Vetores e Strings

Agenda

- Vetores
- Strings
- Exercícios

Motivação para o uso de Vetores

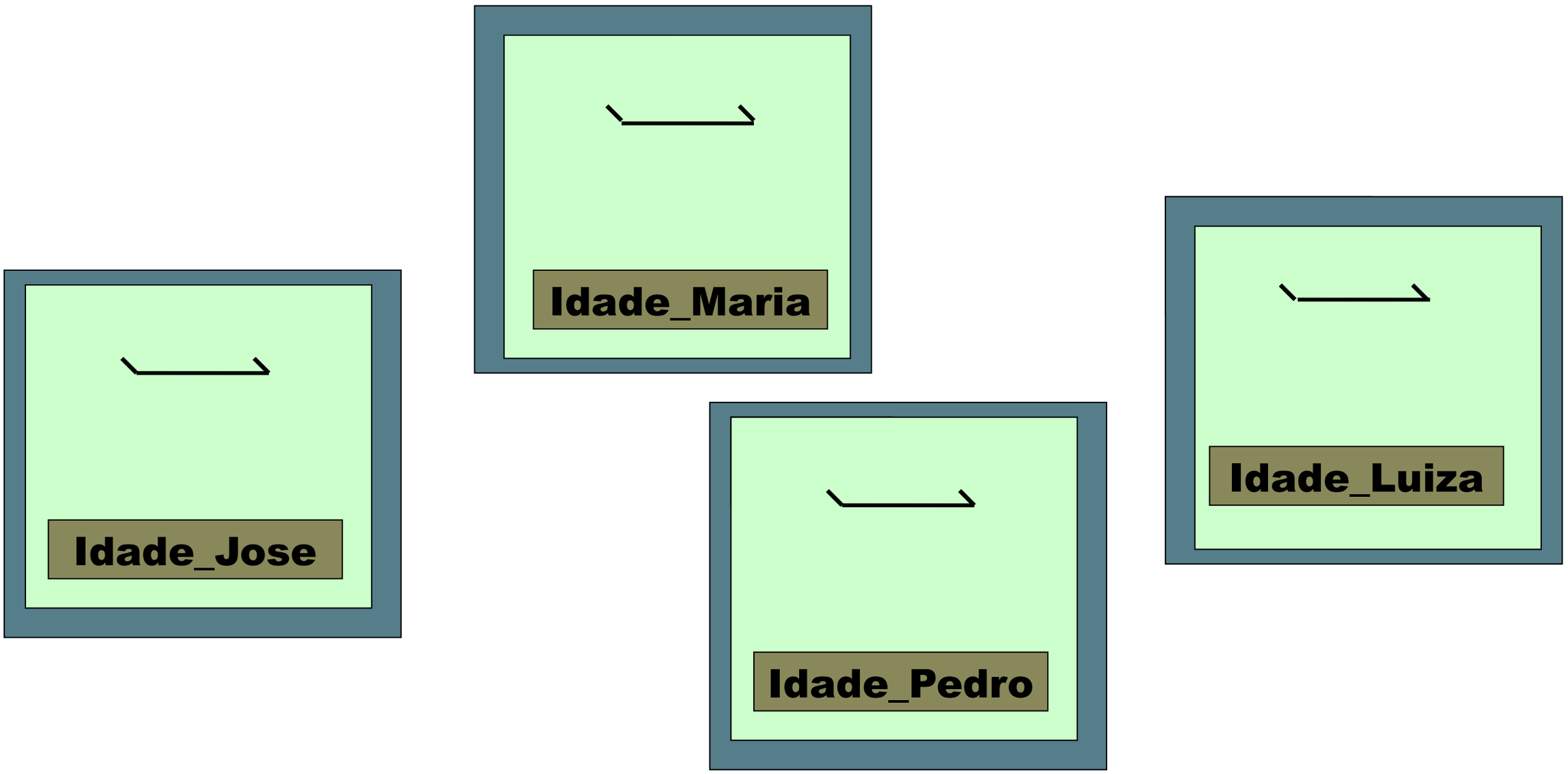
- Quando desejamos utilizar diversas variáveis de um mesmo tipo, pode ser inviável declarar cada uma dessas variáveis.

