
# Algoritmos e Estruturas de Dados I

## Variáveis

versão 3.8

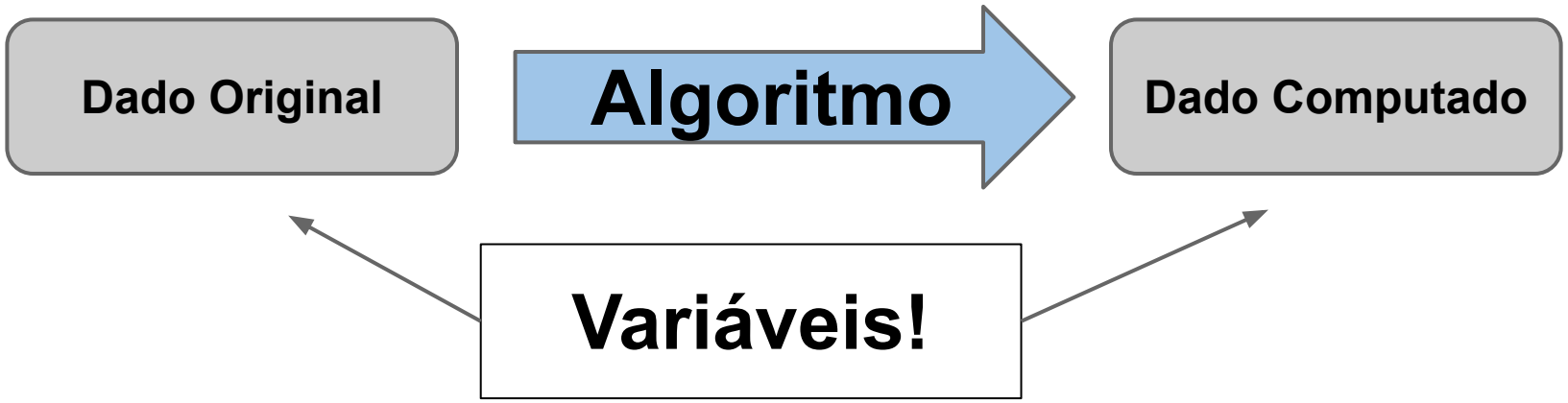
Fabiano Oliveira
fabiano.oliveira@ime.uerj.br

# Variáveis

De maneira geral, computar é explicitar um dado que encontrava-se de outra forma empregando-se para tal um processo de transformação, chamado de ***algoritmo***.

# Variáveis

De maneira geral, computar é explicitar um dado que encontrava-se de outra forma empregando-se para tal um processo de transformação, chamado de ***algoritmo***.



# Variáveis

# Variáveis

Variável é o nome simbólico de um local de armazenamento de um valor

- Variáveis são declaradas (estabelecendo domínio)
- Valores são armazenados pela operação de atribuição
- Valores são lidos pelo uso do nome simbólico

# Variáveis

Tipos:

- Escalar (ou Primitivo)
- Matrizes e Vetores
- Estrutura (ou Registro)
- Ponteiro (ou Referência)

# Variável Escalar
# (ou Variável Primitiva)

# Variável Escalar

- Variável Escalar é aquela que armazena um único valor de determinado tipo

- Adotaremos os seguintes tipos de valores em pseudo-código:
  - Inteiro: números inteiros (ex: -4, 0, 1, 100)
  - Real: números reais (ex: 2.2, 0, 3.1415)
  - Caractere: símbolo alfanumérico ("a", "b", "9", "")
  - Lógico: V (verdadeiro) / F (falso)
  - Cadeia: texto (ex.: "abc", "Fabiano Oliveira", etc.)
  - DataHora: horários (ex: "05/12/1978 20:12")

# Variável Escalar

- O intervalo de valores que se pode armazenar em um tipo de variável é chamado de ***domínio do tipo da variável***

- Cada linguagem dimensiona o tamanho de memória para armazenar cada tipo de valor de sua maneira, o que significa que cada linguagem pode ter um domínio para um tipo de variável eventualmente distinto das outras

# Variável Escalar

- Assume-se que cada tipo de variável possuirá um valor que, no contexto de um algoritmo específico, representa a ausência de valor. Este valor é arbitrário e pode variar conforme o algoritmo. De maneira geral, chamaremos tal valor de "NULO", independente do tipo da variável
- Exemplos de mapeamentos típicos de "NULO" para valores reais, conforme o tipo:
  - Lógico: F
  - Inteiro: 0 (ou -1)
  - Caractere: ""
  - Data/Hora: 01/01/1900 00:00
  - etc.

# Variável Escalar

- Declaração:

```
var <nome variável, ...>: <tipo variável>
```

- Atribuição:

```
<nomevar> ← <valor no domínio do tipo> |
            <expressão de valor>
```

# Variável Escalar

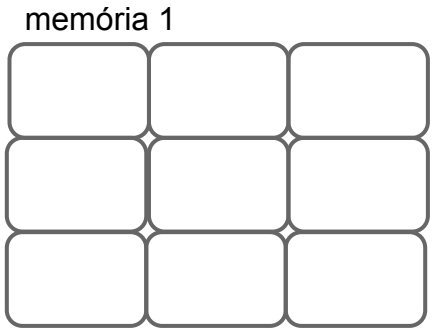
| Operador | Pseudo-código |
|---|---|
| Soma | `a + b` |
| Subtração | `a - b` |
| Multiplicação | `a * b` |
| Divisão | `a / b` |
| Divisão Inteira | `a div b` |
| Resto da Divisão | `a mod b` |
| Exponenciação | `a ^ b` |
| Raiz quadrada | `sqrt(a)` |

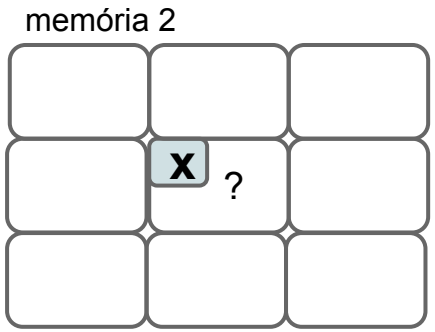
# Variável Escalar

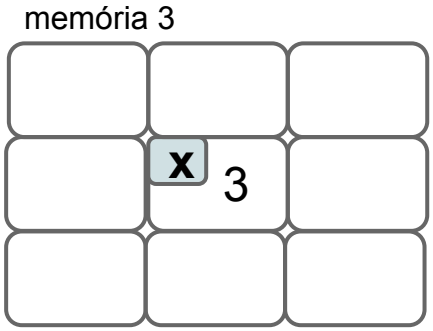
| Operador | Pseudo-código |
|---|---|
| Igual | = |
| Diferente | ≠ |
| Menor | < |
| Maior | > |
| Menor ou Igual | ≤ |
| Maior ou Igual | ≥ |
| E | E |
| OU | OU |
| NÃO | NÃO ou ! ou ¬ |

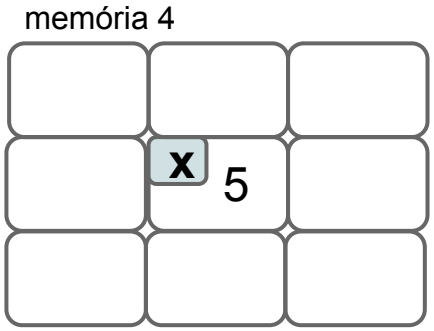
# Variável Escalar

```
programa Exemplo()
{memória 1}
    var x: Inteiro
{memória 2}
    x ← 3
{memória 3}
    x ← x + 2
{memória 4}
```



