

INF101202

Algoritmos e Programação

Modalidade Ead – Turma H

Material de apoio: uso de estruturas de tipo de dados indexadas e homogêneas em C

Seja o seguinte problema :



(problema 1)

Ler 30 valores e calcular a média aritmética dos mesmos.

Pergunta:

Quantas variáveis são necessárias para ler os 30 valores ?

30 ?

Ou 1?



Então, dado o problema 1:

Ler 30 valores e calcular a média aritmética dos mesmos.

E, dada a pergunta:

Quantas variáveis são necessárias para ler os 30 valores ?

30 ?


Ou 1?

A resposta é:

à vontade do freguês, mas uma só é suficiente!!!!.

Ler trinta valores inteiros e calcular a média dos valores

Problema 1, solução 1: com trinta variáveis para a leitura dos valores



```
//Le trinta valores e calcula sua media aritmetica
//Solucao 1
#include <stdio.h>
int main( )
{
    int i, valor1, valor2, valor3...valor30, somatorio;
    float media;
    printf("\nValor 1: ");
    scanf("%d", &valor1);
    printf("\nValor 2: ");
    scanf("%d", &valor2);
    printf("\nValor 3: ");
    scanf("%d", &valor3);
    (...)
    somatorio = valor1+valor2+valor3+ ...valor30;
    media = (float)somatorio / 30;
    printf("\nMedia = %8.2f\n", media);
}
```

Ler trinta valores inteiros e calcular a média dos valores

Problema 1, solução 2: com uma única variável para a leitura dos valores



```
//Le trinta valores e calcula sua media aritmetica
```

```
//Solucao 2
```

```
#include <stdio.h>
```

```
#define MAX 30
```

```
int main( )
```

```
{ int i, valor, somatorio;
```

```
float media;
```

```
somatorio = 0;
```

```
printf("Forneca %d valores (inteiros):\n", MAX);
```

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

```
{
```

```
printf("Valor %d: ",i);
```

```
scanf("%d",&valor);
```

```
somatorio = somatorio + valor;
```

```
}
```

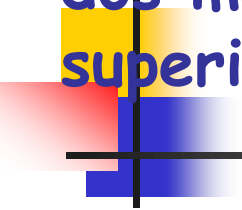
```
media = (float)somatorio / MAX;
```

```
printf("\nMedia = %8.2f\n", media);
```

```
return 0;
```

```
}
```

Problema 2: Ler 30 valores, calcular a média aritmética dos mesmos e imprimir a média e os valores iguais ou superiores à média

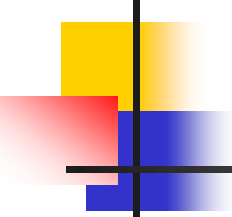


Pergunta:

Quantas variáveis são necessárias para ler os valores?

30?

Ou 1?



Problema 2: Ler 30 valores, calcular a média aritmética dos mesmos e imprimir a média e os valores iguais ou superiores à média

Então, a pergunta:


Quantas variáveis são necessárias para ler os valores?

30?

Ou 1?

Recebe a resposta:

30 variáveis de mesmo tipo!!!

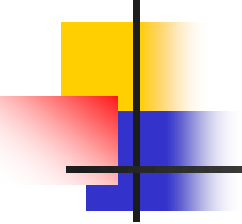


E, se para o mesmo problema 2, em vez de trinta desejar-se processar 60, 100, 200 ou 500 valores?

Resposta:

serão necessárias sucessivamente 60, 100, 200 ou 500 variáveis, cada uma com um nome diferente.

O que obviamente não parece ser um modo razoável de resolver-se o problema.



Solução para o problema
de múltiplas variáveis
de mesmo tipo:

**ARRANJOS UNI
OU
MULTIDIMENSIONAIS**



ARRANJOS UNI OU MULTIDIMENSIONAIS

são conjuntos de dados de mesmo tipo, ou seja, vetores de elementos consecutivos, todos de mesmo tipo. São estruturas de dados homogêneas.

