

Vetores em C e C++

(parte 1)

CI208/UFPR: Programação de Computadores

Prof. Wagner M. Nunan Zola

Setembro/2020

Introdução

- Nessa parte do curso iremos trabalhar com **Vetores** na linguagem **C / C++**
- Primeiro vamos relembrar um conhecido conceito matemático: O conceito de **matrizes**. Assim estaremos prontos para entender melhor os **vetores**.
- **Mais adiante** na disciplina passaremos a trabalhar com matrizes.

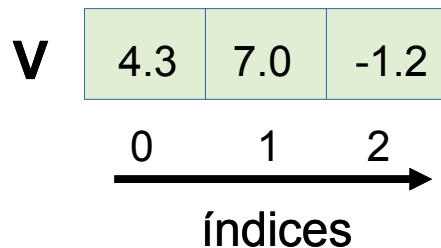
Matrizes x Vetores em C

- Uma matriz é uma coleção de elementos de dados que são do mesmo tipo
 - Exemplos: coleção de inteiros, coleção de caracteres, coleção de floats
- Elementos (ou células) da matriz podem ser indicados pelos índices das células (números indicando posição das células)
- É um conhecido conceito matemático:
 - uma matriz é um arranjo retangular (ou tabela) de números, símbolos ou expressões, organizados em linhas e colunas.
 - Exemplo: a dimensão da matriz M abaixo de 2×3 , ou seja: existem duas linhas e três colunas

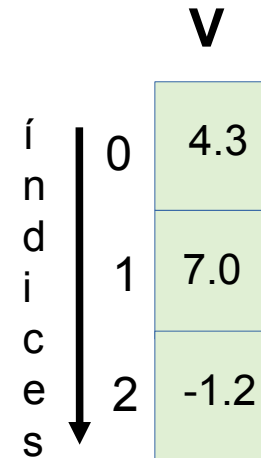
$$\begin{bmatrix} 4.3 & 7.0 & -1.2 \\ -3.0 & 2.2 & 19.1 \end{bmatrix}$$

Matrizes x Vetores em C

- Uma matriz de apenas uma dimensão é também denominada **vetor**
- Podemos **imaginar** um vetor como:
 - uma matriz de apenas uma linha
 - ou
 - uma matriz de apenas uma coluna
- Exemplo de vetor **V** com 3 células do tipo **float**



Note que em C (ou C++) os índices começam em 0



Declaração de Vetores em C/C++

- A declaração de variáveis do tipo **vetor** é feita com a seguinte sintaxe:

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

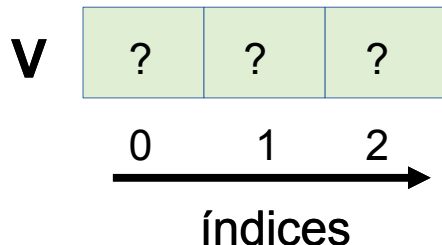
onde:

`<tipo>` É o tipo das células

`<nome>` É o nome do Vetor

`<tamanho>` É a quantidade de células do vetor

Ex: `float V[3];` // declara o vetor V com 3 floats



Note que ao ser declarado (como qualquer variável) o conteúdo de um vetor é **indeterminado**

Inicialização de Vetores

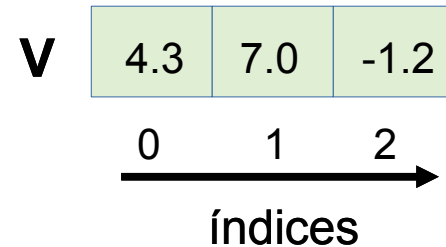
- Podemos inicializar um vetor: célula por célula, por exemplo:

```
float V[3];    // declara o vetor V com 3 floats
```

```
V[0] = 4.3;
```

```
V[1] = 7.0;
```

```
V[2] = -1.2;
```



.

