



Aula 19: Vetores, ponteiros e funções

Introdução a Programação

Túlio Toffolo & Puca Huachi
<http://www.toffolo.com.br>

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?
- 3 Vetores e funções
- 4 Aritmética de ponteiros
- 5 Exercícios

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?
- 3 Vetores e funções
- 4 Aritmética de ponteiros
- 5 Exercícios

Vetores em C/C++

- Conhecidos em C/C++ como **arrays**.
 - Correspondem a posições de memória.
 - São identificados por um nome.
 - Individualizadas por índices.
 - Conteúdo do mesmo tipo.
-
- Resumindo: vetores são **posições de memória** identificadas por um mesmo **nome**, individualizadas por **índices** e cujo conteúdo é do **mesmo tipo**.

Declaração de um vetor

<tipo> identificador [<número de posições>];

- Tipo: int, float, double, etc.
- Identificador: é o nome da variável que identifica o vetor.
- Número de posições: é o tamanho do vetor!

Exemplos:

```
1 int vetor[5];
```

```
1 double notas[50];
```

```
1 char palavra[20];
```

Declaração de um vetor

- Ao declaramos um vetor, os seus elementos não são inicializados.
- Mas é possível atribuir valores iniciais.
- O valores iniciais são colocados entre chaves

Exemplos:

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

```
1 double notas[5] = {0.0, 10.0, 7.5, 8.5, 9.9};
```

Declaração de um vetor

Importante:

- A quantidade de valores entre chaves não deve ser maior que o número de elementos
- A fim de facilitar a inicialização, C/C++ permite deixar o número de elementos em branco [].
- Neste caso, o compilador vai supor que o tamanho do vetor é igual ao número de valores especificados entre chaves

```
1 int vetor[] = {0, 2, 5, 3, 9}; // tamanho = 5
```

```
1 double notas[] = {10.0, 9.5, 7.5}; // tamanho = 3
```

Declaração de um vetor

Diferentes forma de declarar um vetor:

```
1 // declaração sem inicializar os valores do vetor (eles terão 'lixo')
2 int v1[3];
3
4 // declaração inicializando os valores do vetor
5 int v2[3] = {0, 2, 5};
6
7 // declaração alternativa inicializando os valores do vetor
8 int v3[] = {0, 2, 5};
```


Uso de constantes em vetores

```
1  ...
2  #define TAM_MAX 10
3
4  int main()
5  {
6      double vetor[TAM_MAX];
7
8      // coloca os valores {TAM_MAX, TAM_MAX-1, ..., 1} no vetor
9      for (int i = 0; i < TAM_MAX; i++) {
10         vetor[i] = TAM_MAX - i;
11     }
12     ...
13     return 0;
14 }
```

Uso de constantes em vetores (2)

```
1  ...
2  const int TAM_MAX = 10;
3
4  int main()
5  {
6      double vetor[TAM_MAX];
7
8      // coloca os valores {0, 1, ..., TAM_MAX - 1} no vetor
9      for (int i = 0; i < TAM_MAX; i++) {
10         vetor[i] = i;
11     }
12     ...
13     return 0;
14 }
```

Criando uma cópia de um vetor

```
1  ...
2  #define TAM_MAX 20
3
4  int main()
5  {
6      double vetor[TAM_MAX];
7      for (int i = 0; i < TAM_MAX; i++) {
8          vetor[i] = i;
9      }
10
11     ...
12
13     // copiando cada posição do vetor
14     double copia[TAM_MAX];
15     for (int i = 0; i < TAM_MAX; i++) {
16         copia[i] = vetor[i];
17     }
18 }
```

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?**
- 3 Vetores e funções
- 4 Aritmética de ponteiros
- 5 Exercícios

Vetores são ponteiros?

Para responder a esta pergunta, vamos entender como funciona a alocação de um vetor:

- Um vetor ocupa um total de **tamanho do vetor** \times **tamanho do tipo** bytes. Assim, quantos bytes ocupará o vetor a seguir?

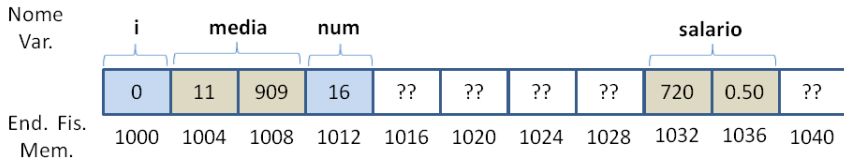
```
1 int vetor[5];
```

- Na linha acima, será alocado um vetor que ocupa $5 \times 4 = 20$ bytes.
- Estes bytes estarão em posições **contíguas!**
- A variável `vetor` será utilizada para acessar a memória alocada.

