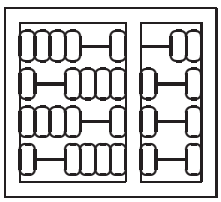

MC-102 Algoritmos e Programação
de Computadores
IC-UNICAMP

Aula 12 - Matrizes

Por: Evandro Cesar Bracht
(Sala IC-86) 1S2005

1 Objetivos

Aprender o uso da estrutura de dados conhecida como matrizes, declaração, inicialização, atribuição de valores e acesso ao conteúdo.

2 Motivação

Utilizar um grande número de variáveis relacionadas sem a necessidade de criar nomes diferentes.

3 Aula e Exemplos

Na aula anterior criamos um programa que lia as notas de uma prova para um conjunto de alunos e então calcular a média da turma. Agora queremos ler as notas de 4 provas para cada aluno e então calcular a média do aluno e a média da classe. O tamanho da turma é no máximo 50 alunos.

Uma solução é criar 4 vetores cada um com 50 posições. E então ler as respectivas informações.

```
int nota0[50],nota1[50],nota2[50],nota3[50];
```

Agora suponha que o número de provas pode ser no máximo 100 provas, se tornaria inviável criar 100 vetores e atribuir 100 nomes diferentes. Para este problema podemos utilizar matrizes.

A matriz para o problema de 50 alunos e 4 notas, pode ser `int notas[50][4]` ou `int notas[4][50]`. Ambas as declarações de matrizes terão 200 espaços da memória destinados a valores de notas. O diferença esta no que significa cada matriz. No primeiro caso temos uma matriz de 50 linhas e 4 colunas, onde cada linha representa um aluno e cada coluna representa uma prova. Já no segundo caso temos uma matriz de 4 linhas com 50 colunas, neste caso as linhas representam as provas e as colunas representam os alunos.

Como vimos, vetores também podem possuir múltiplas dimensões, se declararmos um identificador que é um vetor de vetores, ou seja, uma matriz de duas dimensões. Quando declaramos um vetor de

n posições com `int nota[n]` temos um vetor como ilustra a figura 1. Já se declararmos um vetor de vetores com `int nota[n][n]` teremos uma estrutura semelhante a figura 2

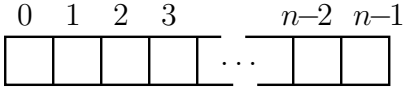

Figura 1: Vetor de n elementos.

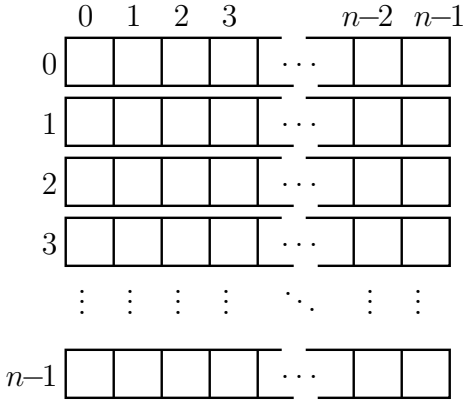

Figura 2: Matriz $n \times n$ elementos.

3.1 A declaração

A sintaxe em C para a criação de um vetor multi dimensional ou matriz é:

tipo identificador[n. de variáveis 1][n. de variáveis 2];

Onde

- *tipo* é o tipo das variáveis que devem ser criados. Exemplo:*int, char, float*, entre outros;
- *identificador* é o nome que será utilizado para referenciar o conjunto de variáveis;
- *n. de variáveis 2* é o número de variáveis de cada vetor que será criado.
- *n. de variáveis 1* é o número de vetores que será criado.

Um exemplo de declaração de matriz em C:

```
int notas[3][8];
```

Neste exemplo criamos 3 vetores onde cada vetor possui 8 variáveis. Podemos interpretar isso como sendo uma matriz 3×8, com um total de 24 variáveis. Esta matriz esta ilustrada na figura 3.

