



<http://www.nickgentry.com/>

Algoritmos e Estruturas de Dados

Disciplina 301477

Programa de Pós-graduação em
Computação Aplicada

Prof. Alexandre Zaghetto

<http://alexandre.zaghetto.com>
zaghetto@unb.br

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

O presente conjunto de *slides* não pode ser reutilizado ou republicado sem a permissão do instrutor.

Módulo 05

Estrutura de dados

Unidimensional Homogênea

Indexadas

(Vetores)

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

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

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

10							
----	--	--	--	--	--	--	--

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

10	5						
----	---	--	--	--	--	--	--

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

10	5	8					
----	---	---	--	--	--	--	--

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

10	5	8	4	2	9	3	1
----	---	---	---	---	---	---	---

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

NOTAS	10	5	8	4	2	9	3	1
-------	----	---	---	---	---	---	---	---

1. Vetores

- Em muitas aplicações precisamos trabalhar com conjuntos de dados que são semelhantes em tipo.
- Por exemplo, o conjunto de notas dos alunos de uma turma.
- Dependendo da natureza do problema, é conveniente colocar estas informações sob um mesmo conjunto e referenciar cada elemento deste conjunto por um número índice.

	0	1	2	3	4	5	6	7
NOTAS	10	5	8	4	2	9	3	1

2. Declaração de Vetores

- Forma geral em C:

```
<tipo> <nome>[<tamanho>];
```



2. Declaração de Vetores

- Exemplo:

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

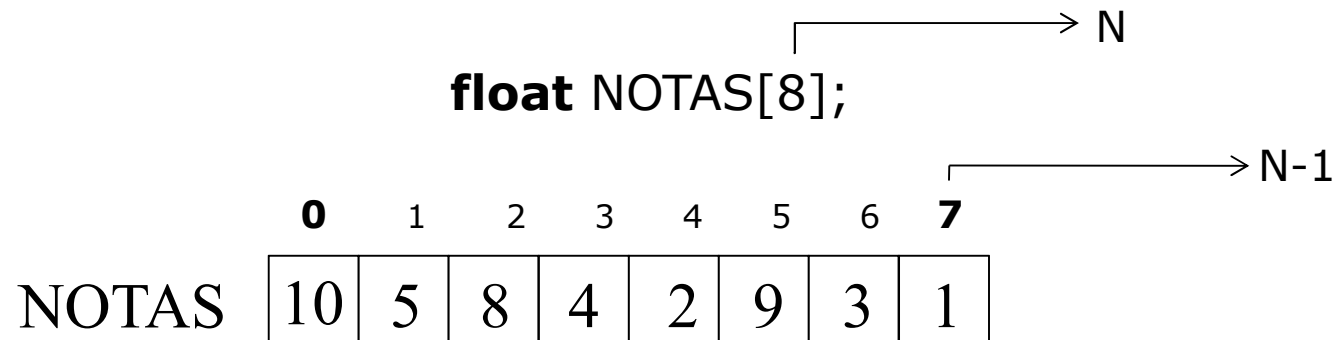
int main()
{
    float SALF[100];
    int CODF[100];
    int FILHOSF[100];

    return 0;
}
```



2. Declaração de Vetores

Um ponto IMPORTANTE que deve ser frisado é que na linguagem C o índice de um vetor de N elementos vai de 0 a N-1, então F[0] é o primeiro elemento, F[N-1] é o último elemento e F[N] é uma variável inválida, pois contando de 0 a N-1 possuímos exatamente N elementos.





3. Preenchimento de Vetores

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

int main()
{
    float notas[3];
    int i;

    for(i=0;i<=2;i++)
    {
        printf("Digite nota:");
        scanf("%f",&notas[i]);
    }

    return 0;
}
```