Problema

- Preciso de um software que armazene a idade, em anos, de quatro pessoas:
  - Maria
  - José
  - Pedro
  - Luiza;

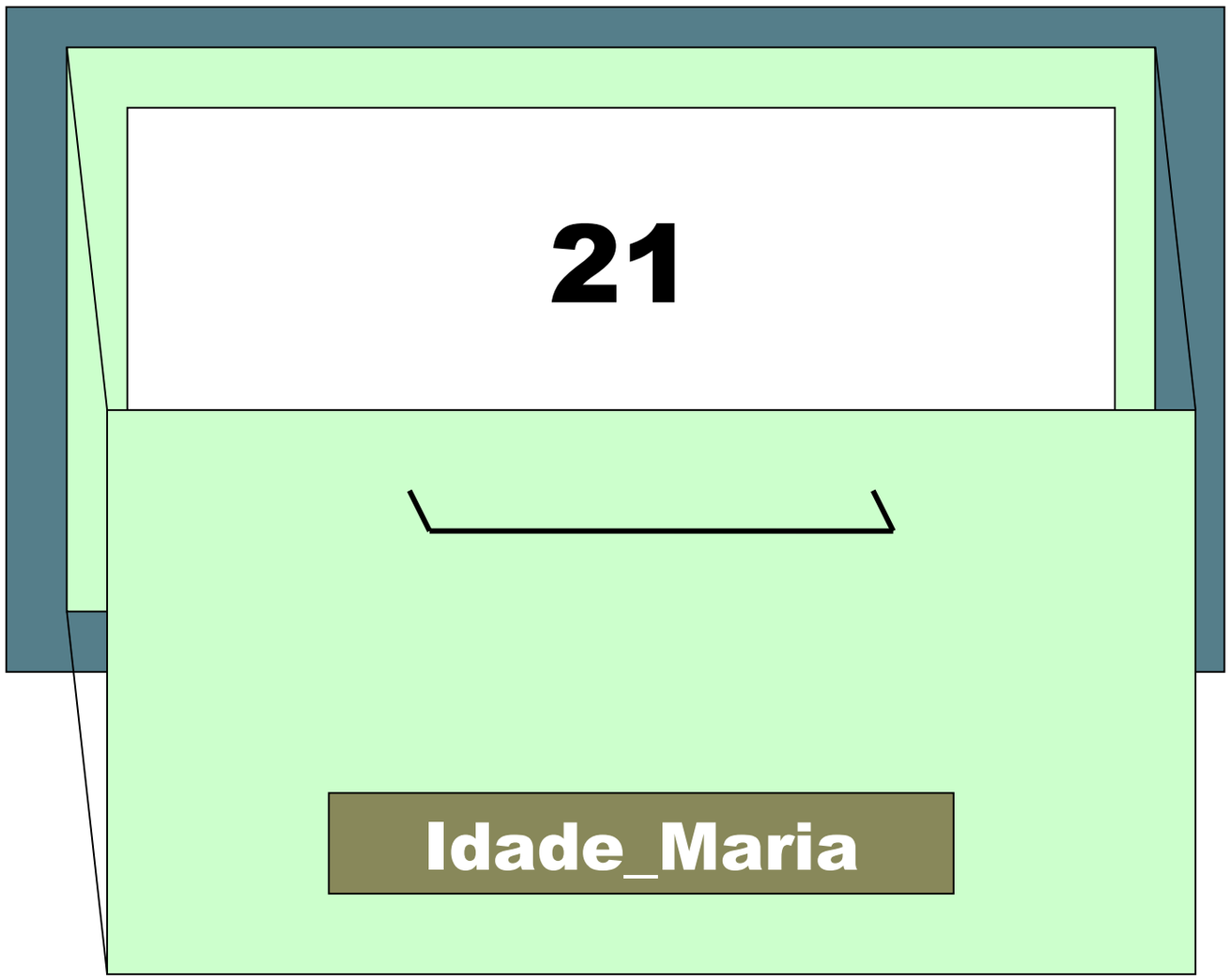
- Neste caso, precisamos então criar quatro lugares (caixinhas) na memória para guardar, em cada uma, a idade de uma das pessoas.