Alocação de memória para vetores

Exemplo (assuma que toda a memória disponível está ilustrada abaixo):

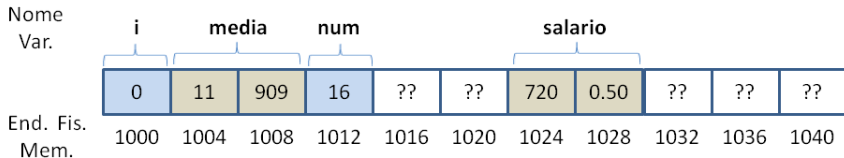
```
float salario;  
int i = 0;  
float media = 11909;  
int num;  
...  
num = 16;  
salario = 7200.50;  
int vet[4]; // ok, existe espaço
```



Alocação de memória para vetores

Exemplo (assuma que toda a memória disponível está ilustrada abaixo):

```
float salario;  
int i = 0;  
float media = 11909;  
int num;  
...  
num = 16;  
salario = 7200.50;  
int vet[4]; // não existe espaço
```



Vetores são ponteiros?

- Mas... o que o código a seguir imprimirá?

```
1  /* O código a seguir vai gerar um 'warning':  
2     * format specifies type 'int' but the argument has type 'int *'  
3     */  
4     int vetor[5];  
5     printf("Vetor: %d\n", vetor);
```

- Imprimirá o **endereço de memória** do início de `vetor`. Mas para esta impressão, devemos utilizar o formato `"%p"` ao invés de `"%d"`.

```
1     int vetor[5];  
2     printf("Vetor: %p\n", vetor);
```


Vetores são ponteiros?

Então, a variável `vetor` do código abaixo armazena um **endereço de memória**:

```
1 int vetor[5];
```

- No entanto, `vetor` tem uma característica especial: é **“read-only”**.
- Portanto o **endereço de memória** para o qual `vetor` aponta não pode ser alterado.

Erros comuns

Note que o código a seguir resultará em um erro de compilação

```
1  int vetor[5];
2  int vetor2[5];
3  vetor2 = vetor;
4  // error: array type 'int[5]' is not assignable
```

- Lembre-se que as variáveis `vetor` e `vetor2` são vetores (ou *arrays*) alocados estaticamente pelo compilador.
- Estas variáveis não podem ter seus valores (que são endereços de memória) alterados.
- Cuidado para não confundir: nós podemos, sim, alterar o **conteúdo** da memória apontada por estas variáveis!!!

Conteúdo de vetores

O que o código abaixo vai imprimir?

```
1 int vetor[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
2 printf("*vetor = %d\n", *vetor);
```

- Vai imprimir o **conteúdo** do primeiro inteiro de **vetor**. Ou seja:

```
1 *vetor = 1
```

Conteúdo de vetores

E o código abaixo?

```
1 int vetor[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
2 printf("vetor[0] = %d\n", vetor[0]);
```

- Vai imprimir o **conteúdo** do inteiro na posição 0 de `vetor`. Ou seja, o **conteúdo** do endereço de memória `vetor+0`.

```
1 vetor[0] = 1
```

Conteúdo de vetores

E o código abaixo?

```
1 int vetor[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };  
2 printf("vetor[5] = %d\n", vetor[5]);
```

- Vai imprimir o **conteúdo** do inteiro na posição 5 de `vetor`. Ou seja, o **conteúdo** do endereço de memória `vetor+5`.
- Note que ao somar 5 em `vetor`, será somado o valor $20 = 5 \times 4$ ao endereço de memória de `vetor`, onde:
 - 5 é o número de elementos;
 - 4 é o tamanho de cada elemento (no caso, de um `int`).

```
1 vetor[5] = 6
```

Conteúdo de vetores

E o código abaixo?

```
1 int nro = 10;
2 int *p = &nro;
3 printf("*p = %d\n", *p);
4 printf("p[0] = %d\n", p[0]);
```

- Vai imprimir o **conteúdo** do ponteiro `p` e, em seguida, o **conteúdo** do endereço de memória `p+0`.

```
1 *p = 10
2 p[0] = 10
```

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?
- 3 Vetores e funções**
- 4 Aritmética de ponteiros
- 5 Exercícios

Exemplo: Busca

Dada uma coleção de n elementos, pretende-se saber se um determinado elemento está presente nessa coleção. Para efeitos práticos, vamos supor que essa coleção é implementada como sendo um vetor de n elementos inteiros:

`vetor[0] . . vetor[n-1]`

Exemplo: Busca

Uma possível solução é percorrer o vetor desde a primeira até a última posição em busca do valor.

