C
Vetores

Adriano Cruz
adriano@nce.ufrj.br

Instituto de Matemática
Departamento de Ciência da Computação
UFRJ

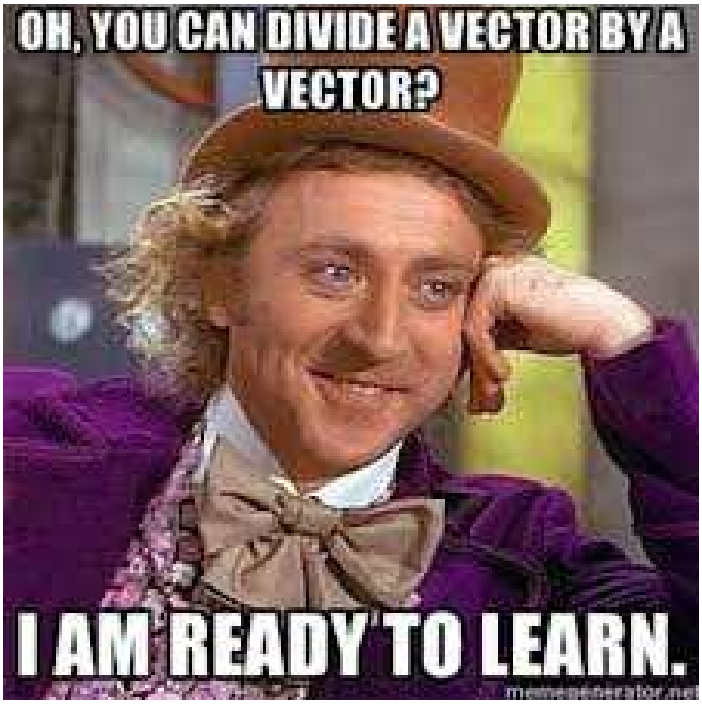
15 de agosto de 2013

Section Summary

Bibliografia

1. Adriano Cruz. *Curso de Linguagem* C, Disponível em `http://equipe.nce.ufrj.br/adriano`
2. Ulysses de Oliveira. *Programando em* C, Editora Ciência Moderna.

Vetores?



O que são?

- Vetores são usados para tratamento de conjuntos de dados que possuem as mesmas características.
- Uma das vantagens de usar vetores é que o conjunto recebe um nome comum e elementos deste conjunto são referenciados através de índices.
- Matemática: v_i, c_1, t_5.
- C: `v[i]`, `c[1]`, `t[5]`
- Pelo nome vetor referenciaremos estruturas que podem ter mais de uma dimensão.
- Por agora estaremos mostrando vetores de tamanhos fixos. Somente após apresentarmos ponteiros iremos abordar alocação de memória para vetores.

Section Summary

Declaração

- A forma geral da declaração de vetores de uma dimensão é: `tipo nome [tamanho];`
- `tipo` é um tipo qualquer de dados.
- `nome` é o nome pelo qual o vetor vai ser referenciado
- `tamanho` é o número de elementos que o vetor vai conter.
- Observar que em `C` o primeiro elemento tem índice `0` e o último `tamanho - 1`.

Examples

```c
int numeros[1000];   /* vetor de 1000 inteiros */
float notas[65];      /* conjunto de 65 numeros reais */
char nome[40];       /* conjunto de 40 caracteres */
```

Espaço de Memória

- O espaço de memória, em bytes, ocupado por um vetor de tipo qualquer é igual a:
  $espaço = tamanho * sizeof(tipo)$
- É importante notar que em C não há verificação de limites em vetores.
- Isto significa que é possível ultrapassar o fim de um vetor e escrever em outras variáveis, ou mesmo em trechos de código.
- É tarefa do programador controlar os limites.