Variáveis



Todas as variáveis guardam um conteúdo de mesmo significado e são do mesmo tipo de dados.

- Variável Idade_Maria



Pergunta

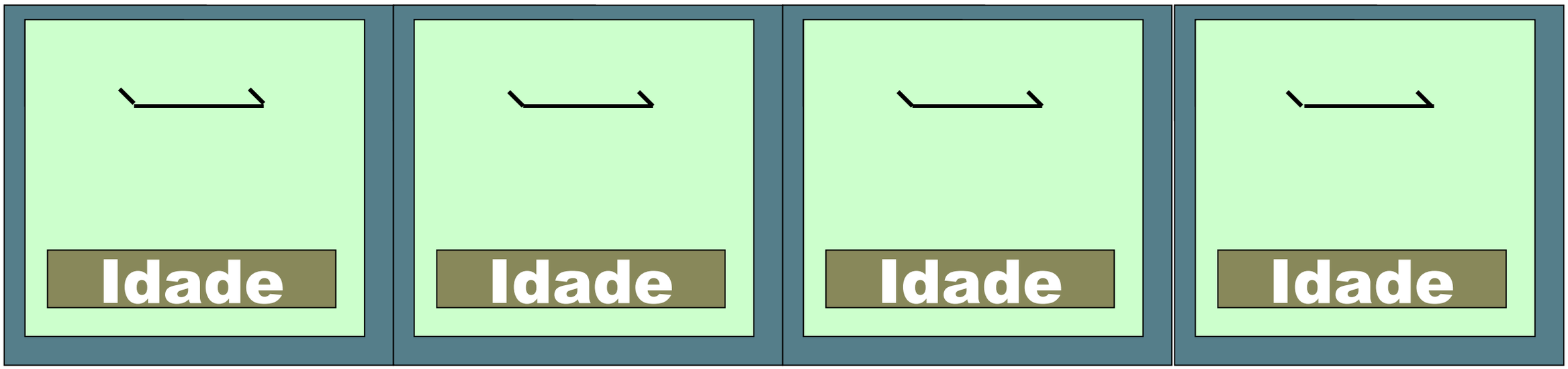
- Para guardar 100 idades de 100 pessoas distintas, o que precisamos fazer?
    - Até então, criar 100 variáveis. Uma para cada pessoa.
- E se tivermos que guardar as idades de 1000 pessoas?
- Será que não existe nada mais prático?

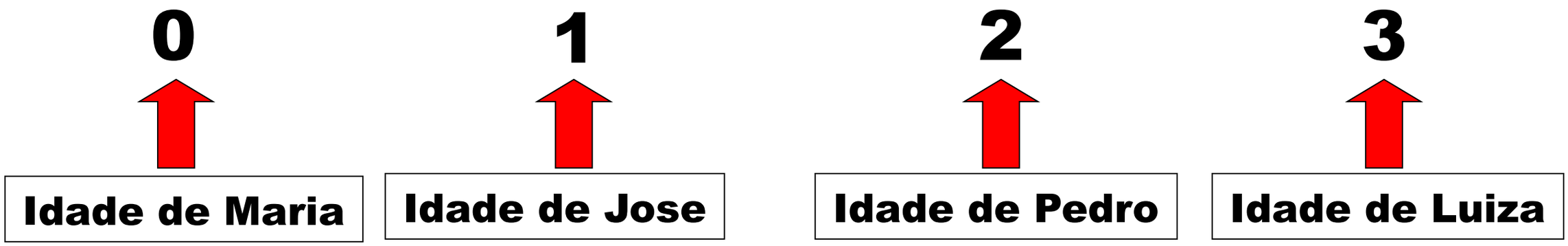
Vetor

- Para situações como esta, apresentada anteriormente, foi criada uma estrutura conhecida como VETOR;

- Um vetor nada mais é do que UMA variável com diversas posições (caixinhas) numeradas. Onde pode-se guardar diversos valores (um em cada caixinha) do mesmo tipo.

