Vetores, Matrizes, Strings

PROF. MAURICIO A DIAS – MACDIASPAE@GMAIL.COM

INTRODUÇÃO A COMPUTAÇÃO PARA ENGENHARIAS

Seja o problema: **Ler 9 valores, calcular a média aritmética dos mesmos e imprimir a média e os valores iguais ou superiores à média.**

Pergunta-se:

Quantas variáveis serão necessárias para ler os valores?

9?

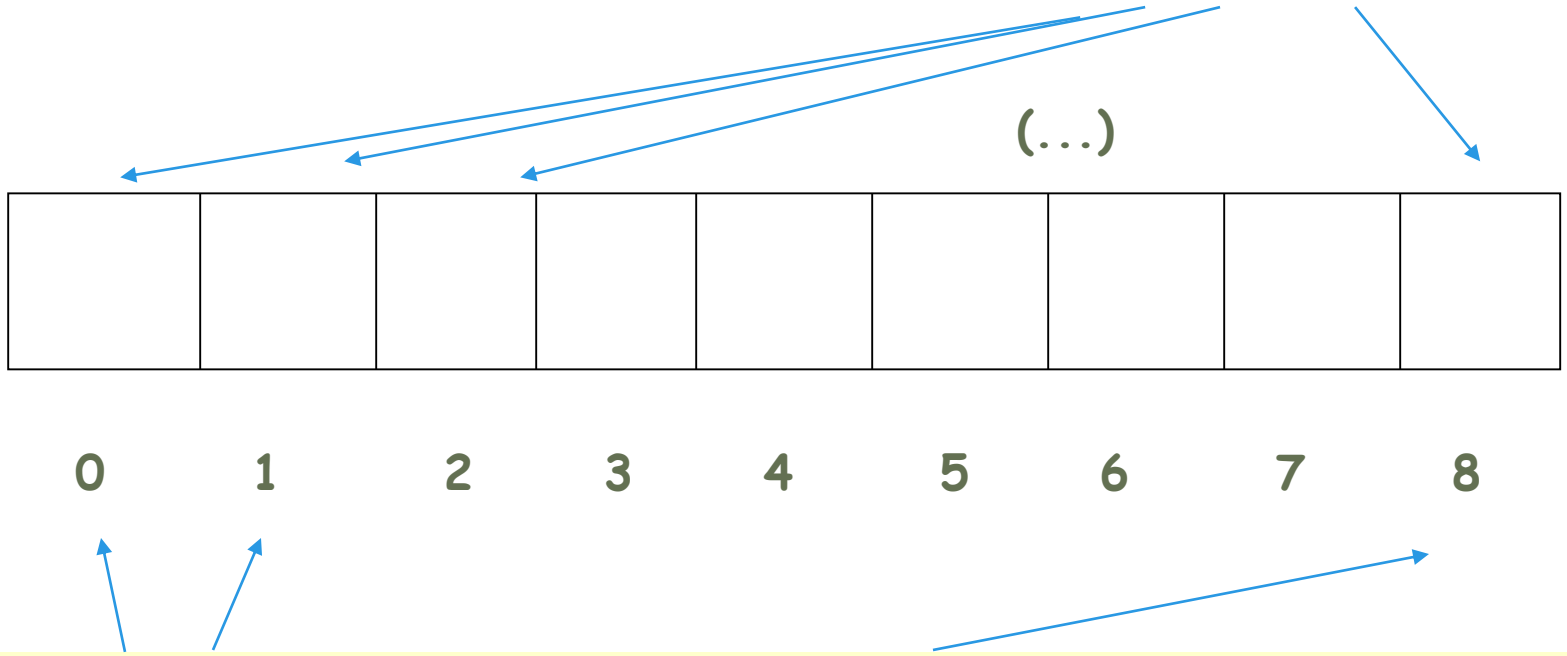
Ou 1?

Resposta: **9 variáveis (de mesmo tipo!)**

Solução para o problema das múltiplas variáveis de mesmo tipo:

um vetor

Nome: comum a todas as variáveis → abc



A posição dentro do vetor (índice) identifica os valores individuais

Ex.: abc[0] ... abc[8]

