Vetores

Introdução à Programação para Biologia Molecular

Rosane Minghim

Apoio na confecção: Danilo Medeiros Eler

Rogério Eduardo Garcia

Renato Rodrigues

Carlos E. A. Zampieri

Baseado na Apostila: Curso Introdutório de Computação por R. Minghim e G. P. Telles

Vetores

- Tipos Compostos
  - Conjunto de vários elementos de mesmo tipo
  - Indexáveis individual e diretamente
- Um único nome. Ex: **V**
- Um tipo. Ex: **V:inteiro**

- Armazenamento em sequência.

  V:inteiro[1..7]

- Um índice.
  - Acesso direto: **V[i]**

Exemplos

```
tipo  frase = caracter[0..80]

variável
    valor: real[1..10]
    linha: frase

valor[1] ← 0.35
linha[0] ← ':'
```

Exemplos – Índice Negativo e Operações

```
variável
   abcissas: real[-5..5]
   j:inteiro
   valor:real
   prox_abs:real

abcissas[4] ← 10
abscissas[-1] ← 3
abcissas[0] ← -2
abcissas[-2] ← 3
j ← -5
abcissas[j] ← 0
abcissas[1] ← abcissas[5] + abcissas[-5]
valor ← raiz(abscissa[-1],2)
prox_abs ← abscissa[ trunca(abscissa[0]) ]
```

Após a sequência completa o valor de prox_abs seria 3

Exemplo: média aritmética de números lidos

```
Algoritmo média
   variável
       soma, n,i: inteiro
       aux:inteiro
       média:real

     leia(n)
     i ← 0
     enquanto (i<n) faça
              leia(aux)
              soma ← soma + aux
              i ← i + 1
     fim
    média ← soma/n
   escreva(soma,média)
fim
```

Exemplo: média aritmética de números lidos

E se também quiséssemos imprimir todos os números, entre aqueles lidos, que sejam maiores ou iguais a média?

```
Algoritmo média

tipo vetor50 = inteiro[1..50]
variável
  soma, n: inteiro
  valor: vetor50
  i:inteiro

  leia(n)
  soma ← 0
  para i de 1 até n passo 1 faça
    leia(valor[i])
    soma ← soma + valor[i]
  fim para
  média ← soma/n
  escreva(média)
  para i de 1 até n passo 1 faça
    se (valor[i] >= média) então
      escreva(valor[i])
    fim se
  fim para
fim
```

Exercício

Desenvolver um programa para ler um vetor **VAL** de números inteiros e criar outro vetor **VAL2** de mesma quantidade de elementos que VAL, onde os elementos tenham o dobro do valor dos elementos de VAL. O número máximo de elementos é 30.

Vetores em subprogramas - Exemplo

Suponha o subprograma dado na aula anterior (subprogramas), que possui o seguinte cabeçalho:

calcule_pontos (questao,resposta,pontos,acerto)

e atualiza os pontos obtidos na prova por um candidato com base na resposta de uma questão em particular.

Desenvolver um subprograma para, dado um vetor de 10 elementos contendo a resposta a cada uma das 10 questões da prova, calcular a soma de pontos de um funcionário. Além disso, o subprograma retorna um vetor de 10 posições contendo, em cada posição, o valor `verdadeiro` se a questão está correta e o valor `falso` se a questão está errada.

Vetores em subprogramas - Exemplo

Dois vetores são necessários: um, de entrada, para armazenar as respostas e um outro, de saída, para conter quais estão corretas.

Vamos definir tipos para eles:

```
constante  N_QUESTOES = 10

tipo
   vetor_respostas = caracter[1..N_QUESTOES]
   vetor_acertos = logico[1..N_QUESTOES]
```

Vetores em subprogramas - Exemplo

O subprograma recebe o vetor de respostas e fornece, como saída, além da soma total de pontos, o vetor de acertos. O cabeçalho para o subprograma é, portanto:

Subprograma calcule_resultado_prova(respostas,acertos,número_de_pontos)

`e: respostas: vetor_resposta` {um vetor contendo, em cada posição,a resposta a uma questão da prova}

`s: acertos: vetor_acertos` {um vetor contendo, em cada posição: `falso` se a resposta à questão associada for errada, e `verdadeiro` caso ela seja correta}

