Programação Estruturada Aula 9 - Matrizes e Strings II

Prof. Luis Carlos Retondaro

Técnico em Telecomunicações
2^o Ano

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Campus Petrópolis

2017

Sumário

Inicialização

- Matrizes bidimensionais podem ser inicializadas usando listas aninhadas de elementos entre chaves;

- **Exemplo:** Uma matriz bidimensional com três linhas e duas colunas pode ser inicializada da seguinte forma:

```
double Mat[3][2] =
       { {1.0,  0.0},      /* linha 0 */
         {-7.8, 1.3},      /* linha 1 */
         {6.5,  0.0}       /* linha 2 */
       };
```

Inicialização

- Quando a matriz é inicializada no momento da declaração, o tamanho da primeira dimensão pode ser omitido;

```
double Mat[][2] =
 { {1.0,  0.0},      /* linha 0 */
   {-7.8, 1.3},      /* linha 1 */
   {6.5,  0.0}       /* linha 2 */
 };
```

Matrizes de strings

- Para criar uma matriz de strings basta usar uma matriz bidimensional de caracteres;
- O primeiro índice indica a quantidade de strings, o segundo índice indica o tamanho de cada string;
- **Exemplo:** Uma matriz que pode armazernar até 30 strings com no máximo 79 caracteres cada uma:

```
char strMatriz[30][80];
```

- Para acessar uma string individual basta apenas especificar o índice esquerdo.

```
gets(strMatriz[2]);
```

Matrizes multidimensionais

- C permite matrizes com mais de duas dimensões, o limite exato, se existe depende do compilador;
- O formato da definição de uma matriz de dimensão k, onde o número de elementos em cada dimensão é $n_0, n_1, \ldots, n_{k-1}$, respectivamente, é:

```
tipo matMulti[n0][n1][n2]...[nk-1];
```

- Esse comando define uma matriz chamada *matMulti* com $n0 \times n1 \times n2 \times \cdots \times nk - 1$ elementos do tipo definido.

Matrizes multidimensionais

- Matrizes com mais de 2 dimensões não são usadas frequentemente devido a quantidade de memória de que elas necessitam;
- **Exemplo:** matriz com 4 dimensões como:

```
char matMulti[10][6][9][4];
```

- Tamanho em memória: $10 * 6 * 9 * 4$ ou 2.160 bytes.

Matrizes multidimensionais

- O armazenamento necessário de matrizes cresce exponencialmente com o número de dimensões;
- Grandes matrizes multidimensionais são geralmente alocadas dinamicamente;
- Durante a manipulação de matrizes multidimensionais, toma-se tempo do computador calculando seu índice, assim é mais lento acessar um elemento de uma matriz multidimensional que de uma com apenas uma dimensão.

Referenciando um elemento

- Após declarar uma matriz precisamos de um modo de referenciar seus elementos individualmente;
- De forma semelhante a matrizes unidimensionais e bidimensionais, a referência é feita através do número entre colchetes para cada dimensão;

```
//Curso, disciplina e alunos
  float notas[2][3][25];

  notas[1][2][10] = 8.5;
```

- Assim como matrizes unidimensionais e bidimensionais pode-se referenciar posições utilizando variáveis inteiras.

```
  int i = 1;
  int j = 2;
  int k = 20;
  notas[i][j][k] = 8.5;
```

Exemplo manipulação de matriz multidimensional

```
#include <stdio.h>
#include <stdlib.h>
#define CURSOS 1
#define TURMAS 1
#define ALUNOS 2
int main(){

  int i, j, k;
  float notas[CURSOS][TURMAS][ALUNOS], nota;

  //inicializando matriz
  for(i=0; i<CURSOS; i++)
    for(j=0; j<TURMAS; j++)
      for(k=0; k<ALUNOS; k++)
        notas[i][j][k] = 0;

  //notas da p1
  for(i=0; i<CURSOS; i++)
    for(j=0; j<TURMAS; j++)
      for(k=0; k<ALUNOS; k++){
        printf("Entre com a nota da P1 para o Curso %d turma %d aluno %d \n",i
    +1,j+1,k+1);
        scanf("%f",&nota);
        notas[i][j][k] += nota;
       }

  //continua...
```

Exemplo manipulação de matriz multidimensional

```
//continuacao...

//notas da p2
for(i=0; i<CURSOS; i++)
  for(j=0; j<TURMAS; j++)
    for(k=0; k<ALUNOS; k++){
      printf("Entre com a nota da P2 para o Curso %d turma %d aluno %d \n",i
   +1,j+1,k+1);
      scanf("%f",&nota);
      notas[i][j][k] += nota;
    }

//media
printf("\nCalculando a media...\n");
for(i=0; i<CURSOS; i++)
  for(j=0; j<TURMAS; j++)
    for(k=0; k<ALUNOS; k++)
      notas[i][j][k] /= 2;

 //continua..
```

