
Fundamentos de Programação

Aula 08 - Vetores

Arnaldo Barreto Vila Nova

Vetores

- Definição
 - Declaração
 - Exemplos
 - Funções com vetores
 - Vetores como ponteiros
-

Vetores - Definição

- Como fazer um programa para guardar a nota de 100 alunos e depois fazer algumas operações com elas?
 - Até então teríamos que usar 100 variáveis para isso
 - Agora, vamos aprender a usar uma “tabela” com 100 posições, sem precisarmos declarar uma quantidade absurda de variáveis
-

Vetores - Definição

- Vetores (ou *arrays*, ou *arranjos*) são estruturas para armazenar um conjunto de valores em um mesmo nome
-

Vetores - Definição

- Vetores (ou *arrays*, ou *arranjos*) são estruturas para armazenar um conjunto de valores em um mesmo nome
 - Um índice identifica a posição no vetor onde um valor está inserido
-

Vetores - Definição

- Vetores (ou *arrays*, ou *arranjos*) são estruturas para armazenar um conjunto de valores em um mesmo nome
 - Um índice identifica a posição no vetor onde um valor está inserido
 - Em C, todos os valores armazenados em um vetor tem que ser do mesmo tipo declarado para o vetor
-

Vetores - Definição

- **Exemplo:** Receber 100 notas usando 100 variáveis



Vetores - Definição

- **Exemplo:** Receber 100 notas usando 100 variáveis

```
float nota1, nota2, /*...*/, 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);
```

Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor



Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor

```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f",&nota[i]);  
}
```

Vetores - Definição

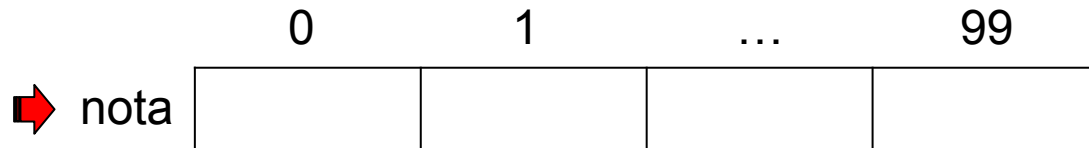
- **Exemplo:** Receber 100 notas usando 1 vetor

A declaração do vetor reserva 100
endereços de memória para
valores float

➡

```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f", &nota[i]);  
}
```

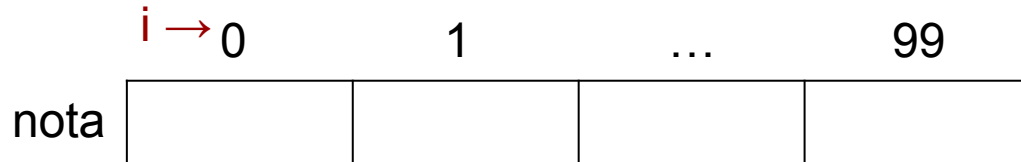
É como se fosse criada na memória
uma tabela com o nome “nota” indo
da posição (ou índice) 0 até 99



Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor

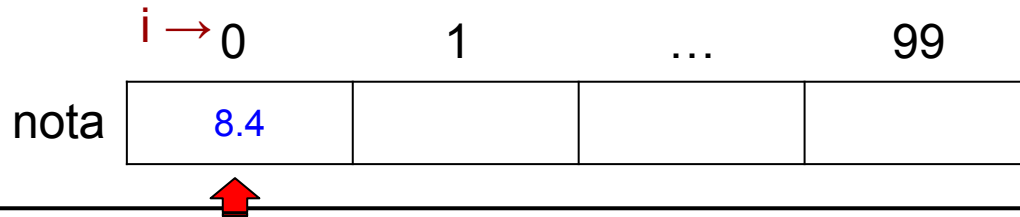
```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f",&nota[i]);  
}
```



Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor

```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f",&nota[i]);  
}
```

