Vetores e Matrizes

Alexandre Mello

Fatec Campinas

2018

Roteiro

Vetores

- Vetores são construções de linguagens de programação que servem para armazenar vários dados de um mesmo tipo de forma simplificada.

Vetores

- Suponha que desejamos guardar notas de alunos.
- Com o que sabemos, como armazenaríamos 3 notas?

```
float nota1, nota2, nota3;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);
printf("Nota do aluno 3: ");
scanf("%f", &nota3);
```

Vetores

- Com o que sabemos, como armazenaríamos 100 notas?

```
float nota1, nota2, nota3,..., nota100;

printf("Nota do aluno 1: ");
scanf("%f", &nota1);
printf("Nota do aluno 2: ");
scanf("%f", &nota2);
...
printf("Nota do aluno 100: ");
scanf("%f", &nota100);
```

- Criar 100 variáveis distintas não é uma solução elegante para este problema.

Definição de Vetores

- Um vetor em C é uma coleção de variáveis de um mesmo tipo que são referenciadas por um **identificador único**.
- Características de um vetor:
  - As variáveis ocupam posições contíguas na memória.
  - O acesso se dá por meio de um índice inteiro.
  - O vetor possui um tamanho pré-definido.
  - O acesso do vetor com um índice fora dos limites, pode causar comportamento anômalo do programa.

Declaração de um vetor

Para se declarar um vetor usamos a seguinte sintaxe:

- **tipo_variável** identificador[**tamanho do vetor**];

Exemplos

```
float notas[100]; //vetor "notas"  corresponde
                  //a 100 variáveis do tipo float

int primos[20]; //vetor "primos"  corresponde
                // a 20 variáveis do tipo int
```

Usando um vetor

- Após declarada uma variável do tipo vetor, pode-se acessar uma determinada posição do vetor utilizando-se um índice de valor inteiro.
- Sendo n o tamanho do vetor, os índices válidos para o vetor vão de 0 até $n - 1$.
  - A primeira posição de um vetor tem índice 0.
  - A última posição de um vetor tem índice $n - 1$.
- A sintaxe para acesso de uma determinada posição é:
  - identificador[**posição**];

O vetor em uma posição específica tem o mesmo comportamento que uma variável simples.

Exemplo

```
int nota[10];
int a;
nota[5] = 95;  //"nota[5]" corresponde a uma var. inteira
a = nota[5];
```

Usando um vetor

- Você deve usar valores inteiros como índice para acessar uma posição do vetor.
- O valor pode ser inclusive uma variável inteira.

Exemplo

```
int g, vet[10];
for(g=0; g<10; g++)
    vet[g]=5*g;
```

- Quais valores estarão armazenados em cada posição do vetor após a execução deste código?

Vetores e a Memória

- Suponha o código:

```
int d;
int vetor[5];
int f;
```

- Na memória temos:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-

Vetores e a Memória

- Ao executar o comando

```
vetor[3]=10;
```

temos:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
					10		

Vetores e a Memória

- O que ocorre se forem executados os comandos abaixo?

```
vetor[3]=10;
vetor[5]=5;
vetor[-1]=1;
```

Vetores e a Memória

- Ao executar

```
vetor[3]=10;
vetor[5]=5;
vetor[-1]=1;
```

teremos:

Nome	d	vetor					f
Índice	-	0	1	2	3	4	-
	1				10		5

- O seu programa estará errado pois você está alterando inadvertidamente valores de outras variáveis.
- Em alguns casos o seu programa será encerrado (**Segmentation Fault**).
- Em outros casos seu programa poderá continuar executando, mas ocorrerão erros difíceis de serem rastreados.

Questões importantes sobre vetores

- O tamanho do vetor é pré-definido (durante a execução do programa não pode ser alterado).
- O uso de índices fora dos limites podem causar comportamento anômalo do programa.

Como armazenar até 100 notas?

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);

for (i = 0; i < n; i++) {
  printf("Digite a nota do aluno %d: ", i);
  scanf("%f", &nota[i]);
}
```

- O programa acima está correto?

Como armazenar até 100 notas?

- Você deve testar se $n > 100$ para evitar erros!!

```
float nota[100];
int n, i;

printf("Número de alunos: ");
scanf("%d", &n);
if(n>100){
  n=100;
  printf("\nNumero máximo de alunos alterado para 100");
}
for (i = 0; i < n; i++) {
  printf("Digite a nota do aluno %d: ", i);
  scanf("%f", &nota[i]);
}
```

Exemplo: Produto Interno de dois vetores

- Ler dois vetores de dimensão 5 e computar o produto interno destes.
- Quais tipos de variáveis usar?

Exemplo: Produto Interno de dois vetores

- Abaixo temos o código para ler dois vetores de dimensão 5.

