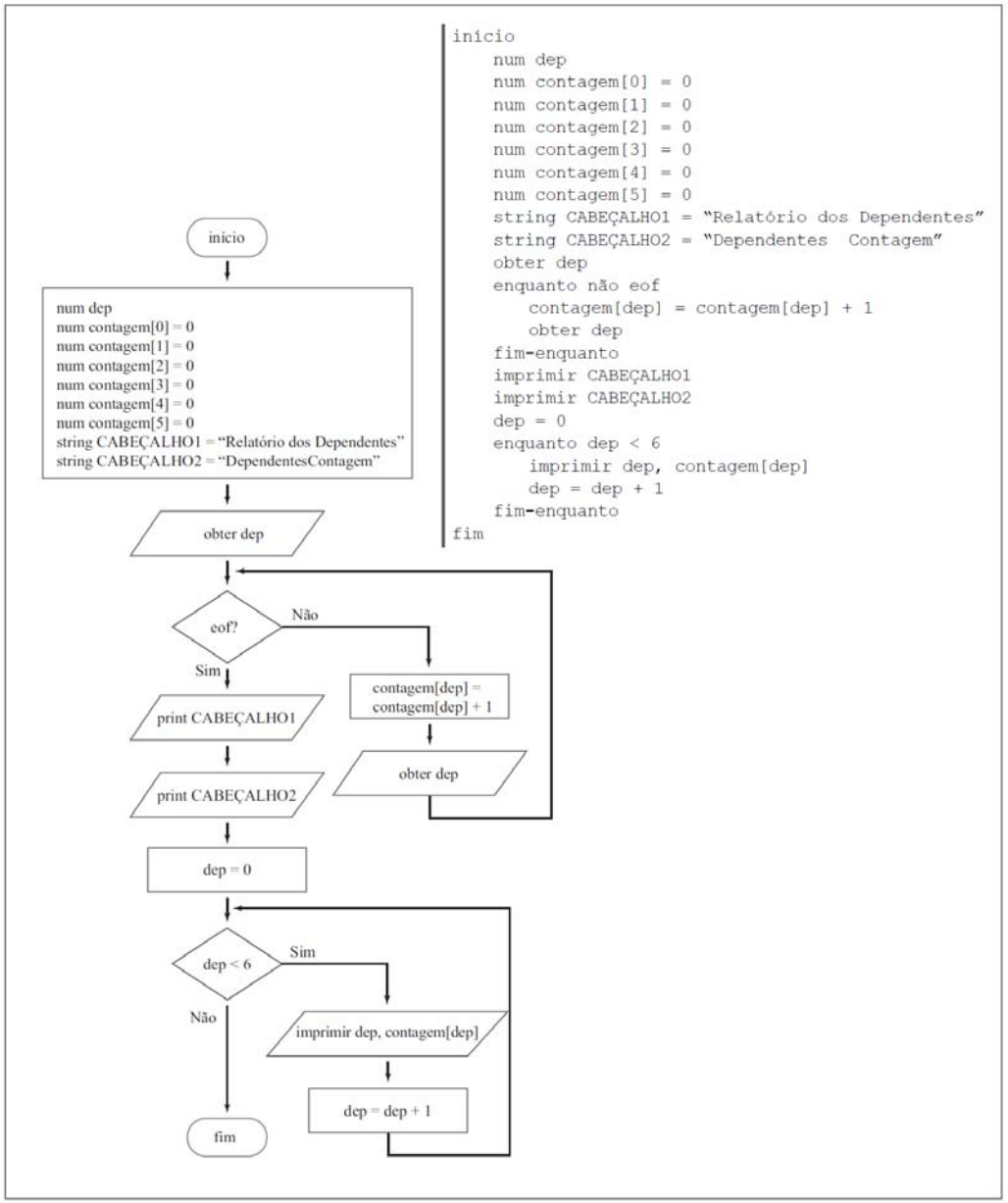


Figura 6-7 Fluxograma e pseudocódigo para o programa de relatório de dependentes

Usando uma constante nomeada para referir-se à dimensão de um array

- Evitar “números mágicos” (constantes não nomeadas)
- Declarar uma constante numérica nomeada para ser usada toda vez que um array for acessado
- Se assegurar de que qualquer subscrito usado continue menor que o valor constante
- Ao declarar um array, uma constante que representa a dimensão do array é automaticamente criada, em muitas linguagens

Declaração e inicialização de arrays

- Declarações em linguagens diferentes têm duas coisas em comum:
  - Nomeiam o array contagem
  - Indicam que existirão 20 elementos numéricos separados

Linguagem de Programação	Declaração de um Array de 20 elementos
Visual Basic	`Dim Contagem[20] As Integer`
Cnº, C++	`int contagem[20]`
Java	`int[] contagem = new int[20]`

Tabela 6-1 Declarar um array de 20 elementos nomeados `contagem` em diversas linguagens comuns

Declaração e inicialização de arrays (cont.)

- Inicializar os elementos do array

```
num contagem[20] = 0
```

- Fazer atribuições individuais

```
contagem[0] = 5
```

- A maioria das linguagens de programação permite que sejam atribuídos menos valores do que os elementos declarados do array
- **Loop de inicialização:** uma estrutura de loop que fornece valores iniciais para todos os elementos de qualquer array
- Use um loop para ajustar todos os elementos do array

Declaração e inicialização de arrays (cont.)