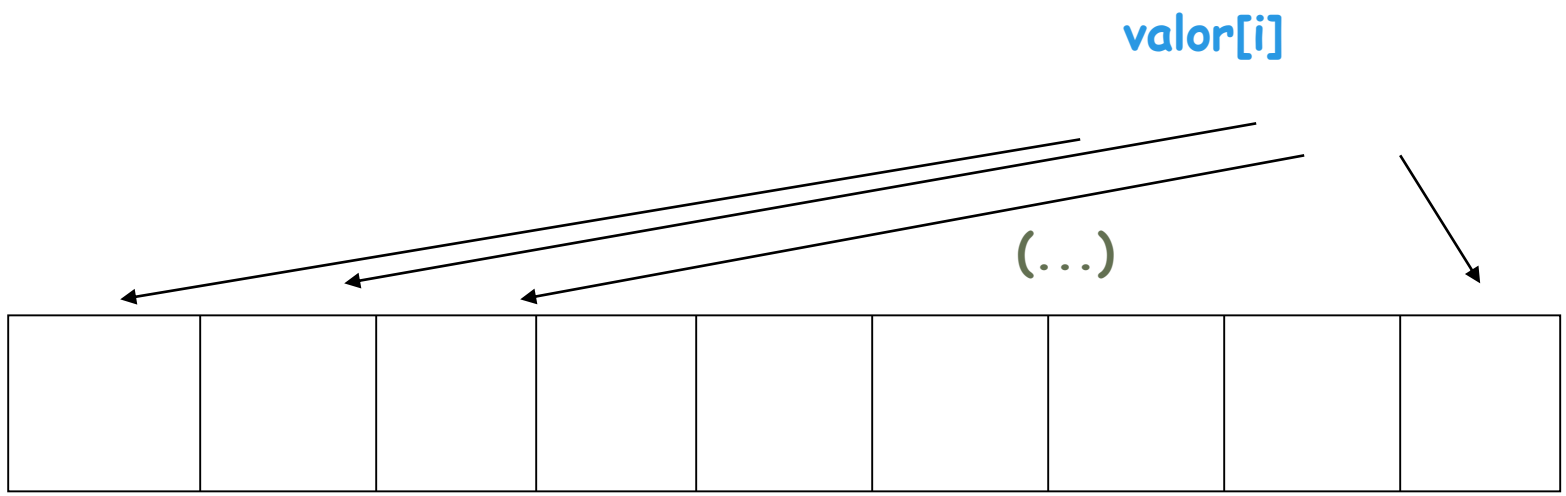
Vetores: variáveis compostas homogêneas

Um só tipo;

Um só nome;

Múltiplas posições de memória identificadas por índices.

Seja um vetor inteiro de 9 elementos chamado valor

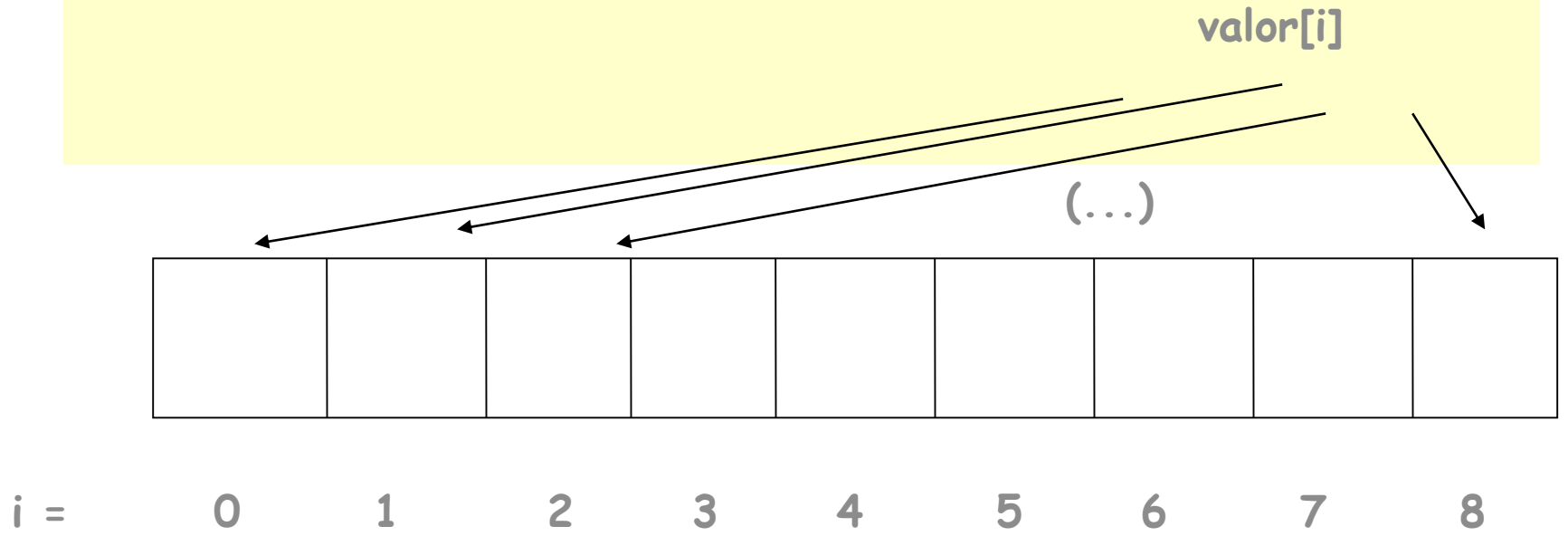


i = 0 1 2 3 4 5 6 7 8

Declaração de um vetor
(= arranjo de uma só dimensão)

Exemplo:

```
int valor[9];
//valor corresponderá a 9 variáveis, com
// índices variando de 0 a 8
```



ATENÇÃO

A primeira posição de um vetor é a posição **zero**.

