9002 — Aula 20
Algoritmos e Programação de Computadores

Instituto de Engenharia – UFMT

Segundo Semestre de 2014

02 de dezembro de 2014

Roteiro

Questões da prova

Como armazenar a nota de cada uma de 10 questões da prova de **um** aluno?

```
float questoes[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d: ", i);
    scanf("%f", &questao[i]);
}
```

E se fossem **100** alunos?

Questões da prova

Como armazenar a nota de cada uma de 10 questões da prova de **um** aluno?

```
float questoes[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d: ", i);
    scanf("%f", &questao[i]);
}
```

E se fossem **100** alunos?

Questões da prova

Como armazenar a nota de cada uma de 10 questões da prova de **um** aluno?

```
float questoes[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d: ", i);
    scanf("%f", &questao[i]);
}
```

E se fossem **100** alunos?

Questões da prova

Como armazenar a nota de cada uma
de 10 questões da prova de **um** aluno?

```
float questoes[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d: ", i);
    scanf("%f", &questao[i]);
}
```

E se fossem **100** alunos?

Cem alunos

Como armazenar a nota de cada uma de 10 questões das provas 100 alunos?

```
float questoes1[10], questoes2[10], ..., questoes100[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 1: ", i);
    scanf("%f", &questao1[i]);
}

...

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 100: ", i);
    scanf("%f", &questao100[i]);
}
```

Cem alunos

Como armazenar a nota de cada uma de 10 questões das provas 100 alunos?

```
float questoes1[10], questoes2[10], ..., questoes100[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 1: ", i);
    scanf("%f", &questao1[i]);
}

...

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 100: ", i);
    scanf("%f", &questao100[i]);
}
```

Cem alunos

Como armazenar a nota de cada uma de 10 questões das provas 100 alunos?

```
float questoes1[10], questoes2[10], ..., questoes100[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 1: ", i);
    scanf("%f", &questao1[i]);
}

...

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 100: ", i);
    scanf("%f", &questao100[i]);
}
```

Cem alunos

Como armazenar a nota de cada uma de 10 questões das provas 100 alunos?

```
float questoes1[10], questoes2[10], ..., questoes100[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 1: ", i);
    scanf("%f", &questao1[i]);
}

...

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 100: ", i);
    scanf("%f", &questao100[i]);
}
```

Cem alunos

Como armazenar a nota de cada uma
de 10 questões das provas 100 alunos?

```
float questoes1[10], questoes2[10], ..., questoes100[10];
int i;

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 1: ", i);
    scanf("%f", &questao1[i]);
}

...

for (i = 0; i < 10; i++) {
    printf("Nota da questao %d do aluno 100: ", i);
    scanf("%f", &questao100[i]);
}
```

Cem alunos – Melhorando

Revendo vetores

Já vimos esse problema:

- Problema: queríamos armazenar 100 variáveis como `float nota`

  Solução: usar um vetor de variáveis: `float notas[100]`

- Problema: agora queremos 100 vetores como `float questoes[10]`

  Solução: usar um vetor de **vetores**: `float questoes[100][10]`

Cem alunos – Melhorando

Revendo vetores

Já vimos esse problema:

- Problema: queríamos armazenar 100 variáveis como `float nota`

  Solução: usar um vetor de variáveis: `float notas`**[100]**

- Problema: agora queremos 100 vetores como `float questoes[10]`

  Solução: usar um vetor de **vetores**: `float questoes`**[100]**`[10]`

Cem alunos – Melhorando

Revendo vetores

Já vimos esse problema:

- Problema: queríamos armazenar 100 variáveis como `float nota`

  Solução: usar um vetor de variáveis: `float notas`**[100]**

- Problema: agora queremos 100 vetores como `float questoes[10]`

  Solução: usar um vetor de **vetores**: `float questoes`**[100]**`[10]`

Cem alunos – Melhorando

Revendo vetores

Já vimos esse problema:

- Problema: queríamos armazenar 100 variáveis como `float nota`

  Solução: usar um vetor de variáveis: `float notas`**[100]**

- Problema: agora queremos 100 vetores como `float questoes[10]`

  Solução: usar um vetor de **vetores**: `float questoes`**[100]**`[10]`

Vetores — Definição

Matriz

Matriz é uma coleção de vetores.

Características

- É um vetor como outro qualquer
- Cada vetor é acessado por por meio de um índice primário
- Cada variável simples é acessada por por meio de um índice secundário

Vetores — Definição

Matriz

Matriz é uma coleção de vetores.