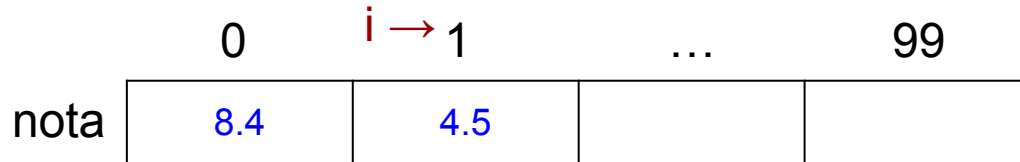


Cada índice poderá guardar 1 valor

Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor

```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f", &nota[i]);  
}
```



Vetores - Definição

- **Exemplo:** Receber 100 notas usando 1 vetor

```
float nota[100];  
int i;  
for (i=0; i<100; i++)  
{  
    printf("Nota do aluno %d:", i);  
    scanf("%f", &nota[i]);  
}
```

	0	1	...	<i>i</i> → 99
nota	8.4	4.5	...	9.75

Vetores - Declaração

- Na declaração de um vetor, identificamos o tipo, o nome e número de posições, que ficará entre colchetes ([])

`<tipo> nome[<número de posições>];`

`float nota[100];`

`int cod[10];`

Vetores - Declaração

- Em C, o primeiro índice sempre será 0, e o último índice será o <número de posições> - 1
 - Ao declarar um vetor, um espaço na memória sequencial na memória é reservado para armazenar cada posição do vetor.
 - Exemplo: se a posição 0 de um vetor de inteiros está no endereço de memória 234, a posição 1 estará no 238, a posição 2 estará no 242 e assim sucessivamente, já que cada inteiro ocupa 4 bytes na memória.
-

Vetores - Declaração

- Se você tentar acessar um valor fora do tamanho do vetor, seu programa irá dar erro e encerrar ou irá usar dados desconhecidos armazenados na memória fora do vetor.
 - Ou seja, se um vetor tem tamanho 100, não existe posição 101 ou superior, assim como não existe posições negativas.
 - Uma vez especificado, o tamanho do vetor não pode ser alterado.
-

Vetores - Declaração

- Se você tentar acessar um valor fora do tamanho do vetor, seu programa irá dar erro e encerrar ou irá usar dados desconhecidos armazenados na memória fora do vetor.
 - Ou seja, se um vetor tem tamanho 100, não existe posição 101 ou superior, assim como não existe posições negativas.
 - Uma vez especificado, o tamanho do vetor não pode ser alterado. (A não ser que você utilize ponteiros e Alocação Dinâmica com funções que serão vistas futuramente)
-

Vetores - Declaração

- Podemos inicializar os elementos de um vetor usando chaves na declaração ({ })
- Verifique os valores impressos na tela com o código

```
int v1[5] = {1, 3, 5, 7, 9};  
int v2[4] = {2, 4, 6, 8};  
  
printf("%i", v1[0]);  
printf("%i", v2[3]);  
printf("%i", v1[4]+v2[2]);  
printf("%i", v1[5]+v2[0]);
```

Vetores - Declaração

- Ao inicializar os elementos de um vetor usando chaves, podemos omitir o tamanho do vetor. O tamanho dele será a quantidade de valores inicializados.

```
int v1[] = {1, 3, 5, 7, 9};  
int v2[] = {2, 4, 6, 8};
```

Vetores - Declaração

- Ao inicializar os elementos de um vetor usando chaves, podemos omitir o tamanho do vetor. O tamanho dele será a quantidade de valores inicializados.

```
int v1[] = {1, 3, 5, 7, 9};  
int v2[] = {2, 4, 6, 8};
```



Mesmo sem especificar o tamanho, v1
terá 5 elementos e v2 terá 4.

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada
-

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

Primeiro, declaramos
nosso vetor



```
int valor[10];
```

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

Precisaremos também de uma variável para percorrermos cada posição do vetor.



```
int valor[10];  
int i;
```

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

Nossa estrutura de repetição
irá percorrer da posição 0 até a
posição 9 para preenchermos





```
int valor[10];  
int i;  
for (i=0; i<10; i++)  
{  
}
```

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

```
int valor[10];
int i;
for (i=0; i<10; i++)
{
    printf("Digite o valor da posição %i: ", i);
    scanf("%i", &valor[i]);
}
```

 
No scanf vou especificar também a
posição no vetor em que quero
guardar o inteiro recebido

Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

```
int valor[10];
int i;
for (i=0; i<10; i++)
{
    printf("Digite o valor da posição %i: ", i);
    scanf("%i", &valor[i]);
}
for (i=9; i>=0; i--)
{

}
```

Para exibirmos de forma inversa, nosso índice começará da posição 9 e irá decrementar até a posição 0.