Exemplo manipulação de matriz multidimensional

```
//continuacao...

 //verificando aprovação
  for(i=0; i<CURSOS; i++)
    for(j=0; j<TURMAS; j++)
      for(k=0; k<ALUNOS; k++)
        if(notas[i][j][k] < 7){
          printf("Entre com a nota da PF para o Curso %d turma %d aluno %d \n",i
    +1,j+1,k+1);
          scanf("%f",&nota);
          notas[i][j][k] += nota;
          notas[i][j][k] /=2;
        }

  //Imprimindo resultado final
  for(i=0; i<CURSOS; i++)
    for(j=0; j<TURMAS; j++)
      for(k=0; k<ALUNOS; k++)
        if(notas[i][j][k] >= 5)
          printf("Aluno %d do Curso %d turma %d foi aprovado com nota %.2f \n",k
    +1,i+1,j+1,notas[i][j][k]);
        else
          printf("Aluno %d do Curso %d turma %d foi reprovado com nota %.2f \n",
    k+1,i+1,j+1,notas[i][j][k]);
  return(0);
}
```

Resultado ...

```
// Resultado para
//#define CURSOS 1
//#define TURMAS 1
//#define ALUNOS 3

Entre com a nota da P1 para o Curso 1 turma 1 aluno 1
9.1
Entre com a nota da P1 para o Curso 1 turma 1 aluno 2
8.3
Entre com a nota da P2 para o Curso 1 turma 1 aluno 1
3.2
Entre com a nota da P2 para o Curso 1 turma 1 aluno 2
6.5

Calculando a media...
Entre com a nota da PF para o Curso 1 turma 1 aluno 2
6.0
Aluno 1 do Curso 1 turma 1 foi aprovado com nota 8.70
Aluno 2 do Curso 1 turma 1 foi reprovado com nota 4.67
```

Matrizes multidimensionais como argumento para funções

- Quando o parâmetro de uma função é uma matriz multidimensional (uma matriz com dimensão maior que um), todas as dimensões desta matriz, exceto a primeira, precisa ser explicitamente especificada no cabeçalho e protótipo da função;

```
tipo_retorno nome_funcao(tipo matMult[][n1][n2]...[nk-1], ...)
```

- A primeira dimensão pode ser incluída se o programador desejar, mas não é obrigatório.

Matrizes multidimensionais como argumento para funções

```
tipo_retorno nome_funcao(tipo matMult[][n1][n2]...[nk-1], ...)
```

- Quando uma função, que possui uma matriz como parâmetro, é chamada, na chamada da função somente o nome da matriz é passado como parâmetro;
- O tipo e a dimensão da matriz passada como parâmetro deve ser consistente com o tipo e a dimensão da matriz que é o parâmetro formal.

Exemplo matriz como parâmetro para função

```
// manipulando matriz
#include <stdio.h>

void exibe(int matriz[][2][3], int linhas){
  int i, j, k;

  for(i = 0; i< linhas; i++)
    for(j=0; j<2; j++)
      for(k=0; k<3; k++)
         printf("elemento[%d][%d][%d] = %d\n", i, j, k, matriz[i][j][k]);
}


int main()
{
     int pesos[2][2][3] = { { { 1, 2, 3 }, { 4, 5, 6 } },
     { { 7, 8, 9 }, { 10, 11, 12 } } };

     exibe(pesos,2);

     return(0);
 }
```

Exemplo matriz como parâmetro para função

Resultado

```
elemento[0][0][0] = 1
elemento[0][0][1] = 2
elemento[0][0][2] = 3
elemento[0][1][0] = 4
elemento[0][1][1] = 5
elemento[0][1][2] = 6
elemento[1][0][0] = 7
elemento[1][0][1] = 8
elemento[1][0][2] = 9
elemento[1][1][0] = 10
elemento[1][1][1] = 11
elemento[1][1][2] = 12
```

Inicialização de matriz não-dimensionada

- Imagine a Inicialização de matriz para construir uma tabela de mensagem de erro;

```
char e1[17] = "erro de leitura\n";
char e2[17] = "erro de escrita\n";
char e3[17] = "arquivo não pode ser aberto\n";
```

- É tedioso contar os caracteres manualmente em cada mensagem.