# Simulação de Execução (fazer o "Chinês")

Para entender a dinâmica de um programa, é útil a técnica de listar as variáveis do programa e simular sua execução, atualizando os valores das variáveis passo a passo

# Simulação de Execução (fazer o "Chinês")

**Exercício**: descrever estado final de memória para o seguinte algoritmo:

```
programa Exercício()
    var x, y: Inteiro, xbemmaior: Lógico
    x ← 2
    y ← 3
    x ← x + y
    y ← y + x
    x ← 2*x + y
    xbemmaior ← (x > (3 * y))
```

# Variável Escalar

**Exercício**: há algo de errado abaixo?

```
programa Exercício()
    var x, y: Inteiro, z: Real
    x ← 2
    y ← x + y
    z ← x / y
```

# Variável Estrutura (ou Registro)

# Variável Escalar

- Variável estrutura é aquela que define um novo tipo de variável, composta pela junção de uma ou mais variáveis em uma única variável

# Variável Estrutura

- **Definição**:

```
estrutura <nome> <tipo1,tipo2,...>:
    <declaração de variável>
    (podendo usar os tipos parametrizáveis tipo1,
    tipo2, etc.)
    ...
```

- **Declaração**:

```
var <nome variável, ...>:
     <nome estrutura><tipo1,tipo2,...>
```

# Variável Estrutura

- **Exemplo 1**:

```
estrutura Aluno:
   Matricula: Inteiro
   Nome: Caractere[50]
   DataNasc: DataHora
   Endereco: Caractere[500]
   Formado: Lógico
   CR: Real
var a: Aluno
```

# Variável Estrutura

- **Exemplo 2**:

```
estrutura Aluno <Tipo1, Tipo2>:
   Matricula: <Tipo1>
   Nome: Caractere[50]
   DataNasc: <Tipo2>
   Endereco: Caractere[500]
   Formado: Lógico
   CR: Real
var Aluno1: Aluno <Inteiro, DataHora>
var Aluno2: Aluno <Caractere[10], Inteiro>
```

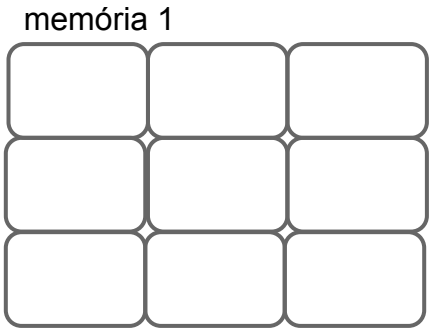
# Variável Estrutura

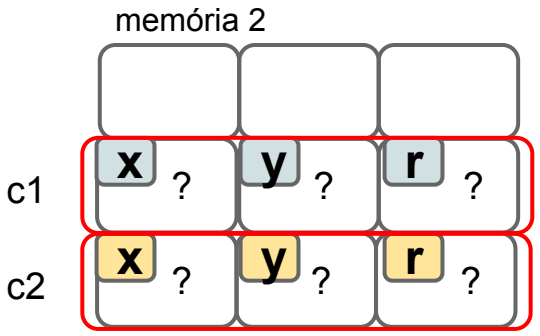
- **Atribuição**: atribuição de valores a cada variável específica da estrutura, na forma:

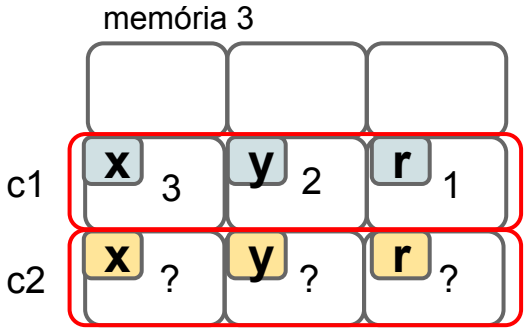
```
<var estrutura>.<var da estrutura> ← <valor>
```

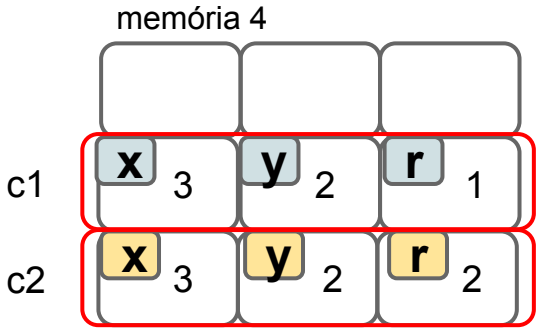
# Variável Estrutura

```
programa CriaCirculos()
    estrutura Circulo:
        x, y: Real
        r: Real
{memória 1}
    var c1, c2: Circulo
{memória 2}
    c1.x ← 3
    c1.y ← 2
    c1.r ← 1
{memória 3}
    c2 ← c1
    c2.r ← 2 * c2.r
{memória 4}
```



# Variável Estrutura

**Exercícios**:

Definir uma estrutura Endereco e preencher valores de duas variáveis deste tipo, e preencher valor para uma variável deste tipo

Definir uma estrutura Contato, com os campos Nome, E-mail, Endereço Comercial e Endereço Residencial, e preencher valor para uma variável deste tipo

# Variável Matriz e Vetor

# Variável Matriz e Vetor

- Variável Matriz é a variável que armazena múltiplos valores de um mesmo tipo, organizados em dimensões (análogo a matriz matemática)

- Vetor é o caso particular da Matriz de uma única dimensão

# Variável Matriz e Vetor

- **Declaração:**

```
var <nome matriz>[<início>..<fim>,...]: <tipo variável>
```

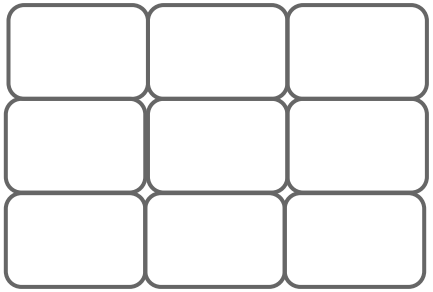
- **Atribuição:**

Idêntico a variáveis, levando-se em conta que `<nome matriz>[<índice>,...]` é uma variável, para <índice> entre <início> e <fim> correspondente à sua dimensão

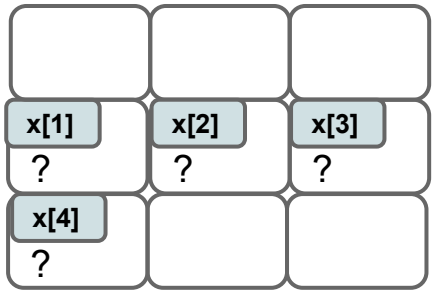
# Variável Matriz e Vetor

```
programa Exemplo()
{memória 1}
var x[1..4]: Inteiro
{memória 2}
    x[1] ← 3
{memória 3}
    x[2] ← x[1] + 2
{memória 4}
```