Arranjos

Podem ser:

- *unidimensionais: vetores* 
- *multidimensionais: matrizes*



Arranjos unidimensionais

*Os arranjos são estruturas ditas
homogêneas, pois todos os seus
componentes são do mesmo tipo*

A seguir uma solução com vetor para o problema 2:
Criar um vetor de 30 elementos para guardar os
valores de entrada !!!!



```
#include <stdio.h> // Problema 2
```

```
#include <stdlib.h>
```

```
#define MAX 30
```

```
int main( )
```

```
{ int i, valor, somatorio;
```

```
int vetor[MAX];
```

```
float media;
```

```
somatorio = 0;
```

```
printf("Forneca %d valores (inteiros):\n", MAX);
```

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

```
{
```

```
    printf("Valor %d: ",i);
```

```
    scanf("%d",&vetor[i]);
```

```
    somatorio = somatorio + vetor[i];
```

```
}
```

```
media = (float)somatorio / MAX;
```

```
printf("\nMedia = %8.2f\n", media);
```

```
printf("\n\nValores iguais ou superiores a media\n");
```

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

```
{
```

```
    if (vetor[i] >= media)
```

```
        printf("%4d", vetor[i]);
```

```
}
```

```
printf("\n");
```

```
system("pause");
```

```
return 0;
```

```
}
```

Declaração de arranjo unidimensional com 30 valores

Ler 30 valores, calcular a média aritmética dos mesmos e imprimir a média e os valores iguais ou superiores à média.



Vetores

Estruturas de armazenamento compostas de múltiplos elementos que ocupam posições contíguas de memória.

Todos os elementos dessas estruturas são de mesmo tipo, têm um só nome, comum a todos, e são identificados de modo único por seu valor de posição (índice).