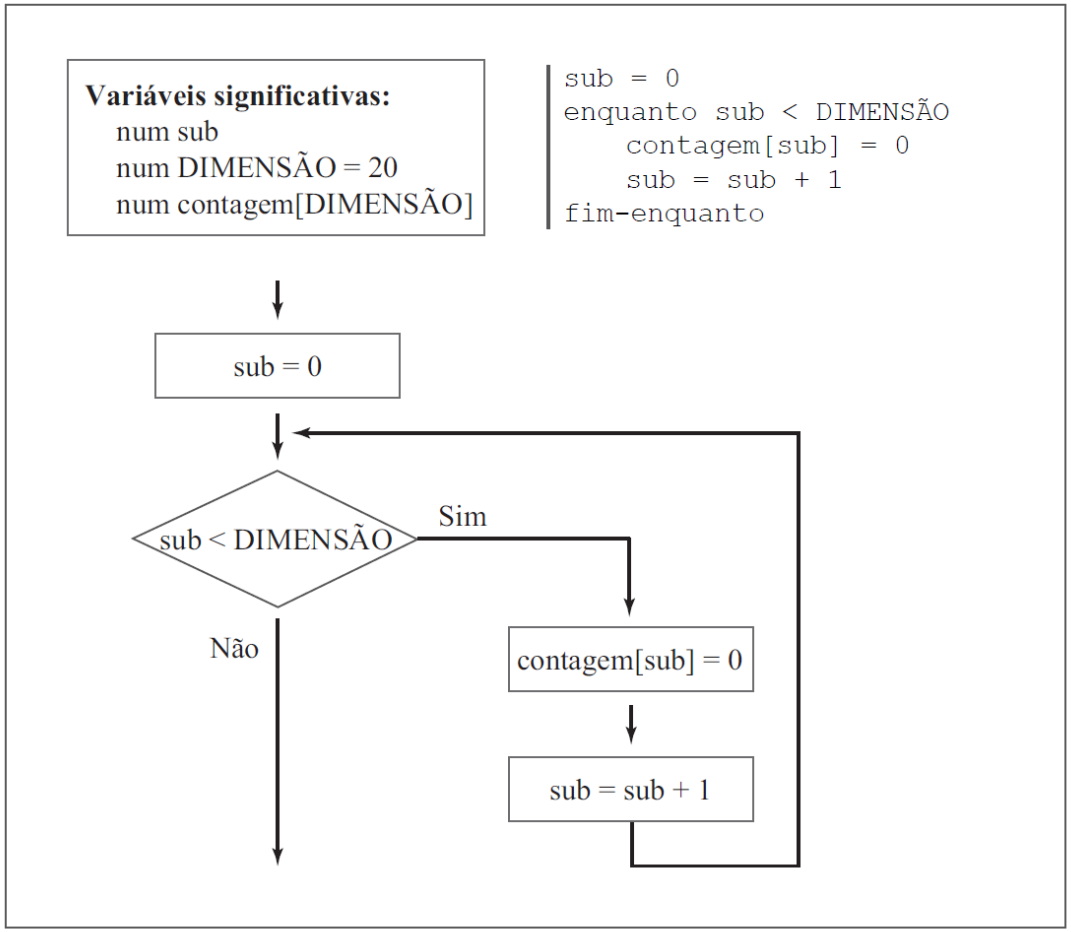


Figura 6-8 Um loop que ajusta valores para todos os elementos de um array

Arrays variáveis e constantes

- **Array variável:** seus valores mudam durante a execução do programa
  - Os valores desejados são criados durante a própria execução da aplicação
- **Array constante**: é possível atribuir a ele valores finais e permanentes ao escrever o código do programa
- **Valores definidos por código-fonte** são explicitamente atribuídos para os elementos do array

Piso	Aluguel em $
0 (subsolo)	350
1	400
2	600
3 (cobertura)	1000

Figura 6-9 Aluguéis por piso

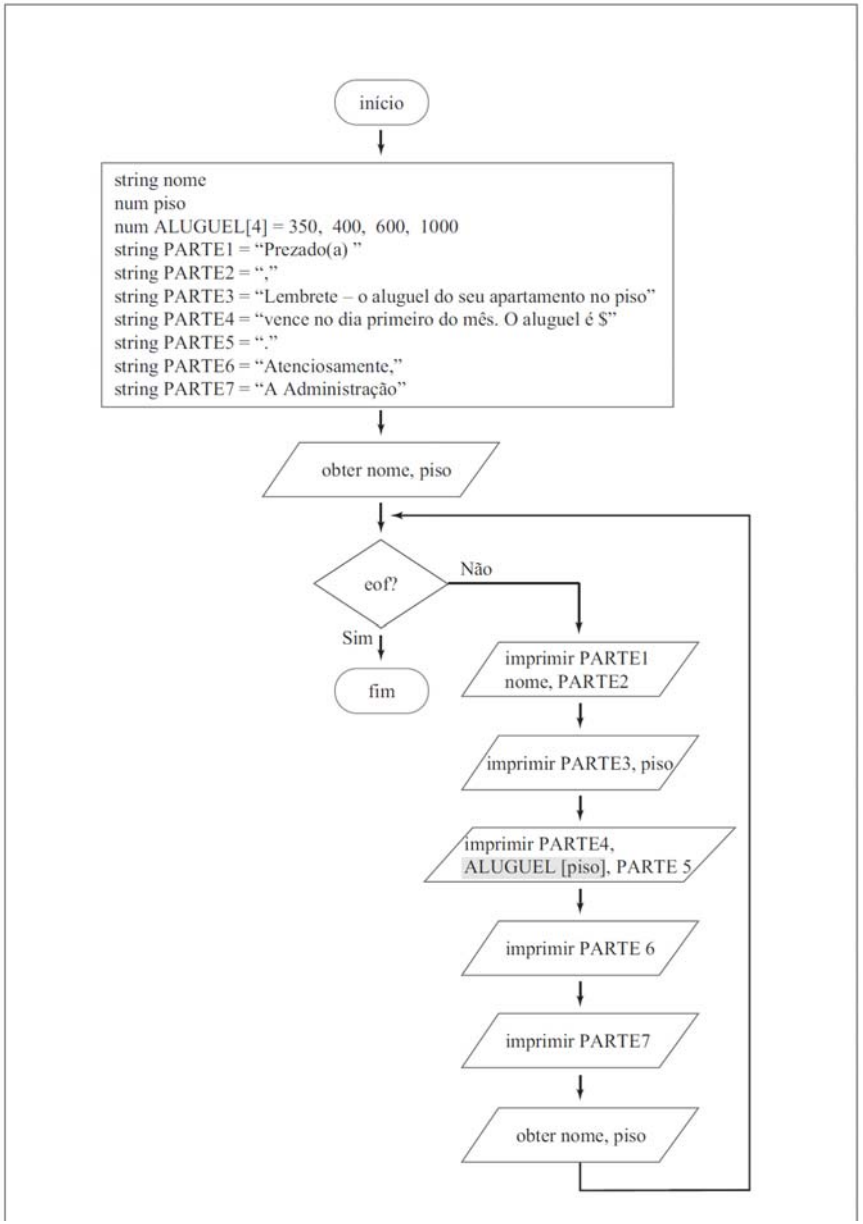


Figura 6-11 Programa que produz as cartas aos inquilinos

Arrays variáveis e constantes (cont.)