Vetores - Exemplo

- Receber 10 valores inteiros e exibí-los em ordem inversa à que foi informada

```
int valor[10];
int i;
for (i=0; i<10; i++)
{
    printf("Digite o valor da posição %i: ", i);
    scanf("%i",&valor[i]);
}
for (i=9; i>=0; i--)
{
    printf("Posição %i do vetor: %i", i, valor[i]);
}
```

No printf também colocamos a
posição do valor que desejamos exibir



Vetores - Exemplo

- Pedir para o usuário dizer o tamanho do vetor e ler essa quantidade de valores
-

Vetores - Exemplo

- Pedir para o usuário dizer o tamanho do vetor e ler essa quantidade de valores

Neste caso, queremos receber
o tamanho do vetor antes de
declará-lo



```
int i, tam;
```

Vetores - Exemplo

- Pedir para o usuário dizer o tamanho do vetor e ler essa quantidade de valores

```
int i, tam;  
  
printf("Por favor, digite o tamanho do vetor: ");  
scanf("%i", &tam);
```

Vetores - Exemplo

- Pedir para o usuário dizer o tamanho do vetor e ler essa quantidade de valores

```
int i, tam;  
  
printf("Por favor, digite o tamanho do vetor: ");  
scanf("%i", &tam);
```

Só podemos declarar o vetor
quando soubermos o tamanho



```
int valor[tam];
```

Vetores - Exemplo

- Pedir para o usuário dizer o tamanho do vetor e ler essa quantidade de valores

```
int i, tam;

printf("Por favor, digite o tamanho do vetor: ");
scanf("%i", &tam);

int valor[tam];

for (i=0; i<tam; i++)
{
    printf("Digite o valor %i: ", i);
    scanf("%i", &valor[i]);
}
```

Nosso índice irá da posição 0
até a posição tam-1



Exercícios

- Leia um vetor de 10 notas e calcule a média delas
 - Leia um vetor de 5 notas, um vetor de 5 pesos e calcule a média ponderada das notas armazenadas no primeiro vetor usando os pesos armazenados no segundo vetor
 - Ler um vetor de 5 números reais e dizer qual o maior valor e qual o menor menor informados
 - Receber do usuário a quantidade de alunos, armazenar em um vetor a nota de cada aluno e calcular a média da turma
-

Exercícios

- Preencher um vetor A com 100 números aleatórios entre -50 e 50, e depois conte quantos números pares e quantos números ímpares foram gerados.
 - Gerar um vetor B apenas com os números pares do vetor A com números aleatórios.
 - Dado um inteiro X, dizer se X é um elemento do vetor A e dizer qual a posição dele no vetor
 - Dizer quantos números de A são menores que X
-

Funções com vetores

- Passando vetores como parâmetros

- Exemplo: função para calcular a média dos valores de um vetor

Para este tipo de caso, é comum recebermos o vetor e o tamanho desta forma.



```
float media(float vetor[], int tam)
{

}
}
```

Funções com vetores

- **Passando vetores como parâmetros**

- Exemplo: função para calcular a média dos valores de um vetor

```
float media(float vetor[], int tam)
{
    int i;
    float soma=0;
    for(i=0; i<tam; i++)
    {
    }

}
```

Assim, podemos percorrer o
vetor normalmente



Funções com vetores

- **Passando vetores como parâmetros**

- Exemplo: função para calcular a média dos valores de um vetor

```
float media(float vetor[], int tam)
{
    int i;
    float soma=0;
    for(i=0; i<tam; i++)
    {
        soma += vetor[i];
    }
    return (float)soma/tam;
}
```

Funções com vetores

- **Passando vetores como parâmetros**
 - Exemplo 2: função para imprimir os valores de um vetor

```
void imprimir(float vetor[], int tam)
{
    int i;
    for(i=0; i<tam; i++)
    {
        printf("%f, ", vetor[i]);
    }
}
```

Funções com vetores

- **Passando vetores como parâmetros**
 - Exemplo 3: preencher um vetor com valores do usuário

```
void preencher(float vetor[], int tam)
{
    int i;
    for(i=0; i<tam; i++)
    {
        scanf("%f", &vetor[i]);
    }
}
```

Note que não usamos o return
para retornar o vetor



Funções com vetores

- Passando vetores como parâmetros

- Exemplo 3: preencher um vetor com valores do usuário



Isso porque a passagem de um vetor é sempre por referência e não por cópia.