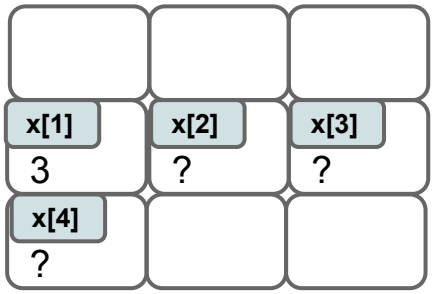
memória 1



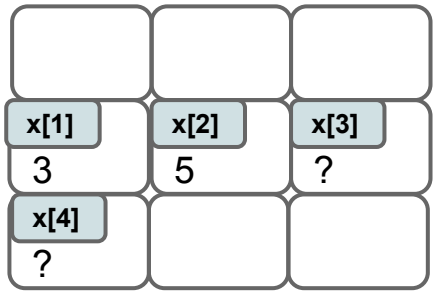
memória 2




memória 3




memória 4



# Variável Matriz e Vetor

**Exercício**: descrever estado final de memória para o seguinte algoritmo:

```
programa Exercício()
    var x[1..2], y[1..2]: Inteiro
    x[1] ← 1
    y[1] ← 2
    x[2] ← x[1] + y[1]
    y[2] ← y[1] + x[2] + x[1]
```

# Variável Matriz e Vetor

**Exercício**: descrever estado final de memória para o seguinte algoritmo:

```
programa Exercício()
    var x[1..2,1..2]: Inteiro
    x[1,1] ← 1
    x[1,2] ← 2
    x[2,1] ← x[1,1] + x[1,2]
    x[2,2] ← x[1,2] + x[2,1] + x[1,1]
```

# Variável Matriz e Vetor

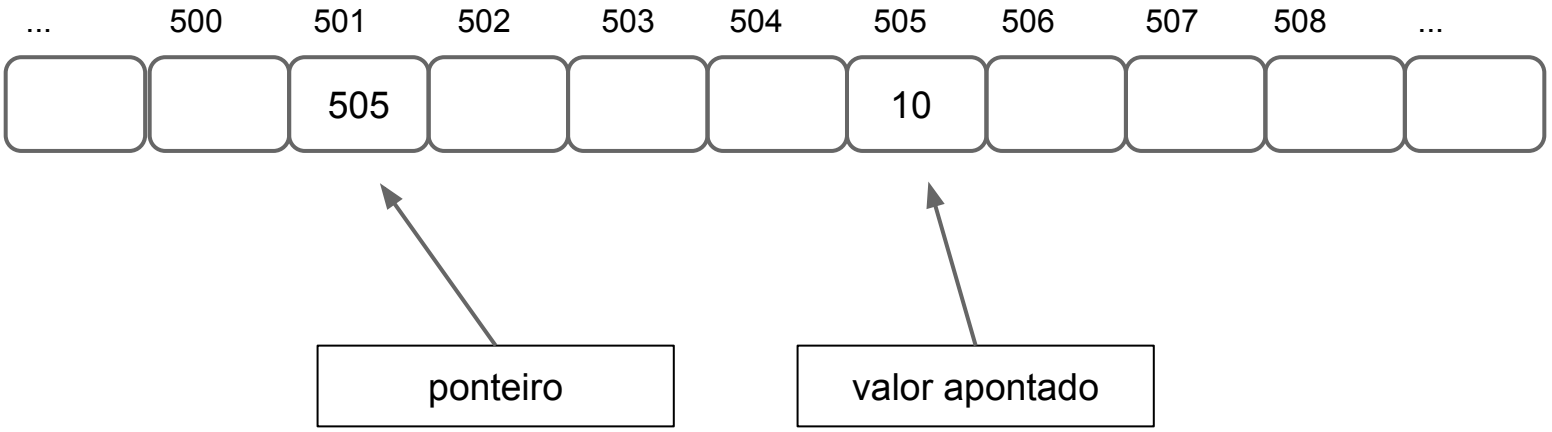
**Exercício**:

Escrever um algoritmo que armazene 3 contatos em um vetor. Usar definição existente de estrutura de Contato.

# Variável Ponteiro (ou Referência)

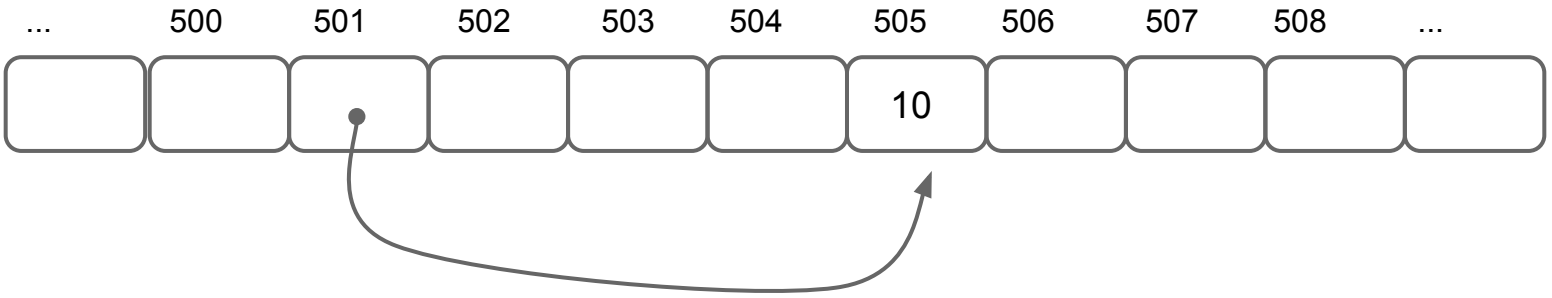
# Variável Ponteiro

- Variável Ponteiro é aquela que armazena não o dado em si, mas o endereço de memória onde este dado se encontra



# Variável Ponteiro

- Variável Ponteiro é aquela que armazena não o dado em si, mas o endereço de memória onde este dado se encontra



# Variável Ponteiro

- **Declaração:**

```
var <nome ponteiro>: ^<tipo variável>
```

# Variável Ponteiro

- **Atribuição:**
  - Atribuir **o ponteiro** como uma variável normal, através de:
    - recebimento do valor de outro ponteiro
    - operador `@<variável>` para capturar o endereço de memória da variável especificada
    - uso do comando **`alocar`**`(<nome ponteiro>)` para obter área de memória adicional (conhecida como ***alocação dinâmica***)
      - deve-se usar o comando **`desalocar`**`(<nome ponteiro>)` após finalizar o uso da variável, caso contrário, ocorre o problema conhecido como "vazamento de memória"

# Variável Ponteiro

- **Atribuição:**
  - Atribuir **o ponteiro** como uma variável normal, através de:
    - recebimento do valor de outro ponteiro

# Variável Ponteiro

- **Atribuição:**
  - Atribuir **o ponteiro** como uma variável normal, através de:
    - operador `@<variável>` para capturar o endereço de memória da variável especificada