Inicialização de matriz não-dimensionada

- Pode-se deixar C calcular automaticamente a dimensão da matriz usando matriz não dimensionada;

- Se na inicialização da matriz a dimensão não é especificada o compilador cria uma matriz grande o bastante para conter todos os inicializadores presentes.

```
char e1[] = "erro de leitura\n";
char e2[] = "erro de escrita\n";
char e3[] = "arquivo não pode ser aberto\n";
```

Inicialização de matriz não-dimensionada

```
char e2[] = "erro de escrita\n";
printf("%s tem comprimento %d \n", e2, sizeof(e2));
```

- **Resultado:** erro de escrita tem comprimento 17;
- Inicialização de matrizes não-dimensionada não estão restritas a matrizes unidimensionais;
- Para matrizes multidimensionais deve-se especificar todas, exceto a primeira dimensão para que o compilador possa indexar a matriz de forma apropriada.

Inicialização de matriz não-dimensionada

- Pode-se construir tabelas de comprimento variável e o compilador aloca automaticamente armazenamento suficiente para guardá-los;

```
int sqrt[][2] = {
 1, 1,
 2, 4,
 3, 9,
 4, 16,
 5, 25,
 6, 36,
 7, 49,
 8, 64,
 9, 81,
 10, 100
};
```

- A vantagem dessa declaração sobre a vesrão que especifica o tamanho é que você pode aumentar ou diminuir a tabela sem alterar as dimensões da matriz;

Exemplo com o jogo da velha

- Matrizes multidimensionais são comumente utilizadas para simular jogos de tabuleiro;
- O canto esquerdo superior é a posição (1, 1) e o canto esquerdo inferior é a posição (3,3);

```
 //Um jogo da velha simples
#include <stdio.h>
#include <stdlib.h>

char matriz[3][3];

//inicializa matriz
void initMat(){
  int i, j;

  for(i=0; i<3; i++)
    for(j=0; j<3; j++)
      matriz[i][j] = ' ';
}
//continua..
```

Exemplo com o jogo da velha

```
//continuacao...
//Obtem a sua jogada
void getPlayerMove(){
  int x,y;

  printf("Digite as coordenadas para o X\n");
  scanf("%d%d",&x, &y);
  x--;
  y--;

  if(matriz[x][y] != ' '){
    printf("Posicao inválida tente novamente!\n");
    getPlayerMove();
  }
  else
    matriz[x][y] = 'X';
}

 //continua...
```

Exemplo com o jogo da velha

```
//continuacao...

//Obtem uma jogada do computador
void getComputerMove(){
  int i, j;

  for(i=0; i<3; i++){
    for(j=0; j<3; j++)
      if(matriz[i][j] == ' ')
        break;
      if(matriz[i][j] == ' ')
        break;
  }
  if(i*j == 9){
    printf("Empate\n");
    exit(0);
  }
  else
    matriz[i][j] = 'O';
}

 //continua...
```

Exemplo com o jogo da velha

```
//continuacao...

//Mostra matriz na tela
void dispMatriz(){
  int t;

  for(t=0; t<3; t++){
    printf("%c  | %c | %c", matriz[t][0], matriz[t][1], matriz[t][2]);
    if(t != 2)
      printf("\n---|---|---\n");
  }
  printf("\n");
}

 //continua...
```

Exemplo com o jogo da velha

```
//continuacao...

//Verifica se ha um vencedor
char check(){
  int i;

  //verifica as linhas
  for(i=0; i<3; i++)
  if((matriz[i][0] == matriz[i][1]) && (matriz[i][0] == matriz[i][2]))
    return(matriz[i][0]);

  //verifica as colunas
  for(i=0; i<3; i++)
    if((matriz[0][i] == matriz[1][i]) && (matriz[0][i] == matriz[2][i]))
      return(matriz[0][i]);

  //testa as diagonais
  if((matriz[0][0]  == matriz[1][1]) && (matriz[0][0]  == matriz[2][2]))
    return(matriz[0][0]);
  if((matriz[0][2]  == matriz[1][1]) && (matriz[0][2]  == matriz[2][0]))
    return((matriz[0][2]));

  return(' ');
}



 //continua...
```

Exemplo com o jogo da velha

```
//continuacao...

void main(){
  char done;

  printf("*** Jogo da velha ***\n");
  printf("Modalidade: Jogador X Computador\n");

  done = ' ';
  initMat();
  do{
    dispMatriz();
    getPlayerMove();
    done = check();
    if(done != ' ')
      break;
    getComputerMove();
    done = check();
  }while(done == ' ');
  if(done == 'X'){
    printf("Parabéns você ganhou!\n");
    dispMatriz();
  }
  else{
    printf("Computador ganhou!!!\n");
    dispMatriz();
  }
}
```

Referências

1. C Completo e Total, Herbert Schidt; Pearson Makron Books; 3a. Ed., 1997.

2. Linguagem C. DAMAS, Luis. 10a. Edição. LTC, 2014.