```
int main(){
  double vetor1[5], vetor2[5], resultado;
  int i;

  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 1:",i);
    scanf("%lf",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 2:",i);
    scanf("%lf",&vetor2[i]);
  }

  //calculando o produto interno
  ......
}
```

Exemplo: Produto Interno de dois vetores

- Abaixo temos a parte do código para computar o produto interno dos vetores.

```
int main(){
  double vetor1[5], vetor2[5], resultado;
  int i;

   ...

  //calculando o produto interno
  resultado = 0.0;
  for(i=0; i < 5; i++){
    resultado = resultado + ( vetor1[i]*vetor2[i] );
  }
  printf("\n\nO produto interno é: %lf\n",resultado);
}
```

Exemplo: Produto Interno de dois vetores

- Agora o código completo.

```
int main(){
  double vetor1[5], vetor2[5], resultado;
  int i;

  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 1:",i);
    scanf("%lf",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 2:",i);
    scanf("%lf",&vetor2[i]);
  }

  //calculando o produto interno
  resultado = 0.0;
  for(i=0; i < 5; i++){
    resultado = resultado + ( vetor1[i]*vetor2[i] );
  }
  printf("\n\nO produto interno é: %lf\n",resultado);
}
```

Exemplo: Elementos Iguais

- Ler dois vetores com 5 inteiros cada.
- Checar quais elementos do segundo vetor são iguais a algum elemento do primeiro vetor.
- Se não houver elementos em comum, o programa deve informar isso.

Exemplo: Elementos Iguais

- Abaixo está o código que faz a leitura de dois vetores.

```
int main(){
  int vetor1[5], vetor2[5];
  int i, j,  umEmComum;

  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 1:",i);
    scanf("%d",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 2:",i);
    scanf("%d",&vetor2[i]);
  }

  ...
}
```

Exemplo: Elementos Iguais

- Para cada elemento do **vetor1** testamos todos os outros elementos do **vetor2** para saber se são iguais.
- Usamos uma variável indicadora para decidir ao final dos laços encaixados, se os vetores possuem ou não um elemento em comum.

```
int main(){
  int vetor1[5], vetor2[5];
  int i, j,  umEmComum;

  ...

  umEmComum = 0; //Assumimos que não hajam elementos comuns
  for(i = 0; i < 5 ; i++)
    for(j = 0; j < 5; j++)
      if(vetor1[i] == vetor2[j]){
         umEmComum = 1;  //Descobrimos que há elemento comum
         printf("Posicao %d do vetor1 igual a posição %d do vetor2.\n",i,j);
      }
  if(!umEmComum)
    printf("Nenhum elemento em comum!\n");
}
```

Exemplo: Elementos Iguais

- Código completo abaixo.

```
int main(){
  int vetor1[5], vetor2[5];
  int i, j,  umEmComum;

  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 1:",i);
    scanf("%d",&vetor1[i]);
  }
  printf("\n\n");
  for(i=0; i<5; i++){
    printf("Entre com valor da posição %d para vetor 2:",i);
    scanf("%d",&vetor2[i]);
  }

  umEmComum = 0;
  for(i = 0; i < 5 ; i++)
    for(j = 0; j < 5; j++)
      if(vetor1[i] == vetor2[j]){
         umEmComum = 1;
         printf("Posicao %d do vetor1 igual a posição %d do vetor2.\n",i,j);
      }
  if(!umEmComum)
    printf("Nenhum elemento em comum!\n");
}
```

Matrizes e Vetores Multidimensionais

- Matrizes e Vetores Multidimensionais são generalizações de vetores simples vistos anteriormente.
- Suponha por exemplo que devemos armazenar as notas de cada aluno em cada laboratório de MC102.
- Podemos alocar 15 vetores (um para cada lab.) de tamanho 50 (tamanho da turma), onde cada vetor representa as notas de um laboratório específico.
- Matrizes e Vetores Multidimensionais permitem fazer a mesma coisa mas com todas as informações sendo acessadas por um nome em comum (ao invés de 15 nomes distintos).

Declaração de Matrizes

- A criação de uma matriz é feita com a seguinte sintaxe:

tipo `nome_da_matriz`[**linhas**][**colunas**];

onde **tipo** é o tipo de dados que a matriz armazenará, **linhas** (respectivamente **colunas**) é um inteiro que especifica o número de linhas (respectivamente colunas) que a matriz terá.

- A matriz criada terá (`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).

Exemplo de declaração de matriz

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

	0	1	2	3
0				
1				
2				
3				

Acessando dados de uma Matriz

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

`nome_da_matriz` [**ind_linha**][**ind_coluna**]

onde **ind_linha** (respectivamente **ind_coluna**) é um índice inteiro especificando a linha (respectivamente coluna) a ser acessada.

- No exemplo abaixo é atribuído para **aux** o valor armazenado na variável da 1^a linha e 11^a coluna da matriz:

```
int matriz[100][200];
int aux;
...
aux = matriz [0][10];
```

Acessando dados de uma Matriz

- Lembre-se que assim como vetores, a primeira posição em uma determinada dimensão começa no índice 0.
- O compilador não verifica se você utilizou valores válidos para a linha e para a coluna!
- Assim como vetores unidimensionais, comportamentos anômalos do programa podem ocorrer em caso de acesso à posições inválidas de uma matriz.

Declarando Vetores Multidimensionais

- Para se declarar um vetor com 3 ou mais dimensões usamos a seguinte sintaxe:

tipo `nome_vetor`$[d_1][d_2]\ldots[d_n]$;

onde d_i, para $i = 1, \ldots, n$, é um inteiro que especifica o tamanho do vetor na dimensão correspondente.

- O vetor criado possuirá $d_1 \times d_2 \times \cdots \times d_n$ variáveis do tipo **tipo**.
- Cada dimensão i é numerada de 0 a $d_i - 1$.

Declarando Vetores Multidimensionais

- Você pode criar por exemplo uma matriz para armazenar a quantidade de chuva em um dado dia, mês e ano, para cada um dos últimos 3000 anos:

```
double chuva[31][12][3000];

chuva[23][3][1979] = 6.0;
```

Exemplo

Criar aplicações 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: Lendo e Imprimindo uma Matriz

- Primeiramente vamos implementar o código para se fazer a leitura e a impressão de uma matriz:

```
#include <stdio.h>
#define MAX 10

int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);

  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }
  ...
}
```

- **MAX** é uma constante inteira definida previamente com valor 10 no nosso exemplo.
- Note porém que o tamanho efetivo da matriz é lido na variável **n**.

Exemplo: Lendo e Imprimindo uma Matriz

- Agora o código da impressão de uma matriz:

```
int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);
  printf("Lendo dados da matriz 1, linha por linha\n");
  ...

  printf("Imprimindo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat1[i][j]);
    }
    printf("\n"); //Após a impressão de uma linha da matriz pula linha
  }

}
```

- Para imprimir linha por linha, fixado uma linha i, imprimimos todas colunas j desta linha e ao final do laço em j, pulamos uma linha, para impressão de uma próxima linha.

Exemplo: Lendo e Imprimindo uma Matriz

- Código completo para ler e imprimir uma matriz:

```
#include <stdio.h>
#define MAX 10

int main(){
  double mat1[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes (max. 10): ");
  scanf("%d", &n);
  printf("Lendo dados da matriz, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }

  printf("Imprimindo dados da matriz, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat1[i][j]);
    }
    printf("\n");
  }
}
```

Exemplo: Soma de Matrizes

- Vamos implementar a funcionalidade de soma de matrizes quadradas.
- Primeiramente lemos as duas matrizes:

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, n;

  printf("Dimensão das matrizes: ");
  scanf("%d", &n);

  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }

  printf("Lendo dados da matriz 2, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat2[i][j]);
    }
  }
  ...
}
```

Exemplo: Soma de Matrizes

- Agora para cada posição (i, j) fazemos

$$\text{mat3}[i][j] = \text{mat1}[i][j] + \text{mat2}[i][j]$$

tal que o resultado da soma das matrizes estará em **mat3**.