- Para cada posição i , verificamos se `vetor[i]` é igual ao valor procurado.
- Se chegarmos ao fim do vetor sem sucesso, podemos afirmar que o valor procurado não está no vetor.

```
1 #define N 10
2
3 int main()
4 {
5     int vetor[N] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
6     int valor;
7     scanf("%d", &valor);
8
9     // variáveis auxiliares
10    int i;
11    int encontrado = 0;
12
13    // buscando elemento
14    for (i = 0; i < N; i++) {
15        if (vetor[i] == valor) {
16            encontrado = 1;
17            break;
18        }
19    }
20
21    if (encontrado)
22        printf("Item encontrado na posição %d\n", i);
23    else
24        printf("Item não foi encontrado");
25 }
```

```
1 #define N 10
2
3 int main()
4 {
5     int vetor[N] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
6     int valor;
7     scanf("%d", &valor);
8
9     // variáveis auxiliares
10    int i;
11    int encontrado = 0;
12
13    // buscando elemento
14    for (i = 0; i < N && !encontrado; i++) {
15        if (vetor[i] == valor)
16            encontrado = 1;
17    }
18
19    if (encontrado)
20        printf("Item encontrado na posição %d\n", i);
21    else
22        printf("Item não foi encontrado");
23 }
```

Vetores e funções

E... se quisermos criar uma função que busca um elemento?

- O que a função retornaria?
- Quais seriam os parâmetros da função?
 - Vamos precisar saber qual **vetor** e qual o **tamanho**.
- Assim, qual seria um possível protótipo para a função?

```
1  /* Função que busca um número em um vetor de inteiros e retorna a
2   * posição em que o número está; caso o número não seja encontrado,
3   * a função retorna -1.
4   */
5  int buscaLinear(int vetor[], int tamanho, int valor);
```

Vetores e funções

Poderíamos também utilizar a notação de ponteiros:

```
1  /* Função que busca um número em um vetor de inteiros e retorna a
2   * posição em que o número está; caso o número não seja encontrado,
3   * a função retorna -1.
4   */
5  int buscaLinear(int *vetor, int tamanho, int valor);
```

Que neste contexto é equivalente à anterior:

```
1  /* Função que busca um número em um vetor de inteiros e retorna a
2   * posição em que o número está; caso o número não seja encontrado,
3   * a função retorna -1.
4   */
5  int buscaLinear(int vetor[], int tamanho, int valor);
```

Vetores e funções

E eis a implementação:

```
1  /* Função que busca um número em um vetor de inteiros e retorna a
2   * posição em que o número está; caso o número não seja encontrado,
3   * a função retorna -1.
4   */
5  int buscaLinear(int vetor[], int tamanho, int valor) {
6      for (int i = 0; i < tamanho; i++){
7          if (vetor[i] == valor) {
8              return i;
9          }
10     }
11
12     return -1;
13 }
```

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?
- 3 Vetores e funções
- 4 Aritmética de ponteiros**
- 5 Exercícios

Ponteiros e vetores

A variável que representa um vetor pode ser vista como um ponteiro.

- Mas... o que o código a seguir vai imprimir?

```
1 int v[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };  
2 printf("*v = %d", *v);
```

```
1 *v = 1
```

- `*v` imprime o conteúdo (do tipo `int`) que está no endereço de memória `v`, que é exatamente igual a `v[0]`.

Ponteiros e vetores

Portanto:

- $v[0]$ é equivalente a $*v$, que é equivalente a $*(v+0)$
- $v[4]$ é equivalente a $*(v+4)$

Aritmética de ponteiros:

- Ao somar 4 em um ponteiro do tipo `int*`, estamos “pulando” 4 inteiros.
- Assim, podemos utilizar **indexação** ($v[4]$) ou aritmética de ponteiros ($*(v+4)$) para ler/escrever na memória.

Ponteiros e vetores

Qual a diferença prática das funções a seguir?

```
1 void imprimeVetor1(int v[], int n) {
2     for (int i = 0; i < n; i++)
3         printf("%d ", v[i]);
4     printf("\n");
5 }
6
7 void imprimeVetor2(int *v, int n) {
8     for (int i = 0; i < n; i++)
9         printf("%d ", v[i]);
10    printf("\n");
11 }
12
13 void imprimeVetor3(int *v, int n) {
14     for (int i = 0; i < n; i++)
15         printf("%d ", *(v+i));
16    printf("\n");
17 }
```

Ponteiros e vetores

Exemplos de utilização:

```
1  int main()  
2  {  
3      int v[5] = { 100, 101, 102, 103, 104 };  
4  
5      imprimeVetor1(v, 5);  
6      imprimeVetor2(v, 5);  
7      imprimeVetor3(v, 5);  
8  
9      return 0;  
10 }
```

Resultado:

```
1  100 101 102 103 104  
2  100 101 102 103 104  
3  100 101 102 103 104
```

Aula de Hoje

- 1 Breve revisão
- 2 Vetores são ponteiros?
- 3 Vetores e funções
- 4 Aritmética de ponteiros
- 5 Exercícios**

Exercícios

Exercício 1

Crie uma função que retorna o maior número em um vetor de inteiros.

Dica: utilize o protótipo a seguir.

```
1 int maior(int *vetor, int tamanho);
```

Exercício 2

Crie uma função que retorna a média dos valores de um vetor de double. Utilize **aritmética de ponteiros** neste exercício.

Dica: utilize o protótipo a seguir.

```
1 double media(double vetor[], int tamanho);
```



Perguntas?