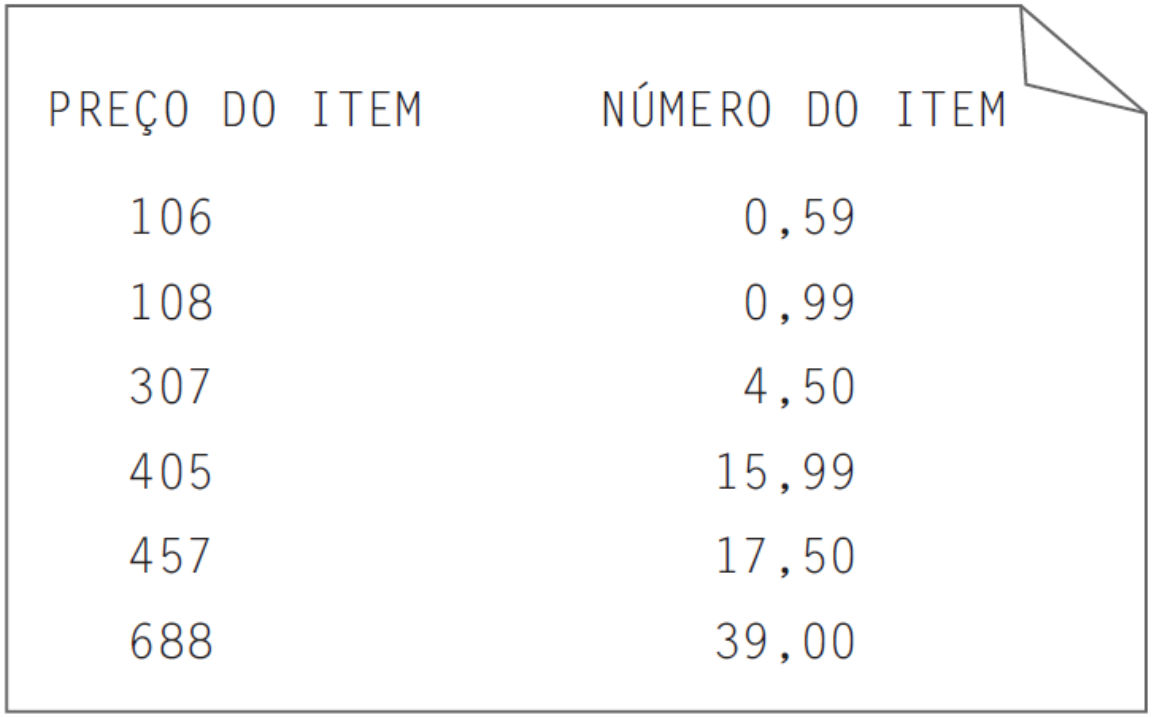
```
início
    string nome
    num piso
    num ALUGUEL[4] = 350,  400,  600,  1000
    string PARTE1 = "Prezado(a) "
    string PARTE2 = ","
    string PARTE3 = "Lembrete - o aluguel do seu apartamento no piso "
    string PARTE4 = "vence no dia primeiro do mês. O aluguel é $"
    string PARTE5 = "."
    string PARTE6 = "Atenciosamente,"
    string PARTE7 = "A Administração"
    obter nome, piso
    enquanto não eof
       imprimir PARTE1, nome, PARTE2
       imprimir PARTE3, piso
       imprimir PARTE4, ALUGUEL[piso], PARTE5
       imprimir PARTE6
       imprimir PARTE7
       obter nome, piso
    fim-enquanto
fim
```

Figura 6-11 Programa que produz as cartas aos inquilinos

Procurando um casamento exato em um array

- Às vezes, é necessário procurar no array para encontrar o valor que você precisa
- Exemplo: serviço de vendas por reembolso postal
  - Os códigos dos itens sejam números de três dígitos, não necessariamente consecutivos
  - Quando o cliente encomenda um item, verificar se o número do item é válido
  - Crie um array com número de itens válidos
  - Procure o array pela combinação exata

Procurando um casamento exato em um array (cont.)



PREÇO DO ITEM	NÚMERO DO ITEM
106	0,59
108	0,99
307	4,50
405	15,99
457	17,50
688	39,00

Figura 6-12 Itens disponíveis em empresa de vendas por reembolso postal

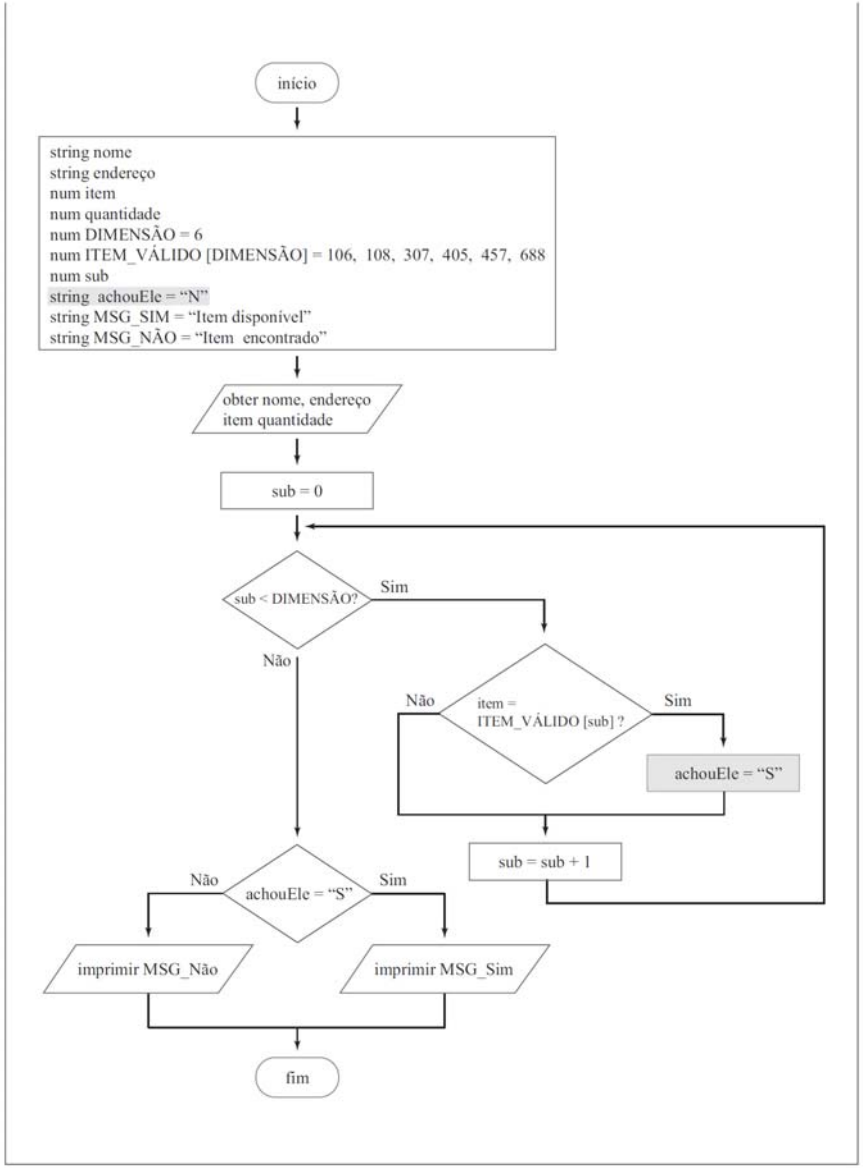


Figura 6-13 Fluxograma e pseudocódigo para programa que verifica a disponibilidade de itens

```
início
    string nome
    string nome
    string endereço
    num item
    num quantidade
    num DIMENSÃO = 6
    num ITEM_VÁLIDO[DIMENSÃO] = 106,  108,  307,  405,  457,  688
    num sub
    string achouEle = "N"
    string MSG_SIM = "Item disponível"
    string MSG_NÃO = "Item não encontrado"
    obter nome, endereço, item, quantidade
    sub = 0
    enquanto sub < DIMENSÃO
       achouEle = "N"
       se item = ITEM_VÁLIDO[sub] então
          achouEle = "S"
       fim-se
       sub = sub + 1
    fim-enquanto
    se achouEle = "S"
       imprimir MSG_SIM
    senão
       imprimir MSG_NÃO
    fim-se
fim
```

Figura 6-13 Fluxograma e pseudocódigo para programa que verifica a disponibilidade de itens (cont.)

Procurando um casamento exato em um array (cont.)