4. Acessando o Conteúdo de Vetores

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

int main()
{
    float notas[3], media=0;
    int i;

    for(i=0;i<=2;i++)
    {
        printf("Digite nota:");
        scanf("%f",&notas[i]);
        media +=notas[i];
    }

    media /= 3;
```




4. Acessando o Conteúdo de Vetores

```
printf("As notas digitadas foram: \n");  
  
for(i=0;i<=2;i++)  
    printf("NOTA[%d]: %f \n", i, notas[i]);  
  
printf("A media eh: %f \n", media);  
  
return 0;  
  
}
```

5. Alocação Estática

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

```
int main() {
int i, quant;
```

```
printf("Quantas notas deseja entrar?");
scanf("%d",&quant);
```

```
float nota[quant];
```

```
for(i=0;i<=quant-1;i++)
{
    printf("Digite nota:");
    scanf("%f",&nota[i]);
}
```

```
return 0;
```

```
}
```

→ Não façam!
Vamos estudar
isso mais tarde
em **alocação
dinâmica**.

6. Inicialização

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

int main( )
{
    float nota[3]={8,9,10};
    int i;

    for (i=0; i<=2; i++)
        printf ("Nota: %.1f\n", nota[i]);

    return 0;

}
```



6. Inicialização

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

int main( )
{
    float nota[]={8,9,10};
    int i;

    for(i=0;i<=2;i++)
        printf("Nota:%.1f\n",nota[i]);

    return 0;
}
```



6. Inicialização

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

```
int main( )
```

```
{
```

```
float nota[];
```

```
int i;
```

→ Não deve ser utilizado da forma abaixo. Vamos resolver mais tarde.

```
for (i=0; i<=2; i++)
```

```
    printf("Nota:%.1f\n", nota[i]);
```

```
return 0;
```

```
}
```

7. Exemplo

Exemplo 1: Escrever um programa que solicita ao usuário um conjunto de 10 valores reais e verifica quantos estão acima da média.

7. Exemplo

- **Exemplo 1 (boa prática de programação):**

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 10

int main()
{
    float media = 0, valores[MAX];
    int i, conta = 0;

    for(i=0; i<=MAX-1; i++){
        printf("Escreva um valor:");
        scanf("%f", &valores[i]);
        media += valores[i];
    }
```

7. Exemplo

- **Exemplo 1 (boa prática de programação):**

```
media = media/MAX;  
printf("A media eh: %.2f \n\n", media);  
  
for(i=0;i<=MAX-1; i++)  
    if(valores[i] > media) conta++;  
  
printf("Acima de %.2f: %d \n\n", media, conta);  
  
return 0;  
  
}
```


7. Exemplo

Exemplo 2: Um dos mais comuns geradores de números pseudoaleatórios é o *linear congruential generator*, que utiliza a recorrência abaixo. A série de valores gerados por este algoritmo é determinada por um número fixo chamado *semente*.

$$X_{n+1} = (aX_n + b) \bmod m$$

(a) Escreva um algoritmo que gere duas sequências R1 e R2 de números pseudoaleatórios de comprimento 100000 cada uma. Normaliza as sequências de forma que os valores fiquem entre 0 e 1. Considere:

R1 → $X_0 = 5$, $a = 22695477$, $b = 1$ e $m = 1013904223$

R2 → $X_0 = 23$, $a = 22695477$, $b = 1$ e $m = 1013904223$



7. Exemplo

As equações abaixo permitem a obtenção de uma distribuição normal padronizada a partir de duas sequencias R_1 e R_2 geradas aleatoriamente.

$$z_0 = \sqrt{-2 \ln R_1} \cos(2\pi R_2)$$

$$z_1 = \sqrt{-2 \ln R_1} \sin(2\pi R_2)$$

(b) Escreva um algoritmo para gerar uma sequência de valores segundo uma distribuição normal padronizada. Calcule o histograma e o valor da média e da variância do conjunto de valores gerados.

“Uma nova verdade científica não triunfa convencendo seus opositores e fazendo com que vejam a luz, mas porque seus oponentes finalmente morrem e uma nova geração cresce familiarizada.”

Max Plank, em sua Scientific Autobiography