`número_de_pontos:inteiro` {o número de pontos conseguido na prova}

{este subprograma utiliza o subprograma calcule_pontos}

Vetores em subprogramas - Exemplo

O código do subprograma deve, para cada elemento do vetor de respostas, chamar o subprograma calcule_pontos. O resultado do acerto, devolvido por ele, deve ser armazenado no vetor de acertos. O código fica:

```
variável
    i,resp:inteiro
    certo:logico
início
    numero_de_pontos ← 0
    para i de 1 até N_QUESTOES passo 1 faça
       resp ← respostas[i]
       calcule_pontos(i, resp, numero_de_pontos,certo)
       acertos[i] ← certo
    fim para
fim
```

Vetores em subprogramas - Exemplo

Subprograma calcule_resultado_prova(respostas,acertos,numero_de_pontos)

e: respostas: vetor_resposta {um vetor contendo, em cada posição, a resposta a uma questão da prova}

s: acertos: vetor_acertos {um vetor contendo, em casa posição,falso se a resposta à questão associada for errada, e verdadeiro caso ela seja correta}

numero_de_pontos:inteiro {o número de pontos conseguido na prova}

{este subprograma utiliza o subprograma calcule_pontos}

```
variável
   i:inteiro
início
  numero_de_pontos ← 0
  Para i de 1 até N_QUESTOES passo 1 faça

calcule_pontos(i,respostas[i],numero_de_pontos,acertos[i])
  fim para
fim
```

Vetores - Inicialização

O pseudo-código admite a definição de vetores constantes, da seguinte forma:

```
constante
  nome:tipo = {c1,c2,...c_max}
```

Onde:

`tipo` é a definição de um vetor e

`c1,c2,...c_max` são constantes do tipo elementar armazenado no vetor.

Alguns exemplos:

```
tipo
  vet = inteiro[1..6]
constante
  v: vet = {4,5,4,3,3,1}
  c: caracter[1..3] = {'A', 'B', 'C`}
```

Vetores em subprogramas - Exercícios

1. Fazer um algoritmo principal que, utilizando o subprograma calcule_resultado_prova, leia do usuário as respostas para os funcionários da empresa que prestaram a prova, imprimindo, para cada um deles, a pontuação total e as o número das perguntas respondidas corretamente.
2. Alterar o algoritmo acima para, além das impressões solicitadas, imprimir o número do funcionário que tirou a maior nota.
3. Fazer um subprograma para ler do usuário um vetor genérico de números reais com, no máximo, 50 elementos.
4. Fazer um subprograma para escrever o conteúdo de um vetor genérico com, no máximo, 50 elementos.

Vetores em subprogramas - Exercícios

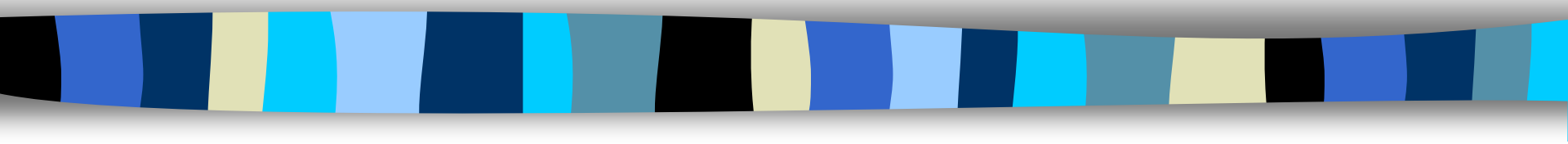
5. Desenvolver um subprograma para determinar o maior valor armazenado num vetor de n elementos
6. Desenvolver um algoritmo principal para testar o subprograma acima
7. Desenvolver um conjunto de subprogramas para determinar certos valores para os dados armazenados num vetor de números reais:
    1. A média
    2. A mediana
    3. O desvio padrão
    4. O número de valores acima da média

    Obs. Esses subprogramas podem utilizar subprogramas definidos anteriormente, e também uns aos outros.
8. Desenvolver um algoritmo principal para testar os subprogramas desenvolvidos no item acima.