- **Flag:** uma variável ajustada para indicar se determinado evento ocorreu
- Técnicas para procurar um array:
  - Defina uma variável como 0 para começar no primeiro item do array
  - Inicialize uma variável flag como falsa para indicar o valor desejado que não foi encontrado
  - Examine cada item no array
  - Se os valores casarem, estabeleça a flag como **`verdadeira`**
  - Se o valor não casar, incremente o subscrito e examine o próximo item do array

Usando arrays paralelos

- Exemplo: serviço de vendas por reembolso postal
  - Dois arrays, cada um com seis itens
    - Números válidos dos itens
    - Preços válidos dos itens
  - Cada preço do array está propositada e convenientemente na mesma posição que o número do item correspondente no outro array

Usando arrays paralelos (cont.)

- **Arrays paralelos**
  - Cada elemento de um array é associado ao elemento da mesma posição relativa no outro
- Olhe através de um item válido do array para o item do cliente
  - Quando é encontrado um casamento para o número do item, ele retira o preço correspondente da lista de valores no array

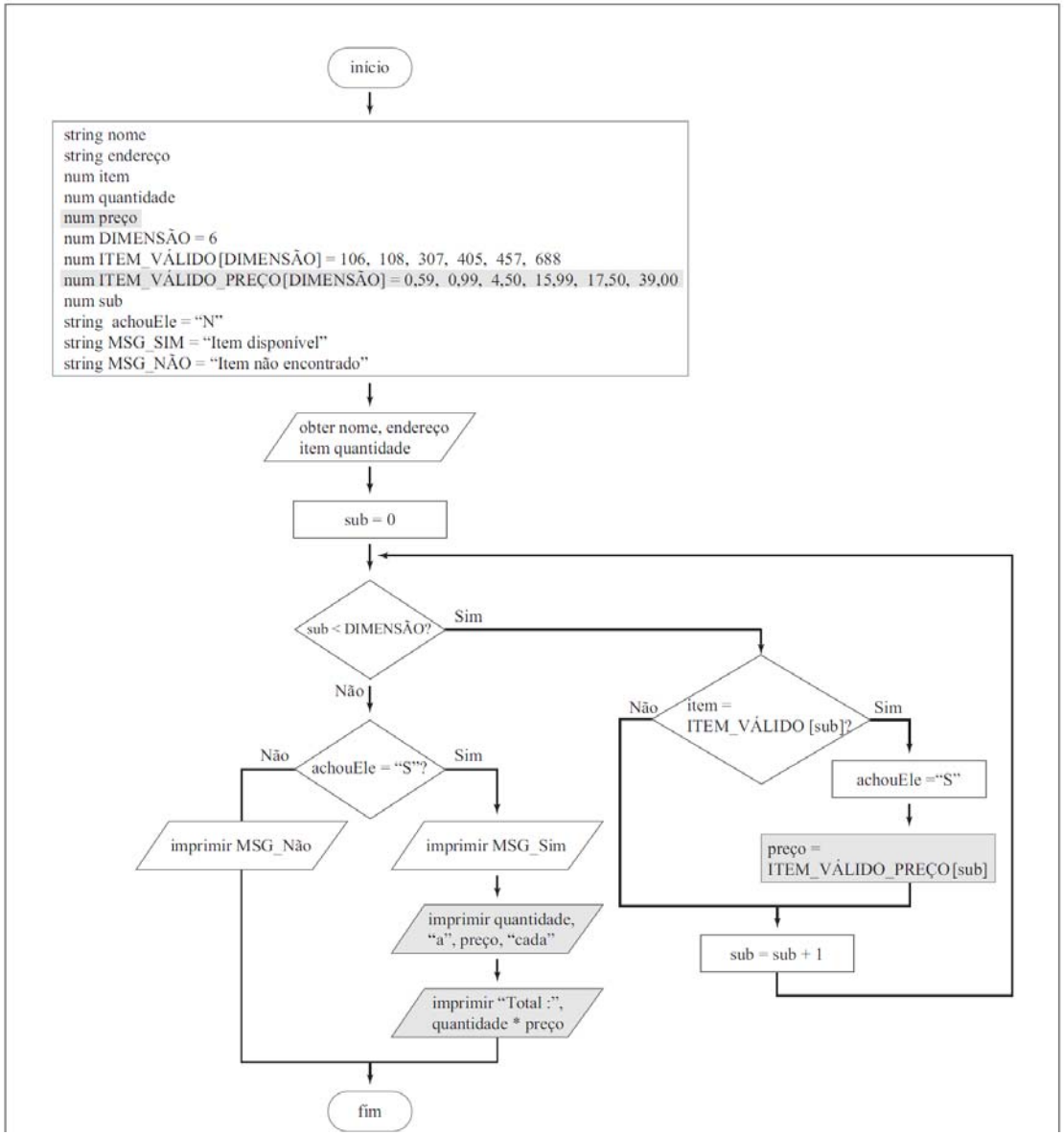


Figura 6-14 Fluxograma e pseudocódigo de programa que encontra o preço de um item

```
início
    string nome
    string endereço
    num item
    num quantidade
    num preço
    num DIMENSÃO = 6
    num ITEM_VÁLIDO[DIMENSÃO] = 106,  108,  307,  405,  457,  688
    num ITEM_VÁLIDO_PREÇO[DIMENSÃO] = 0,59,  0,99,  4,50,  15,99,  17,50,  39,00
    num sub
    string achouEle = "N"
    string MSG_SIM = "Item disponível"
    string MSG_NÃO = "Item não encontrado"
    obter nome, endereço, item, quantidade
    sub = 0
    enquanto sub < DIMENSÃO
       se item = ITEM_VÁLIDO[sub] então
          achouEle = "S"
          preço = ITEM_VÁLIDO_PREÇO[sub]
       fim-se
       sub = sub + 1
    fim-enquanto
    se achouEle = "S" então
       imprimir MSG_SIM
       imprimir quantidade, " a ", preço, " cada"
       imprimir "Total ", quantidade * preço
    senão
       imprimir MSG_NÃO
    fim-se
fim
```

Figura 6-14 Fluxograma e pseudocódigo de programa que encontra o preço de um item (cont.)

Usando arrays paralelos (cont.)