- Variável Idade (Vetor)





Algoritmo

Quantidade de idades

```
Início
inteiro Idade[4]
  Idade[0] = 21
  Idade[1] = 42
  Idade[2] = 55
  Idade[3] = 10
  escreva (“Idade de Maria: ”, Idade[0])
  escreva (“Idade de José: ”, Idade[1])
  escreva (“Idade de Pedro: ”, Idade[2])
  escreva (“Idade de Luiza: ”, Idade[3])
fim
```

Idade	21	42	55	10
	0	1	2	3

Posição de Armazenagem

Algoritmo – Entrada Via Teclado

```
início
  inteiro Idade[4]
  escreva (“Digite a idade de Maria: ”)
  leia (Idade[0])
  escreva (“Digite a idade de José: ”)
  leia (Idade[1])
  escreva (“Digite a idade de Pedro: ”)
  leia (Idade[2])
  escreva (“Digite a idade de Luiza: ”)
  leia (Idade[3])

   escreva (“Idade de Maria: ”, Idade[0])
   escreva (“Idade de José: ”, Idade[1])
   escreva (“Idade de Pedro: ”, Idade[2])
   escreva (“Idade de Luiza: ”, Idade[3])
fim
```

Idade

50	25	10	5
0	1	2	3

Algoritmo – Entrada Via Teclado (Loop)

```
Início
  inteiro Idade[4]
  inteiro cont

  para cont de 0 até 3 passo 1 faça
     escreva ("Digite a idade: ")
     leia (Idade[cont])
  fim para

  cont = 0
  enquanto (cont < 4) faça
     escreva ("Idade: ", Idade[cont])
     cont = cont + 1
  fim enquanto
fim
```

cont
0
1
2
3
4

Idade

10	20	15	18
0	1	2	3

Algoritmo – Entrada Via Teclado (Loop)

- Vamos resolver o problema proposto anteriormente. Ou seja, um algoritmo para guardar e imprimir 1000 idades distintas:

```
início
    inteiro Idade[1000]
    inteiro i
    para i de 0 até 999 passo 1 faça
        escreva (“Digite a idade: ”)
        leia (Idade[i])
    fim para

    i = 0
    enquanto (i < 1000) faça
        escreva (“Idade: ”, Idade[i])
        i = i + 1
    fim enquanto
fim
```