Características

- É um vetor como outro qualquer
- Cada vetor é acessado por por meio de um índice primário
- Cada variável simples é acessada por por meio de um índice secundário

Vetores — Definição

Matriz

Matriz é uma coleção de vetores.

Características

- É um vetor como outro qualquer
- Cada vetor é acessado por por meio de um índice primário
- Cada variável simples é acessada por por meio de um índice secundário

Vetores — Definição

Matriz

Matriz é uma coleção de vetores.

Características

- É um vetor como outro qualquer
- Cada vetor é acessado por por meio de um índice primário
- Cada variável simples é acessada por por meio de um índice secundário

Declarando uma matriz

Matriz de tamanho $M \times N$

<tipo> identificador [<linhas>] [<colunas>] ;

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo <tipo>
- As linhas são numeradas de 0 a *linhas* $- 1$
- As colunas são numeradas de 0 a *colunas* $- 1$

Geralmente denotamos

- o número de *linhas* por m
- o número de *colunas* por n

Declarando uma matriz

Matriz de tamanho $M \times N$

<tipo> identificador [<linhas>] [<colunas>] ;

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo <tipo>
- As linhas são numeradas de 0 a *linhas* $-$ 1
- As colunas são numeradas de 0 a *colunas* $-$ 1

Geralmente denotamos

- o número de *linhas* por *m*
- o número de *colunas* por *n*

Declarando uma matriz

Matriz de tamanho $M \times N$

<tipo> identificador [<linhas>] [<colunas>] ;

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo <tipo>
- As linhas são numeradas de 0 a *linhas* $- 1$
- As colunas são numeradas de 0 a *colunas* $- 1$

Geralmente denotamos

- o número de *linhas* por m
- o número de *colunas* por n

Declarando uma matriz

Matriz de tamanho $M \times N$

<tipo> identificador [<linhas>] [<colunas>] ;

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo <tipo>
- As linhas são numeradas de 0 a *linhas* $- 1$
- As colunas são numeradas de 0 a *colunas* $- 1$

Geralmente denotamos

- o número de *linhas* por *m*
- o número de *colunas* por *n*

Declarando uma matriz

Matriz de tamanho $M \times N$

<tipo> identificador [<linhas>] [<colunas>] ;

- Uma matriz possui *linhas* $\times$ *colunas* variáveis do tipo <tipo>
- As linhas são numeradas de 0 a *linhas* $-$ 1
- As colunas são numeradas de 0 a *colunas* $-$ 1

Geralmente denotamos

- o número de *linhas* por *m*
- o número de *colunas* por *n*

Exemplo de declaração de matriz

```
int matriz [5][4];
```

	0	1	2	3
0				
1				
2				
3				
4				

Declarando uma vetor de múltiplas dimensões

Vetor com n dimensões

<tipo> identificador [< dim_1 >] [< dim_2 >] ... [< dim_n >]

- Possui $dim_1 \times dim_2 \times \cdots \times \dim_N$ variáveis do tipo <tipo>
- Cada dimensão é numerada de 0 a $dim_i - 1$

Declarando uma vetor de múltiplas dimensões

Vetor com n dimensões

<tipo> identificador [< dim_1 >] [< dim_2 >] ... [< dim_n >]

- Possui $dim_1 \times dim_2 \times \cdots \times \dim_N$ variáveis do tipo <tipo>
- Cada dimensão é numerada de 0 a $dim_i - 1$

Declarando uma vetor de múltiplas dimensões

Vetor com n dimensões

<tipo> identificador [< dim_1 >] [< dim_2 >] ... [< dim_n >]

- Possui $dim_1 \times dim_2 \times \cdots \times \dim_N$ variáveis do tipo <tipo>
- Cada dimensão é numerada de 0 a $dim_i - 1$

Acessando uma matriz

Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[1][10]` — Refere-se a variável na 2^a linha e na 11^a coluna da matriz.

O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

Acessando uma matriz

Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[1][10]` — Refere-se a variável na 2^a linha e na 11^a coluna da matriz.

O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

Acessando uma matriz

Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[1][10]` — Refere-se a variável na 2^a linha e na 11^a coluna da matriz.

O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

Acessando uma matriz

Em qualquer lugar onde você escreveria uma variável no seu programa, você pode usar um elemento de sua matriz, da seguinte forma:

```
nome_da_matriz [<linha>] [<coluna>]
```

Ex: `matriz[1][10]` — Refere-se a variável na 2^a linha e na 11^a coluna da matriz.

O compilador não verifica se você utilizou valores válidos para a linha e para a coluna.