Vetores

Índices

Valor de posição,
identifica de modo
único cada elemento

0 1 2 3 4 5 6 7 8

vet

--	--	--	--	--	--	--	--	--

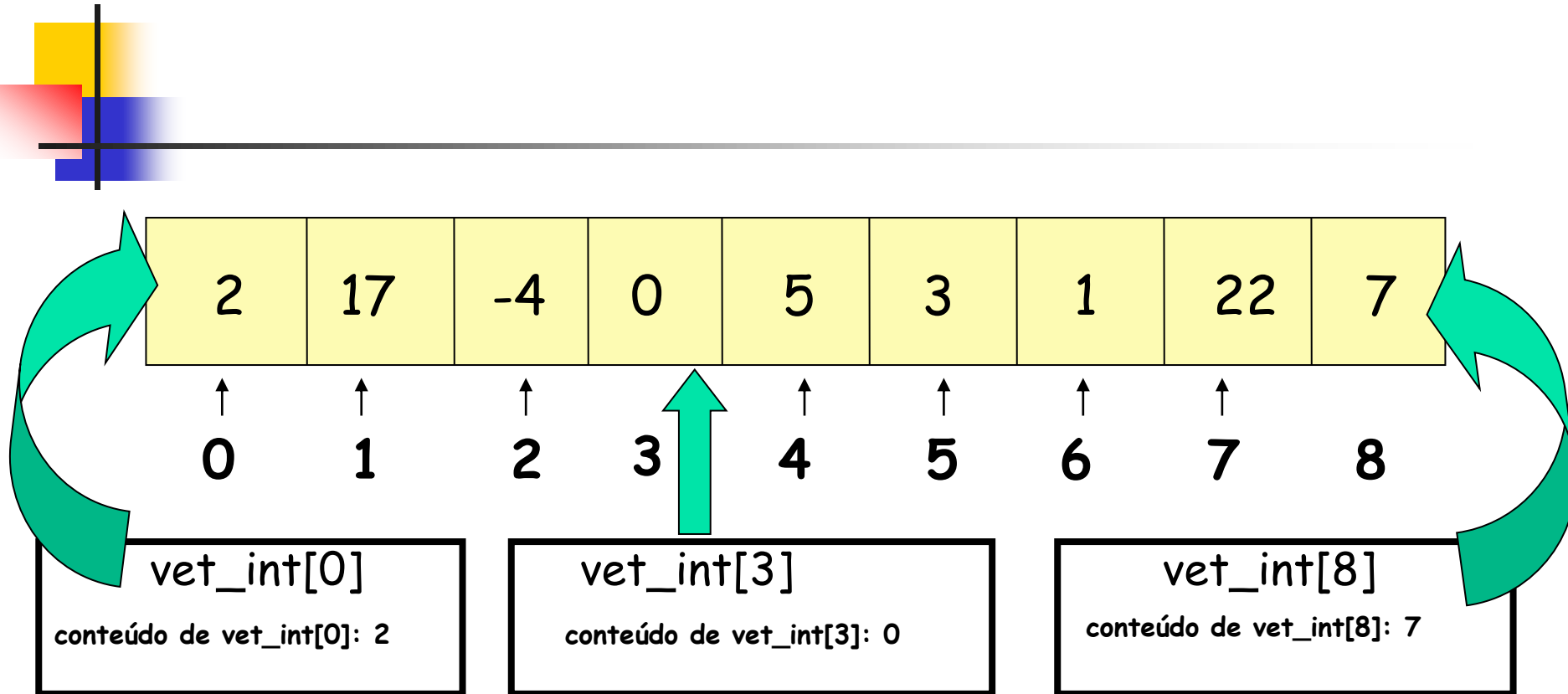
nome

Comum para todos os
elementos

Valor

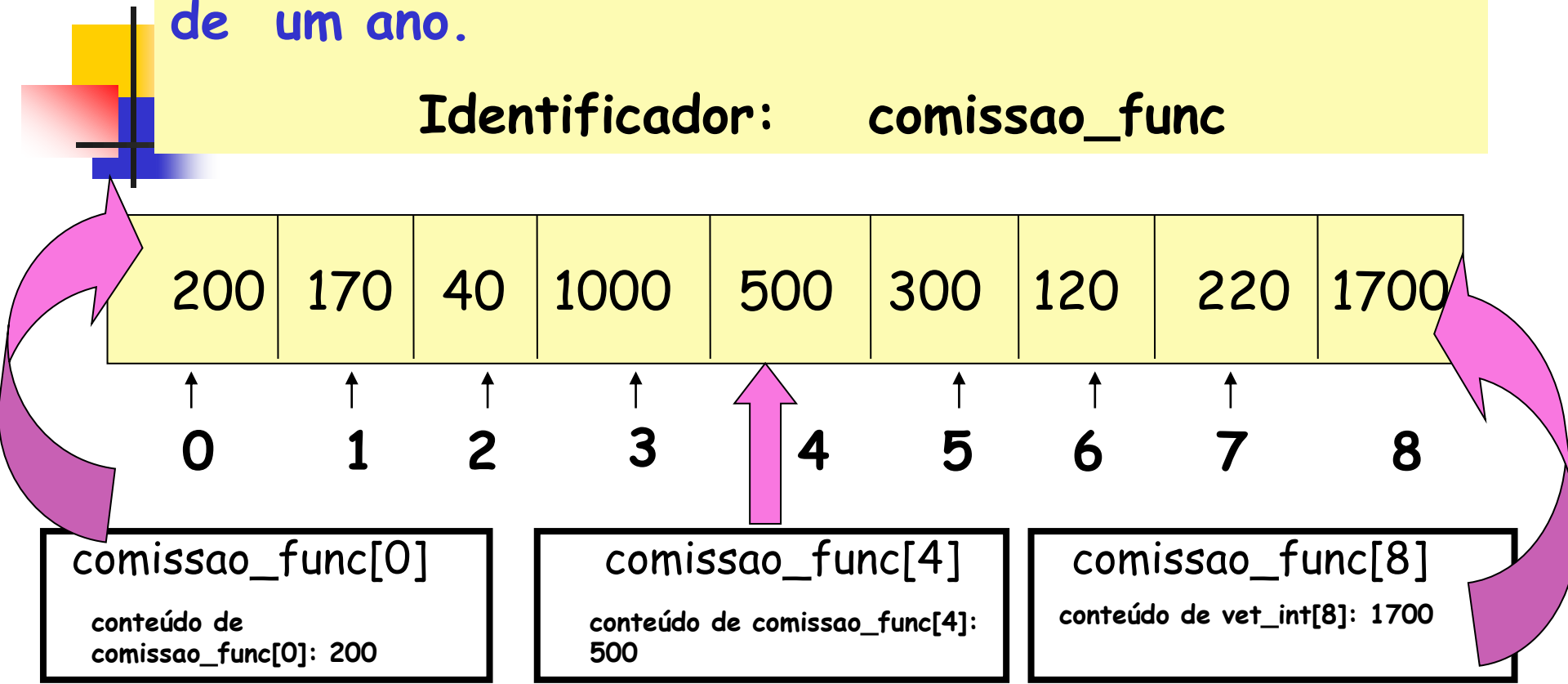
Semelhante a uma
variável simples

Exemplo de vetor: vet_int



Exemplo de vetor: conjunto de comissões mensais associadas a um determinado empregado ao longo de um ano.