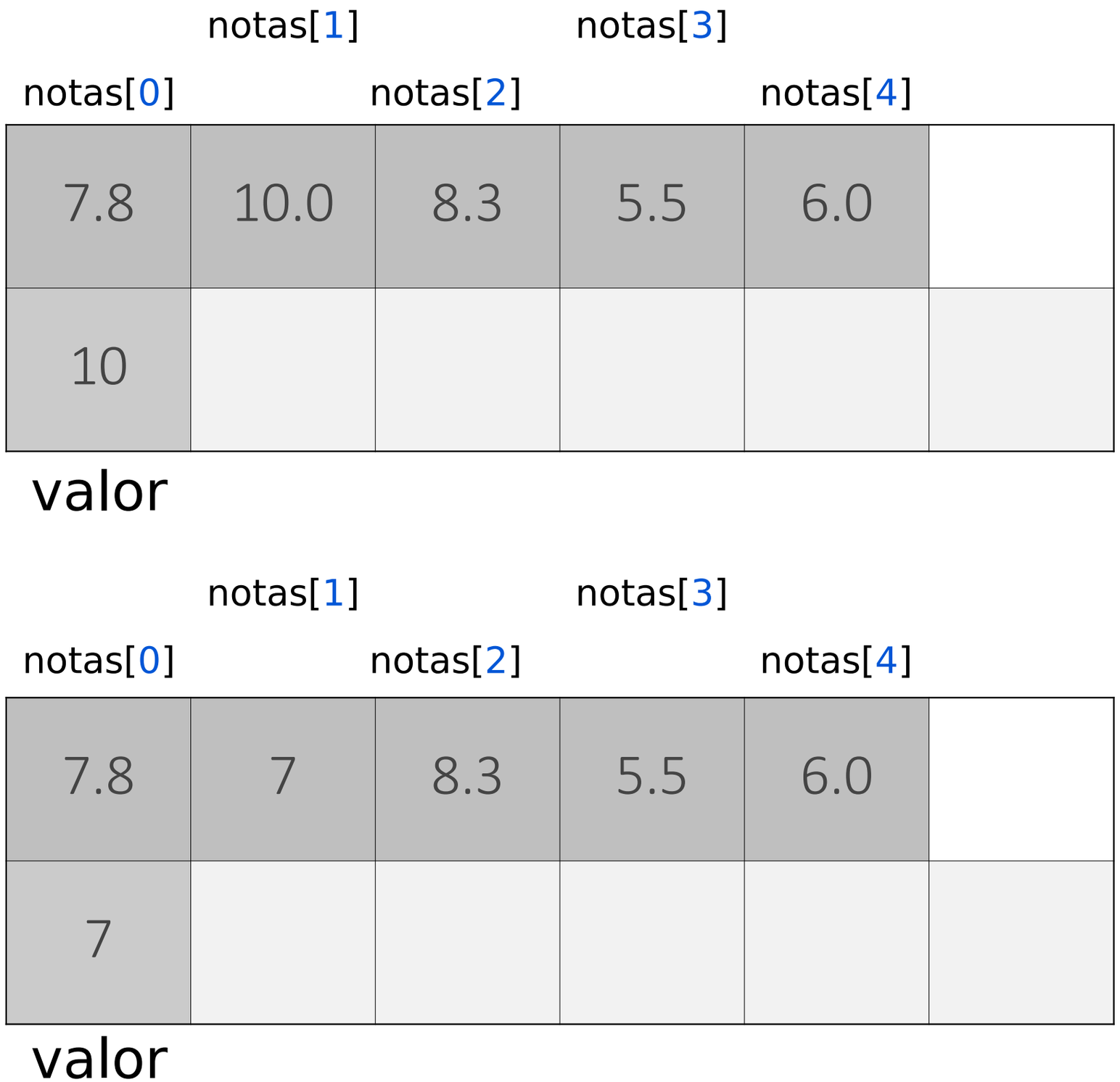
Declaração de Vetor

tipo_vetor nome_do_vetor [tamanho_do_vetor];

- tamanho_do_vetor é um número inteiro ou uma variável do tipo int
- Esta declaração cria tamanho_do_vetor variáveis do tipo tipo_vetor.
- As variáveis criadas pelo vetor são acessadas por:
  - nome_do_vetor[0]
  - nome_do_vetor[1]
  - ...
  - nome_do_vetor[tamanho_do_vetor -1]