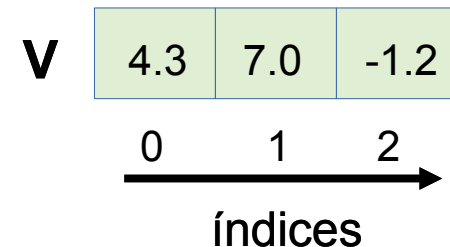
Inicialização de Vetores

- Podemos inicializar um vetor: célula por célula, por exemplo:

```
float V[3]; // declara o vetor V com 3 floats
```

```
V[0] = 4.3;  
V[1] = 7.0;  
V[2] = -1.2;
```

- Podemos inicializar um vetor na própria declaração, por exemplo:



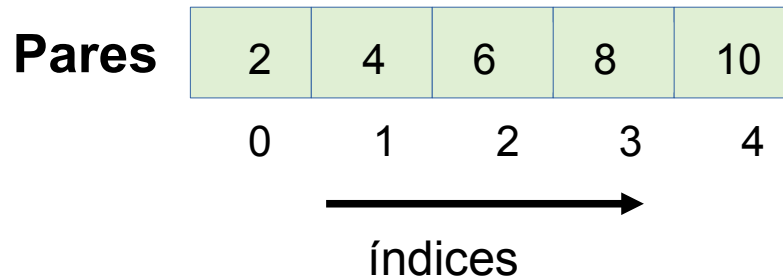
```
float V[3] = {4.3, 7.0, -1.2}; // declara o vetor V inicializando...  
                                // ... suas células conforme figura
```

Inicialização de Vetores

- Podemos inicializar um vetor: por códigos com repetições, por exemplo:

```
int Pares[5]; // declara o vetor Pares com 5 inteiros
int i;

i = 0;
while( i < 5 ) {
    Pares[i] = 2*i + 2;    // inicializa posição i com valor 2*i + 2
    i = i + 1;
}
```



Inicialização de Vetores

- Podemos inicializar um vetor: por códigos com repetições, por exemplo:

```
int Pares[5]; // declara o vetor Pares com 5 inteiros
int i;
```

```
i = 0;
```

```
while( i < 5 ) {
```

```
    Pares[i] = 2*i + 2;    // inicializa posição i com valor 2*i + 2
```

```
    i = i + 1;
```

```
}
```

Note que `i = i + 1` é muito usado.

Vamos aprender uma abreviação:

`i++;` // isso é o mesmo que `i = i + 1`

Pares

2	4	6	8	10
---	---	---	---	----

0 1 2 3 4



índices

Inicialização de Vetores

- Podemos inicializar um vetor: por códigos com repetições, por exemplo:

```
int Pares[5]; // declara o vetor Pares com 5 inteiros
int i;
```

```
i = 0;
while( i < 5 ) {
    Pares[i] = 2*i + 2;    // inicializa posição i com valor 2*i + 2
    i++;
}
```

Note que $i = i + 1$ é muito usado.
Vamos aprender uma abreviação:

$i++$; // isso é o mesmo que $i = i + 1$
// vamos substituir

Pares

2	4	6	8	10
0	1	2	3	4

→
índices

Inicialização de vetores via leitura de dados

- Podemos inicializar um vetor: lendo **cada célula** via cin
- Também vamos precisar de repetições:

```
int Pares[5]; // declara o vetor Pares com 5 inteiros
int i;
```

```
i = 0;
while( i < 5 ) {
    cin >> Pares[i]; // inicializa posição i do vetor V com valor um lido
    i++;
}
```

Supondo que ao executar, o usuário digitou os 5 primeiros pares, teremos o mesmo resultado anterior no vetor **Pares**

Pares

2	4	6	8	10
0	1	2	3	4

índices

Hora para um exemplo um pouco mais complicado!

- Fazer um programa para ler dois vetores A e B, ambos de 5 elementos float.
O programa deve calcular o vetor C com a soma vetorial: $C = A + B$
Em seguida o programa deve imprimir o vetor C resultante.
- Note que, a soma vetorial acima é uma **notação matemática**
 - Não podemos somar dois vetores diretamente assim em C/C++
 - Devemos usar a repetição para somar cada posição equivalente de A e B e colocar resultado em C.
 - Ou seja, devemos fazer uma **repetição**:
para cada célula i (**dentro do while**) devemos calcular:
 $C[i] = A[i] + B[i];$
- O programa pode ser visto no slide seguinte

Programa exemplo para soma vetorial

/* Fazer um programa para ler dois vetores A e B, ambos de 5 elementos float. O programa deve calcular o vetor C com a soma vetorial: $C = A + B$. Em seguida o programa deve imprimir o vetor C resultante. */