# Variável Ponteiro

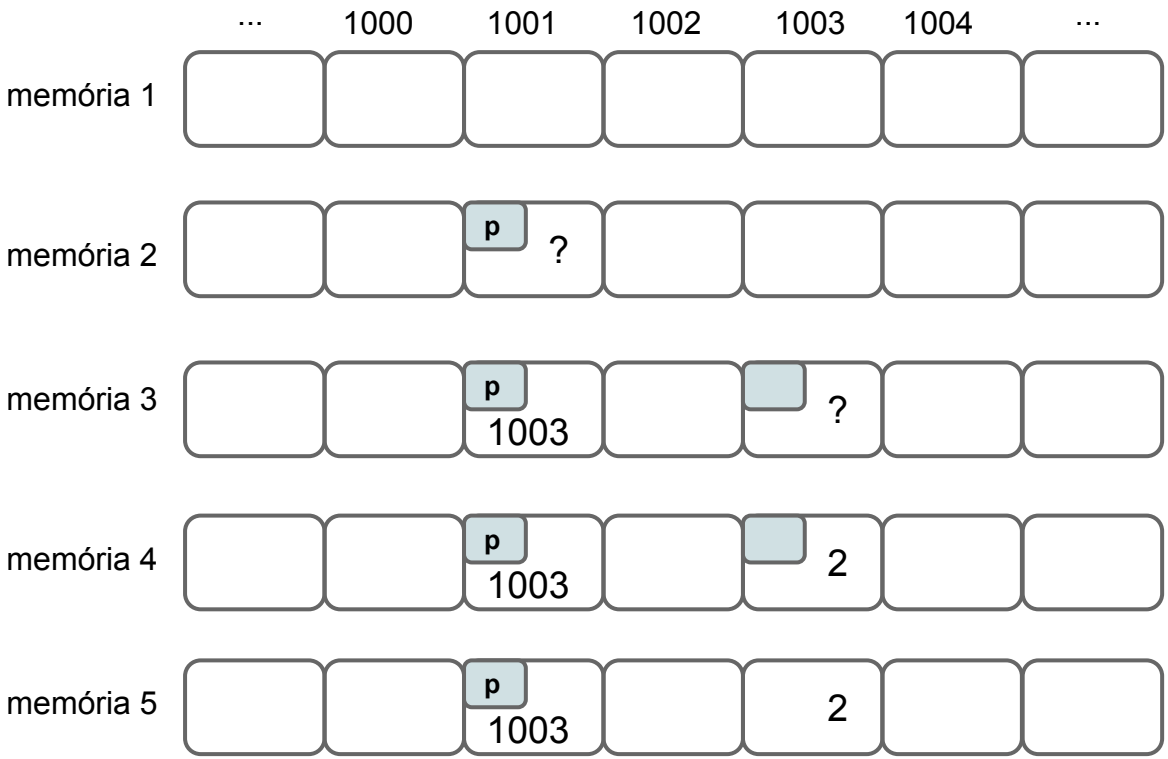
- **Atribuição:**
  - Atribuir **o ponteiro** como uma variável normal, através de:
    - uso do comando **`alocar`**`(<nome ponteiro>)` para obter área de memória adicional (conhecida como ***alocação dinâmica***)
      - deve-se usar o comando **`desalocar`**`(<nome ponteiro>)` após finalizar o uso da variável, caso contrário, ocorre o problema conhecido como "vazamento de memória"

# Variável Ponteiro

- **Atribuição:**
  - Atribuir **o valor apontado** pelo ponteiro p como uma variável normal, através do uso do nome p^
    - p^ equivale a uma variável do tipo de p cujo valor está alocado no endereço p

# Variável Ponteiro

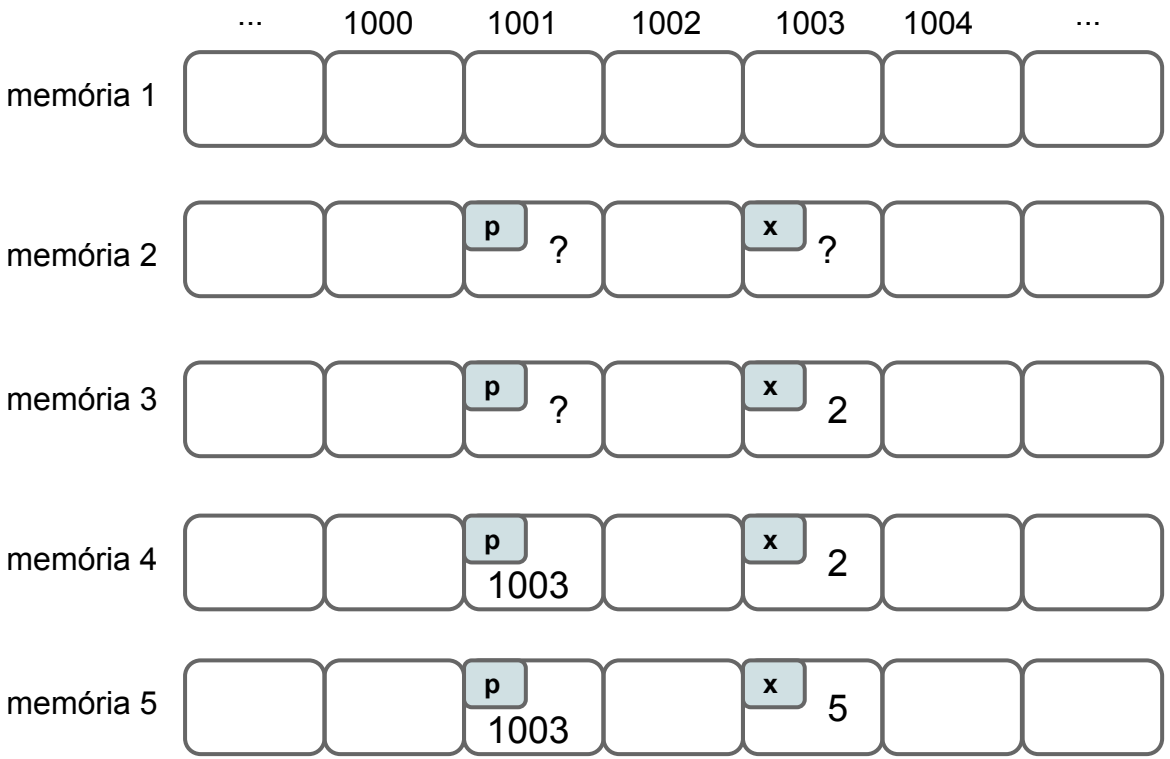
```
programa Exemplo()
{memória 1}
    var p: ^Inteiro
{memória 2}
    alocar(p)
{memória 3}
    p^ ← 2
{memória 4}
    desalocar(p)
{memória 5}
```



# Variável Ponteiro

```
programa Exemplo()
{memória 1}
    var p: ^Inteiro,
        x: Inteiro
{memória 2}
    x ← 2
{memória 3}
    p ← @x
{memória 4}
    p^ ← p^ + 3
{memória 5}
    escrever("x=",x)

    escrever(p)
    escrever(p^)
    escrever(@p)
```