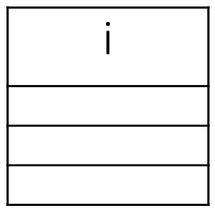
Exemplo de Vetor com índices inteiros

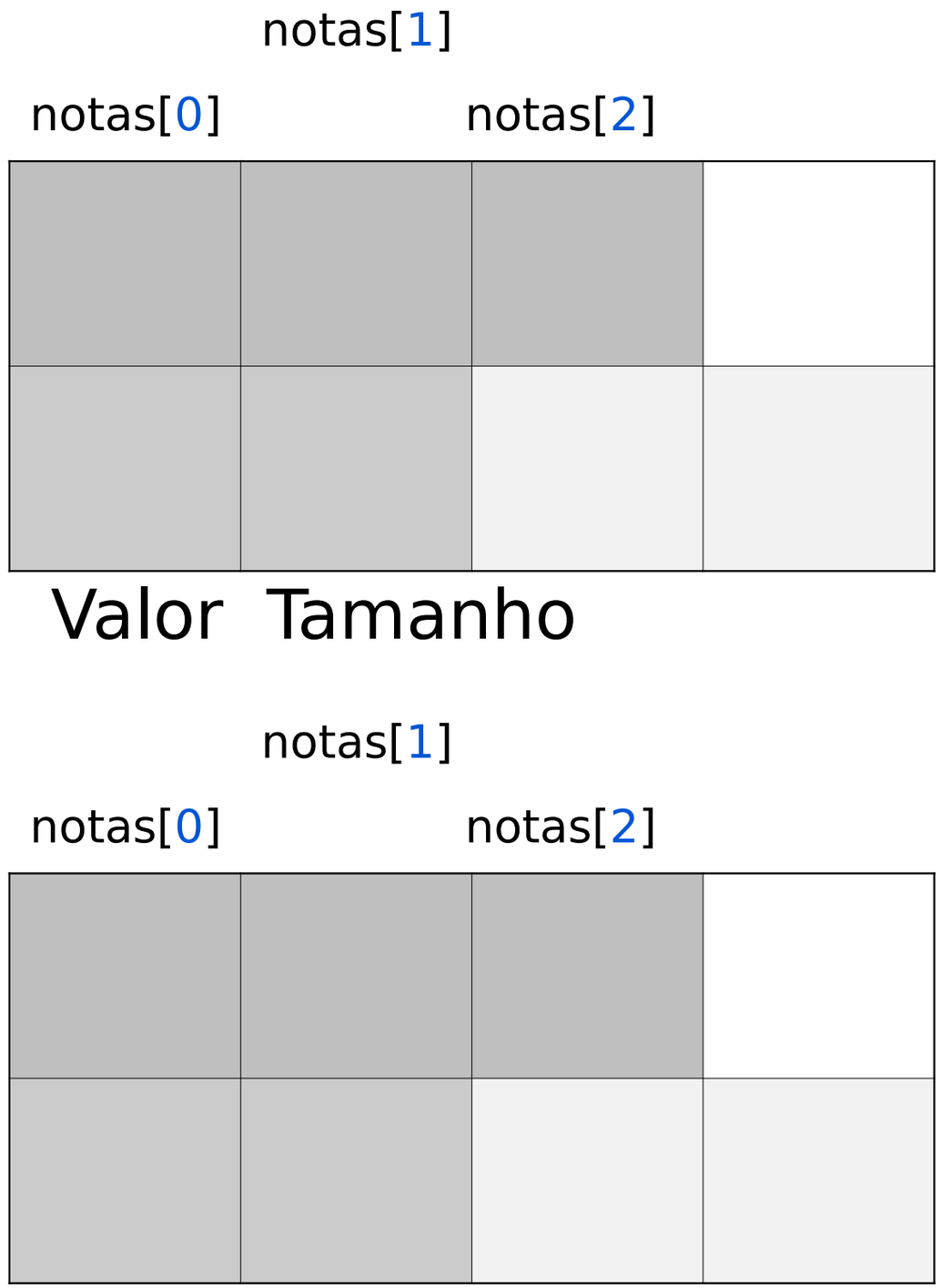
```
float notas [ 5 ];
notas[0] = 7.8;
notas[1] = 10.0;
notas[2] = 8.3;
notas[3] = 5.5;
notas[4] = 6.0;
float valor = notas[1];
valor = valor - 3;
notas[1] = valor;
```