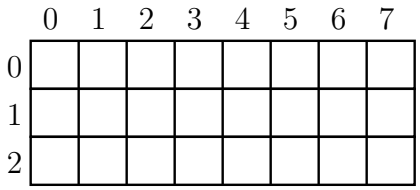

Figura 3: Exemplo de uma matriz 3×8.

Podemos criar também vetor de vetores de vetores ..., ou seja, podemos ter matrizes de múltiplas dimensões. Um exemplo de declaração de uma matriz de 3 dimensões em C:

```
int notas[3][8][4];
```

3.2 Referenciando um elemento do vetor

Após declarar uma matriz precisamos de um modo de referenciar seus elementos individualmente. Isto é feito através do numero entre colchetes para cada dimensão seguindo o nome do vetor.

Note a diferença nas linhas que se segue.

```
int notas[50][4];
notas[5][3] = 85;
```

Na primeira linha temos a declaração da matriz bidimensional *notas* do tipo inteiro contendo 50 vetores de 4 posições. Como vimos anteriormente as linhas da matriz representam os alunos e as colunas representam as notas das provas. Na segunda linha temos a atribuição do valor 85 a quarta posição do sexto vetor da matriz *notas*. Para o problema de notas de alunos, estamos atribuindo a nota 85 para a quarta prova do sexto aluno.

Vale lembrar, assim como em vetores, os indicies de uma matriz variam entre 0 e $n-1$. Além disso, assim como em vetores, podemos referenciar posições da matriz utilizando variáveis inteiras. O exemplo que segue tem o mesmo resultado que o exemplo anterior.

```
int notas[50][4];
int i = 5;
int j = 3;
notas[i][j] = 85;
```

3.3 Armazenando dados em uma matriz

Em seguida temos um exemplo de como fazer a leitura das notas de cada prova para cada aluno.

```
for(i=0; i<50; i++)
  for(j=0; i<4; j++){
    printf("Digite a nota da prova %d do aluno %d: ",j,i);
    scanf("%d",&notas[i][j]);
}
```

3.4 Lendo dados de um matriz

Em seguida temos um exemplo de como é feita a leitura dos dados já armazenados em uma matriz.

```
soma_turma = 0;
for(i=0; i<50; i++){
  soma_aluno = 0;
  for(j=0; j<4; j++)
    soma_aluno = soma_aluno + notas[i][j];
```

```
  soma_turma = soma_turma + soma_aluno/4;
  printf("Media do aluno %d é %f",i,soma_aluno/5);
}
printf("Media da turma %f.",soma_turma/50);
```

3.5 Inicializando matrizes

Assim como em vetores podemos atribuir um valor inicial a uma matriz, esta inicialização deve ser feita no momento da criação da matriz. O trecho de código a seguir exemplifica a inicialização do vetor *teste*.

```
int identidade[4][4]={ {1,0,0,0},
                       {0,1,0,0},
                       {0,0,1,0},
                       {0,0,0,1} };
```

4 Exemplos e/ou Exercícios

1. Uma posição de uma matriz é referenciada por linha e coluna, faça um programa que leia uma matriz 4×4. O programa deve perguntar o valor de cada posição da matriz, em seguida imprima esta matriz na tela.

   R:

   ```
   #include <stdio.h>
   #define LIM 4
   int main(){
     int mat[LIM][LIM];
     int i=0,j=0;
     for (i=0; i<LIM; i++) {
       for (j=0; j<LIM; j++) {
         printf("Digite o valor da posição (%d,%d): ",i,j);
         scanf("%d",&mat[i][j]);
       }
     }
     for (i=0; i<LIM; i++){
       for (j=0; j<LIM; j++)
         printf("%d ",mat[i][j]);
       printf("\n");
     }
     return 0;
   }
   ```

