

Vetores

Rafael Beserra Gomes

UFRN

Material compilado em 9 de setembro de 2018.

Licença desta apresentação:



<http://creativecommons.org/licenses/>

Vetores

Vetores

- **Problema:** armazenar em uma variável mais de um valor do mesmo tipo
- Por exemplo:
 - uma sequência de números que o usuário digitou (int)
 - o histórico de valor do dólar (float)
 - uma frase como sequência de caracteres (char)
- Criar inúmeras variáveis é inviável

```
1 #include <stdio.h>
2
3 int main() {
4
5     int n1, n2, n3, n4, n5;
6     scanf("%d %d %d %d %d", &n1, &n2, &n3, &n4, &n5);
7
8     return 0;
9 }
```

Vetores

- **Arranjos (array)**: conjunto de elementos identificáveis por um índice
- Arranjos unidimensionais: **vetores**
- Arranjos bidimensionais: **matrizes** (em aula posterior)

Vetores

Representações de vetores:

- Matematicamente: $v = (v_1, v_2, \dots, v_{n-1}, v_n)$
- Computacionalmente:
 - Os elementos são indexados por um índice (geralmente $v[i]$)
 - Geralmente começam do índice **0**

Exemplo:

- Vetor tamanho 4, índices de 0 a 3

vetor	4	6	2	1
índice	0	1	2	3

Vetores em C

Declarando um vetor em C

Há várias opções:

```
1 #include <stdio.h>
2
3 int main() {
4
5     int vetor1[100];
6     int vetor2[] = {5, 1, 3, 9};
7     int vetor3[5] = {1, 2, 3, 4, 5};
8
9     return 0;
10 }
```

- **único tipo**
- tamanho do vetor: **suficiente para caber os dados**
- tamanho do vetor será especificado no problema
- tamanhos flexíveis: (**VLA** ou **alocação dinâmica**)

Acesso ao índice

Basta identificar o elemento usando o seu **índice** entre [] (lembre-se de que começa com 0):

```
1 #include <stdio.h>
2
3 int main() {
4
5     int vetor[] = {4, 6, 2, 1};
6
7     printf("%d\n", vetor[2]);
8
9     return 0;
10 }
```

vetor	4	6	2	1
índice	0	1	2	3



Exercício em sala

Declare um vetor de 5 inteiros, inicializando seus valores em 1, 4, 5, 7 e 9. Acessando o 2º número do vetor, modifique seu valor de 4 para 3. Depois escreva seus valores na tela utilizando uma estrutura de repetição.

Lembre-se de que não necessariamente precisa usar todo o vetor:

numeros	8	7	8	6	5	8	16
índice i	0	1	2	3	4	5	6
n = 5					↑		

```
1 #include <stdio.h>
2
3 int main() {
4
5     int numeros[10];
6     int n;
7
8     scanf("%d", &n);
9     for(int i = 0; i < n; i++)
10         scanf("%d", &numeros[i]);
11
12     printf("Os numeros digitados foram: ");
13     for(int i = 0; i < n; i++)
14         printf("%d", numeros[i]);
15
16     return 0;
17 }
```



Exercício em sala

Escreva um programa em C com os seguintes passos:

- declarar um vetor de 10 inteiros
- ler um número inteiro **n** (assuma que o usuário digita $n \leq 10$)
- ler **n** inteiros, armazenando-os no vetor
- ler um número inteiro **x**
- escrever na tela quantos dos **n** números são iguais a **x**

Vetores também podem ser usados para **contar/marcar**:

```
1 #include <stdio.h>
2
3 int main() {
4
5     int numOcorrencias[10];
6     int n;
7
8     for(int i = 0; i < 10; i++)
9         numOcorrencias[i] = 0;
10
11     printf("Digite 20 numeros entre 0 e 9: ");
12     for(int i = 0; i < 20; i++) {
13         scanf("%d", &n);
14         numOcorrencias[n]++;
15     }
16
17     for(int i = 0; i < 10; i++)
18         printf("%d ocorreu %d vezes\n", i, numOcorrencias[i]);
19
20     return 0;
21 }
```



Exercício em sala


Escreva um programa que leia números inteiros até o usuário digitar 0. Depois, escreva na tela quantas vezes cada número de 1 a 9 foi lido do usuário.

Detalhes Implementacionais

VLA

VLA: variable length array (arranjo de tamanho variável)

```
1 int n;  
2 scanf("%d", &n);  
3 int vetorVLA[n];
```

- **vantagem:** rápido de declarar e mais fácil de entender
- **desvantagens:**
 - risco de **stack overflow**: usa memória da pilha (aprx 8MB)
 - **fraca portabilidade**: nem todos os compiladores implementam
-  Pelas desvantagens, **não use VLA!**, use **alocação dinâmica**

Falha de segmentação

o que acontece acessando fib[4]?

Endereço									valor	tipo	identificação
0xbffff22f	1	1	0	1	1	1	0	1	3.2	real	precoGasolina
0xbffff230	1	1	0	0	1	1	0	0			
0xbffff231	0	1	0	0	1	1	0	0			
0xbffff232	0	1	0	0	0	0	0	0			
0xbffff233	0	1	0	0	1	0	0	1	1	inteiro curto	fib[0]
0xbffff234	0	1	0	0	1	1	0	1	1	inteiro curto	fib[1]
0xbffff235	0	1	0	0	0	1	0	0	2	inteiro curto	fib[2]
0xbffff236	0	0	1	1	0	0	0	0	3	inteiro curto	fib[3]
0xbffff237	0	0	0	0	0	1	0	1	5	inteiro curto	valorIndice
0xbffff238	0	1	0	0	0	0	1	0	B	caractere	letra1
0xbffff239	0	1	0	0	0	0	1	1	C	caractere	letra2



Falha de segmentação (segmentation fault) : acesso indevido de memória ou a um endereço inválido