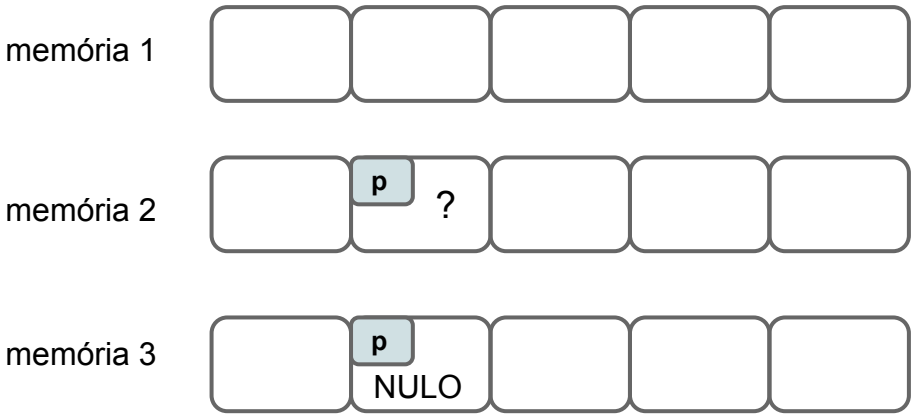


# Variável Ponteiro

- Pode-se atribuir uma variável ponteiro com um valor especial (NULO) para denotar que esta variável ainda não aponta para nenhuma memória atribuída pelo algoritmo

# Variável Ponteiro

```
programa Exemplo()
{memória 1}
    var p: ^Inteiro
{memória 2}
    p ← NULO
{memória 3}
```



# Variável Ponteiro

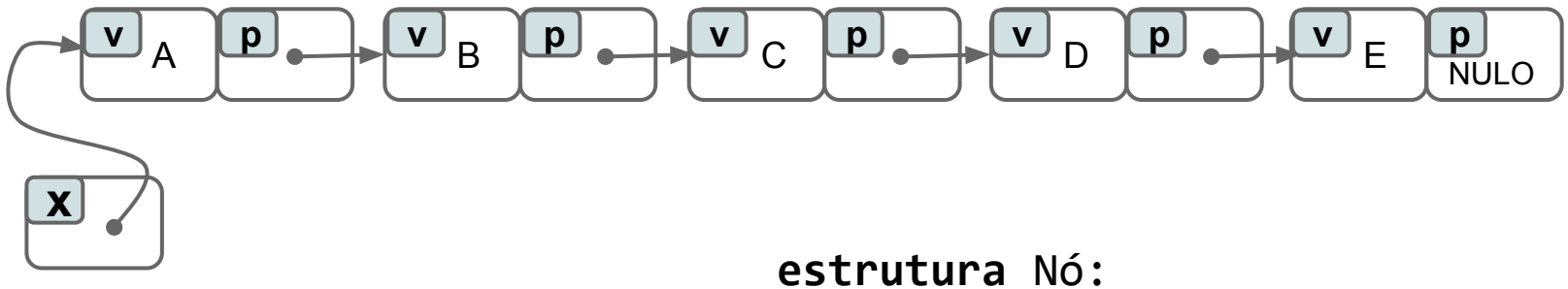
Quando o uso de ponteiros se faz necessário:

- A quantidade de memória necessária é imprevisível
  - Ex: programa que armazena um banco de dados em memória
  - a maior parte dos programas de uso prático possui esta característica

(continua...)

# Variável Ponteiro

(... continuação)




```
estrutura Nó:
    v: Caractere
    p: ^Nó

var x: ^Nó
```

(continua...)

# Variável Ponteiro

(... continuação)

- O desempenho de transferência de dados em memória possui relevância
  - Neste caso, movimenta-se apenas os ponteiros aos dados ao invés dos dados

# Variável Ponteiro

Cuidados com o uso de ponteiros:

- Alocação e desalocação de memória é de responsabilidade do programador
  - Perigo de vazamento de memória!

```
p: ^Inteiro
alocar(p)
p^ ← 10
alocar(p)   // "perdido" memória alocada anteriormente!
```

(continua...)

# Variável Ponteiro

(... continuação)

- Operações sem sentido podem ser feitas com consequências desastrosas

```
programa SemSentido()
    ...
    var p: ^Inteiro
    alocar(p)
    p ← p + 1
    desalocar(p)
    ...
```

# Variável Ponteiro

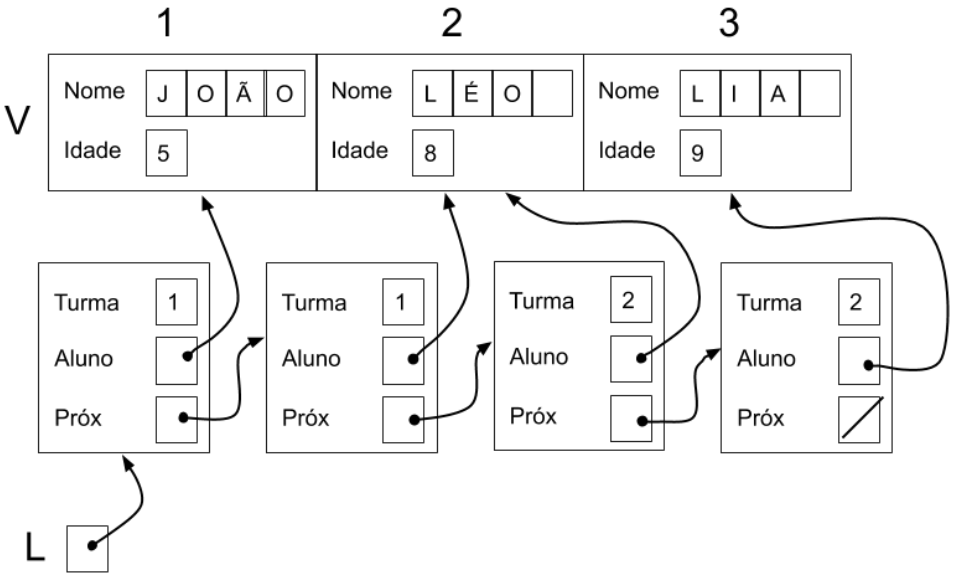
**Exercício**:

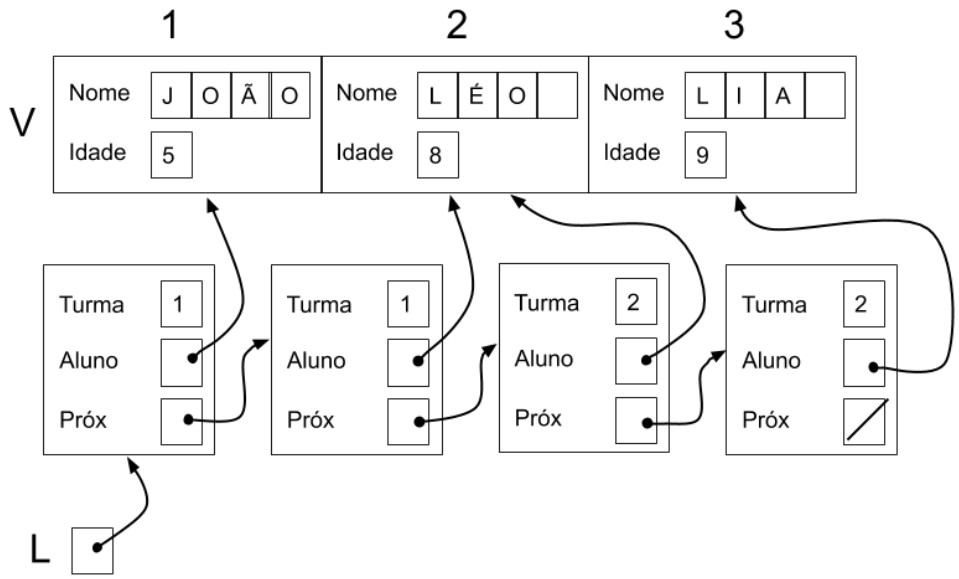
Escrever um algoritmo que armazene 3 contatos em um vetor, no qual cada elemento é um ponteiro para um contato. Usar definição existente de estrutura de Contato.