Ex.: maior = vet[0]; // inicializacao de variável maior
//com o primeiro valor do vetor vet

- O sistema não controla a correção dos índices usados.

  Quem deve garantir que os índices estejam dentro do intervalo correto é o programador.

- *Vetores* não são inicializados automaticamente pelo sistema. Inicialmente contém valores quaisquer (= "lixo").

Algumas formas de inicialização de um vetor:

Por leitura:

```
for (i = 0; i <MAX;i++)
   scanf("%d", &valor[i]);
```

Por atribuição, para garantir valor inicial zero a posições que funcionarão como contadores ou acumuladores:

```
for (i = 0; i <MAX;i++)

    cont_valores[i]= 0;
```

O que pode ser usado como índice de um *vetor*?

Tanto variáveis inteiras quanto constantes numéricas.

Ex.:

```
printf("Escore do aluno %d: ", escore[i]);
printf("Escore do primeiro aluno %d: ", escore[0]);
```

Variáveis diferentes podem ser usadas para acessar um *vetor* em momentos diferentes de um programa:

Ex.:

```
scanf("%d",&valor[i]);
...
printf("Valor: %d ", valor[j]);
```

Uma mesma variável pode ser usada no mesmo momento ou em momentos diferentes para acessar *vetores* diferentes.

Ex.:

```
printf("\n%d   %d", gabarito[i] , resultado[i]);
```

Ainda sobre índices de vetores:

Não existe vinculação permanente entre um valor ou variável e um *vetor*.

Qualquer índice (variável ou constante) usado para acessar um *vetor* deve corresponder a um valor dentro do intervalo de índices válidos para o *vetor*.

Arranjo multidimensional
(ou matriz, tabela)

Um arranjo multidimensional é o arranjo que necessita de mais de um índice para referenciar seus elementos.

Ex.: arranjo bidimensional ou matriz de duas dimensões.

Declaração de notas como matriz bidimensional:

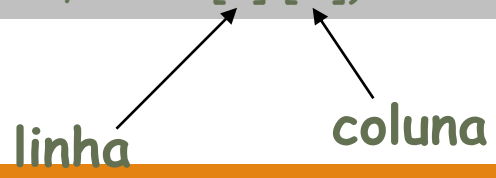
```
…
float notas [7] [6];
```

Outra forma:

```
#define  MAXLIN  7
#define  MAXCOL  6
…
float notas [MAXLIN] [MAXCOL];
```

Acesso a um elemento de Notas:

```
printf("%6.2f" , notas[1] [2]);
```



Arranjos bidimensionais em C

Primeiro índice : linha;

Segundo índice : coluna.

Acesso a elementos determinados da matriz:

Considerando uma matriz definida como *notas[7][6]* com as notas de alunos, em que cada linha corresponde a um aluno e cada coluna a uma prova:

Primeira nota do primeiro aluno:

```
printf("Nota 1 do Primeiro Aluno: %6.2f", notas[0] [0]);
```

Primeira nota do terceiro aluno , ou seja, i = 2 j = 0:

```
scanf("%f", &notas[i] [j]);
```

Última nota do último aluno:

```
if (notas[6] [5] > 9.5)
        ...
```

ATENÇÃO:

Seja qual for o número de dimensões*

os elementos de um arranjo são sempre de mesmo tipo!

* (número de índices necessários para acessar um elemento da matriz = número de dimensões da matriz)

Strings

Não existe um tipo String em C.

Strings em C são vetores do tipo char que terminam com ‘\0’.

Para literais string, o próprio compilador coloca ‘\0’.

```
#include <stdio.h>
#include <stdlib.h>
main(){
   char re[8] = "lagarto"; //tamanho máximo de 7 caracteres
   printf ("%s", re);
   system("pause");
}
```

Para ler uma String

Comando gets

```
#include <stdio.h>
#include <stdlib.h>
main(){
   char re [80];
   printf ("Digite o seu nome: ");
   gets(re);
   printf ("Oi %s\n", re);
   system("pause");
}
```

Tratamento de strings

#include <string.h>

Principais funções para manipulação de strings:

- strcmp (s1, s2): comparação de strings (0 p/ iguais)
- strlen(s1): devolve o tamanho da string
- strcpy(para, de): copia string
- strcat(str1,str2): concatena duas strings
- strupr(str): coloca *str* em letras maiúsculas
- strlwr(str): coloca *str* em letras minúsculas

Tratamento de strings

Como os strings são armazenados como um vetor de caracteres, eles podem ser manipulados como um vetor normal, acessando cada caracatere pelo respectivo índice

Vetores, Matrizes, Strings

PROF. MAURICIO A DIAS – MACDIASPAE@GMAIL.COM

INTRODUÇÃO A COMPUTAÇÃO PARA ENGENHARIAS