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( ) {
```

```
    float A[5], B[5], C[5];    // declara os 3 vetores
```

```
    int i;
```

```
    cout << "Digite o vetor A: ";
```

```
    i = 0;
```

```
    while( i < 5 ) {    // leitura do Vetor A
```

```
        cin >> A[i];
```

```
        i++;
```

```
    }
```

```
    cout << "Digite o vetor B: ";
```

```
    i = 0;
```

```
    while( i < 5 ) {    // leitura do Vetor B
```

```
        cin >> B[i];
```

```
        i++;
```

```
    }
```

```
    // continua
```

```
    i = 0;
```

```
    while( i < 5 ) {    // Soma vetorial
```

```
        C[i] = A[i] + B[i];
```

```
        i++;
```

```
    }
```

```
    cout << "A soma vetorial é: ";
```

```
    i = 0;
```

```
    while( i < 5 ) {    // imprime do Vetor C
```

```
        cout << C[i] << " ";
```

```
        i++;
```

```
    }
```

```
    cout << endl;    // troca de linha no final
```

```
    return 0;
```

```
} // fim do programa
```

Simplificando a notação com o comando **for**

- Como vimos o padrão comum de uso do comando **while** é:

```
    Inicialização ;  
  
while( expressão lógica ) {  
    comando1 ;  
    ...  
    comando n ;  
    passo ;  
}
```

// **inicializa** variáveis do while
// (logo acima do while!)

// comandos a serem repetidos

// dá o **passo** para a próxima iteração
// (o passo é dado ao final do while!)

- O comando **for** é uma simplificação sintática equivalente ao padrão de repetição acima:

```
for( Inicialização ; expressão lógica ; passo ) {  
    comando1 ;  
    ...  
    comando n ;  
}
```

// comandos a serem repetidos

Vantagens na utilização do **for**

- É uma notação mais simples (programas mais compactos)
- Visualização da inicialização, expressão lógica e passo **na mesma linha**
- **Fácil de ler** (quando nos acostumamos à notação)

```
for( Inicialização ; expressão lógica ; passo ) {  
    comando1 ;  
    ...  
    comando n ;  
}
```

// comandos a serem repetidos

Por exemplo, podemos facilmente calcular:

$$S = \sum_{i=0}^{i=N} i$$

Assim:

```
S = 0;  
for( i=0; i<=N ; i++ )  
    S = S + i;
```

Agora podemos reescrever o programa usando **for**

/* Fazer um programa para ler dois vetores A e B, ambos de 5 elementos float. O programa deve calcular o vetor C com a soma vetorial: $C = A + B$. Em seguida o programa deve imprimir o vetor C resultante. */

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( ) {
```

```
    float A[5], B[5], C[5];    // declara os 3 vetores
```

```
    int i;
```

```
    cout << "Digite o vetor A: ";
```

```
    for( i=0; i < 5; i++ )    // leitura do Vetor A
```

```
        cin >> A[i];
```

```
    cout << "Digite o vetor B: ";
```

```
    for( i=0; i < 5; i++ )    // leitura do Vetor B
```

```
        cin >> B[i];
```

```
    for( i=0; i < 5; i++ )    // Soma vetorial
```

```
        C[i] = A[i] + B[i];
```

```
    cout << "A soma vetorial é: ";
```

```
    for( i=0; i < 5; i++ )    // imprime do Vetor C
```

```
        cout << C[i] << " ";
```

```
    cout << endl;    // troca de linha no final
```

```
    return 0;
```

```
} // fim do programa
```

Erros e problemas comuns em programas com vetores

- Vários erros comuns podem ocorrer com programas que trabalham com vetores. Alguns desses erros podem ser detectados mas dependendo do caso podem ocorrer sem nosso conhecimento. Devemos estar atentos a esses erros.
- Índices inválidos: Quando os índices (posições) das células de vetores usados nos programas tem valores errados dizemos que o programa trabalha com **índices inválidos**. São programas que apresentam erros de programação (bugs).
- Em **casos menos graves**, ao rodar o programa com índices inválidos, o programa cancela por mecanismos do sistema operacional. Nesses casos o programa termina abruptamente e imprime na tela uma mensagem do tipo “**segmentation Fault**” (ou **falha de segmentação**). Esses casos são menos graves pois o “bug” pode ser percebido.
- Em alguns casos o programa com índices inválidos pode entrar em **loops infinitos**. Nesses casos o o programa fica em loop e não termina nunca, a não ser que o usuário cancele o programa.
- **No pior caso**, o programa com índices inválidos pode não terminar com falha de segmentação, nem entrar em loop. Nesse pior caso, o programa apenas **faz cálculos e gera respostas erradas** (sem conhecimento do usuário). Esse caso é o pior porque pode gerar prejuízos ou desastres.
- **Como resolver?** O programador deve ter o cuidado de **produzir programas corretos**, sem índices inválidos. O teste dos programas pode ajudar, mas não garante correção.