2. Faça um programa que leia uma matriz 10×10. O usuário terá de digitar o índice da linha e o índice da coluna e em seguida o valor da posição especificada. A matriz deve ser inicializada com 0 em todas as posições. A leitura será feita enquanto os índices forem não negativos. Após a leitura imprima a matriz na tela. **R:**

```
#include <stdio.h>
#define LIM 10
int main(){
  int mat[LIM][LIM];
  int i=0,j=0;
  for (i=0; i<LIM; i++)
    for (j=0; j<LIM; j++)
      mat[i][j]=0;
  printf("Digite o índice da coluna e o índice da linha: ");
  scanf("%d %d",&i,&j);
  while((i>=0) && (j>=0)){
    printf("Digite o valor da posição (%d,%d): ",i,j);
    scanf("%d",&mat[i][j]);

    printf("Digite o índice da coluna e o índice da linha: ");
    scanf("%d %d",&i,&j);
  }
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",mat[i][j]);
    printf("\n");
  }
  return 0;
}
```

3. Leia 2 matrizes 5×5, mostre elas na tela e então calcule e mostre a soma entre elas.

   R:

```
#include <stdio.h>
#define LIM 5
int main(){
  int mat1[LIM][LIM];
  int mat2[LIM][LIM];
  int i=0,j=0;

  for (i=0; i<LIM; i++) {
    for (j=0; j<LIM; j++) {
      printf("Digite o valor da posição (%d,%d) da matriz 1: ",i,j);
      scanf("%d",&mat1[i][j]);
      printf("Digite o valor da posição (%d,%d) da matriz 2: ",i,j);
      scanf("%d",&mat1[i][j]);
    }
  }

  printf("Matriz 1\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
```

```
      printf("%d ",mat1[i][j]);
    printf("\n");
  }

  printf("Matriz 2\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",mat2[i][j]);
    printf("\n");
  }

  printf("\n Matriz 1 + Matriz 2\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",mat1[i][j]+mat2[i][j]);
    printf("\n");
  }
  return 0;
}
```

4. Leia 2 matrizes 8×8, mostre elas na tela e então calcule e mostre a multiplicação entre elas.

   R:

```
#include <stdio.h>
#define LIM 8
int main(){
  int mat1[LIM][LIM];
  int mat2[LIM][LIM];
  int resp[LIM][LIM];
  int i,j,k;

  for (i=0; i<LIM; i++) {
    for (j=0; j<LIM; j++) {
      printf("Digite o valor da posição (%d,%d) da matriz 1: ",i,j);
      scanf("%d",&mat1[i][j]);
      printf("Digite o valor da posição (%d,%d) da matriz 2: ",i,j);
      scanf("%d",&mat1[i][j]);
    }
  }
  printf("Matriz 1\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",mat1[i][j]);
    printf("\n");
  }
  printf("Matriz 2\n");
  for (i=0; i<LIM; i++){
```

```
    for (j=0; j<LIM; j++)
      printf("%d ",mat2[i][j]);
    printf("\n");
  }
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      resp[i][j]=0;

  for (i=0; i<LIM; i++)
    for (j=0; j<LIM; j++)
      for (k=1; k<LIM; k++)
        resp[i][j]= resp[i][j]+mat1[i][k]*mat2[k][j]

  printf("\n Matriz 1 * Matriz 2\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",resp[i][j]);
    printf("\n");
  }
  return 0;
}
```

5. Leia uma matriz 4×4, e calcular sua transposta.

   R:

```
#include <stdio.h>
#define LIM 4
int main(){
  int mat[LIM][LIM];
  int i=0,j=0,k=0;

  for (i=0; i<LIM; i++)
    for (j=0; j<LIM; j++){
      printf("Digite o valor da posição (%d,%d): ",i,j);
      scanf("%d",&mat[i][j]);
    }
  printf("Transposta\n");
  for (i=0; i<LIM; i++){
    for (j=0; j<LIM; j++)
      printf("%d ",mat[j][i]);
    printf("\n");
  }
  return 0;
}
```

5 Referências

Estas aulas foram baseadas em:

- Notas de aula do **prof. Alexandre Falcão**
  (http://www.dcc.unicamp.br/ afalcao/mc102/notas-aula.pdf)
- Apostila do **prof. Flávio Keidi Miyazawa**
- Livro *Treinamento em Linguagem C, Curso Completo, Módulo 1.* de Victorine, Viviane e Mizrahi, Makron books.