Programação Procedimental

Vetores e Matrizes

Aula 03

Prof. Felipe A. Louza

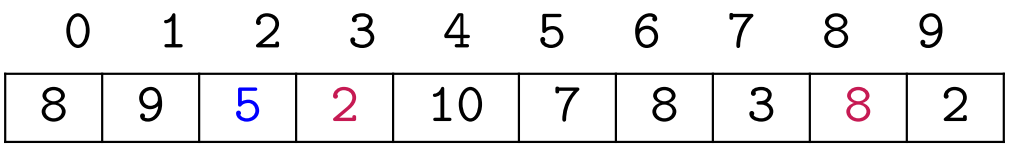
Roteiro

Roteiro

Vetores

Podemos armazenar coleções de itens utilizando vetores/arrays:

- Em linguagem C:
  - Todos os elementos são do mesmo tipo.
  - Ocupam posições contíguas na memória.
  - Vetores têm tamanho fixo, definido na declaração de variável.




```
int notas[10];
```

Vetores

Sintaxe:

```
1  tipo nome[tamanho];
```

- Acesso por meio de índice
- A primeira posição de um vetor tem índice zero (0)
- A última posição de um vetor tem índice tamanho - 1

0	1	2	3	4	5	6	7	8	9
8	9	5	2	10	7	8	3	8	2

```
int notas[10];
notas[0] = 8;
```

Vetores

Você deve usar apenas valores inteiros como **índice** para acessar uma posição do vetor.

```
1  int notas[10];
2  notas[0];
3  notas[1];
4  notas[2];
5  ...
```

- Esse valor pode ser uma variável inteira.

```
1  int notas[10];
2  int i;
3  for(i=0; i<10; i++){
4    notas[i] = 5*i;
5  }
```

Vetores

Uma boa prática é declarar o tamanho máximo de um vetor utilizando uma constante:

```c
#include <stdio.h>

#define TAM_MAX 10

int main(){

  int notas[TAM_MAX];
  int i;
  for(i=0; i<TAM_MAX; i++){
    notas[i] = 5*i;
  }

return 0;
}
```

Vetores

Exemplo:

```
1   int d;
2   int notas[5];
3   int f;
```

- Na memória, temos:

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

Vetores

Exemplo:

```
1   int d;
2   int notas[5];
3   int f;
```

- Ao executar `notas[3]=10;` temos:

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

Vetores

O que ocorre com os seguintes comandos:

```
1  notas[5]=5;
2  notas[-1]=1;
```

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

Segmentation Fault

- O seu programa estará errado pois você está alterando inadvertidamente valores de outras variáveis.

Vetores

Importante

- Em alguns casos, o programa é encerrado (**Segmentation Fault**).
- Em outros casos seu programa poderá continuar executando, mas ocorrerão erros difíceis de serem rastreados.

Roteiro

Leitura e Escrita com vetores

Leitura e Escrita:

- Cada posição `nota[i]` é um `int`:
  - Imprimir:

```
1 printf("%d", nota[i]);
```

  - Ler um número e guardar em `nota[i]`:

```
1 scanf("%d", &nota[i]);
```

0	1	2	3	4	5	6	7	8	9
0	0	0	7	0	0	0	0	0	0

`scanf("%d", &lista[3]);`

Exemplo

Vamos armazenar n ($<= 100$) notas?

```c
#include <stdio.h>

int main(){

  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]);
  }

return 0;
}
```

- O programa acima está correto? Sempre irá funcionar?

Exemplo

Precisamos testar se $n > 100$ para evitar erros!!

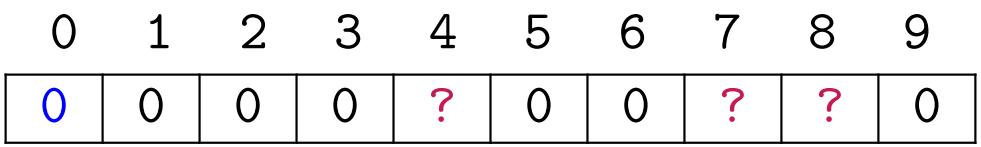
```
1   printf("Número de alunos: ");
2   scanf("%d", &n);
3
4   if(n > 100){
5     n = 100;
6     printf("\nNumero máximo de alunos alterado para 100");
7   }
8
9   for (i = 0; i < n; i++) {
10    printf("Digite a nota do aluno %d: ", i);
11    scanf("%f", &nota[i]);
12  }
```

Roteiro

Inicialização de um vetor

Importante:

- Ao declararmos um vetor, seus valores podem conter lixo:




```
int notas[10];
```

Inicialização de um vetor

Podemos declarar e já atribuir um conjunto de valores para um vetor.