# Variável Ponteiro

## **Exercício**:

Elabore um algoritmo cujo estado de memória, ao final, poderia ser representado pela figura abaixo.




1
2
3
V
Nome
J O Ã O
Idade
5
Nome
L É O
Idade
8
Nome
L I A
Idade
9
Turma
1
Aluno
Próx
Turma
1
Aluno
Próx
Turma
2
Aluno
Próx
Turma
2
Aluno
Próx
L

# Variáveis como Parâmetros de Procedimentos e Funções

# Variável como Parâmetro

- As variáveis podem ainda ser declaradas como parâmetros de procedimentos e funções

- Nestes casos, deve-se indicar se as variáveis são passadas **por valor** ou **por referência**

# Variável como Parâmetro

- **Declaração**:

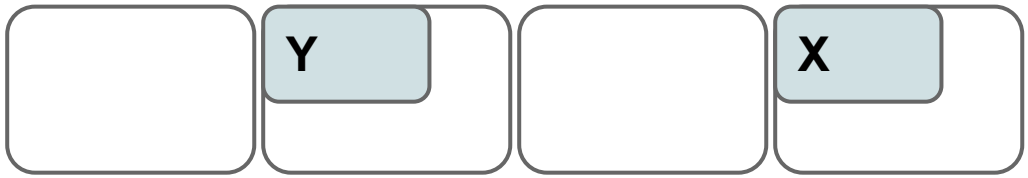
```
procedimento <nome> (val | ref
<declaração de variável>, ...)
```

se **ref** não for especificado, assume-se **val** por padrão

# Variável como Parâmetro

```
função CalculaDobro(val X: Inteiro): Inteiro
    X ← X*2
    retorna (X)

var Y: Inteiro
Y ← 10
escrever (CalculaDobro(Y))
escrever (Y)
```




**Resultado:**

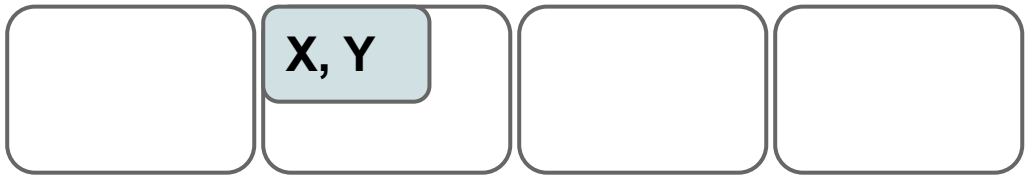
20

10

# Variável como Parâmetro

```
função CalculaDobro(ref X: Inteiro): Inteiro
    X ← X*2
    retorna (X)

var Y: Inteiro
Y ← 10
escrever (CalculaDobro(Y))
escrever (Y)
```




**Resultado:**

20

20

# Variável como Parâmetro

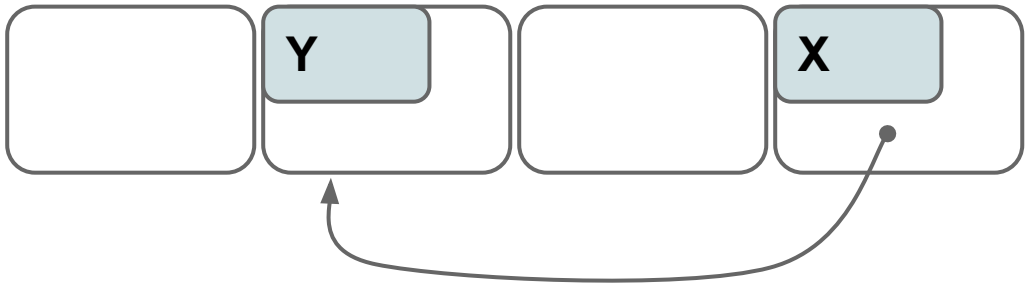
- Note que é possível transformar uma variável que está sendo passada por valor para que ela se comporte como se estivesse sendo passado por referência, e vice-versa

- Isto é útil pois nem todas as linguagens implementam os dois mecanismos de passagem de parâmetros

# Variável como Parâmetro

```
função CalculaDobro(val X: ^Inteiro): Inteiro
    X^ ← X^ * 2
    retorna (X^)

var Y: Inteiro
Y ← 10
escrever (CalculaDobro(@Y))
escrever (Y)
```




**Resultado:**

20

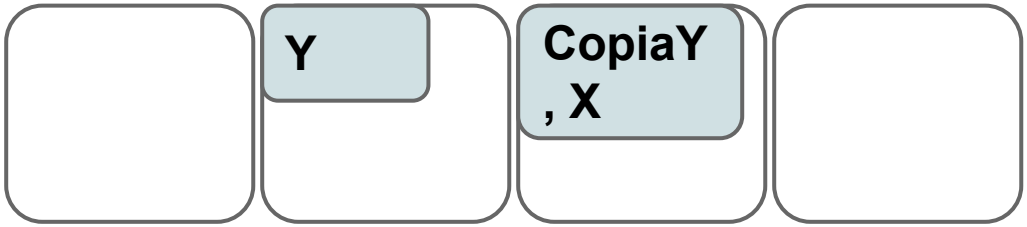
20

Note que a passagem é **por valor**, mas o resultado é como se a passagem fosse **por referência**!

# Variável como Parâmetro

```
função CalculaDobro(ref X: Inteiro): Inteiro
    X ← X*2
    retorna (X)

var Y, CopiaY: Inteiro
Y ← 10
CopiaY ← Y
escrever (CalculaDobro(CopiaY))
escrever (Y)
```




**Resultado:**

20

10

Note que a passagem é **por referência**, mas o resultado é como se a passagem fosse **por valor**!

# Alocação de Espaço

# Análise de Complexidade

- Como medimos complexidade de espaço?

| Tipo de Variável | Complexidade de Espaço |
|---|---|
| Escalar | $\theta(1)$ |
| Vetor/Matriz A, cada elemento de tipo X | $\theta(\|A\|) \cdot$ <Espaço de X> |
| Estrutura E, com campos de tipos $T_1$, $T_2$, …, $T_K$ | $\sum$ { <Espaço de $T_i$> : i = 1,...,K } |
| Ponteiro para tipo X | $\theta(1)$ |

# Exercícios

# Exercícios

1. Em cada item abaixo, um algoritmo precisa de guardar os dados especificados. Para cada item, decida que tipo de variável é o mais apropriado e faça a declaração da variável correspondente:

   (a) um número, num algoritmo que calcula o quadrado de tal número
   (b) uma frase, num algoritmo que inverte tal frase
   (c) uma tabela de números, num algoritmo que guarda o índice pluviométrico de determinada região mês a mês por 12 anos
   (d) as notas dos alunos de determinado curso, num algoritmo que calcula a média da turma
   (e) os lados de um triângulo, num algoritmo que calcula a área de tal triângulo
   (f) os lados de diversos triângulos, num algoritmo de computação gráfica que manipule diversos triângulos para formar poliedros
   (g) o nome, idade, e endereço de um cliente, num algoritmo que registra uma dada venda de produtos de uma loja

# Exercícios

2. Para cada um dos problemas abaixo, a entrada deve ser armazenada em variáveis e somente após toda a entrada ser lida é que o processamento deve começar a ocorrer.

   a. Elabore um algoritmo que peça ao usuário o nome dele e o cumprimente usando tal nome.

   b. Modifique o programa anterior de modo que apenas os usuários Ana e João sejam cumprimentados de forma especial; para os demais, um cumprimento genérico deve ser dado.

   c. Escreva um programa que imprima uma tabela de multiplicação para números até 12. O formato de saída deve ser como uma tabela com 12 colunas e 12 linhas, na qual a célula de linha i coluna j deve possuir o valor i*j.

   d. Escreva um programa que peça ao usuário um número n e imprima a soma dos números 1 a n.