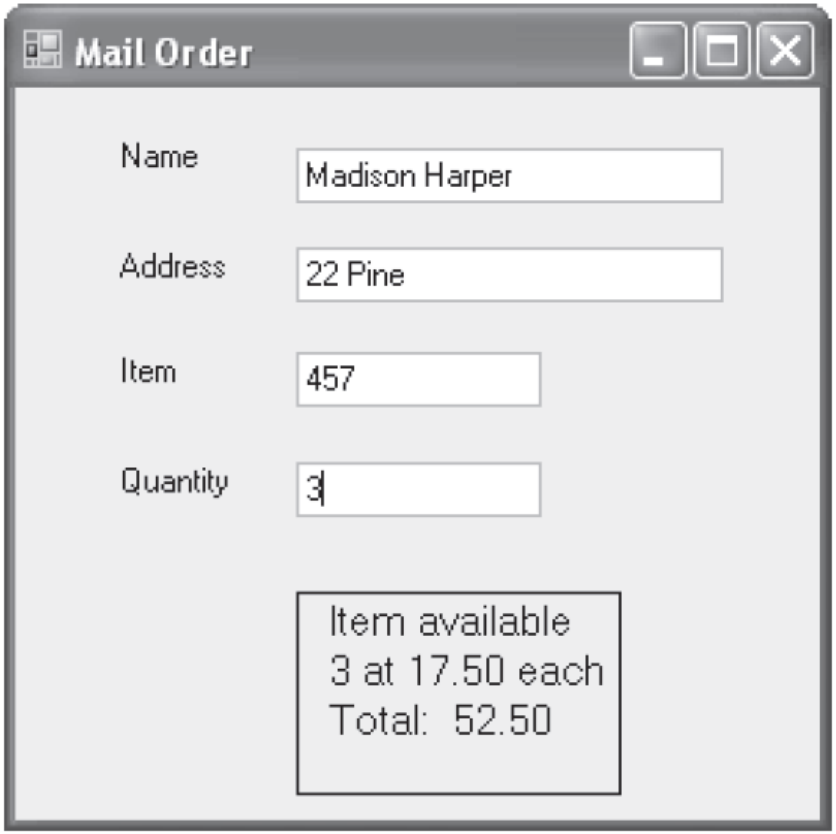


Figura 6-15 Execução típica do programa que encontra o preço dos itens

Melhorando a eficiência da busca usando uma saída antecipada

- O programa deve parar a busca pelo array quando o casamento é encontrado
- Estabelecer uma variável para um valor específico ao invés de deixar que o processo normal o faça
- **Saída antecipada**: sair de um loop assim que o casamento é encontrado
  - Melhora a eficiência
- Quanto maior o array, mais benéfico é sair do loop de busca assim que se encontrar aquilo que se está procurando

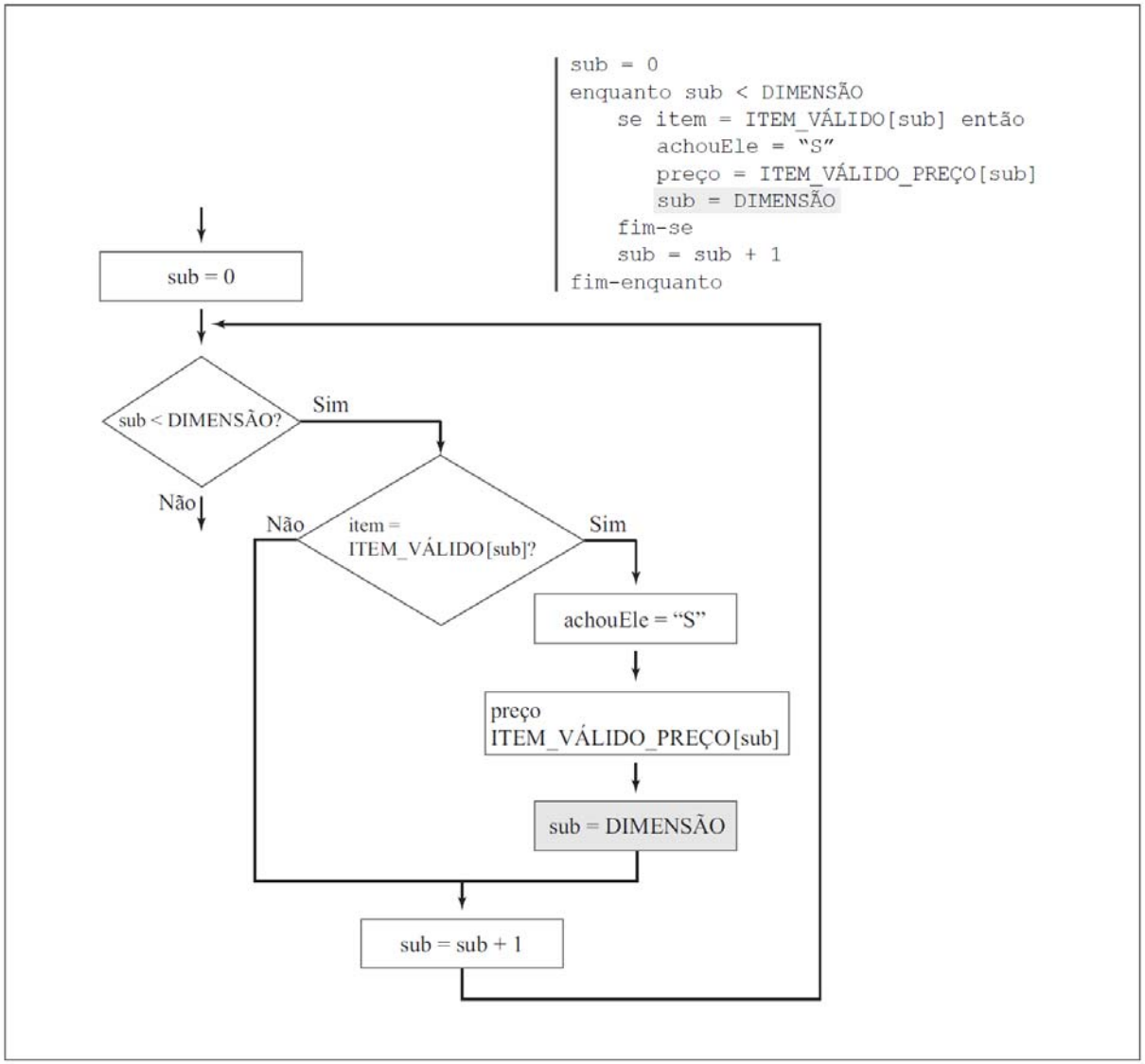


Figura 6-16 Fluxograma e pseudocódigo do loop que encontra o preço dos itens, saindo do loop assim que ele é encontrado

Buscando um casamento de faixas em um array

- Algumas vezes os programadores querem trabalhar com faixas de valores dos arrays
- Exemplo: serviço de vendas por reembolso postal
  - Ler os dados do pedido do cliente e determinar a porcentagem de desconto com base no valor do campo `quantidade`
- Primeira abordagem:
  - Array com tantos elementos quanto um cliente poderia encomendar e armazenar o desconto apropriado para cada número possível

Buscando um casamento de faixas em um array (cont.)

```
numeric DISCOUNT[76]
 = 0, 0, 0, 0, 0, 0, 0, 0, 0,
   0.10, 0.10, 0.10, 0.10,
   0.15, 0.15, 0.15, 0.15, 0.15,
   0.15, 0.15, 0.15, 0.15, 0.15,
   0.15, 0.15, 0.15,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20,
   0.20, 0.20, 0.20, 0.20, 0.20
```

NÃO FAÇA ISSO

Apesar de esse array ser utilizável, ele é repetitivo, propenso a erros e difícil de usar.

Figura 6-18 Array de desconto utilizável – porém, ineficiente

Buscando um casamento de faixas em um array (cont.)

