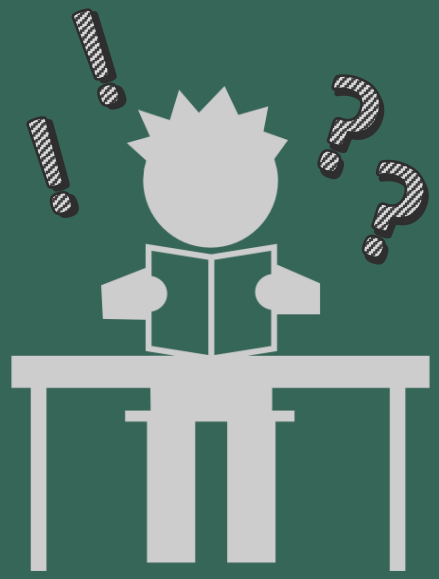
ESTRUTURA DE DADOS



AULA 01 – VETORES

(ARRAY UNIDIMENSIONAL)

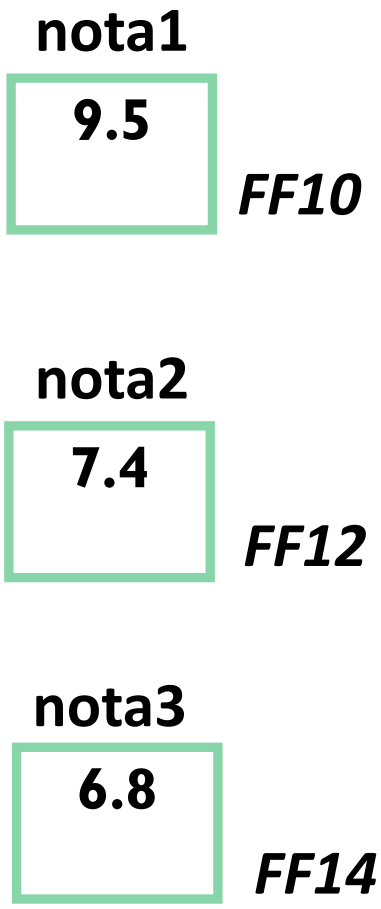
DO INÍCIO ...VARIÁVEIS SIMPLES

QUIZ: O que acontece quando você tem uma única variável simples inteira e precisa "receber" 3 notas de alunos que serão informados pelo usuário??

( ) Todos os valores são armazenados nessa variável e podem ser acessados a qualquer momento do programa.
( ) ao receber um número a partir do segundo, o anterior será "perdido"/sobreposto.
( ) os 3 valores permanecem na memória principal do computador desde que haja espaço suficiente.
( ) Apenas o último valor informado pelo usuário permanecerá armazenado nesta variável.
( ) todos os 3 valores informados pelo usuário serão sobrescritos.
( ) Não é possível armazenar todos esses valores em uma variável simples, pois só há um endereço de memória associado ao nome desta variável.
( ) esta variável simples será sobrescrita 3 vezes (a cada vez que o usuário digitar um novo número).
( ) Nenhuma das respostas anteriores.
( ) Não faço a mínima ideia.

SOLUÇÕES???

```
float nota1, nota2, nota3;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);
printf("Nota do aluno 3: ");
scanf("%f", &nota3);
```



E SE FOREM 100 NOTAS DE 100 ALUNOS QUE PRECISAM SER TODAS ACESSADAS???

```
float nota1, nota2, nota3, /* .... */ nota100;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);

/* ... */

printf("Nota do aluno 100: ");
scanf("%f", &nota100);
```

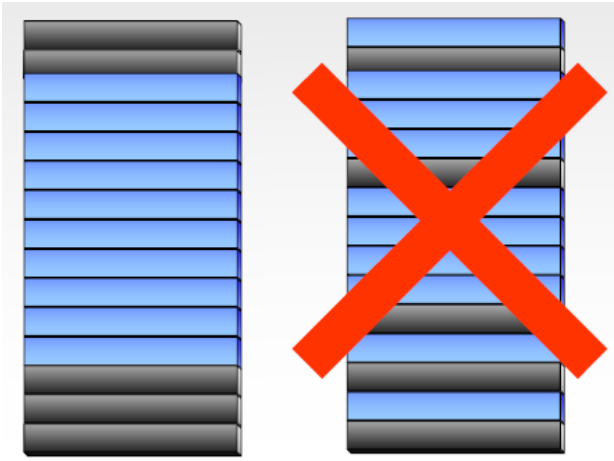
É eficiente ?