Identificador: `comissao_func`



200	170	40	1000	500	300	120	220	1700
↑	↑	↑	↑	↑	↑	↑	↑	↑
0	1	2	3	4	5	6	7	8

`comissao_func[0]`

conteúdo de
`comissao_func[0]`: 200

`comissao_func[4]`

conteúdo de `comissao_func[4]`:
500

`comissao_func[8]`

conteúdo de `vet_int[8]`: 1700

Declaração de vetores

Forma geral:

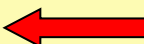
`tipo_da_variável nome_da_variável [tamanho];`

Exemplos:

Info é um vetor real de 10 elementos, cujos índices podem variar de 0 a 9:

`float info[10];`

Notas é um vetor inteiro de 50 elementos, cujos índices podem variar de 0 a 49:

`#define LIM_SUP 50` 
`int notas[LIM_SUP];`

Temperaturas é um vetor real de 101 elementos, cujos índices podem variar de 0 a 100:

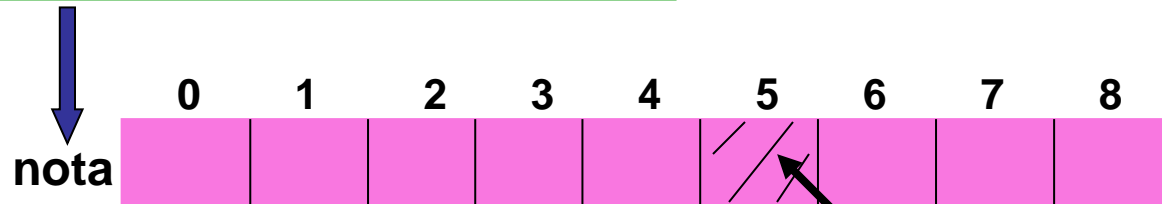
`#define LIM_TEMP 101` 
`float temperaturas [LIM_TEMP];`

Atenção: as declarações de constantes com os *defines*, não são obrigatórias, embora seja fortemente recomendado seu uso.

Utilização de um vetor

Variável indexada

nome do arranjo [índice]



Ex:

```
float nota[9]; // declaração  
scanf("%f", &nota[5]); //leitura de 1 elemento
```

nota[5]

```
nota[ 1 ] = 7.5 ;  
nota[ 2 ] = nota [ 1 ] + 2 ;  
if (nota[ 1 ] > 6.0)  
    printf ("Aprovado");
```

...

Apresentação do conteúdo de elementos de um vetor

O elemento do vetor **valor** que será apresentado dependerá do valor de **i** no momento da execução do comando de escrita:

```
printf("\n\nValor %d: " , valor[i]);
```

Está sendo solicitada a apresentação do conteúdo da posição zero do vetor **info**:

```
printf("\n\nPrimeiro valor = %6.2f: " , info[0]);
```

Obs.: Referências a elementos de vetores podem ser usadas em todas as situações em que variáveis do tipo do vetor podem ser usadas.

Inicialização de vetores



Vetores, como as demais variáveis, ao serem criados contêm lixo.

Vetores podem ser inicializados:

- a) na declaração;
- b) por atribuição, em algum momento da execução;
- c) por leitura.

a) inicialização de vetores na declaração

Forma geral:

Tipo básico var[n] = {valor₀, valor₁, valor_{n-1}};

n=total de
elementos

`int v[3] = {10, 20, 30}; //declaração de v`

v

10	20	30
0	1	2

Posições não inicializadas são preenchidas com zero:

`int x[5] = {10, 20, 30}; // declaração de x`

x

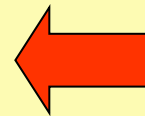
10	20	30	0	0
0	1	2	3	4

a) inicialização de vetores na declaração (cont.)

```
float z[5] = {2.7, 3.5, 4.2, 9.23 , 4.7};
```

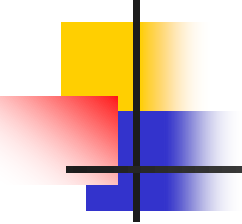
o mesmo que escrever:

```
int z[5];  
z[0] = 2.7;  
z[1] = 3.5;  
z[2] = 4.2;  
z[3] = 9.23;  
z[4] = 4.7;
```



Inicialização por
atribuição, durante
a execução.