- Sintaxe:

```
1  tipo nome[] = {v1, v2, ..., vn} ;
```

Exemplos:

```
1  double vet1[ 4 ] = {2.3, 3.4, 4.5, 5.6};
```

ou:

```
1  int vet2[] = {5, 4, 3, 10, -1, 0};
```

  - Note que, no primeiro exemplo, automaticamente é criado um vetor com tamanho igual ao número de dados da inicialização.

Inicialização de um vetor

```c
#include <stdio.h>

int main(){
  double vet1[4] = {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]);

return 0;
}
```

Roteiro

Soma de números

Exemplo 1:

- Realize a soma de todos os valores armazenados em um vetor:

```
int vetor[] = {1, 2, ..., n}
```

```c
#include <stdio.h>

int main(){

  const int MAX = 100;
  int n, i, vetor[MAX];

  printf("Digite o tamanho do vetor: ");
  scanf("%d", &n);

  if(n>MAX) n = MAX;

  for(i=0; i<n; i++){
    vetor[i]=i+1;
  }

  int soma=0;
  for(i=0; i<n; i++){
    soma += vetor[i];
  }

  printf("Soma é: %d\n", soma);

return 0;
}
```

Busca sequencial

Exemplo 2:

- Escreve um programa que busca um valor x em um vetor fornecido pelo usuário:

```
int vetor[100];
```

```c
#include <stdio.h>

int main(){

  const int MAX = 100;
  int i, vetor[MAX];

  for(i=0; i<MAX; i++)
    scanf("%d", &vetor[i]);

  int x;
  scanf("%d", &x);

  for(i=0; i<n; i++){
    if(vetor[i]==x)
      break;
  }

  if(i==n)
    printf("Valor não encontrado :(");
  else
    printf("Valor na posição %d\n", i);

return 0;
}
```

Menor valor

Exemplo 3:

- Escreve um programa que encontra o menor valor em um vetor fornecido pelo usuário:

```
int vetor[100];
```

```c
#include <stdio.h>

int main(){

  const int MAX = 100;
  int i, vetor[MAX];

  for(i=0; i<MAX; i++)
    scanf("%d", &vetor[i]);

  int min = vetor[0];

  for(i=1; i<n; i++){
    if(vetor[i] < min){
      min = vetor[i];
    }
  }

  printf("Menor valor encontrado: %d\n", min);

return 0;
}
```

Roteiro

Matrizes

Matrizes são vetores bi-dimensionais, definidos da seguinte forma:

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

	0	1	2	3
0				
1				
2				
3				
4				

Acessando valores em uma matriz

Acessando valores em uma matriz:

```
1  int matriz[5][4];
2
3  matriz[0][0] = 2;
4  matriz[4][3] = matriz[0][0]*3;
```

	0	1	2	3
0	2			
1				
2				
3				
4				6

Leitura e Escrita com matrizes

Lendo uma matriz do teclado:

```
1  int matriz[5][4];
2  for (i = 0; i < 5 ; i++){
3    for (j = 0; j < 4; j++) {
4      scanf ("%d", &matriz[i][j]);
5    }
6  }
```

	0	1	2	3
0				
1				
2				
3				
4				

Leitura e Escrita com matrizes

Imprimindo os valores de uma matriz:

```
1  int matriz[5][4];
2  for (i = 0; i < 5; i++){
3    for (j = 0; j < 4; j++) {
4      printf("%d\t", matriz[i][j]);
5    }
6    printf("\n");
7  }
```

	0	1	2	3
0				
1				
2				
3				
4				

Acessando dados de uma Matriz

Importante

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

```
1  double matriz[5][4];
2
3  matriz [5] [0] = 3;
```

  - Assim como vetores unidimensionais: comportamentos anômalos podem ocorrer.