- Enfoques da primeira abordagem:
  - Exige um array muito grande, que utiliza muita memória
  - Armazena o mesmo valor repetidas vezes
  - Como saber se o array tem elementos suficientes?
    - O cliente sempre pode encomendar mais
- Abordagem melhor:
  - Criar um array de descontos de apenas quatro elementos um para cada uma das taxas possíveis de desconto
  - Array paralelo com faixa de desconto
- Use loop para fazer comparações

Buscando um casamento de faixas em um array (cont.)

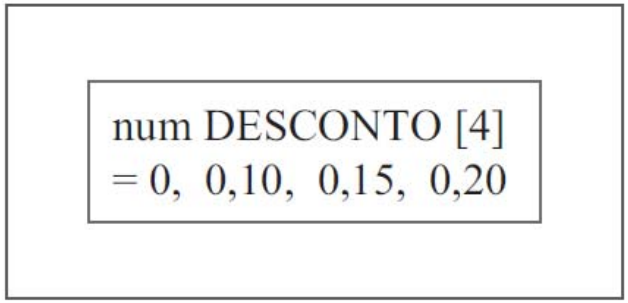


Figura 6-19
Array de descontos melhorado

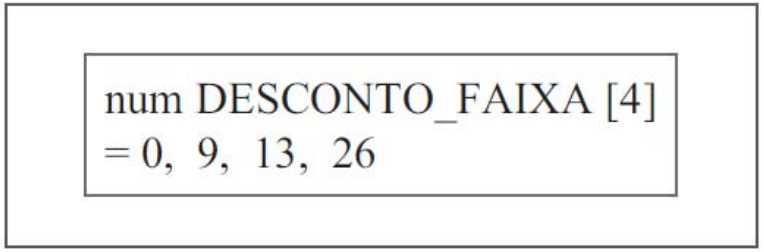


Figura 6-20
O array `DESCONTO_FAIXA` usando o fim inferior de cada faixa

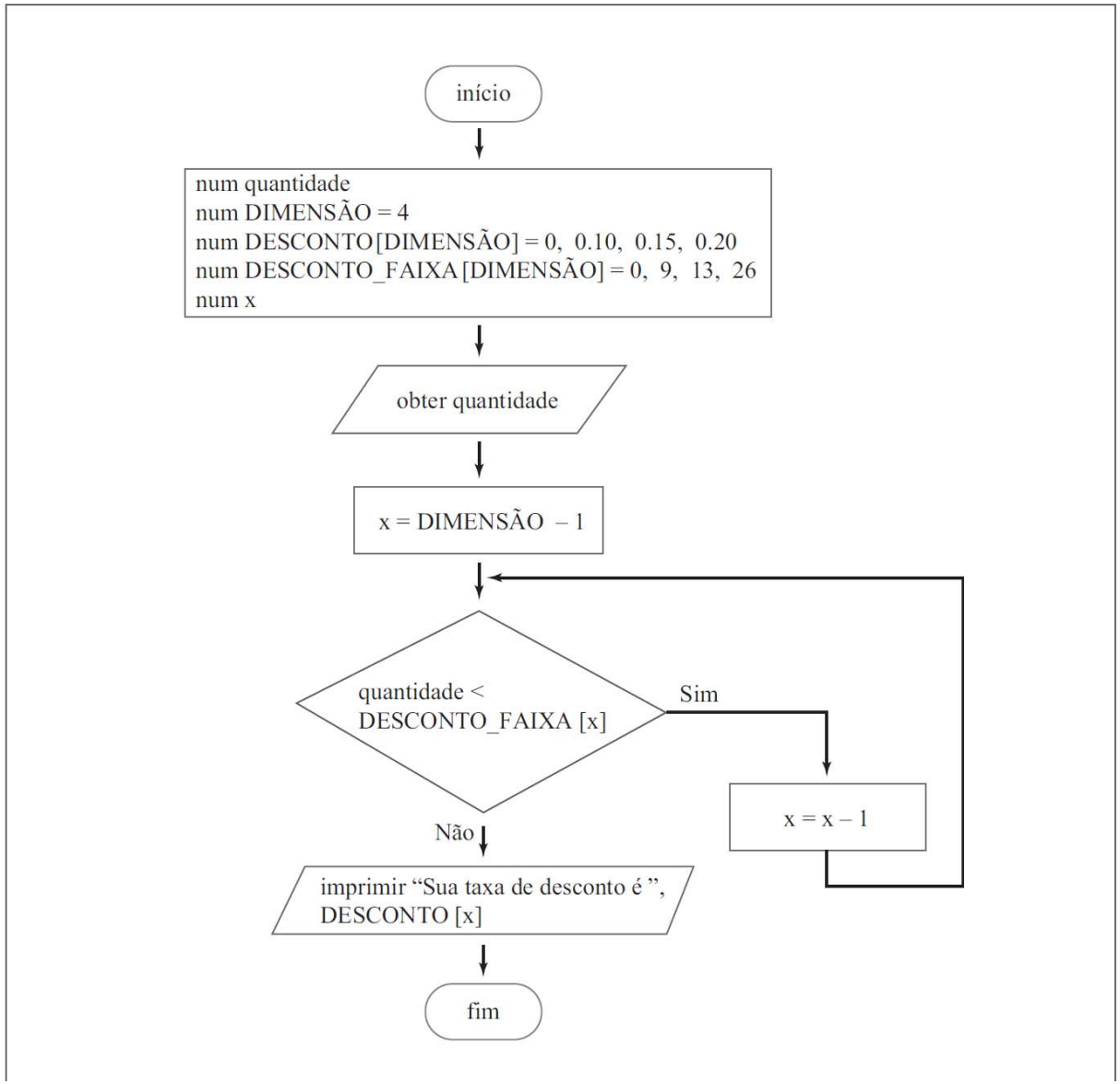


Figura 6-21 Programa que determina a faixa de desconto

Permanecendo dentro dos limites dos arrays

- Todo array têm um tamanho finito
  - Número de elementos no array
  - Número de bytes no array
- Arrays são sempre compostos de elementos do mesmo tipo de dado
- Elementos do mesmo tipo de dado sempre ocupam o mesmo número de bytes da memória

Permanecendo dentro dos limites dos arrays (cont.)