(se você passar um vetor de 50 mil posições, não vai criar uma cópia dele na memória)

```
void preencher(float vetor[], int tam)
{
    int i;
    for(i=0; i<tam; i++)
    {
        scanf("%f", &vetor[i]);
    }
}
```

Funções com vetores

- Passando vetores como parâmetros

- Exemplo 3: preencher um vetor com valores do usuário



Isso porque a passagem de um vetor é sempre por referência e não por cópia.

Ou seja, é a mesma coisa de um ponteiro (internamente é um ponteiro, declarado de forma diferente).

```
void preencher(float vetor[], int tam)
{
    int i;
    for(i=0; i<tam; i++)
    {
        scanf("%f", &vetor[i]);
    }
}
```

Funções com vetores

- Passando vetores como parâmetros

- Exemplo 3: preencher um vetor com valores do usuário



Note que usando a declaração padrão de ponteiros, o código funcionará da mesma forma.

Este ponteiro irá receber o endereço de memória em que o primeiro elemento do vetor está alocado.

```
void preencher(float *vetor, int tam)
{
    int i;
    for(i=0; i<tam; i++)
    {
        scanf("%f", &vetor[i]);
    }
}
```

Funções com vetores

- Passando vetores como parâmetros
 - Exemplo 4: usando as funções anteriores

```
int main()
{
    float vetor[10];
    preencher(vetor, 10);
    imprimir(vetor, 10);
    printf("\nMedia = %f", media(vetor, 10));

    return 0;
}
```

Exercícios

- Refaça os exercícios anteriores usando funções



Vetores como Ponteiros

- Podemos trabalhar com ponteiros como se fossem vetores
 - Na verdade, os vetores são ponteiros com Alocação Estática de memória.
 - Dessa forma o computador aloca a memória, mas você não pode gerenciá-la, como, por exemplo alterar o tamanho ou liberar a memória quando não necessita mais.
-

Vetores como Ponteiros

- Lembra da função **malloc** que vimos em ponteiros?
- Ela aloca uma determinada quantidade de memória em bytes para um ponteiro

```
int main()  
{  
    float *a = malloc( sizeof(float) );  
  
    free(a);  
    return 0;  
}
```

Aqui estou alocando espaço
para 1 número **float**



Vetores como Ponteiros

- Lembra da função **malloc** que vimos em ponteiros?
- Ela aloca uma determinada quantidade de memória em bytes para um ponteiro

```
int main()  
{  
    float *a = malloc(10 * sizeof(float));  
  
    free(a);  
    return 0;  
}
```

Agora estou alocando espaço para
10 números **float** (um vetor)



Vetores como Ponteiros

- Lembra da função **malloc** que vimos em ponteiros?
- Ela aloca uma determinada quantidade de memória em bytes para um ponteiro

```
int main()
{
    float *a = malloc(10 * sizeof(float));
    preencher(a, 10);
    imprimir(a, 10);
    printf("\nMedia = %f", media(a, 10));
    free(a);
    return 0;
}
```



Note que o código anterior que fizemos com um vetor vai funcionar igualmente com ponteiro. Isso porque um vetor é um ponteiro com alocação estática de memória.

Vetores como Ponteiros

- Então qual a diferença?
 - A memória alocada estaticamente fica reservada até acabar seu escopo.
 - A memória alocada pelo **malloc** fica reservada até a utilização do comando **free** ou até o programa finalizar.
 - Só é possível modificar o tamanho de um vetor por alocação dinâmica de memória. (Veremos como fazer isso em aulas futuras)
-