Exemplo de Vetor com variáveis nos índices

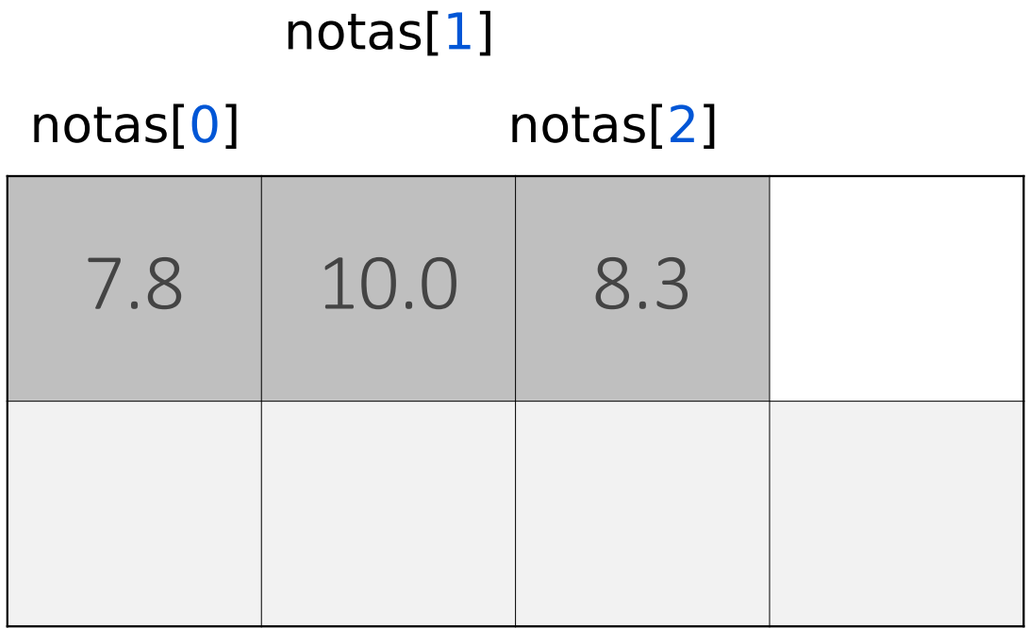
```
int tamanho = 3;

int indice = 0;

float notas [ tamanho ];

notas[indice++] = 7.8;

notas[indice++] = 10.0;

notas[indice++] = 8.3;

float valor = notas[1];

valor = valor - 3;

notas[1] = valor;
```





Acesso a índice fora do limite

```
int tamanho = 3;

int indice = 0;

float notas [ tamanho ];

notas[0] = 7.8;

notas[1] = 10.0;

notas[2] = 8.3;

notas[3] = 9.0; // Este acesso altera uma posição de memória indevida e pode causar um erro se a posição de memória não estiver reservada para o programa atual.
```



Ler, preencher e imprimir um vetor

```
início
    inteiro i, tamanho =5;
    real  notas [ tamanho ];

    para i de 0 até tamanho - 1 passo 1 faça
        escreva (“Digite o valor para o vetor notas[i])
        leia(notas[i])
    fim para

    para i de 0 até tamanho - 1 passo 1 faça
        escreva notas[i])
    fim para
fim
```

Ler, preencher e imprimir um vetor

```
int main(void){
    int i, tamanho =5;
    float notas [ tamanho];

    for (i=0; i< tamanho; i++){
        printf(“Digite o valor da posição notas[%d]:\n”,i);
        scanf(“%f”, &notas[i]);
        }

    printf(“Valores lidos:\n”);

    for (i=0; i< tamanho; i++)
        printf(“notas[%d] = %.2f\n”, i, notas[i]);

        return 0;
}
```

Exemplo: Produto interno de vetores

- Crie 2 vetores de dimensão 5, leia os valores para estes vetores e calcule o produto interno destes vetores.

- Produto interno de dois vetores é a soma dos produtos entre os elementos em posições equivalentes dos vetores.

- Por exemplo, o produto interno dos vetores:
  <2,3,4,5,6> e <3,5,6,1,5>
  é igual a
  2*3 + 3*5 + 4*6 + 5*1+ 6*5

Algoritmo : Produto interno de vetores

```
Inicio

      inteiro i
      real vetor1[5], vetor2[5], resultado

    para i de 0 até 4 passo 1 faça
        escreva ("Digite o valor para vetor1[i])
        leia (vetor1[i]))
    fim para

    para i de 0 até 4 passo 1
        escreva ("Digite o valor para vetor2[i])
        leia (vetor2[i]))
    fim para

    resultado = 0

     para i de 0 até 4 passo 1
        resultado = resultado + ( vetor1[i] * vetor2[i] )
    fim para

    escreva (resultado);

fim
```

Código: Produto interno de vetores

```
int main(void){

    int i; double vetor1[5], vetor2[5], resultado;

    for(i=0; i<5; i++){
    printf("Digite o valor para vetor1[%d]:\n",i);
    scanf("%lf", &vetor1[i]);
    }

    for(i=0; i<5; i++){
    printf("Entre com valor para vetor2[%d]:\n",i);
    scanf("%lf", &vetor2[i]);
    }

    resultado = 0.0;

    for(i=0; i < 5; i++)
    resultado = resultado + ( vetor1[i] * vetor2[i] );

    printf("\nO produto interno é: %lf\n", resultado);

    return 0;
}
```

