Arranjos Unidimensionais

Introdução

Rafael Beserra Gomes

Universidade Federal do Rio Grande do Norte

Material compilado em 13 de março 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)
  - os códigos dos produtos em um carrinho de compras (int)
  - o histórico de valor do dólar (float)
  - uma frase como sequência de caracteres (char)
- Criar inúmeras variáveis é inviável

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**

Representação do vetor na memória

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:

```
1  #include <stdio.h>
2
3  int main() {
4
5      int vetor1[100];
6      int vetor2[] = {5, 1, 3, 9};
7
8      int i = 50;
9      //isso eh chamado de VLA: variable length array
10     //valido a partir de C99, use-o com cautela
11     int vetorVLA[i];
12
13     return 0;
14 }
```

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;
}
```

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**

Strings em C

Strings em C

- **Strings** é o conceito abstrato de uma sequência de caracteres.
- Representação em C: vetor de caracteres ou literais (entre aspas duplas)
- **printf**: use %s no especificador de formato, a escrita é feita até o caractere nulo ('\0')
- **scanf**: use %s no especificador de formato, a leitura é realizada até um caractere separador (espaço, tab, enter)

```
1 char palavra[100] = "IMD0012";
2
3 //passamos o endereco base
4 scanf("%s", &palavra[0]);
5 printf("%s\n", palavra);
6
7 //forma mais facil: palavra ja eh o endereco base do vetor!
8 scanf("%s", palavra);
9 printf("%s\n", palavra);
```

Strings em C

- **Cuidado** com o buffer de entrada! O que ocorre no seguinte código?

```
1 char palavra[100];
2 char t;
3 scanf("%s", &palavra[0]);
4 printf("%s\n", palavra);
5 scanf("%c", &t);
6 printf("(%c)\n", t);
7 scanf("%s", palavra);
8 printf("%s\n", palavra);
```

- Opções para ler uma string com espaços:

```
1 char frase[100];
2 char t;
3 //opcao 1: scanf("%[^\n]s\n", frase);
4 //opcao 2: gets(frase);
5 //opcao 3: fgets(frase, 100, stdin);
6 scanf("%c", &t);
7 printf("(%c)\n", t);
8 printf("%s\n", frase);
```

Ordenação de vetores

Ordenação de vetores

Um dos problemas clássicos da computação é a **ordenação de vetores**. Exemplos de algoritmos:

- Bogosort
- **Bubble sort**
- Insertion sort
- **Selection sort**
- Quicksort
- Merge sort

Há uma diferença significativa de tempo de processamento entre eles.