Algoritmos - TADS

Algoritmos – Matrizes em C

Professor: Victor Hugo L Lopes

Agenda

- Definições
- Declaração da Matriz
- Matrizes de outros tipos de elementos
- Verificação de limites
- Inicialização de matrizes
- Matrizes multidimensionais
- Matriz como argumentos de funções
- Strings

Programação em Linguagem C

- Definição:

Segundo Cormem, "uma matriz é um arranjo retangular de números".

Ex.: Matriz A, 2x3

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

Linhas, Colunas e Células

Programação em Linguagem C

- Definição:

Segundo MIZRAHI, “Matriz é uma coleção de variáveis do mesmo tipo, que compartilham um mesmo nome”.

Uma matriz é composta por elementos dispostos em linhas e colunas.

Um elemento específico em uma matriz é acessado por meio de um índice, associado ao nome da matriz.

Matrizes podem ter uma ou várias dimensões.

Programação em Linguagem C

- Definição:

Ainda Segundo Cormem, “um vetor é um arranjo unidimensional de números”.

Ex.: Matriz B

$$x = \begin{pmatrix} 2 \\ 3 \\ 5 \end{pmatrix}$$

Linhas

Programação em Linguagem C

- Definição:

Em resumo, uma matriz, seja ela unidimensional ou multidimensional nada mais é do que uma coleção de variáveis de um mesmo tipo que, normalmente, dizem respeito a um mesmo conjunto de dados.

Imagine o caso de um programa para cálculo de média de um aluno, que precisa de 4 variáveis para guardar as notas. Você deveria criar algo como:

```
int n1, n2, n3, n4;
```

Certo??

Programação em Linguagem C

...

```
int n1, n2, n3, n4;
float media= 0;
Scanf("%d,%d,%d,%d",&n1,&n2,&n3,&n4);
media = (n1+n2+n3+n4)/4;
Printf("%2.0f",media);
```

Isto resolve o problema proposto.

Mas imagine o crescimento deste problema, gerando a necessidade da construção deste programa para o calculo de dezenas de alunos?

Programação em Linguagem C

Diante desta problemática, precisamos de uma forma conveniente para referenciar tais coleções de dados similares.

A matriz é um recurso oferecido pela linguagem C para este propósito, sendo um conjunto de variáveis do mesmo tipo e referenciada por um único nome.

Cada variável é referenciada por meio de um número, o índice.

Sintaxe:

tipo nome[tamanho];

Programação em Linguagem C

```
Int main(){
        int notas[4];
        int i;
        float media = 0;
        scanf("%d%d%d%d",&notas[0], &notas[1], &notas[2],
            &notas[3]);

        for(i=0;i<4;i++){
                media += notas[i];
        }
        printf("media=%f",media/4);
}
```

Programação em Linguagem C

Declaração da matriz:

Em C, matrizes precisam ser declaradas como quaisquer outras variáveis, para que o compilador possa reservar espaços na memória suficientes para armazenar os dados.

Os elementos da matriz são guardados numa sequência contínua de memória, isto é, um seguido do outro, o que não ocorre quando criamos variáveis separadas.

Economizamos em nomes e na organização destas informações em memória.

Int notas[4];

Estamos criando uma matriz chamada notas, com 4 “gavetas”, que serão todas do tipo int.

Programação em Linguagem C

Declaração da matriz:

float valores[50]; //equivale a termos 50 variáveis do tipo float

char nome[60];//60 caracteres

Ou seja, o valor inteiro dentro dos colchetes indica a quantidade de informações da coleção podem ser guardadas.

Programação em Linguagem C

Referenciação dos elementos da matriz:

Cada um dos elementos da matriz é referenciado individualmente por meio de um número inteiro, entre colchetes, após o nome da matriz.

Quando referencia um elemento, esse número tem um significado diferente daquela da criação da matriz, o qual indica sua dimensão.

Ao referenciarmos um elemento de uma matriz, sua posição é especificada por este número.