VARIÁVEIS COMPOSTAS

✓ **São um conjunto de variáveis identificadas por um mesmo nome que correspondem a posições de memória distintas.**

- Homogêneas (vetores e matrizes)
- Heterogêneas (estruturas)

✓ **Aloca-se uma porção contígua da memória para armazenar os elementos de vetores e matrizes.**



PORÇÃO CONTÍGUA DE MEMÓRIA

Porção contígua da memória para armazenar os elementos de vetores e matrizes.

Característica **fundamental** destas estruturas de dados

Define como os dados estarão **organizados** na **memória**

Permite a manipulação por meio de **índices**, **associados** aos **endereços** de memória

Permite a manipulação por meio de **aritmética** para o **deslocamento** sobre **endereços** de memória

VETORES >> ARRAYS UNIDIMENSIONAIS

- ✓ São ED homogêneas (compostas por elementos de um único tipo, ou do mesmo tipo);
- ✓ Os elementos são representados por um único nome de variável;
- ✓ Necessitam de um índice para identificar (individualizar) cada elemento do conjunto;
- ✓ Muito utilizado para trabalhar com conjuntos de dados que são semelhantes em tipo.

- ✓ ***Ex: o conjunto das alturas dos alunos de uma turma, ou um conjunto de seus nomes.***
- ✓ Objetivo: colocar as informações sob um mesmo conjunto, e poder referenciar cada dado individualmente por um índice.

VETORES >> ARRAYS UNIDIMENSIONAIS

Para cada valor do vetor:

- ✓ **haverá um índice diferente para acessá-los, mesmo que os valores armazenados sejam repetidos;**
- ✓**Cada índice do vetor representa um endereço de memória diferente e consecutivo aos demais elementos desse vetor.**
- ✓ ***Ex: vetor de 5 elementos***

V[0]=4	V[1]=7	V[2]=2	V[3]=5	V[4]=3
FF10	*FF11*	*FF12*	*FF13*	*FF14*

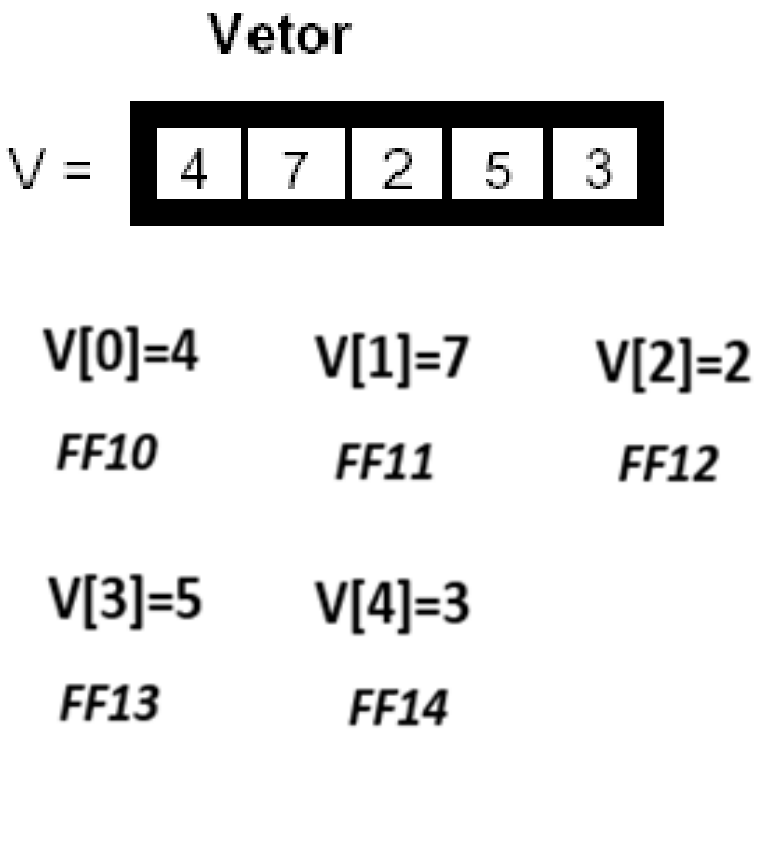
VETORES >> ARRAYS UNIDIMENSIONAIS

✓**Características**:

✓**vetor é um arranjo que permite acesso direto ao elemento;**

✓**Os elementos são independentes >>> para acessar qualquer elemento de um vetor, basta referenciar o índice desejado, não é necessário acessar todos os elementos e nem os elementos anteriores ou posteriores;**

✓ **Uma variável vetor armazena vários valores simultaneamente sob o mesmo nome de variável.**