Roteiro

Inicialização de matrizes

Inicialização de matrizes:

- Lembre-se de que posições não inicializadas podem conter lixo!

```
1   int matriz[5][4];
2
3   for (i = 0; i < 5; i++)
4     for (j = 0; j < 4; j++)
5       matriz[i][j] = 0;
```

	0	1	2	3
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

Inicialização de matrizes

Inicialização de matrizes:

- Podemos inicializar uma matriz durante a sua declaração:

```
1  int matriz[5][4] = {{1, 1, 1, 1},
2                      {2, 2, 2, 2},
3                      {3, 3, 3, 3},
4                      {4, 4, 4, 4},
5                      {5, 5, 5, 5}};
```

	0	1	2	3
0	1	1	1	1
1	2	2	2	2
2	3	3	3	3
3	4	4	4	4
4	5	5	5	5

Soma de linhas

Exemplo:

- Escreve um programa que soma as linhas de uma matriz:

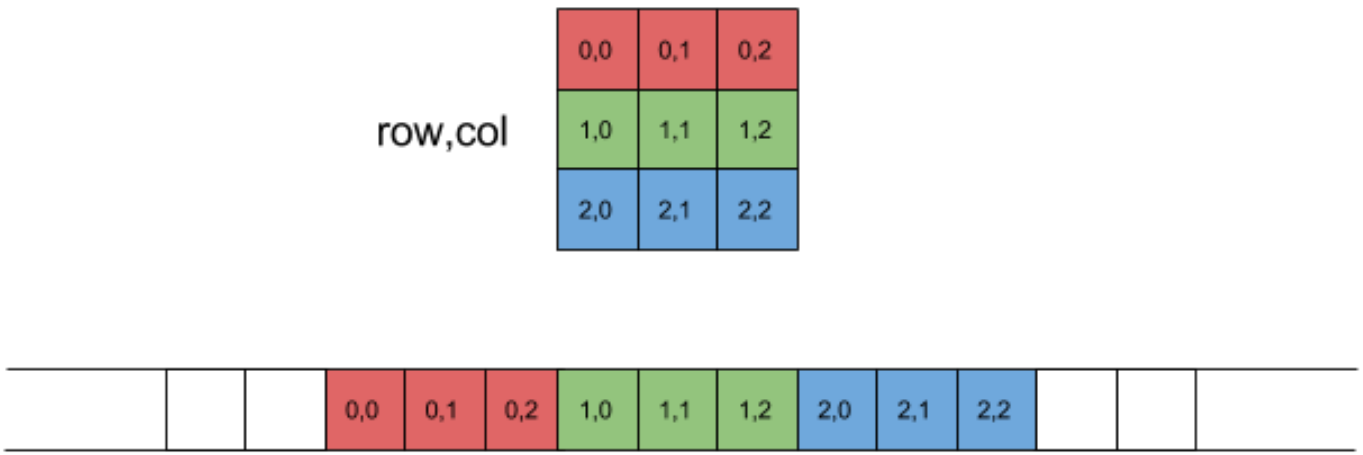
	0	1	2	3
0	1	1	1	1
1	2	2	2	2
2	3	3	3	3
3	4	4	4	4
4	5	5	5	5

Soma de linhas

```c
#include <stdio.h>

int main(){

  int matriz[5][4] = {{1, 1, 1, 1},
                      {2, 2, 2, 2},
                      {3, 3, 3, 3},
                      {4, 4, 4, 4},
                      {5, 5, 5, 5}};
  int soma = 0;

  for(i=0; i<5; i++){
    for(j=0; j<4; j++){
        soma += matriz[i][j];
    }
    printf("%d\n", soma);
    soma = 0;
  }

return 0;
}
```

Representação em memória

Representação em memória:

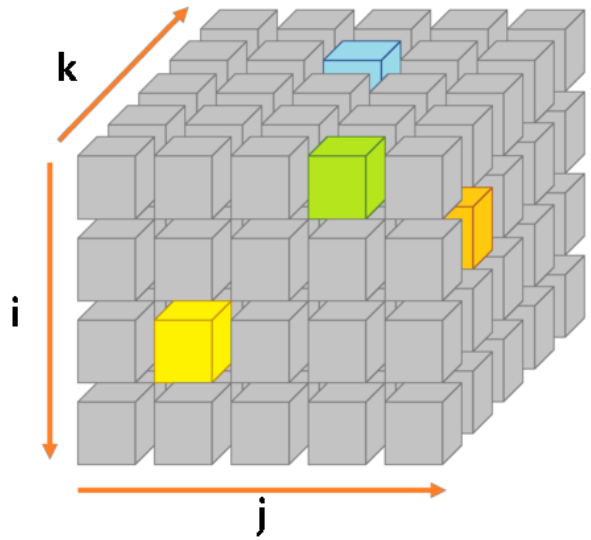


Roteiro

Vetores Multidimensionais

Vetores Multidimensionais são generalizações de vetores simples.

```
1   int nome[x][y][z]...[w];
```

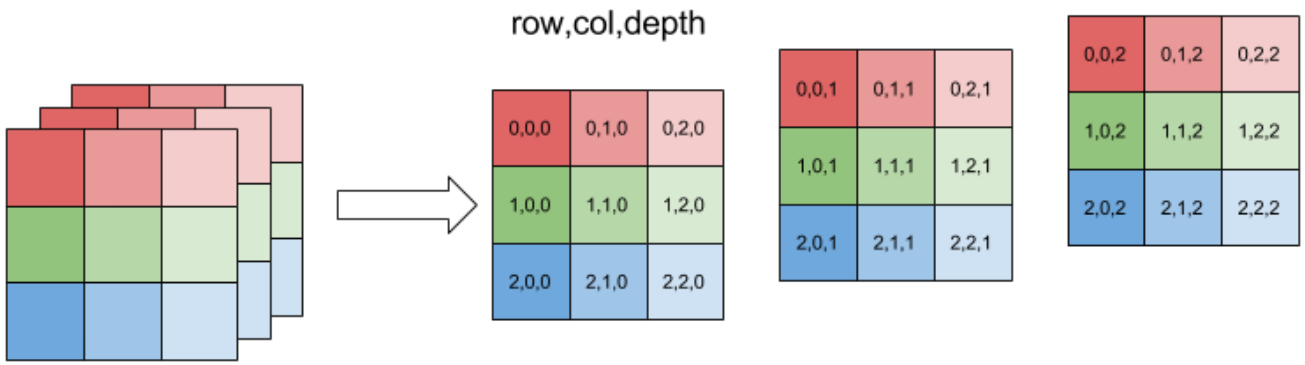



```
int cubo[4][5][5];
```

Vetores Multidimensionais

Acessando valores em uma matriz:

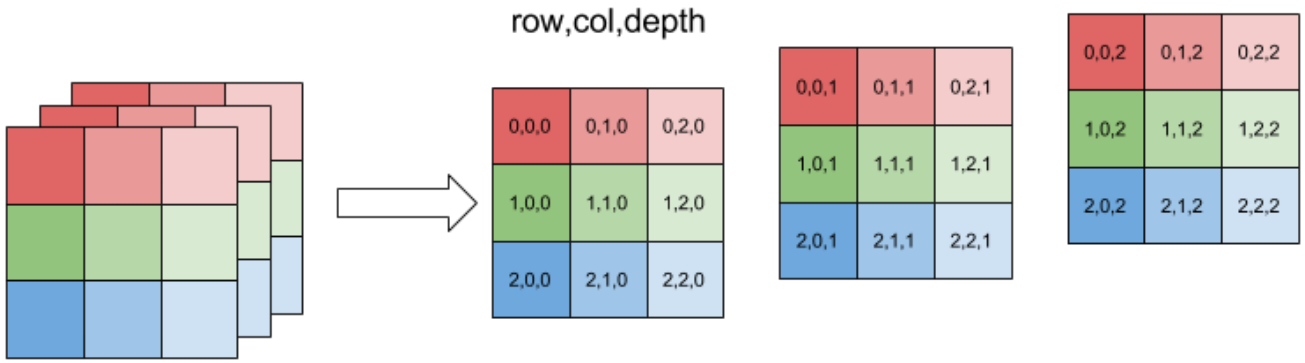
```
1   int cubo[3][3][3];
2
3   cubo[0][0][0] = 1;
4   cubo[2][2][0] = 9;
5   cubo[2][2][2] = 27;
```