- O número de bytes em um array é sempre um múltiplo do número de elementos do array
- Ao acessar dados armazenados em um array, é importante usar um subscrito que contenha um valor que acesse a memória ocupada pelo array

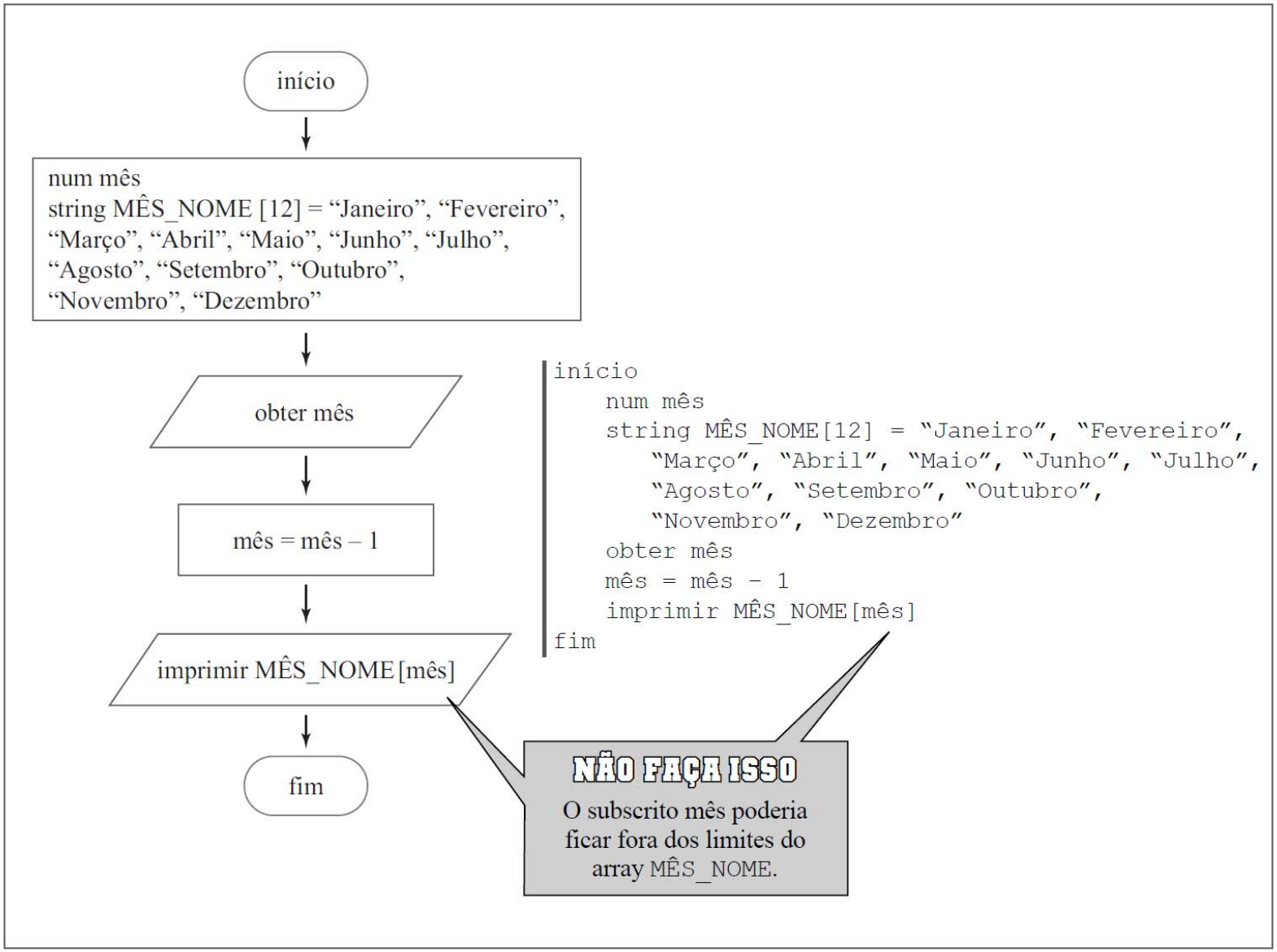


Figura 6-22 Determinando o string do mês a partir da entrada numérica do usuário

Permanecendo dentro dos limites dos arrays (cont.)

- A lógica de programação entende que todo número inserido pelo usuário é um número válido
- Ao usar um valor de subscrito que é negativo ou maior que o número de elementos em um array:
  - Algumas linguagens de programação param a execução do programa e emitem uma mensagem de erro
  - Outras linguagens de programação não emitem uma mensagem de erro, mas acessam um valor em uma localidade da memória externa à área ocupada pelo array

Permanecendo dentro dos limites dos arrays (cont.)

- De qualquer forma, ocorre um erro de lógica
- **Fora dos limites**: quando se usa um subscrito que não está dentro da faixa de subscritos aceitáveis
- Um bom programa deve ser capaz de lidar com esses erros e não permitir que o subscrito esteja fora dos limites

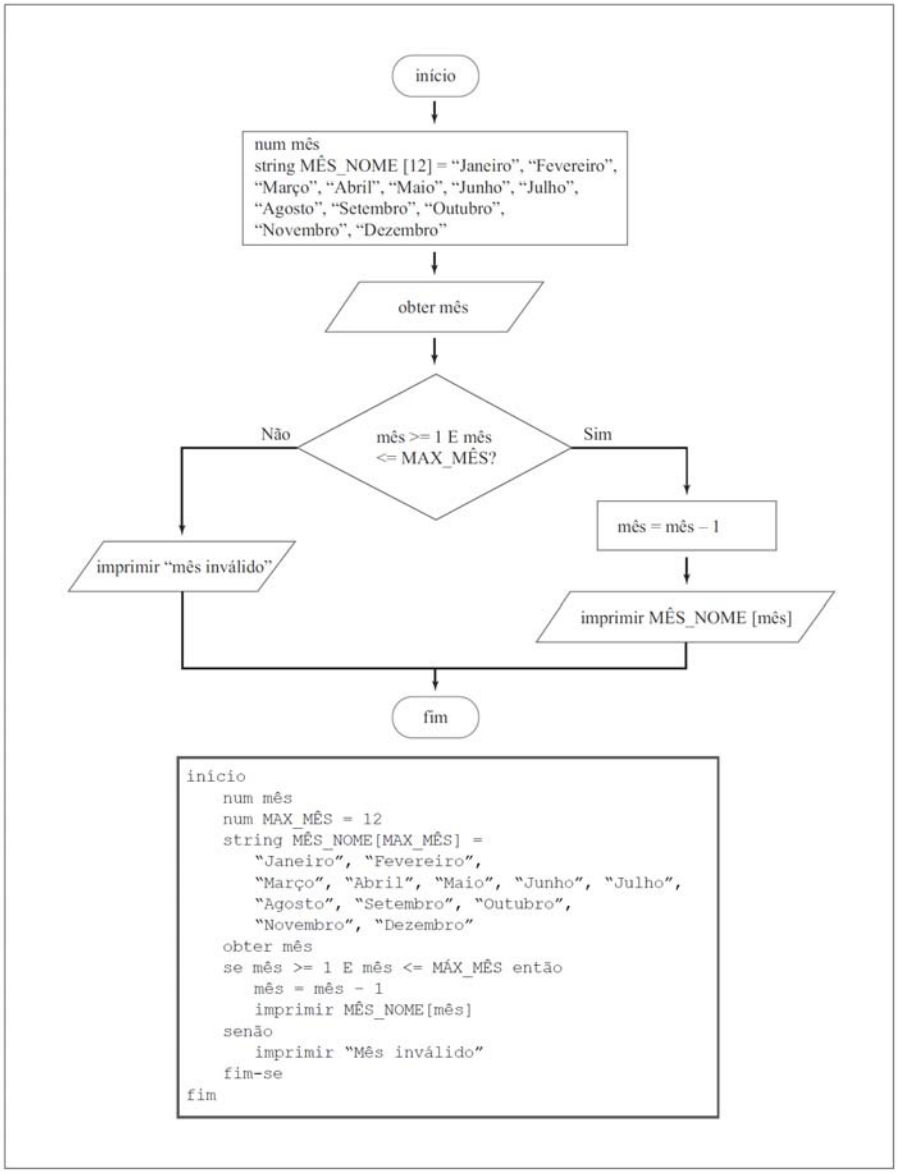


Figura 6-23 Programa que usa uma seleção para garantir que um subscrito seja válido