VETORES >> ARRAYS UNIDIMENSIONAIS

- ✓ **Os vetores são listas ordenadas, pois possuem índices que variam de 0 a n;**
- ✓ **O índice inicia na posição 0 (1° elemento do vetor) e vai até o último elemento declarado na variável.**
- ✓ **O primeiro elemento de vetor tem índice 0 e o último tem índice *tamanho - 1*, onde *<valor>* é o tamanho do vetor.**
- ✓ **Sintaxe:**

<tipo_de_dado> <nome_da_variável> [<valor>];

- ✓ **Exemplos:**

int valor [10] ; char palavra [5]; float numero [3];

VETORES >> ARRAYS UNIDIMENSIONAIS

✓ **Para referenciar um elemento do vetor, deve-se explicitar o nome do vetor seguido de um índice inteiro.**

✓ **Quando faz-se a declaração de um vetor é reservado espaço de memória para os seus elementos.**

✓ **Essa quantidade de memória (em *bytes*) varia de acordo com o tipo de dado e pode ser calculada como:**

***quantidade de memória = tamanho do tipo * tamanho do vetor*.**

✓ **Os vetores também podem ser inicializados no momento da declaração. Sintaxe da inicialização (apenas na declaração):**

tipo nome_vetor[tam] = {lista de valores};

***ATENÇÃO! A lista de valores* é uma lista, separada por vírgulas, dos valores de cada elemento do vetor.**

VETORES >> ARRAYS UNIDIMENSIONAIS

✓ **Ex.: a inicialização de um vetor pode ser feita manualmente.**

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

main()
{
	int i, vetor [5]={10,20,30,40,50};
	for (i=0;i<5;i++)
	{
		printf("Posicao%d: %d\n", i, vetor[i]);
	}
	system("pause");
}
```

VETORES >> ARRAYS UNIDIMENSIONAIS

QUIZ: O que ocorre quando um vetor é declarado e não inicializado???

( ) Sintaticamente, não há problema apenas declarar e não inicializar um vetor.
( ) Manipular valores de um vetor não inicializado pode ocasionar erros em tempo de execução ou de processamento.
( ) Vetores não inicializados possuem valores desconhecidos considerados "lixo".
( ) Não é possível manipular vetores que não foram inicializados.

( ) Um vetor não inicializado possui valores nulos, branco ou 0 de acordo com o tipo de dado que ele foi declarado.
( ) Um vetor não inicializado na declaração pode ter valores armazenados pelo usuário em tempo de execução.
( ) Nenhuma das alternativas anteriores.
( ) Não faço a mínima ideia.

VETORES >> ARRAYS UNIDIMENSIONAIS

- ✓ **Exemplos:**

  int dia[3] = {12,30,14}; //ou dia[0]=12; dia[1]=30; dia[2]=14;

  float nota[5] = {8.4,6.9,4.5,4.6,7.2};

  char vogal[5] = {'a', 'e', 'i', 'o', 'u'};

- ✓ **A forma clássica de inicializar um vetor pode ser feito por meio de um laço *for*, mas qualquer outra estrutura de repetição poderá ser utilizada.**
- ✓**a variável que representa o vetor, é uma constante que armazena o endereço inicial do vetor, isto é, a variável "*nota*", sem indexação, aponta para o primeiro elemento do vetor.**
- ✓**Uma vez definido o tamanho do vetor, este tamanho não poderá ser alterado em tempo de execução. Ou seja, é uma estrutura de dados estática >> vamos expandir e flexibilizar mais à frente.**

VETORES >> ARRAYS UNIDIMENSIONAIS

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);

for (i = 0; i < n; i++) {
  printf("Nota do aluno %d: ", i+1);
  scanf("%f", &nota[i]);
}
```

Como armazenar 100 notas?

VETORES >> EXEMPLOS

Ex1:

```
main()
{
int vetor[10];
int x;
for(x=0; x<10; x++)
        {
        vetor[x]=x;
        printf("%d\n", vetor[x]);
        }
}
```

Ex2:

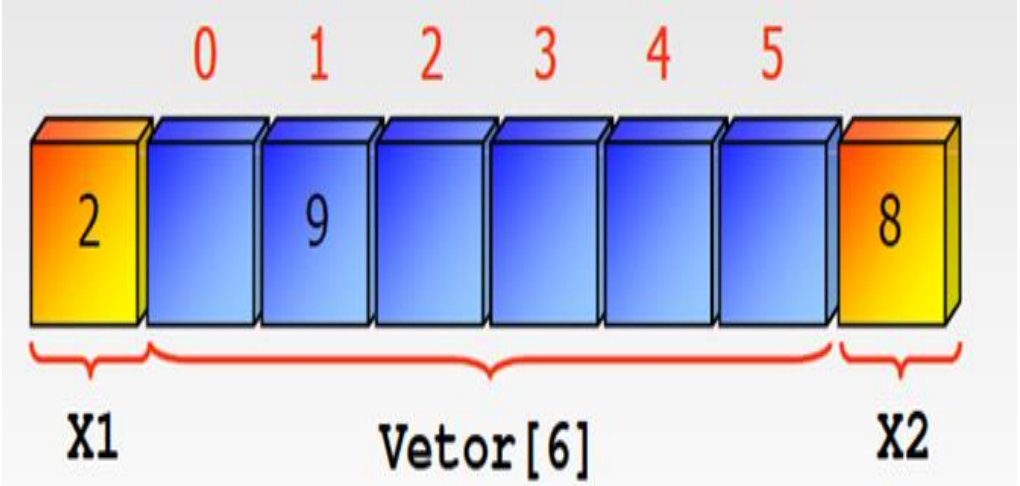
```
main()
{
float nota[20], soma;
int  x;

for( x=0; x<20; x++) // instrução composta
        {
        printf("Digite a nota do aluno");
        scanf("%f",&nota[x]);
        }
soma=0.0;
for(x=0; x<20; x++) //instrução simples
        soma+=nota[x];
printf("Média das notas %3.2f ",soma/20);
}
```

ATENÇÃO!

✓ **1. Índices fora do limite podem causar comportamento anômalo do código, pois o compilador vai tentar referenciar um espaço em memória que não foi reservado para o vetor quando este foi declarado.**

✓ **2. O programa poderá causar instabilidade ou até mesmo, travar o compilador.**

✓ **3. Isto implica, que pode-se atribuir valores a outros dados (até mesmo a uma parte do código do programa) acarretando resultados imprevisíveis.**




```
int X1;
int Vetor[6];
int X2;
```

```
Vetor[1]  = 9;
Vetor[-1] = 2;
Vetor[6]  = 8;
```

PONTOS IMPORTANTES!

1. A linguagem C não verifica se você está utilizando índice inválido! Você pode acabar lendo "lixo" ou alterando outras variáveis do seu programa!

2. A inicialização manual de um vetor pode ser feita apenas no momento da sua declaração.

3. Pode-se omitir o tamanho do vetor quando ele é inicializado. Neste caso, o compilador vai calcular automaticamente que a quantidade de posições usadas pelo vetor pela inicialização. Ex:

```
int vetor[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int m[][2] = {23, 45, 54, 55,77, 65} // como a 2ª dimensão é 2, o compilador calcula que a 1ª dimensão é 3.
```

PONTOS IMPORTANTES!

4. Vetores e matrizes precisam ser declarados, de modo que o compilador conheça o tipo de dados e reserve espaço de memória suficiente para armazená-la.

5. Os elementos dos arrays são guardados em uma sequência consecutiva/contígua de memória.

6. Para a definição do tamanho do vetor (e da matriz) o índice deve ser sempre um n° inteiro; mas o conteúdo do vetor pode ter qualquer tipo suportado pela linguagem.

7. Neste material, todos os vetores foram alocados estaticamente.

PRIMEIROS PASSOS

1º :

Criar um vetor de 10 posições de inteiro;

2º :

Inicializar o vetor com valores fornecidos pelo usuário;

3º :

Exibir todos valores da última até a 1ª posição;

4º :

Calcular e exibir a quantidade de números pares;

5º :

Calcular e exibir a média de todos os valores.

DESAFIOS...

1) Escreva um programa que leia um vetor de duas posições e troque o conteúdo entre elas.

2) Calcular a média final de um aluno, sendo que foram dadas 10 notas.

3) Faça um programa que:

a) leia um vetor de 100 números reais;

b) leia um outro valor real;

c) verifique se este outro valor real existe ou não no vetor de 100 elementos. Caso exista, exibir a posição do vetor. Caso o valor não seja encontrado, exiba uma mensagem ao usuário.

d) imprimir na tela todos os números maiores que o valor real fornecido pelo usuário e sua respectiva posição.

DESAFIOS...

4) Faça um programa para ler 20 números inteiros, calcular a média dos mesmos e exibir todos os números que estiverem acima da média. Considere a média como sendo 7.0.

5) Escreva um algoritmo que verifique se dois vetores são iguais. Os vetores devem ter 10 elementos cada.

6) Ler dois vetores A e B e dizer quantos elementos de A são maiores que qualquer elemento de B. Imprima estes elementos na tela.

7) Dado um conjunto de 20 valores reais, fornecidos pelo usuário:

a) Imprima os valores que são positivos;

b) Calcule e imprima a média dos valores negativos.

8) Faça um programa que verifique se um determinado número fornecido pelo usuário esta em um vetor e quantas vezes ele se repete. Considere o vetor contendo 30 elementos fornecidos pelo usuário.