

MC-102 — Aula 15

Matrizes e Vetores Multi-Dimensionais

Instituto de Computação – Unicamp

2 de Maio de 2013

Roteiro

- 1 Matrizes e Vetores Multi-Dimensionais
 - Inicialização de Vetores
- 2 Exemplo
- 3 Informações Extras: Representação de Matrizes por Linearização
- 4 Exercícios

Inicialização de Vetores

- Em algumas situações, ao criarmos uma matriz, pode ser útil atribuir valores já na sua criação.
- No caso de vetores, a inicialização é simples: Basta atribuir uma lista de valores constantes de mesmo tipo separados por vírgulas e entre chaves.

Exemplo

```
int vet[5] = {10, 20, 30, 40, 50};
```

- No caso de strings, você pode atribuir diretamente uma constante string.

Exemplo

```
char st1[100] = "sim isto é possível";
```

Inicialização de Vetores

- No caso de matrizes, usa-se chaves para delimitar as linhas:

Exemplo

```
int vet[2][5] = { {10, 20, 30, 40, 50} , {60, 70, 80, 90, 100} } ;
```

- No caso tridimensional, cada índice da primeira dimensão se refere a uma matriz inteira:

Exemplo

```
int v3[2][3][4] = {  
  { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },  
  { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} },  
};
```

Inicialização de Vetores

```
int main(){
    int i,j,k;
    int v1[5] = {1,2,3,4,5};
    int v2[2][3] = { {1,2,3}, {4,5,6}};
    int v3[2][3][4] = {
        { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },
        { {0, 0, 0, 0}, {5, 6, 7, 8}, {0, 0, 0, 0} }
    };
    char st1[100] = "olha que coisa mais linda, mais cheia de graça";

    .
    .
    .
    .
}
```

Inicialização de Vetores

```
int main(){
    .
    .
    .
    char st1[100] = "olha que coisa mais linda, mais cheia de graça";

    printf("\n\n v1 \n");
    for(i=0; i<5; i++){
        printf("%d, ",v1[i]);
    }
    printf("\n\n v2 \n");
    for(i=0; i<2; i++){
        for(j=0; j<3; j++){
            printf("%d, ",v2[i][j]);
        }
        printf("\n");
    }
    .
    .
    .
}
```

Inicialização de Vetores

```
int main(){
    .
    .
    .
    printf("\n\n v3 \n");
    for(i=0; i<2; i++){
        for(j=0; j<3; j++){
            for(k=0; k<4; k++){
                printf("%d, ",v3[i][j][k]);
            }
            printf("\n");
        }
        printf("\n");
    }

    printf("%s",st1);
}
```

Exemplo

Criar uma aplicação com operações básicas sobre matrizes quadradas:

- Soma de 2 matrizes com dimensões $n \times n$.
- Subtração de 2 matrizes com dimensões $n \times n$.
- Cálculo da transposta de uma matriz de dimensão $n \times n$.
- Multiplicação de 2 matrizes com dimensões $n \times n$.

Exemplo

Dentre as funcionalidades, vamos implementar a multiplicação:

- Vamos multiplicar duas matrizes M_1 e M_2 (de dimensão $n \times n$).
- O resultado será uma terceira matriz M_3 .
- Lembre-se que uma posição (i, j) de M_3 terá o produto interno do vetor linha i de M_1 com o vetor coluna j de M_2 :

$$M_3[i, j] = \sum_{k=1}^n M_1[i, k] \cdot M_2[k, j]$$

Exemplo

Em C temos:

```
for(i=0; i<n; i++){
    for(j=0; j<n; j++){
        resp[i][j] = 0;
        for(k=0; k<n; k++){
            resp[i][j] = resp[i][j] + (mat1[i][k]*mat2[k][j]);
        }
    }
}
```

Linearização de Índices

- Podemos usar sempre vetores simples para representar matrizes (na prática o compilador faz isto por você).
- Ao declarar uma matriz como **int mat[3][4]**, sabemos que serão alocadas 12 posições de memória associadas com a variável **mat**.
- Poderíamos simplesmente criar **int mat[12]**. Mas perdemos a simplicidade de uso dos índices em forma de matriz.
 - ▶ Você não mais poderá escrever **mat[1][3]** por exemplo.

Linearização de Índices

- A *linearização de índices* é justamente a representação de matrizes usando-se um vetor simples.
- Mas devemos ter um padrão para acessar as posições deste vetor como se sua organização fosse na forma de matriz.

Linearização de Índices

- Considere o exemplo:
`int mat[12]; // ao invés de int mat[3][4]`
- Fazemos a divisão por linhas como segue:
 - ▶ Primeira linha: **mat[0]** até **mat[3]**
 - ▶ Segunda linha: **mat[4]** até **mat[7]**
 - ▶ Terceira linha: **mat[8]** até **mat[11]**
- Para acessar uma posição $[i][j]$ usamos:
 - ▶ **mat[i*4 + j];**
onde $0 \leq i \leq 2$ e $0 \leq j \leq 3$.

Linearização de Índices

- De forma geral, seja matriz **mat[n*m]**, representando **mat[n][m]**.
- Para acessar a posição correspondente à $[i][j]$ usamos:
 - ▶ **mat[i*m + j];**
onde $0 \leq i \leq n - 1$ e $0 \leq j \leq m - 1$.
- Note que i pula de blocos de tamanho m , e j indexa a posição dentro de um bloco.

Linearização de Índices

- Podemos estender para mais dimensões. Seja matriz $\mathbf{mat}[n*m*q]$, representando $\mathbf{mat}[n][m][q]$.
 - ▶ As posições de 0 até $(m * q) - 1$ são da primeira matriz.
 - ▶ As posições de $(m * q)$ até $(2 * m * q) - 1$ são da segunda matriz.
 - ▶ Etc...
- De forma geral, seja matriz $\mathbf{mat}[n*m*q]$, representando $\mathbf{mat}[n][m][q]$.
- Para acessar a posição correspondente à $[i][j][k]$ usamos:
 - ▶ $\mathbf{mat}[i*m*q + j*q + k]$;

Linearização de Índices

```
int main(){
    int mat[40]; //representando mat[5][8]
    int i,j;

    for(i=0; i<5; i++)
        for(j=0;j<8; j++)
            mat[i*8 + j] = i*j;

    for(i=0; i<5; i++){
        for(j=0;j<8; j++)
            printf("%d, ",mat[i*8 + j]);
        printf("\n");
    }
}
```

Exercício

- Escreva um programa que lê uma string de até 50 caracteres, e imprime “Palindromo” caso a string seja um palindromo e “Nao Palindromo” caso contrário.
- OBS: Um palindromo é uma palavra ou frase, que é igual quando lida da esquerda para a direita ou da direita para a esquerda (acentos e espaços em brancos são descartados).
- Exemplo de palindromo: Saudável leva duas.