Strings

Disciplina de Programação de Computadores I
Universidade Federal de Ouro Preto

Cadeias de Caracteres

- A linguagem C não possui o tipo string (cadeia de caracteres) explicitamente, mas podemos considerar um vetor de caracteres como uma string.
- Em C, uma string é sempre terminada pelo caractere especial: ’\0’
- Logo, ao declararmos um vetor de caracteres, devemos somar 1 à quantidade desejada de caracteres!
- Exemplo:
  char st1[7] = “string”;

Declaração e inicialização de cadeiras de caracteres

- Podemos declarar e inicializar um vetor de caracteres de duas formas:

```
char st1[ ] = “string”;
char st2[ ] = {‘s’, ‘t’, ‘r’, ‘i’, ‘n’ , 'g’, '\0'};
```

- O tamanho da variável será a quantidade de caracteres atribuídos MAIS UM, devido ao caractere ‘\0’.

Leitura e impressão de cadeias de caracteres

- Uma cadeia de caracteres é lida ou impressa com o modificador %s
- O armazenamento da leitura é interrompido ao se encontrar um espaço, mesmo que existam mais caracteres depois do espaço.
- Para ler uma cadeia de caracteres contendo espaços, indique que a cadeia deve terminar com a quebra de linha, assim: %[^\n]s
- A impressão da cadeia de caracteres é feita até o último caractere antes de ‘\0’

Cópia de strings

- Strings podem ser copiadas através da função strcpy da biblioteca string.h

```
char st1[ ] = “string”;

char st2[10];

char st3[31];

strcpy(st2, st1);

strcpy(st3, “Programacao de Computadores 1”);
```

Funções para manipulação de Strings em Linguagem C

- **strlen ():** Número de caracteres antes do '\0'
  - Ex: int len = strlen(Nome);

- **strcpy ():** atribui a uma variável do tipo string uma constante ou o valor de outra string;
  - Ex: strcpy(Nome2, Nome1);

- **strcmp ()**:
  - Ex: int result = strcmp(Nome1, Nome2)
    - Pode retornar: maior que 0 (Nome1 maior que Nome2), 0 (Nome1 igual a Nome2) ou menor que 0 (Nome1 menor que Nome2);

- **strcat ():** concatenação;
  - Ex: strcat("saudacoes ", Nome);

Exemplo 1

Fazer um programa para contar o número de vogais de uma string.

Exemplo 1 - Resposta

```
int main()
{
    char texto[100] = "Programacao de Cumputadores 1";
    int i;
    int contador = 0;
    for (i=0; texto[i]!= '\0'; i++) {
        if (texto[i] == 'a' || texto[i] == 'A' ||
            texto[i] == 'e' || texto[i] == 'E' ||
            texto[i] == 'i' || texto[i] == 'I' ||
            texto[i] == 'o' || texto[i] == 'O' ||
            texto[i] == 'u' || texto[i] == 'U')
                contador++;
    }

    printf("A frase tem %i vogais.", contador);

    return 0;
}
```

Exemplo 2

Fazer um programa que peça ao usuário para digitar seu nome completo e depois imprima o nome na tela.

Exemplo 2 - Resposta

```
int main()
{
    char nome[100];

    printf("Digite seu nome completo: ");
    gets(nome);

    printf("Seu nome completo é: ");
    puts(nome);

    return 0;
}
```

Referências Bibliográficas

- Material de aula da disciplina Algoritmos, UFJF: https://sites.google.com/site/algoritmosufjf
- Material de aula do Prof. Ricardo Anido, da UNICAMP: http://www.ic.unicamp.br/~ranido/mc102/
- Material de aula da Profa. Virgínia F. Mota: https://sites.google.com/site/virginiaferm/home/disciplinas
- DEITEL, P; DEITEL, H. *C How to Program*. 6a Ed. Pearson, 2010.

Agradecimentos

- Professores do Departamento de Ciência da Computação da UFJF que gentilmente permitiram a utilização das videoaulas elaboradas por eles.