Vetores Multidimensionais

Lendo do teclado:

```
1   int cubo[3][3][3], i, j, k;
2   for (k = 0; k < 3; k++)
3     for (i = 0; i < 3; i++)
4       for (j = 0; j < 3; j++)
5         scanf ("%d", &cubo[i][j][k]);
```



Vetores Multidimensionais

Imprimindo os valores:

```
1   int i, j, k;
2   for (k = 0; k < 3; k++){
3     for (i = 0; i < 3; i++){
4       for (j = 0; j < 3; j++){
5         printf("%d\t",cubo[i][j][k]);
6       }
7       printf("\n");
8     }
9     printf("##\n");
10  }
```

Vetores Multidimensionais

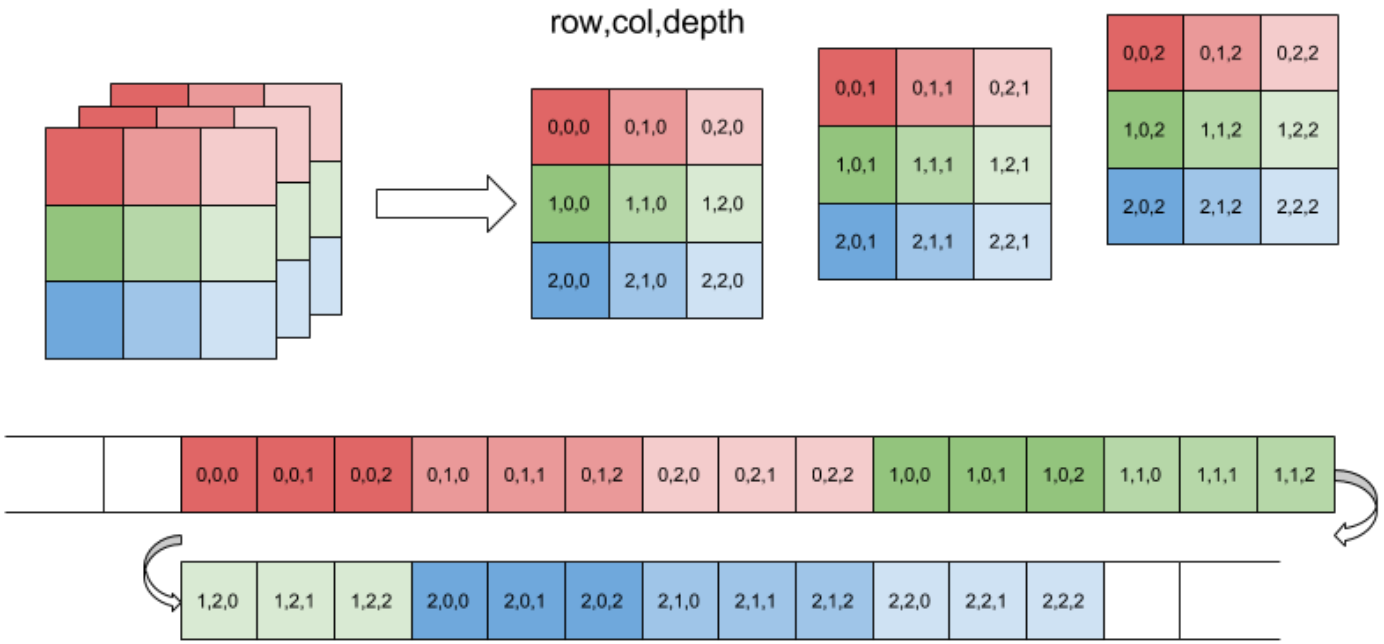
Exemplo:

- Você pode criar um vetor com 3 dimensões para armazenar a quantidade de chuva em um dado **ano**, **mês** e **dia**, para cada um dos últimos 3000 anos:

```
1   double chuva[3000][12][31];
2
3   chuva[1979][3][23] = 6.0;
```

Representação em memória

Representação em memória:



Fim

Dúvidas?

Roteiro

Referências

1. Feofiloff, Paulo. Algoritmos em linguagem C. Elsevier Brasil, 2009.
2. Materiais adaptados dos slides do Prof. Eduardo C. Xavier, da Universidade Estadual de Campinas.