Exemplo: imprimir sequência reversa

```
/* Fazer um programa para ler sequencia de 10 inteiros e imprimir em ordem reversa. */  
#include <iostream>  
using namespace std;  
  
int main ( ) {  
    int Vet[ 10 ];    // declara um vetor de 10 ints  
    int i;  
  
    cout << "Digite uma sequencia de 10 inteiros: ";  
    for( i=0; i < 10; i++ )  
        cin >> Vet[i];  
  
    cout << "A sequencia reversa é: ";  
    for( i=9; i >= 0; i-- )    // percorre o vetor em ordem reversa (da posição 9 → posição 0)  
        cout << Vet[i] << " ";  
    cout << endl;            // troca de linha ...  
                                // ... depois de imprimir a sequência  
  
    return 0;  
} // fim do programa
```

Exemplo: ler, copiar em ordem reversa, imprimir vetor

/ Fazer um programa para ler sequencia de 10 inteiros armazenando no vetor **Vet**.*

*Copiar o vetor **Vet** no vetor **Rev** em **ordem reversa**. Imprimir o vetor **Rev** */*

```
#include <iostream>
```

```
using namespace std;
```

```
int main ( ) {
```

```
    int Vet[ 10 ];           // declara um vetor de 10 ints
```

```
    int Rev[ 10 ];           // declara um vetor de 10 ints para conter a cópia reversa
```

```
    int i;
```

```
    cout << "Digite uma sequencia de 10 inteiros: ";
```

```
    for( i=0; i < 10; i++ )
```

```
        cin >> Vet[i];
```

```
    for( i=0; i < 10; i++ )           // percorre o vetor em ordem, MAS ...
```

```
        Rev[9-i] = Vet[i];           // copia para Rev nas posições reversas
```

```
    cout << "O vetor Rev contém: ";
```

```
    for( i=0; i < 10; i++ )
```

```
        cout << Rev[i] << " ";
```

```
    cout << endl;           // troca de linha ...
```

```
                                // ... depois de imprimir a sequência
```

```
    return 0;
```

```
} // fim do programa
```

Exercícios adicionais

1) Faça um programa que leia DOIS vetores (**A** e **B**) de **N** posições com números reais (**N** definido por #define). O programa deve calcular e imprimir o produto escalar dos dois vetores.

2) Faça um programa que usa um vetor **Vet** de até **N** posições com inteiros positivos (**N** definido por #define). O programa deve ler os números e colocá-los no vetor **Vet**. O último número digitado será negativo e apenas indicará o fim da digitação do vetor **Vet**.

O programa deve contar quantos inteiros positivos foram colocados no vetor.

Em seguida o programa deve ler números do teclado e para cada número lido indicar ao usuário se ele está ou não no vetor **Vet**. Nesse ponto, a digitação de um número negativo deve causar o término do programa.

3) Faça um programa que leia DOIS vetores (**A** e **B**) de **N** posições com números inteiros (**N** definido por #define). Suponha que **A** e **B** representam conjuntos e seus elementos estão em ordem qualquer.

O programa deve calcular e imprimir o vetor **C** tal que $C = A \cap B$, ou seja **C** deve conter a interseção entre os conjuntos **A** e **B**.

Bibliografia adicional:
favor consultar também

“Linguagem C++ - Notas de Aula”. Revisão para C++ por Armando Luiz N. Delgado, a partir de material de Carmem S. Hara e Wagner N. Zola. 2018.

http://www.inf.ufpr.br/ci208/NotasAula/notas-13_Vetores_ou_Arrays.html#SECTION01013700000000000000

Créditos: O conteúdo original deste documento é de autoria do Prof. Wagner M. Nunan Zola, para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183)

Compartilhe este documento de acordo com a licença abaixo



Este documento está licenciado com uma Licença *Creative Commons Atribuição-NãoComercial-SemDerivações* 4.0 Internacional.
<https://creativecommons.org/licenses/by-nc-sa/4.0/>