Programação em Linguagem C

Referenciação dos elementos da matriz:

ex.:

```
float precos[20];

precos[2] = 12.99; // referenciando o elemento de índice 2
```

Programação em Linguagem C

Referenciação dos elementos da matriz:

Os elementos são sempre numerados por índices iniciados por zero.

ex.:

int notas[4];

gera os elementos:

notas[0]

notas[1]

notas[2]

notas[3]

Programação em Linguagem C

Referenciação dos elementos da matriz:

- **tendo em vista que para a referenciação da matriz precisamos utilizar um índice inteiro de tamanho máximo n-1, onde n é o tamanho da matriz, este índice pode ser uma constante, uma variável, uma expressão ou função int.**

ex.:

```
notas[i] = 50;
notas[0] = 90;
notas[i+1]=78;
dias[greg2july(10,10,2014)] = 10;
```

```
#include <stdio.h>
#include <stdlib.h>
#define TAMANHO 50
int main(){
    float notas[TAMANHO], media = 0.0;
    int i=0, j;
    do{
        printf("Digite a nota do aluno %d",i+1);
        scanf("%f",&notas[i]);
    }while(notas[i++] >= 0.0);
    i--; //retirando a última nota, que é negativa
    for (j=0;j<i;j++)
        media += notas[j];
    media /= i;
    printf("media=%.2f\n",media);
    system("PAUSE");
    return 0;
}
```

Programação em Linguagem C

Inicialização de matriz:

- **Você pode fornecer valores a cada elemento da matriz na mesma instrução de sua declaração:**

ex.:

```
int dmes[12] = { 31,28,31,30,31,30,31,31,30,31,30,31};
dmes[0] -> 31
dmes[1] -> 28
dmes[2] -> 31
...
```

Programação em Linguagem C

Inicialização de matriz:

- **A declaração de uma matriz inicializada pode suprimir a dimensão da matriz, restando apenas o par de colchetes vazio:**

ex.:

```
int dmes[] = { 31,28,31,30,31,30,31,31,30,31,30,31};
```

dmes[0] -> 31

dmes[1] -> 28

dmes[2] -> 31

...

Programação em Linguagem C

Exercitando:

Crie um novo projeto C com o nome testesMatrizes. Salve-o na pasta matrizes/testesMatrizes.

Na função principal, declare a matriz:

```
Int matriz1[10];
```

Agora vamos inserir valores em suas posições:

```
matriz1[0] = 1;
matriz1[1] = 2;
...
matriz1[9] = 10;
```

Programação em Linguagem C

Exercitando:

para listar automaticamente todos os valores guardados na matriz, utilizaremos um laço de repetição:

```
Int i;
for(i=0;i<10;i++){
      printf("O valor da posicao %d = %d",i,matriz1[i]);
}
```

Programação em Linguagem C

STRINGS:

Um uso bastante comum de matrizes unidimensionais é para a utilização de cadeias de caracteres.

Relembrando, não há um tipo de dados básico em C que suporte múltiplos caracteres:

Char letra='a';

Char nome = “Victor Hugo”; ??????????????

Programação em Linguagem C

STRINGS:

Diversas linguagem de programação modernas possuem um tipo de dados para uma cadeia de caracteres, como o nome “Victor Hugo”.

Em C, temo uma matriz de caracteres para guardarmos uma cadeia:

Char nome[11] = “Victor Hugo”;

Printf(“Meu nome é %s”, nome);

“Victor Hugo” é uma constante string.

Programação em Linguagem C

STRINGS:

A inclusão do cabeçalho string.h fornece um conjunto de funções para manipulação de matrizes de caracteres.

strcpy(s1, s2); //copia s2 em s1

strcat(s1, s2); //concatena s2 ao final de s1

strlen(s1); //retorna o tamanho de s1

Strch(s1, 'r'); //busca um caractere em s1

Strstr(s1, 'tor'); //busca uma string em s1

Strcmp(s1, s2); // compara as duas strings

Programação em Linguagem C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
Int main(){
    char s1[80], s2[80];
    gets(s1);
    gets(s2);
  printf("\nComprimentos s1 =  %d  s2  =  %d", strlen(s1),strlen(s2));

      if (strstr(s1,"tor")) printf("\nParte encontrada em s1");
}
```

Programação em Linguagem C

Matrizes de mais de uma dimensão/ bidimensional:

Até agora, vimos matrizes de uma única dimensão, ou simplesmente vetores.

Diz-se que uma matriz possui duas dimensões se ela possui, além da dimensão do vetor, uma outra dimensão.

Normalmente encaramos este tipo de matriz como tendo linhas e colunas:

int matriz1[2][2];

0,0	0,1
1,0	1,1

Onde matriz1[linhas][colunas]

Programação em Linguagem C

Matrizes de mais de uma dimensão/ bidimensional:

```
float notas[2][4];
notas[0][0] = 1.00;
notas[0][1] = 2.00;
notas[0][2] = 3.00;
notas[0][3] = 4.00;

notas[1][0] = 5.00;
notas[1][1] = 6.00;
notas[1][2] = 7.00;
notas[1][3] = 8.00;
float medias[2];
medias[0] = (notas[0][0] + notas[0][1] + notas[0][2] + notas[0][3])/4;
medias[1] = (notas[1][0] + notas[1][1] + notas[1][2] + notas[1][3])/4;
```

Programação em Linguagem C

Inicialização de Matrizes de mais de uma dimensão/ bidimensional:

Tendo em vista que a inicialização de uma matriz unidimensional se dá pelo preenchimento de valores nas suas posições no ato da sua criação, como em:

Float notas[4] = {9.0, 8.5, 5.6, 7.7};

Em uma matriz bidimensional, a ideia continua a mesma, mas agora considerando a quantidade de linhas:

Float notas[linhas][colunas] = { {primeira linha}, {segunda linha} };

Programação em Linguagem C

Matrizes de mais de uma dimensão/ bidimensional:

```
float notas[2][4];
notas[0][0] = 1.00;
notas[0][1] = 2.00;
notas[0][2] = 3.00;
notas[0][3] = 4.00;

notas[1][0] = 5.00;
notas[1][1] = 6.00;
notas[1][2] = 7.00;
notas[1][3] = 8.00;
```

```
float notas[2][4] = {{1.00,2.00,3.00,4.00}, {5.00,6.00,7.00,8.00}};
```

```
float medias[2];
medias[0] = (notas[0][0] + notas[0][1] + notas[0][2] + notas[0][3])/4;
medias[1] = (notas[1][0] + notas[1][1] + notas[1][2] + notas[1][3])/4;
```

Programação em Linguagem C

Varrendo uma Matrizes de mais de uma dimensão/ bidimensional:

Ao proceder com a varredura de uma matriz, iremos novamente utilizar laços de repetição, mas agora devemos notar que temos uma nova dimensão para nos preocupar:

```
Float notas[4] = {4, 5, 8, 9};
For (i=0;i<4;i++){
    Printf("\n%.2f",notas[i]);
}
float notas[2][4] = {{4,5,6,8},{3,5,8,9}};
for(i=0;i<2;i++)
   for(j=0;j<4;j++)
      printf("\n%.2f",notas[i][j]);
```

Programação em Linguagem C

Passando matrizes como parâmetros de funções

Tendo em vista que um parâmetro de uma função é uma variável, e que uma matriz é uma variável composta, é possível utilizarmos matrizes como parâmetros de funções:

```
Float media(float lista[ ], int c){
        return (lista[0] + lista[1] ... + lista[c]) / c;
}

int main(){
        float notas[4] = {4,5,6,7};
        printf("%f",media(notas, 4));
        ...
}
```

Programação em Linguagem C

Passando matrizes como parâmetros de funções

Matrizes são passadas para funções por referência!!!

Matrizes são consideradas um tipo de dados bastante grande.

Assim sendo, a linguagem C determina que é melhor manter somente uma única cópia dos dados na memória.

Para isso, ao invés de se passar os valores da matriz à uma nova variável na função, o parâmetro, é passado somente o endereço da memória.

Programação em Linguagem C

Exercitando

Crie um programa que receba valores de uma matriz unidimensional qualquer, envie para uma função, que vai imprimir a ordem inversa dos valores inseridos na matriz.

O programa abaixo utiliza uma matriz bidimensional, passando-a para uma função como parâmetro referencial. Forma 1:

```
1.  void media(float lista[2][4], int c, int d){
2.      int i,j;
3.      float m;
4.      for(j=0;j<c;j++){
5.          m=0.0;
6.          for(i=0;i<d;i++){
7.                  m+=lista[j][i];
8.          }
9.          printf("\n Aluno %d: %.2f",j+1,m/d);
10.     }
11. }
12. int main() {
13.     float notas[2][4] = {{4,5,6,7},{4,4,8,9}};
14.     media(notas,2,4);
15.         System(“PAUSE”):
•       Return 0;
1.  }
```

A função media recebe exatamente a matriz conforme ela foi definida, passando também c e d como sendo as dimensões linhas e colunas, respectivamente. A função calcula a média varrendo as colunas de cada linha, apresentando a média do aluno da linha, voltando para a próxima linha, até o fim da matriz.

O programa abaixo utiliza uma matriz bidimensional, passando-a para uma função como parâmetro referencial. Forma 2:

```
1.  float media2(float lista[], int c){
2.      float m = 0.0;
3.      int i;
4.      for(i=0;i<c;i++){
5.          m+=lista[i];
6.      }
7.      return m/=c;
8.  }
```

A função média2 recebe a matriz interna de cada linha enviada pela chamada, calcula todas as notas em suas colunas, devolvendo a média de cada linha.

```
1.  int main() {
2.      printf("\n\nAluno 1:%.2f",media2(notas[0],4));
3.      printf("\n\nAluno 2:%.2f",media2(notas[1],4));
4.          System(“PAUSE”):
5.      Return 0;
6.  }
```

Programação em Linguagem C

Matriz de muitas dimensões:

Estendendo a ideia de que uma matriz bidimensional é o conjunto de matrizes unidimensionais, onde em cada linha temos uma matriz de uma dimensão (vetor), uma matriz de muitas dimensões, ou multidimensional, é o conjunto de matrizes de imediata ordem inferior:

Uma matriz de 4 dimensões é o conjunto de matrizes de 3 dimensões.

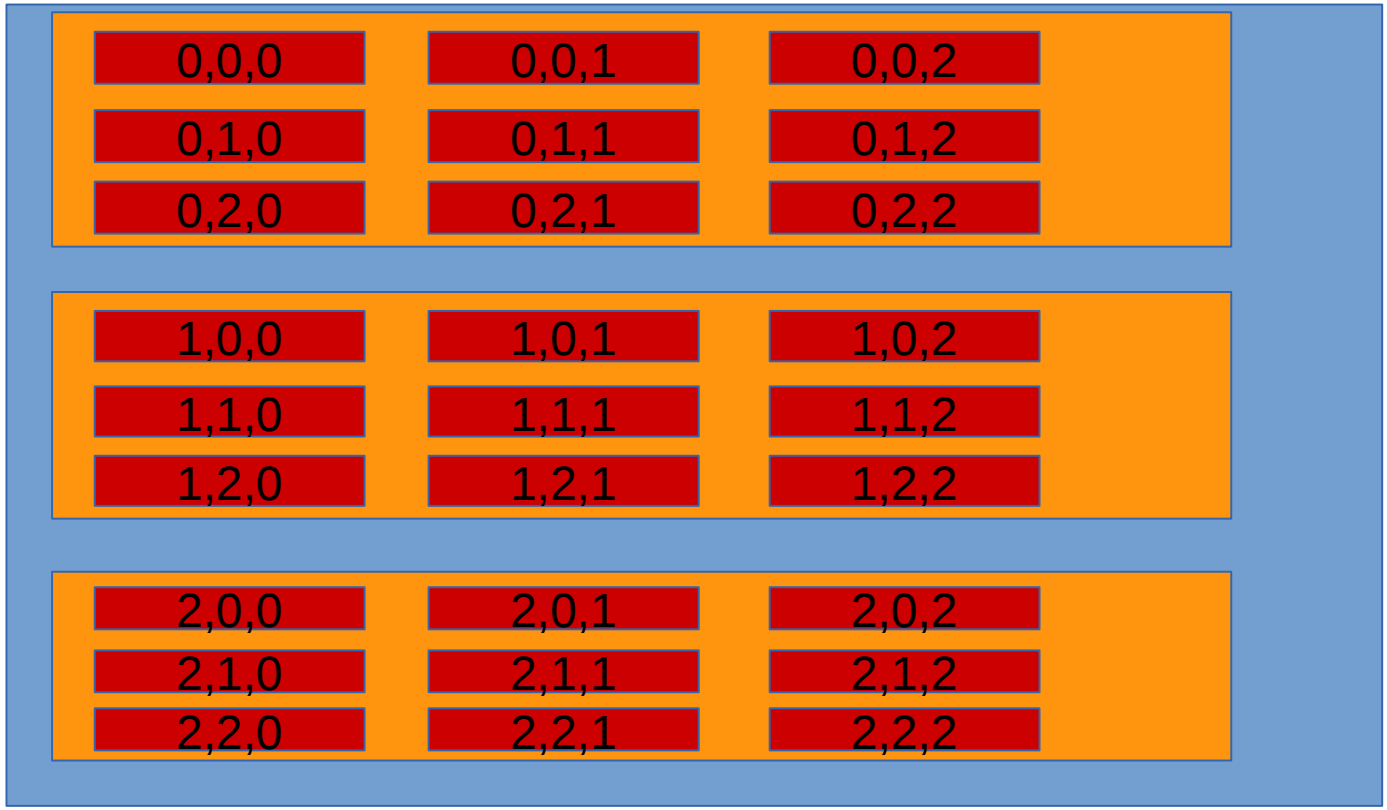
Uma matriz de 3 dimensões é o conjunto de matrizes de 2 dimensões.

Programação em Linguagem C

Matriz de muitas dimensões:

Uma matriz de 3 dimensões é o conjunto de matrizes de 2 dimensões.

Int codigos[3][3][3];



Programação em Linguagem C

Matriz de muitas dimensões:

Inicialização de matrizes tridimensionais:

```
int codigos[3][3][3]={
      {{1,2,3},{4,5,6},{7,8,9}},
      {{10,11,12},{13,14,15},{16,17,18}},
      {{19,20,21},{22,23,24},{25,26,27}}
  };
```

Programação em Linguagem C

Matriz de muitas dimensões:

Varrendo uma matriz tridimensional:

```
int codigos[3][3][3]={
      {{1,2,3},{4,5,6},{7,8,9}},
      {{10,11,12},{13,14,15},{16,17,18}},
      {{19,20,21},{22,23,24},{25,26,27}}
   };
   int i,j,k;
   for(i=0;i<3;i++){
      printf("Elemento %d\n_____________________________________",i);
      for(j=0;j<3;j++){
          printf("\n\tMatriz Bidimensional %d\n",j);
         for(k=0;k<3;k++){
            printf("\t%d”,codigos[i][j][k]);
         }
         printf("\n");
      }
   }
```

```
char produtos[3][50]={{"Alimenticios"},{"Beleza"},{"Hortifruti"}};
   char dadosProdutos[3][4][50]={
    {
      {"Arroz Carnavex"},
      {"Feijao do joao"},
      {"Cafe dubao"},
      {"Macarrao"}
    },
    {
      {"Oh Be"},
      {"Pasta de dente"},
      {"Bah tom"},
      {"Sabao de barra"}
    },
    {
      {"Alface"},
      {"Couve"},
      {"Batata"},
      {"Mandioca"}
    }
   };
```

Listando todos os produtos

```
printf("\n%s \t %s",produtos[0],dadosProdutos[0][0]);
    printf("\n%s \t %s",produtos[0],dadosProdutos[0][1]);
    printf("\n%s \t %s",produtos[0],dadosProdutos[0][2]);
    printf("\n%s \t %s",produtos[0],dadosProdutos[0][3]);
    printf("\n_____");
    printf("\n%s \t %s",produtos[1],dadosProdutos[1][0]);
    printf("\n%s \t %s",produtos[1],dadosProdutos[1][1]);
    printf("\n%s \t %s",produtos[1],dadosProdutos[1][2]);
    printf("\n%s \t %s",produtos[1],dadosProdutos[1][3]);
     printf("\n_____");
    printf("\n%s \t %s",produtos[2],dadosProdutos[2][0]);
    printf("\n%s \t %s",produtos[2],dadosProdutos[2][1]);
    printf("\n%s \t %s",produtos[2],dadosProdutos[2][2]);
    printf("\n%s \t %s",produtos[2],dadosProdutos[2][3]);;
```

A saída do programa

Alimenticios	Arroz Carnavex
Alimenticios	Feijao do joao
Alimenticios	Cafe dubao
Alimenticios	Macarrao

Beleza	Oh Be
Beleza	Pasta de dente
Beleza	Bah tom
Beleza	Sabao de barra

Hortifruti	Alface
Hortifruti	Couve
Hortifruti	Batata
Hortifruti	Mandioca