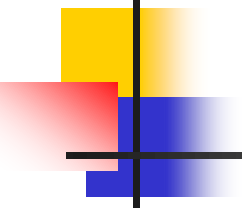
b) inicialização de vetores por atribuição



Por exemplo, garantindo valor inicial zero a posições que funcionarão como contadores ou acumuladores:

```
for (i = 0; i < MAX; i++)  
    cont_valores[i] = 0;
```


c) inicialização de vetores por leitura



```
for (i = 0; i < MAX; i++)  
{  
    printf("Valor %d: ", i+1);  
    scanf("%d", &valor[i]);  
}
```



foram lidos MAX valores para o vetor valor

O que pode ser usado como índice de um vetor?

Respostas.....

1. Tanto variáveis quanto constantes.

Ex.:

```
printf("Escore do primeiro aluno %d: " , escore[1]);  
printf("Escore do aluno %d: " , escore[i]);
```

2. 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]);
```

3. 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]);
```

Lembretes importantes!!!

O índice da **primeira** posição de um vetor é zero.

Ex.: `num[0] = 10;`

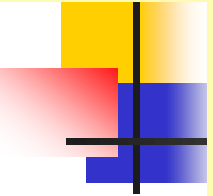
O sistema **não controla** a correção dos índices usados. Quem deve garantir que os índices estejam dentro do intervalo correto é Você, o programador!!!!!!

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

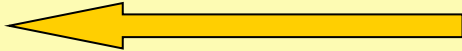
Qualquer índice (variável ou constante) usado para acessar um vetor **deve corresponder** a um valor **dentro do intervalo** de índices **válidos** para o vetor.

Lembra do Problema 2 ??? -> Ler 30 valores, calcular a média aritmética dos mesmos e imprimir a média e os valores iguais ou superiores à média.

Solução: criar um vetor de 30 posições para armazenar os valores



```
#include <stdio.h>
#include <stdlib.h>
#define MAX 30
int main( )
{
    int i, valor, somatorio;
    int vetor[MAX];
    float media;
    somatorio = 0;
    printf("Forneca %d valores (inteiros):\n", MAX);
    for(i=0;i<MAX;i++)
    {
        printf("Valor %d: ",i);
        scanf("%d",&vetor[i]);
        somatorio = somatorio + vetor[i];
    }
    media = (float)somatorio / MAX;
    printf("\nMedia = %8.2f\n", media);
    printf("\n\nValores iguais ou superiores a media\n");
    for(i=0;i<MAX;i++)
    {
        if (vetor[i] >= media)
            printf("%4d", vetor[i]);
    }
    printf("\n");
    system("pause");
    return 0;
}
```



Se em vez de 30 valores, forem 60, 100, 200, 500, etc., basta trocar o valor de MAX. A mesma solução vale para todos os demais casos.

Declaração de constantes simbólicas com *define*

#define identificador valor

Ex.:

#define MINIMO 2

- identificador regra geral em maiúsculas;
- identificador será substituído no programa, em tempo de execução, pelo valor indicado;
- entre identificador e valor nada além de espaços deve ser colocado.

Uso de constantes simbólicas

Podem ser usadas na declaração de vetores, em expressões, ter seu valor apresentado ou atribuído a variáveis.

Não podem ser alteradas por atribuição ou leitura.

```
#include <stdio.h>
#define LIMITE 30
int main ( )
{
    float nota [LIMITE];
    int indice;
    for (indice = 0; indice < LIMITE; indice++)
        scanf("%f", &nota[indice]);
    printf("%d", LIMITE);
    indice = LIMITE;
    (...)
}
```

~~LIMITE = 7;
scanf("%d", &LIMITE);~~