Exemplo I

```c
#define DIM 5
#include <stdio.h>
int main(void)
{
    int vetor[DIM], i, num;

    puts("Entre com o numero inicial do conjunto. ");
    scanf("%d", &num);

    /* Geracao do conjunto */
    for (i = 0 ; i < DIM; i++) vetor[i] = num++;

    /* Impressao do conjunto */
    for (i = 0; i < DIM; i++)
        printf("Elemento %d = %d\n", i, vetor[i]);

    return 0;
}
```

Exemplo II

```c
#define DIM 5
#include <stdio.h>
int main ( void )
{
    int vetor1[DIM], vetor2[DIM], i, prod=0;

    for (i = 0; i < DIM; i++) {
        scanf("%d", &vetor1[i]);
    }
    for (i = 0; i < DIM; i++) {
        scanf("%d", &vetor2[i]);
    }
    for (i = 0; i < DIM; i++) {
        prod += vetor1[i] * vetor2[i];
    }
    printf("O produto vale %d", prod);
    return 0;
}
```


Ordenação Bolha Passo 1

3	9	5	1	2	Compara
3	9	5	1	2	Compara
3	5	9	1	2	Troca
3	5	9	1	2	Compara
3	5	1	9	2	Troca
3	5	1	9	2	Compara
3	5	1	2	9	Troca
3	5	1	2	9	Fim passo 1

Ordenação Bolha Passo 2

- | 3 | 5 | 1 | 2 | 9 | Compara |
- | 3 | 5 | 1 | 2 | 9 | Compara |
- | 3 | 1 | 5 | 2 | 9 | Troca |
- | 3 | 1 | 5 | 2 | 9 | Compara |
- | 3 | 1 | 2 | 5 | 9 | Troca |
- | 3 | 1 | 2 | 5 | 9 | Fim Passo 2 |

Ordenação Bolha Passo 3

- | 3 | 1 | 2 | 5 | 9 | Compara |
- | 1 | 3 | 2 | 5 | 9 | Troca |
- | 1 | 3 | 2 | 5 | 9 | Compara |
- | 1 | 2 | 3 | 5 | 9 | Troca |
- | 1 | 2 | 3 | 5 | 9 | Fim passo 3 |

Ordenação Bolha Passo 4

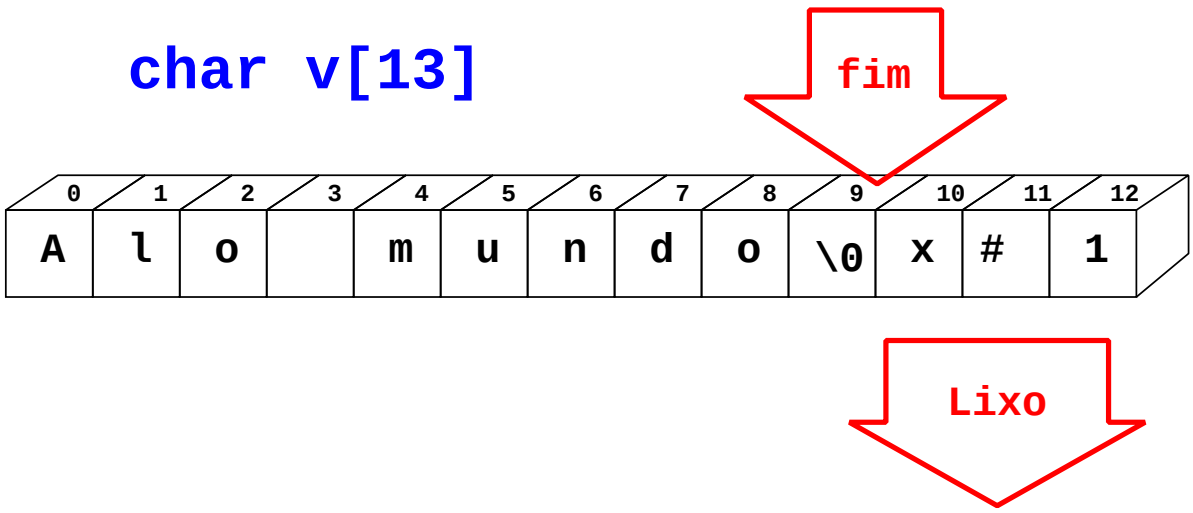
- | 1 | 2 | 3 | 5 | 9 | Compara |
- | 1 | 2 | 3 | 5 | 9 | Fim Passo 4 |