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, n;
  ...
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      mat3[i][j] = mat1[i][j] + mat2[i][j];
    }
  }

  printf("Imprimindo dados da matriz 3, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat3[i][j]);
    }
    printf("\n");
  }
}
```

Exemplo: Multiplicação de Matrizes

- Vamos implementar a funcionalidade de multiplicação de matrizes quadradas.
- 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:

$$M_3[i,j] = \sum_{k=0}^{n-1} M_1[i,k] \cdot M_2[k,j]$$

Exemplo: Multiplicação de Matrizes

- O código da multiplicação está abaixo: para cada posição (i, j) de **mat3** devemos computar

$$\text{mat3}[i,j] = \sum_{k=0}^{MAX-1} \text{mat1}[i,k] \cdot \text{mat2}[k,j]$$

```
...
for(i=0; i<n; i++){
  for(j=0; j<n; j++){
    mat3[i][j] = 0;
    for(k=0; k<n; k++){
       mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
    }
  }
}
...
```

Exemplo: Multiplicação de Matrizes

```
int main(){
  double mat1[MAX][MAX], mat2[MAX][MAX], mat3[MAX][MAX];
  int i, j, k, n;
  printf("Dimensão das matrizes: ");
  scanf("%d", &n);
  printf("Lendo dados da matriz 1, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat1[i][j]);
    }
  }
  printf("Lendo dados da matriz 2, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      scanf("%lf", &mat2[i][j]);
    }
  }

  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      mat3[i][j] = 0;
      for(k=0; k<n; k++){
         mat3[i][j] = mat3[i][j] + (mat1[i][k] * mat2[k][j]);
      }
    }
  }
  printf("Imprimindo dados da matriz 3, linha por linha\n");
  for(i=0; i<n; i++){
    for(j=0; j<n; j++){
      printf("%.2lf \t", mat3[i][j]);
    }
    printf("\n");
  }
}
```

Exercício

- Escreva um programa que lê 10 números inteiros e os salva em um vetor. Em seguida o programa deve encontrar a posição do maior elemento do vetor e imprimir esta posição.

Exercício

- Escreva um programa que lê 10 números ponto flutuante e os salva em um vetor. Em seguida o programa deve calcular a média dos valores armazenados no vetor e imprimir este valor.

Exercício

- Escreva um programa que lê 10 números inteiros e os salva em um vetor. Em seguida o programa deve ler um outro número inteiro C. O programa deve então encontrar dois números de posições distintas do vetor cuja multiplicação seja C e imprimi-los. Caso não existam tais números, o programa deve informar isto.
- Exemplo: Se $vetor = (2, 4, 5, -10, 7)$ e $C = 35$ então o programa deve imprimir "5 e 7". Se $C = -1$ então o programa deve imprimir "Não existem tais números".

Exercícios

- Faça um programa para realizar operações com matrizes que tenha as seguintes funcionalidades:
  - Um menu para escolher a operação a ser realizada:
    1. Leitura de uma matriz_1.
    2. Leitura de uma matriz_2.
    3. Impressão da matriz_1 e matriz_2.
    4. Cálculo da soma de matriz_1 com matriz_2, e impressão do resultado.
    5. Cálculo da multiplicação de matriz_1 com matriz_2, e impressão do resultado.
    6. Cálculo da subtração de matriz_1 com matriz_2, e impressão do resultado.
    7. Impressão da transposta de matriz_1 e matriz_2.

Exercícios

Escreva um programa que leia todas as posições de uma matriz 10×10. O programa deve então exibir o número de posições não nulas na matriz.

Exercícios

- Escreva um programa que lê todos os elementos de uma matriz 4×4 e mostra a matriz e a sua transposta na tela.

Matriz

$$\begin{bmatrix} 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 1 & 0 & 2 \end{bmatrix}$$

Transposta

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Exercícios

- Escreva um programa leia uma matriz do teclado e então imprime os elementos com menor e maior frequência de ocorrência na matriz.

Informações Extras: Inicialização de um vetor

- Em algumas situações é necessário declarar e já atribuir um conjunto de valores constantes para um vetor.
- Em C, isto é feito atribuindo-se uma lista de elementos para o vetor na sua criação da seguinte forma:

tipo identificador[] = {elementos separados por vírgula} ;

- Exemplos:

```
double vet1[] = {2.3, 3.4, 4.5, 5.6};
int vet2[] = {5, 4, 3, 10, -1, 0};
```

- Note que automaticamente é criado um vetor com tamanho igual ao número de dados da inicialização.

Informações Extras: Inicialização de um vetor

```
#include <stdio.h>

int main(){
  double vet1[] = {2.3, 3.4, 4.5, 5.6};
  int vet2[] = {5, 4, 3, 10, -1, 0};
  int i;

  for(i=0; i<4; i++)
    printf("%lf\n", vet1[i]);

  for(i=0; i<6; i++)
    printf("%d\n", vet2[i]);

}
```

Informações Extras: 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} },
};
```

Informações Extras: 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} }
  };

  .
  .
  .
  .
}
```