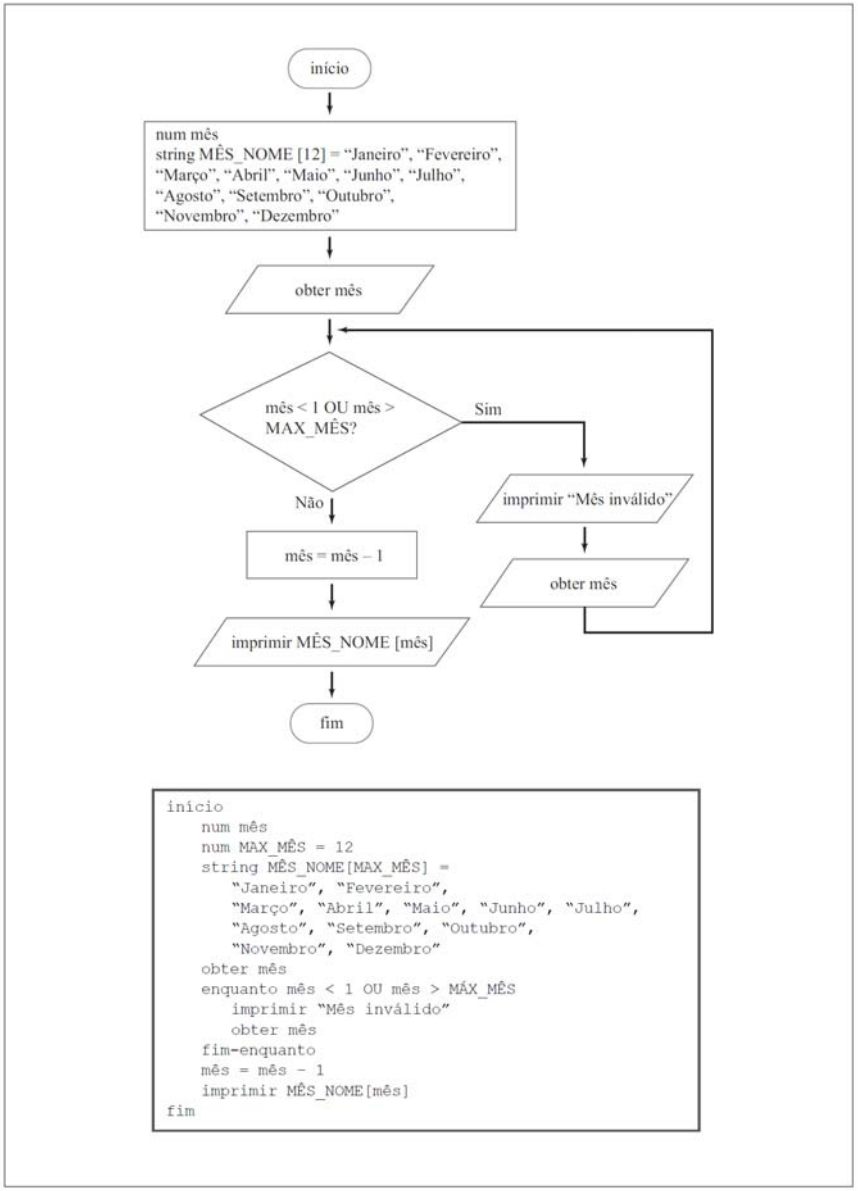


Figura 6-24 Programa que usa um loop para garantir que um subscrito seja válido

Usando um loop `for` para processar arrays

- Loop `for` – em uma única sentença
  - Inicializa uma variável de controle do loop
  - A compara a um limite
  - A altera
- O loop `for` é uma ferramenta especialmente conveniente quando se trabalha com arrays
  - Processa todos os elementos do array do começo ao fim
- Deve-se ter cuidado para se manter dentro dos limites do array
- O mais alto subscrito utilizável do array é uma unidade menor que a dimensão do array

Usando um loop `for` para processar arrays (cont.)

```
início
    num mês
    num MÁX_MÊS = 12
    string MÊS_NOME[MÁX_MÊS] =
        "Janeiro", "Fevereiro",
        "Março", "Abril", "Maio", "Junho", "Julho",
        "Agosto", "Setembro", "Outubro",
        "Novembro", "Dezembro"
    for mês = 0 to MÁX_MÊS -1
        imprimir MÊS_NOME[mês]
    fim-for
fim
```

Figura 6-25 Pseudocódigo que usa um loop `for` para imprimir nomes de meses

Usando um loop `for` para processar arrays (cont.)

```
início
    num mês
    num MÁX_MÊS = 12
    num ARRAY_LIMITE = MÁX_MÊS - 1
    string MÊS_NOME[MÁX_MÊS] =
       "Janeiro", "Fevereiro",
       "Março", "Abril", "Maio", "Junho", "Julho",
       "Agosto", "Setembro", "Outubro",
       "Novembro", "Dezembro"
    for mês = 0 to ARRAY_LIMITE
       imprimir MÊS_NOME[mês]
    fim-for
fim
```

Figura 6-26 Pseudocódigo que usa um loop for mais eficiente para imprimir nomes de meses

Resumo do Capítulo

- Array: uma série ou lista de variáveis na memória
  - Mesmo nome e tipo de dado
  - Subscritos diferentes
- É possível usar uma variável como um subscrito para substituir múltiplas decisões embutidas
- Pode-se declarar e inicializar todos os elementos em um array usando uma única sentença
- É possível inicializar valores do array dentro de um loop de inicialização
- Alguns arrays contêm valores determinados durante a execução de um programa
  - Outros arrays são mais úteis quando seus valores finais desejados são definidos pelo código-fonte quando se escrever o programa

Resumo do Capítulo (cont.)

- Alguns arrays contêm valores determinados durante a execução de um programa
  - Outros arrays são mais úteis quando seus valores finais desejados são definidos pelo código-fonte quando se escrever o programa
- Buscar em um array:
  - Inicie o subscrito
  - Teste cada elemento do array no loop
  - Defina um flag quando o casamento é encontrado
- Arrays paralelos: cada elemento de um array é associado ao elemento de mesma posição relativa do outro array

Resumo do Capítulo (cont.)

- Para comparar um valor a uma faixa de valores em um array, pode-se armazenar o valor do limite inferior ou do superior de cada faixa para comparação
- Para acessar dados armazenados em um array
  - Use um subscrito que contenha um valor que acesse a memória ocupada pelo array
- Ao usar um subscrito que não está dentro da faixa definida de subscritos aceitáveis, diz-se que o subscrito está fora dos limites
- O loop `for` é uma ferramenta especialmente conveniente ao se trabalhar com arrays
  - Frequentemente é necessário processar todos os elementos de um array do começo ao fim