Ordenando

```c
#include <stdio.h>
int main (void) {
    int vetor[5], i, j, temp;
    for (i = 0; i < 5; i++) {
        scanf("%d", &vetor[i]);
    }
    for (i = 0; i < 5 -1; i++) {
        for (j=0; j < 5 - 1 - i; j++) {
            if (vetor[j]>vetor[j+1]) {
                temp = vetor[j];
                vetor[j] = vetor[j+1];
                vetor[j+1] = temp;
            }
        }
    }
    for (i=0; i < 5; i++) printf("%d\n", vetor[i]);
    return 0;
}
```

Section Summary

Cadeia de Caracteres



Cadeia de Caracteres

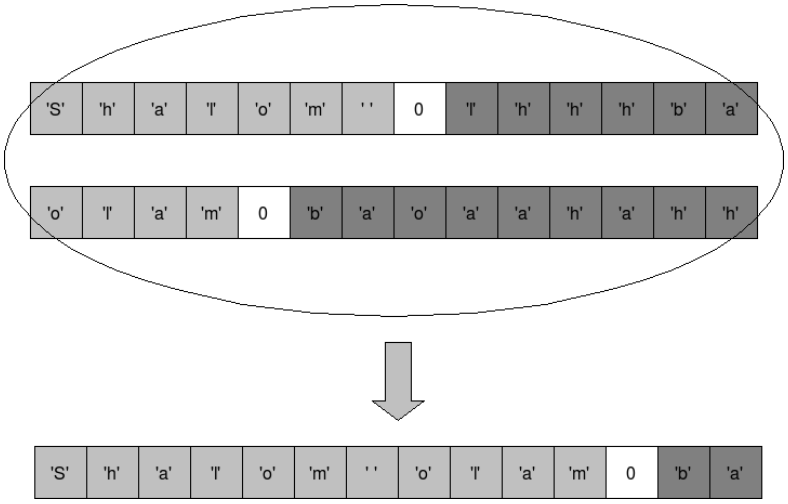
- Um cadeia de caracteres (*string*) é um conjunto de caracteres terminado por um caractere nulo, que é representado como `'\0'`.
- Para especificar um vetor para armazenar um cadeia deve-se sempre reservar um espaço para este caractere.
- Para armazenar um cadeia de 40 caracteres deve-se reservar um vetor de 41 de caracteres.
- Em `C` é possível haver constantes cadeia, que são definidas como uma lista de caracteres entre aspas. Por exemplo,
  `"programando em C"`
- Não é necessário a colocação do caractere nulo ao final da cadeia, o `C` coloca automaticamente.

Sem strings! E agora?

Para facilitar a programação foram criadas algumas funções para manipular cadeias. Algumas das funções mais comuns estão resumidamente descritas a seguir.

Funções I

- char *strcat(char *dest, const char *orig): Concatena cadeia orig ao final de dest. O primeiro caractere de orig substitui o caractere nulo de dest. A função retorna o valor de dest.



Funções II

- char *strcmp (const char *cad1, const char *cad2): Compara lexicograficamente as duas cadeias. Retorna zero se as cadeias são iguais, menor que 0 se cad1 < cad2, maior que 0 se cad1 > cad2.
- char *strncmp (const char *cad1, const char *cad2, size_t n): Compara lexicograficamente até n caracteres das duas cadeias. Retorna zero se as cadeias são iguais, menor que 0 se cad1 < cad2, maior que 0 se cad1 > cad2.
- size_t strlen(const char *cad): Calcula o comprimento da cadeia sem contar o caracter nulo. O comprimento da cadeia é determinado pelo caractere nulo. Não confundir o tamanho da cadeia com o tamanho do vetor que armazena a cadeia.
- char *strcpy(char *dest, const char *orig): Copia cadeia orig para dest. A cadeia destino deve ter espaço suficiente para armazenar orig. O valor de dest é retornado.