Exemplos com Matrizes

Lendo uma matriz 4×5 do teclado:

Lendo matriz

```
int i, j;
int matriz[4][5]

for (i = 0; i < 4; i++) {
    for (j = 0; j < 5; j++) {
        printf ("Valor da linha %d, coluna %d: ", i, j);
        scanf ("%d", &matriz[i][j]);
    }
}
```

Exemplos com Matrizes

Imprimindo uma matriz 4×5 do teclado:

Escrevendo uma matriz

```
int i, j;
int matriz[4][5]

for (i = 0; i < 4; i++) {
    printf("Linha %d: ", i);

    for (j = 0; j < 5; j++) {
        printf("%d ", matriz[i][j];
    }

    printf("\n");
}
```

Inicialização de Matrizes

- 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 Matrizes

```
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} }
  };
  .
  .
  .
  .
}
```

Matrizes e funções

- Ao passar um **vetor simples** como parâmetro, não é necessário fornecer o seu tamanho na declaração da função.
- Quando o **vetor é multi-dimensional** a possibilidade de não informar o tamanho na declaração se restringe à primeira dimensão apenas.

```
void mostra_matriz(int mat[][10], int n_linhas) {
   ...
}
```

Matrizes e funções

- Pode-se criar uma função deixando de indicar a primeira dimensão:

```
void mostra_matriz(int mat[][10], int n_linhas) {
   ...
}
```

- Ou pode-se criar uma função indicando todas as dimensões:

```
void mostra_matriz(int mat[5][10], int n_linhas) {
   ...
}
```

- Mas não pode-se deixar de indicar outras dimensões (exceto a primeira):

```
void mostra_matriz(int mat[5][], int n_linhas) {
  //ESTE NÃO FUNCIONA
   ...
}
```

Vetores multi-dimensionais em funções

```
void mostra_matriz(int mat[][10], int n_linhas) {
  int i, j;

  for (i = 0; i < n_linhas; i++) {
    for (j = 0; j < 10; j++)
      printf("%2d ", mat[i][j]);
    printf("\n");
  }
}

int main() {
  int mat[][10] = { { 0,  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, 28, 29},
                    {30, 31, 32, 33, 34, 35, 36, 37, 38, 39},
                    {40, 41, 42, 43, 44, 45, 46, 47, 48, 49},
                    {50, 51, 52, 53, 54, 55, 56, 57, 58, 59},
                    {60, 61, 62, 63, 64, 65, 66, 67, 68, 69},
                    {70, 71, 72, 73, 74, 75, 76, 77, 78, 79}};

  mostra_matriz(mat, 8);
  return 0;
}
```

Vetores multi-dimensionais em funções

Lembre-se que vetores (multi-dimensionais ou não) são alterados quando passados como parâmetro em uma função

```
void   teste (int mat[2][2]) {
  int i, j;

  for (i = 0; i < 2; i++) {
    for (j = 0; j < 2; j++){
         mat[i][j] = -1;
    }
  }
}

int main() {
  int mat[2][2] = { { 0, 1},
                    { 2, 3} };

  teste(mat);
  //Neste ponto mat tem todas as suas posições com valor -1

  return 0;
}
```

Exercício

Caca-palavras

Escreva um programa que leia uma matriz de caracteres de dimensões 15×20 e depois leia uma palavra. O programa deverá contar o número de vezes que a palavra aparece.

```
OEAIAGBOOL
IIWAXHHLHN
PADUCAPNOC
ZBMOUIZSAS
OXEZOKOEUA
QCRMAAPAOH
DHOMEMTUFO
HOOAJCMVGM
NMFOANGMAE
JEVJVCCSNM
```

Exercício

Caca-palavras

Escreva um programa que leia uma matriz de caracteres de dimensões 15×20 e depois leia uma palavra. O programa deverá contar o número de vezes que a palavra aparece.

```
OEAIAGBOOL
IIWAXHHLHN
PADUCAPNOC
ZBMOUIZSAS
OXEZOKOEUA
QCRMAAPAOH
DHOMEMTUFO
HOOAJCMVGM
NMFOANGMAE
JEVJVCCSNM
```

Exercício

1. Escreva uma programa que lê duas matrizes, uma matriz A de dimensões 4×6 e outra matriz B de dimensões 6×10 e imprima uma terceira matriz $C = A \times B$.
2. Escreva uma programa que lê uma matriz de dimensões 5×5 e verifique se ela é simétrica.
3. Escreva uma programa que lê dois números m e n menores que 100 e depois lê uma matriz de dimensões $m \times n$. O programa deverá imprimir a matriz transposta.