Arranjos Unidimensionais

Rafael Beserra Gomes

UFRN

Material compilado em 13 de dezembro de 2017.

Arranjos Unidimensionais

Arranjos Unidimensionais

- **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 }
```

Arranjos Unidimensionais

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

Arranjos Unidimensionais

Representações de arranjos unidimensionais:

- Matematicamente: $v = (v_1, v_2, ..., v_{n-1}, v_n)$
- Computacionalmente:
  - Em geral há um armazenamento contíguo na memória
  - Os elementos são indexados por um índice (geralmente v[i])
  - Geralmente começam do índice **0**

Exemplo de dados na memória:

Endereço									valor	tipo	identificação
0xbffff22c	0	0	0	0	0	1	0	1	5	**inteiro curto**	valorIndice
0xbffff22d	0	1	0	0	0	0	1	0	B	**caractere**	letra1
0xbffff22e	0	1	0	0	0	0	1	1	C	**caractere**	letra2
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	I	**caractere**	frase[0]
0xbffff234	0	1	0	0	1	1	0	1	M	**caractere**	frase[1]
0xbffff235	0	1	0	0	0	1	0	0	D	**caractere**	frase[2]
0xbffff236	0	0	1	1	0	0	0	0	0	**caractere**	frase[3]
0xbffff237	0	0	1	1	0	0	0	0	0	**caractere**	frase[4]
0xbffff238	0	0	1	1	0	0	0	1	1	**caractere**	frase[5]
0xbffff239	0	0	1	1	0	0	1	0	2	**caractere**	frase[6]
0xbffff23a	0	0	0	0	0	0	0	0	\0	**caractere**	frase[7]
0xbffff23b	0	1	1	0	0	1	1	0	f	**caractere**	frase[8]

Vetores em C

Declarando um vetor em C

Há várias opções:

```c
#include <stdio.h>

int main() {

    int vetor1[100];
    int vetor2[] = {5, 1, 3, 9};
    int vetor3[5] = {1, 2, 3, 4, 5};

    return 0;
}
```

- coloque uma constante suficiente para caber todos os dados (o limite máximo deve estar claro no problema a ser abordado)
- vetores com tamanhos flexíveis: (**VLA** ou **alocação dinâmica**)

Declarando um vetor em C

PERIGO AO USAR[1]

VLA: variable length array (arranjo de tamanho variável)

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

- **vantagens**: rápido de declarar e mais fácil de entender
- **desvantagem**: usa a memória da pilha (geralmente 8MB)[2], podendo ocasionar **stack overflow**
- **desvantagem**: nem todos os compiladores implementam ocasionando problemas de portabilidade
- caso queira utilizar vetor de tamanho variável use os recursos que veremos no tópico **alocação dinâmica** 👍

---

[1] sugiro nem usar 🙂

[2] use ulimit -s para verificar

Representação do vetor na memória

Cuidado! em C não há verificação de índices válidos
No exemplo abaixo, o que acontece acessando frase[5]?

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	I	**caractere**	frase[0]
0xbffff234	0	1	0	0	1	1	0	1	M	**caractere**	frase[1]
0xbffff235	0	1	0	0	0	1	0	0	D	**caractere**	frase[2]
0xbffff236	0	0	0	0	0	0	0	0	\0	**caractere**	frase[3]
0xbffff237	0	1	1	0	0	1	1	0	f	**caractere**	frase[4]
0xbffff238	0	0	0	0	0	1	0	1	5	**inteiro curto**	valorIndice
0xbffff239	0	1	0	0	0	0	1	0	B	**caractere**	letra1
0xbffff23a	0	1	0	0	0	0	1	1	C	**caractere**	letra2

Se houver um acesso indevido de memória ou a um endereço inválido, ocorre uma **Falha de segmentação (segmentation fault)**

Acesso ao índice

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

```c
#include <stdio.h>

int main() {

    int vetor[] = {4, 6, 2, 1};

    printf("%d\n", vetor[2]);

    return 0;
}
```

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					↑		

```c
#include <stdio.h>

int main() {

    int numeros[10];
    int n;

    scanf("%d", &n);
    for(int i = 0; i < n; i++)
        scanf("%d", &numeros[i]);

    printf("Os numeros digitados foram: ");
    for(int i = 0; i < n; i++)
        printf("%d", numeros[i]);

    return 0;
}
```

✎ 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**:

```c
#include <stdio.h>

int main() {

    int numOcorrencias[10];
    int n;

    for(int i = 0; i < 10; i++)
        numOcorrencias[i] = 0;

    printf("Digite 20 numeros entre 0 e 9: ");
    for(int i = 0; i < 20; i++) {
        scanf("%d", &n);
        numOcorrencias[n]++;
    }

    for(int i = 0; i < 10; i++)
        printf("%d ocorreu %d vezes\n", i, numOcorrencias[i]);

    return 0;
}
```

✎ 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.