# Exercícios

2.

e. Modificar o programa anterior de modo que apenas múltiplos de 3 ou 5 sejam considerados na soma. Por exemplo, se n=17, o somatório deve considerar apenas os números 3,5,6,9,10,12,15.

f. Modificar o programa anterior de modo que além de n, seja pedido um valor k seguido de k inteiros que consistem da lista dos múltiplos que devem ser considerados. Por exemplo, se n=17, k=3 seguido dos números 2,3,5, então o somatório deve considerar apenas os números 2,3,4,5,6,8,9,10,12,14,15,16.

g. Modificar o programa anterior de modo que ao invés de um único valor n, o usuário entre com a quantidade q de somas que deseja fazer, seguido de q valores $n_1,...,n_q$, seguido do valor k e dos k múltiplos a considerar. A saída deve ser o valor de cada um dos q somatórios (o i-ésimo somatório é a soma dos números entre 1 e $n_i$ que são múltiplos de qualquer um dos k múltiplos informados).

# Exercícios

2.

h. Escreva um programa que imprime todos os números primos até certo número n solicitado ao usuário.

i. Escreva um jogo de adivinhação onde o usuário tem que adivinhar um número secreto, escolhido aleatoriamente pelo programa. Depois de cada palpite o programa diz ao usuário se seu número era maior ou menor que o escolhido pelo programa. Quando finalmente o número secreto for adivinhado,, o número de tentativas deve ser impresso. Tentativas do mesmo número devem ser contabilizadas apenas como uma tentativa. [Naturalmente, este exercício deve ser interativo, isto é, a uma entrada segue de processamento, seguido de nova entrada, em seguida por processamento, e assim por diante, ao contrário do dito no enunciado da questão geral]

j. Escreva um programa que compute, dado um inteiro n, o valor da expressão:
$4 \cdot \sum \{ (-1)^{k+1} / (2k-1) : k = 1..10^n \}$

# Exercícios

3. Resolva os seguintes problemas:

   a. Escreva uma função que recebe um vetor de inteiros, um inteiro N, e retorne o maior elemento entre os N primeiros.

   b. Escreva um procedimento que recebe um vetor L de inteiros, um inteiro N, e inverta os N primeiros elementos de L (isto é, o primeiro se tornará o último, o segundo se tornará o penúltimo, etc.). A complexidade de espaço auxiliar deve ser constante.

   c. Escreva uma função que recebe um vetor L de inteiros, um inteiro N, um inteiro x, e retorne um valor lógico indicando se x ocorre entre os primeiros N elementos de L.

   d. Escreva uma função que que recebe um vetor L de caracteres, um inteiro N, e retorne um valor lógico indicando se a cadeia formada pelos N primeiros elementos de L é um palíndromo.

# Exercícios

3.

e. Escreva uma função que receba dois vetores A,B de caracteres, dois inteiros N,M, e retorne um vetor C com a concatenação dos N primeiros elementos de A com os M primeiros de B. Ex: A = [ "R","E","S", "X"], B = ["U","L","T","A","D","O","Y","Z"], N=3, M=6, deve resultar no vetor ["R","E","S","U","L","T","A","D","O"]

f. Escreva uma função que receba dois vetores A,B de caracteres, dois inteiros N,M, e retorne um vetor C com os N primeiros elementos de A e os M primeiros de B alternados. Ex: A = [ "R","S","L","X"], B = ["E","U","T","A","D","O","Y","Z"], N=3, M=6, deve resultar no vetor ["R","E","S","U","L","T","A","D","O"]

g. Escreva um procedimento que receba um vetor A de caracteres, dois inteiros N,K, e gire os N primeiros elementos de A de forma circular em K posições à direita. Ex: A = ["R","E","S","U","L","T","A","D","O","X","Y"], N=9, K=3, deve resultar na transformação de A em ["A","D","O","R","E","S","U","L","T","X","Y"]. A complexidade espaço auxiliar deve ser constante a de tempo θ(N).

# Exercícios

3.

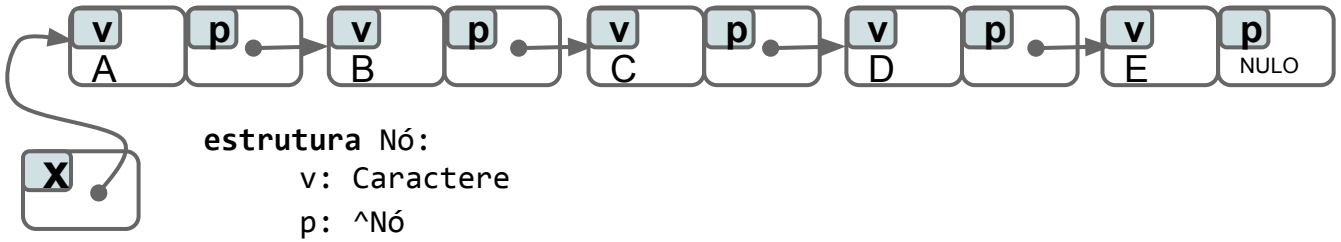
h. Escreva uma função que receba um inteiro N e retorne um vetor cujos N primeiros elementos são os N primeiros números de Fibonacci (o primeiro número de Fibonacci é 0, o segundo é 1, e o terceiro em diante é calculado como a soma dos dois anteriores). Ex.: N=6 deve resultar em [0,1,1,2,3,5].

i. Escreva uma função que recebe um inteiro N e retorne um vetor de inteiros no qual os elementos, do primeiro ao último, consistem dos dígitos de N. Ex.: N=7023 deve resultar em [7,0,2,3].

j. Escreva funções que adicionam, subtraem e multiplicam dois números passados como dois vetores A, B e dois inteiros N,M, de modo que os N (resp. M) primeiros elementos do vetor A (resp. B) contém os dígitos do primeiro (resp. segundo) número. O retorno deve consistir de um vetor com o resultado também como vetor de dígitos. Ex: A=[2,4,9], N=3, B = [7,2], M=2, deve resultar para soma no vetor [3,2,1], para a subtração no vetor [1,7,7] e para a multiplicação no vetor [1,7,9,2,8].

