1 Introdução

ESQUEMA DO CAPÍTULO

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (arranjo, "array")

- um ***único*** nome identifica uma ***série*** de posições de memória;
- cada elemento da série é ***referenciado*** individualmente por um ***índice*** que identifica sua posição relativa na série;

- atribuição:
  nome[índice] ← constante, variável ou expressão
- declaração:
  declare w[1:3] numérico

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Ex. 1:

```
algoritmo
    declare i, w[1:3] numerico
    w[1] ← 5
    w[2] ← 1
    w[3] ← 3
    i ← 2
    escreva w[2], w[i+1], w[w[i]]
fim algoritmo
```

Qual é o resultado da execução desse algoritmo?

Solução:

O algoritmo deverá escrever os seguintes valores numéricos (por quê?):
1, 3, 5

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Ex. 2:

Fazer um algoritmo que leia três notas, calcule e escreva a média das notas e o número de notas acima da média.

Solução:

Uma forma bastante simples de resolver o problema encontra-se desenvolvida ao lado.

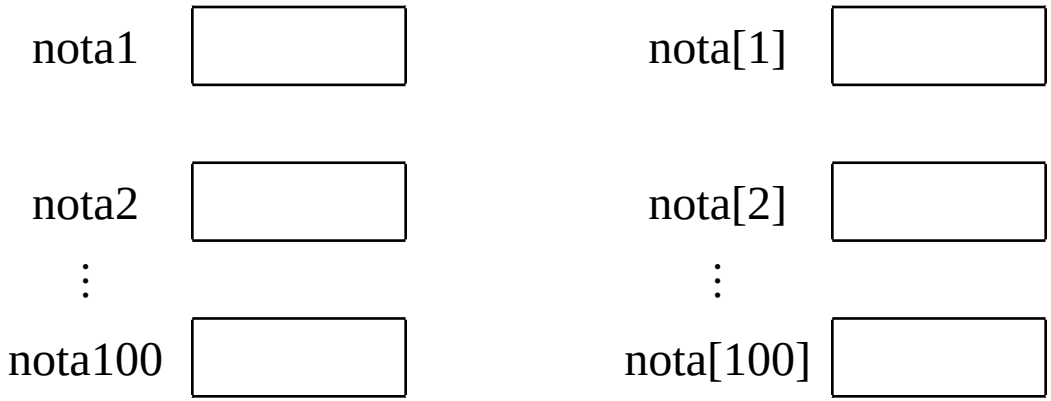
```
algoritmo
        {defina os tipos das variáveis}
    declare nota1, nota2, nota3, media, acima numérico
        {leia notas}
    leia nota1, nota2, nota3
        {calcule média}
    media <- (nota1+nota2+nota3)/3
        {calcule número de notas acima da média}
    acima <- 0
    se nota1 > media então
        acima ← acima+1
    fim se
    se nota2 > media então
        acima ← acima+1
    fim se
    se nota2 > media então
        acima ← acima+1
    fim se
        {escreva resultados}
    escreva "A media é", media
    escreva "Número de notas acima da media é",
acima
fim algoritmo
```

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Ex. 3:

Fazer um algoritmo que leia ***cem*** notas, calcule e escreva a média das notas e o número de notas acima da média.



Solução:

Uma solução possível é apresentada a seguir:

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução:

```
algoritmo
        defina os tipos das variáveis
                {leia notas e calcule média}
        media ← 0
        soma ← 0
        i ← 0
        repita
                i ←  i+1
                leia nota[i]
                        {contribua para a média}
                soma ← soma + nota[i]
        até i >=100
        media ← soma/100
                {calcule número de notas acima da média}
        acima ← 0
        i <- 1
        enquanto i <= 100 faca
                se nota[i] > media então
                        acima ← acima+1
                fim se
                i ← i+1
        fim enquato
                {escreva resultados}
        escreva "A media é", media
        escreva "Número de notas acima da media é", acima
fim algoritmo
```

```
ref.: defina tipo das variáveis
        declare media, soma, nota[1:100], i, acima numérico
fim ref.
```

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Ex. 4:

Escrever um algoritmo que faça reserva de passagens aéreas de uma companhia. Além da leitura do número dos voos e quantidade de lugares disponíveis, ler vários pedidos de reserva, constituídos do número de carteira de identidade do cliente e do número do voo desejado.

Para cada cliente, verificar se há disponibilidade no voo desejado. Em caso afirmativo, imprimir o número da identidade do cliente, e o número do voo, atualizando o número de lugares disponíveis. Caso contrário, avisar ao cliente da inexistência de lugares.

Indicando o fim dos pedidos de reserva, existe um passageiro cujo número da carteira de identidade é -1. Considerar fixo e igual a 37 o número de voos da companhia.