Exemplo

```c
#include<string.h>
#include<stdio.h>

int main( void )
{
    char c, nome[81], sobrenome[41];
    int i;

    printf("Entre com um nome ");
    gets(nome);
    printf("Entre com um sobrenome ");
    gets(sobrenome);

    strcat(nome, " "); /* Para que isto? */
    strcat(nome, sobrenome);
    puts(nome);
    return 0;
}
```


Exercício?

1. Leia e imprima duas cadeias de até 80 caracteres.
2. Compare-as e imprima-as em ordem alfabética.
3. Imprima também quantas letras 'a' (minúscula) existem nas duas cadeias.
4. Imprima agora quantas vogais minúsculas existem nas duas cadeias.

Section Summary

Vetores Multidimensionais

1. A forma geral da declaração é a seguinte:
   `tipo nome [dim1][dim2][dim3]...[dimN];`
2. `dimI` é o tamanho da dimensão `I`.
3. Deve-se tomar cuidado com armazenamento de matrizes multidimensionais, por que a memória necessária para guardar estes dados é igual a
   `sizeof(tipo)*dim1*dim2*dim3*...*dimN`
4. `int matriz[10][20];` define uma matriz de 10 linhas por 20 colunas, que ocupa `sizeof(int)*10*20`.
5. O comando `c = matriz[3][8];` armazena o conteúdo do elemento que está na quarta linha e nona coluna na variável `c`.
6. Observar que o primeiro índice indica a linha e o segundo a coluna.

Exemplos

```c
#define DIML 3
#define DIMC 5
#include <stdio.h>
int main( void ) {
    int i, j;
    int matriz[DIML][DIMC];

    for (i=0; i < DIML; i++)
        for (j=0; j < DIMC; j++)
            scanf("%d", &matriz[i][j]);

    for (i=0; i < DIML; i++) {
        for (j=0; j < DIMC; j++)
            printf("%4d", matriz[i][j]);
        printf("\n");
    }
    return 0;
}
```


Exercícios

1. Escreva um programa que leia uma matriz de 3×3 que contém somente os algarismos 0 e 1 e procure linhas que contenham somente um dos dois caracteres. O programa deve imprimir os números das linhas com caracteres iguais.
2. Modifique o programa anterior para que ele também procure colunas.
3. Modifique o programa para que ele procure também nas diagonais da matriz.
4. Modifique o programa para que ele procure em matrizes de 20×20.
5. Modifique o programa para que a matriz contenha não algarismos, mais sim os caracteres 0 e X.

Exemplos

1. Uma operação muito comum em matemática é a multiplicação de matrizes.
2. Considere a matriz $M1$ com $L1$ linhas e $C1$ colunas e a matriz $M2$ com $L2$ linhas e $C2$ colunas.
3. O número de colunas $C1$ de $M1$ deve ser igual ao número de linhas $L2$ de $M2$.
4. O elemento MR_{ij} da matriz resultado MR do produto destas matrizes é definido pela equação

$$MR_{ij} = \sum_{k=1}^{C1} M1_{ik} \times M2_{kj}$$

Multiplicando Matrizes

```c
#include<stdio.h>
int main ( void ) {
    float m1[3][3], m2[3][3], mr[3][3], m;
    int i, j, k;
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            printf ("%d, %d ", i,j);
            scanf("%f", &m1[i][j]);
        }
    }
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            printf ("%d, %d ", i,j);
            scanf("%f", &m2[i][j]);
        }
    }
```

Multiplicando Matrizes

```
    for (i=0; i<3; i++) {
        for (j=0; j<3; j++) {
            m = 0;
            for (k=0; k<3; k++) {
                m += m1[i][k]*m2[k][j];
            }
            mr[i][j] = m;
        }
    }
    for (i=0; i<3; i++ ) {
        for (j=0; j<3; j++) {
            printf("%.3f ", mr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```


The End