Sugestão

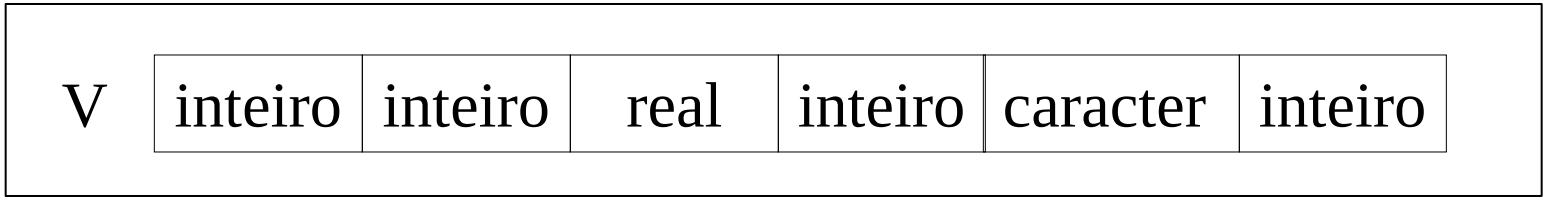
- Variáveis compostas (vetores e matrizes) com um número razoável de elementos devem ser passadas em subprogramas sempre por referência, mesmo quando forem parâmetros apenas de entrada
- A passagem de parâmetros deve ser acompanhada de comentário adequado, logo abaixo do cabeçalho do subprograma, indicando quais parâmetros são de entrada, quais são de saída, e quais são de entrada e saída

Vetores em PYTHON



Vetores

- Não existem vetores (como tal) em Python, por isso usam-se listas
- Tipos Compostos
  - Conjunto de vários elementos de mesmo tipo
- Um único nome. Ex: **V**
- No Python a tipagem é dinâmica, permitindo elementos de tipos diferentes na lista.



- Armazenamento em sequência **V: V[0], V[1], V[2],...,V[5]**
- Um índice.
  - Acesso direto: **V[i]**
- Exemplo:
  - **V = [3,10,4.5,9,'s',8]**

Exemplos

frase = caracter[0..80]

variável

valor: real[1..10]

linha: frase

valor[1] ← 0.35

linha[0] ← ‘:’

Exemplos

```
algoritmo maior
declarações

      leia (n)
      para j de 1 até n faça
        leia nro[j]
      fim para
      maior ← nro[1]
      para i de 2 até n passo 1 faça
        se nro[i] > maior então
          maior ← nro[i]
        fim se
    fim para
    escreva (maior)
fim
```

Exemplos

```
#programa vetor1


nro = []
n = input('forneca a quantidade de numeros: ‘)
for i in range(n+1):
    nro=nro + [input('forneca o ‘+str(i+1)+'o. numero: ')]

maior = nro[0]
for i in range(1,n+1):
    if (nro[i] > maior):
        maior = nro[i]

print 'maior numero: ',maior
```

Vetores em Subprogramas

- Em Python, para que um vetor possa ser modificado em um subprograma ele deve ser retornado como resultado da função
- Exemplo:

```
#programa EXEMPLO

def func(vet,valor):
      vet = vet + [valor]
      return vet

vetor = func(vetor,10)
```

Exemplo

```
constante
  max_el = 100
tipo
  vet_int = inteiro[1..max_el];

Subprograma maior_elemento
(valores,n):inteiro
e: valores:vet_int
     n: inteiro
r: o maior elemento do vetor, inteiro
declarações
início
  maior ← valores[1]
  para i de 2 até n passo 1 faça
     se valores[i] > maior então
        maior ← valores[i]
     fim se
  fim para
  retorne(maior)
fim
```

Exemplo

```
#programa VETOR1

def maior_elemento(valores)
    maior = valores[0]
    for i in range(1,len(valores)):
            if valores[i]>maior:
                    maior = valores[i]
    return maior



nro = []
n = input('FORNECA A QUANTIDADE DE NUMEROS: ')
for i in range(n+1):
    nro=nro + [input('FORNECA O ' + str(i+1)+'o. NUMERO: ')]

print 'MAIOR NUMERO: ', maior_elemento(nro)
```

Inicialização de Vetores em Python

O Python admite inicialização de vetores constantes como no exemplo abaixo:

```
nome = [c1,c2,…,c_max]
```

onde:
c1,c2,...c_max são constantes armazenadas no vetor.

Alguns exemplos:

```
v = [4,5,4,3,3,1]
c = ['A', 'B', 'C']
```

Em Python não há a necessidade de definir o tamanho do vetor, basta inicializá-lo como uma lista vazia e acrescentar elementos depois:

```
nome = []
```

Exemplo

```
#programa vetores

v1 = v2 = [0,1,2,3,4,5,6,7,8,9]
c = ['A','B','C‘]

print ‘v1=’,v1[1], v1[2], v1[3]
print ‘v2=‘,v2[1], v2[2], v2[3]
print ‘c=‘,c[1],c[2],c[3]
```

Saída:

>>> v1=0 1 2

>>> v2=0 1 2

>>> c=A B C