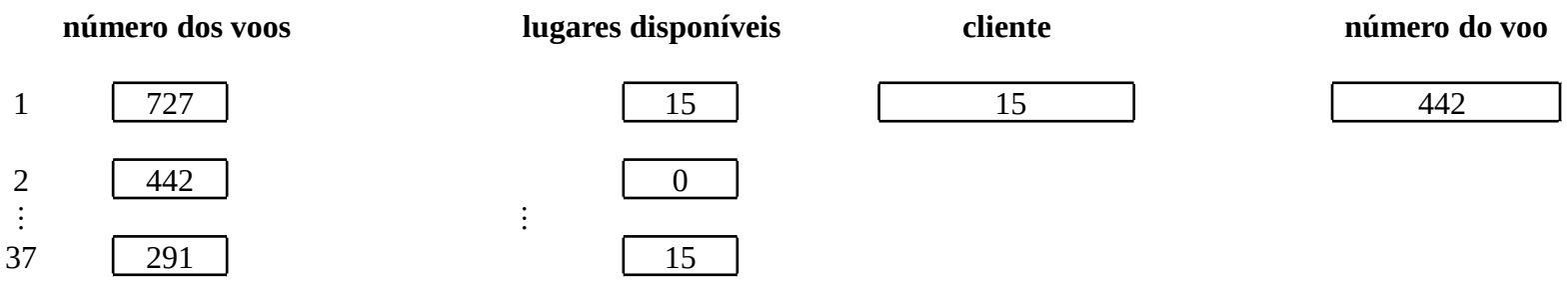
1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução:

Uma solução está descrita abaixo.

Primeiramente, procuramos definir a estrutura de dados necessária:



1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução (cont.):

Em seguida, desenvolvemos o algoritmo:

```
algoritmo
    defina os tipos das variáveis
    leia voos e lugares disponíveis
    leia cliente, nvoo
    enquanto cliente > 0 faça
        verifique a existência do voo
        verifique a existência de lugar
        leia cliente, nvoo
    fim enquanto
fim algoritmo
```

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução (cont.):

E os refinamentos sucessivos:

```
ref.: leia voos e lugares disponíveis
        para i de 1 até 37 faça
                leia voos[i], ldisp[i]
        fim para
fim ref.
```

```
ref.: verifique a existência do voo
        i <- 1
        enquanto (i<37) e (voos[i] <> nvoo) faça
                i ← i+1
        fim enquanto
        se (voos[i] = nvoo) então
                chave ← i   {a pesquisa foi bem sucedida}
        senão
                chave ← 0  {a pesquisa foi mal sucedida}
                escreva “Erro: voo inexistente"
        fim se
fim ref.
```

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução (cont.):

```
ref.: verifique a existência de lugar
        se chave <> 0 então      {isto é, a pesquisa foi bem sucedida}
                se ldisp[chave] > 0 então       {há lugar disponível}
                        escreva cliente, nvoo
                        atualize lugares disponíveis
                senão                           {não há lugar disponível}
                        escreva "voo", nvoo, "lotado"
                fim se
        fim se
fim ref.
```

```
ref.: atualize o número de lugares disponíveis
        ldisp[chave] ← ldisp[chave] - 1
fim ref.
```

```
ref.: defina tipo das variáveis
        declare voos, ldisp[1:37] numerico
        declare chave, cliente, i, nvoos numerico
fim ref.
```

1.4 Estruturas de Dados

1.4.1. variáveis subscritas (cont.)

Solução (final):

Inserindo-se, então, os refinamentos nos devidos lugares, tem-se o algoritmo completo:

```
algoritmo
                {defina os tipos das variáveis}
        declare voos, ldisp[1:37] numérico
        declare chave, cliente, i, nvoos numérico
                {leia voos e lugares disponíveis}
        para i de 1 até 37 faça
                leia voos[i], ldisp[i]
        fim para
        leia cliente, nvoo
        enquanto cliente > 0 faça
                        {verifique a existência do voo}
                i ← 1
                enquanto (i<37) e (voos[i] <> nvoo) faça
                        i <- i+1
                fim enquanto
                se (voos[i] = nvoo) então
                        chave ← i       {a pesquisa foi bem sucedida}
                senão
                        chave ← 0       {a pesquisa foi mal sucedida}
                        escreva "voo inexistente"
                fim se
                        {verifique a existência de lugar}
                se chave <> 0 então             {isto é, a pesquisa foi bem sucedida}
                        se ldisp[chave] > 0 então       {há lugar disponível}
                                escreva cliente, nvoo
                                        {atualize lugares disponíveis}
                                ldisp[chave] ← ldisp[chave] - 1
                        senão                           {não há lugar disponível}
                                escreva "voo", nvoo, "lotado"
                        fim se
                fim se
                leia cliente, nvoo
        fim enquanto
fim algoritmo
```