# Exercícios

4. Escrever um algoritmo que carregue dados para a memória através de declaração e atribuição de variáveis, de modo que os dados carregados em memória possam ser representados pelo esquema abaixo. Em seguida, o programa deve desalocá-los.




```
estrutura Nó:
      v: Caractere
      p: ^Nó
var x: ^Nó
```

Note que:

(a) as setas representam ponteiros; quadrados azuis marcam memória que está alocada para o algoritmo (quando rotulados, existe uma variável com aquele nome com conteúdo naquela porção de memória)

(b) As porções que aparecem em duplas (ex: "A" e ponteiro para uma outra dupla) são mapeadas em áreas contíguas de memória

(c) O algoritmo pode usar variáveis extras de modo que a porção alocada de memória seja ainda maior que a exibida acima

# Exercícios

5. O que há de errado com o programa abaixo?

```
programa ErroAlocacao()

    estrutura Aluno:
        matr: Inteiro
        prox: ^Aluno

    função CarregarLista(): ^Aluno
        var Alunos[1..3]: Aluno
        Alunos[1].matr ← 1234
        Alunos[2].matr ← 233
        Alunos[3].matr ← 555
        ...
        ...

        Alunos[1].prox ← @Alunos[2]
        Alunos[2].prox ← @Alunos[3]
        Alunos[3].prox ← NULO
        retornar @Alunos[1]

var p: ^Aluno

p ← CarregarLista()
escrever(p^.prox^.prox^.matr)
```

# Exercícios

6. Faça um algoritmo que leia o estado de um jogo da velha e escreva ou o símbolo ("X" ou "O") do jogador que ganhou, ou escreva "Velha" se não é possível que nenhum jogador ganhe, ou "Em andamento" para indicar que o resultado da partida ainda é indefinida.

7. Faça um algoritmo como no exercício anterior, mas considere que o jogo da velha está sendo jogado em um tabuleiro de NxN quadrados (o original é jogado em 3x3) e vence o jogador que conseguir executar N jogadas alinhadas em horizontal, vertical ou diagonal.

8. Faça um algoritmo que estime o valor da constante matemática π apenas utilizando soma/subtração, multiplicação/divisão e o gerador de números aleatórios. (Dica: considere o experimento de sortear um ponto qualquer p interno a um quadrado de lado 2, onde qualquer ponto interno tem igual probabilidade de ser sorteado. Qual a probabilidade de que p pertença ao círculo inscrito no quadrado? Note que, na realização de um número grande de tais sorteios, a razão de pontos sorteados que pertencem ao círculo em relação ao total de tais pontos se aproximará do valor da probabilidade calculada.)

# Exercícios

9. Dado um vetor com N elementos:
   a. permutar em tempo θ(N) e espaço θ(1) seus elementos de forma que resulte com igual probabilidade em qualquer das N! permutações possíveis.
   b. dado natural M, escolher M inteiros do vetor em tempo θ(M) e espaço θ(1) de forma que qualquer elemento tenha igual probabilidade de ser escolhido.

10. Dado natural N, determinar o número de zeros no fim de N!. Exemplo: **Entrada:** N = 11; **Saída:** 2; **Entrada:** N = 100; **Saída:** 24; **Entrada:** N = 500; **Saída:** 124.

11. Dado o número H representando as horas e M os minutos, determinar o menor ângulo entre os ponteiros de um relógio que marca este horário.

12. Dado um natural N, crie uma função que calcule $\sqrt{(N)}$ utilizando-se apenas as 4 operações básicas de aritmética. (Dica: se $x = \sqrt{(N)}$, então $N/x = x$. Se $x \neq \sqrt{(N)}$, um algoritmo iterativo pode incrementalmente utilizar o valor de N/x para aproximar o valor de $\sqrt{(N)}$.)

# Exercícios

13. Escreva um algoritmo que leia N alunos (matrícula, nome e idade). Em seguida, o algoritmo deve ler M matrículas e, para cada matrícula, deve escrever o nome do aluno correspondente, buscando nos registros previamente lidos.

14. Dado um vetor de N caracteres, determinar se há repetição de caracteres.

15. Duas palavras são anagramas se uma palavra pode se tornar igual a outra por um rearranjo na ordem de suas letras. Por exemplo, as palavras computação e taãopmoçuc são anagramas. Dados dois vetores A[1..N]: Caractere e B[1..N]: Caractere representando duas palavras com N caracteres, determinar se são anagramas. Crie algoritmos com as seguintes ideias:
    a. marcando a ocorrência de cada caractere de A em um correspondente em B
    b. ordenando-se os caracteres de ambos os vetores A e B
    c. contando-se o número de ocorrência de cada caractere de A e B

# Exercícios

16. Imagens *bitmap* são armazenadas em memória em geral usando-se uma matriz, onde cada elemento da matriz representa um pixel da imagem. Cada pixel constitui de três valores naturais, chamados r, g, b, no intervalo de 0 a 255, indicando a intensidade de cada uma das cores básicas (vermelho, verde e azul). Dado uma imagem com N pixels de largura por N de altura, crie um algoritmo para rotacionar a figura em 90 graus usando espaço auxiliar constante.

17. Remover valores duplicados de um vetor não-ordenado A[1..N] usando espaço auxiliar constante. Atualizar o valor de N para delimitar onde terminam os elementos que permaneceram em A (os elementos na porção A[1..N] tornaram-se todos distintos). Ex.: **Entrada:** A = [1 3 2 2 1 3 4 1]; N = 8; **Saída:** A = [1 3 2 4 0 0 0 0]; N = 4.

18. Usando o comando de gerar números aleatórios com probabilidade uniforme existente na sua linguagem de programação, faça uma função que emule um dado viciado, retornando os números de 1 a 6 com probabilidade de 40%, 20%, 20%, 10%, 9% e 1%, respectivamente. Para certificar que a função esteja funcionando, obtenha 1 milhão de resultados desta função, determine a frequência de retorno de cada número e compare com o percentual requisitado.

# Exercícios

19. Dada uma tabela M[1..N, 1..N] de inteiros, zerar todos os valores numa linha e numa coluna de M que possua originalmente algum valor 0, ou seja, atribuir 0 a todos os valores da linha i e da coluna j se M[i, j] = 0. O algoritmo deve executar em tempo $O(N^2)$ e espaço auxiliar O(N). Ex.:

| **Entrada:** N = 5, | **Saída:** |
|---|---|
| M = (1 1 2 1 0) | M = (0 0 0 0 0) |
| (4 0 1 3 4) | (0 0 0 0 0) |
| (2 5 7 9 1) | (2 0 7 9 0) |
| (2 5 3 8 1) | (2 0 3 8 0) |
| (2 0 1 1 1) | (0 0 0 0 0) |

20. Dados N pontos em $\mathbb{R}^2$, descobrir o maior número de pontos colineares em tempo $O(N^2)$.

# Exercícios

21. Dado um vetor A com N caracteres, um caractere X e um vetor S de M caracteres, substituir cada ocorrência de X em A[1..N] por S[1..M] usando espaço auxiliar constante. Atualizar N com o novo tamanho da cadeia. Ex.: **Entrada:** N = 5, A = [a,b,c,a,b,x,y,x,y,x,y], X = "a", S = [d,e,f], M = 3, **Saída:** A = [d,e,f,b,c,d,e,f,b,x,y], N = 9

22. Usando por base uma função supostamente existente que retorna um número entre 0 (inclusive) e 5 (inclusive) com distribuição uniforme (i.e., a probabilidade de qualquer retorno é a mesma), projetar uma função que retorne um número entre 0 (inclusive) e 7 (inclusive) com distribuição uniforme.