Aula 29: Alocação Dinâmica (Parte 3)

Introdução a Programação

Túlio Toffolo & Puca Huachi
http://www.toffolo.com.br

BCC201 – 2019/2
Departamento de Computação – UFOP

Aulas anteriores

- Estruturas de memórias heterogêneas
- Arquivos de texto
- Alocação dinâmica (Partes 1 e 2)

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Exercícios da última aula

Exercício 1

Defina uma função que retorne a transposta de uma matriz `mat` com `lins`$\times$`cols` elementos. Utilize o protótipo a seguir:

```
1  int ** transposta(int **mat, int lins, int cols);
```

Dica: dada uma matriz A de tamanho $l \times c$, lembre-se que sua transposta T tem tamanho $c \times l$ e cada célula $T_{i,j} = A_{j,i}$.

Exercício 1 - Solução

Código da função:

```
1  int ** transposta(int **mat, int lins, int cols) {
2      int **t = malloc(cols * sizeof(int*));
3      for (int j = 0; j < cols; j++) {
4          t[j] = malloc(lins * sizeof(int))
5          for (int i = 0; i < lins; i++) {
6              t[j][i] = mat[i][j];
7          }
8      }
9
10     return t;
11 }
```

Exemplo de uso da função:

```c
int main() {
    // alocando matriz
    int n = 3, m = 2;
    int **matriz = malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++)
        matriz[i] = malloc(m * sizeof(int));

    ... // código omitido: lê os dados de matriz

    int **transp = transposta(matriz, n, m);

    ... // código omitido: imprime ou faz algo com transp

    // libera a memória alocada
    for (int i = 0; i < n; i++)
        free(matriz[i]);
    for (int j = 0; j < m; j++)
        free(transp[j]);
    free(matriz);
    free(transp);

    return 0;
}
```

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Matrizes dinâmicas

Matrizes dinâmicas são, na verdade, vetores de vetores!

- Criamos um vetor de ponteiros
- Criamos, para cada ponteiro, um vetor!

Matrizes dinâmicas

Exemplo:

```
1  // função que cria uma matriz de tamanho n x m
2  int ** criaMatriz(int n, int m)
3  {
4      int **matriz;
5      matriz = malloc(n * sizeof(int*));
6      for (int i = 0; i < n; i++) {
7          matriz[i] = malloc(m * sizeof(int));
8      }
9      return matriz;
10 }
```

Mas... como liberar a memória alocada para a matriz?

Matrizes dinâmicas

Como liberar a memória alocada por uma matriz?

- Lembre-se: um `free` para cada `mallloc`!

Exemplo:

```
1  void liberaMatriz(int **matriz, int n, int m)
2  {
3      for (int i = 0; i < n; i++)
4          free(matriz[i]);
5      free(matriz);
6  }
```

- Obs: note que a variável `m` não é necessária neste exemplo!

Erros comuns

O que está errado no código abaixo?

```
1  void liberaMatriz(int **matriz, int n, int m)
2   {
3       free(matriz);
4       for (int i = 0; i < n; i++)
5           free(matriz[i]);
6   }
```

Note que neste contexto **a ordem** faz toda a diferença!

- Após liberar a memória apontada por `matriz`, não podemos mais acessar `matriz[i]`.

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Referência de ponteiros

Em alguns casos, pode ser útil passar a referência de um ponteiro para uma função.

Exemplo:

- Alocar memória para dois vetores
- Armazenar o endereço em ponteiros passados por parâmetro

```
1 /* Função fictícia que aloca memória para dois vetores de tamanho n */
2 void alocaVetores(int **vetor1, int **vetor2, int n);
```

- Note o tipo `int**`: trata-se de um ponteiro para ponteiro
- Como implementar a função acima?

```c
/* Função fictícia que aloca memória para dois vetores de tamanho n */
void alocaVetores(int **vetor1, int **vetor2, int n) {
    *vetor1 = malloc(n * sizeof(int));
    *vetor2 = malloc(n * sizeof(int));
}
```

Observações:

- Note que o conteúdo de um ponteiro para ponteiro é um **ponteiro**!
- Note o exemplo abaixo, de como chamar a função acima:

```c
int main() {
    int n = 100;
    ...
    int *vetor1, *vetor2;
 → alocaVetores(&vetor1, &vetor2, n); // passagem por referência
    ...
    free(vetor1);
    free(vetor2);
    return 0;
}
```

Referência de ponteiros

Atenção: cuidado para não confundir com matrizes:

- A referência de um ponteiro `int*` é do tipo `int**`
- A referência de uma matriz `int**` é do tipo `int***`

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

```c
#include <stdlib.h>

void criaVetor(int **vetor, int n)
{
    *vetor = malloc(n * sizeof(int));
    for (int i = 0; i < n; i++)
        (*vetor)[i] = 0;

    // poderíamos fazer tb:
    // *vetor = calloc(n, sizeof(int));
}

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

    int *vetor;
    criaVetor(&vetor, n); // passagem por referência

    ... // faz algo com o vetor

    free(vetor);
    return 0;
}
```

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Exercícios

Exercício 1

Crie um procedimento `alocaMatriz` que recebe por parâmetro:

1. referência de um ponteiro para alocar e retornar uma matriz de inteiros;
2. número de linhas (`n`);
3. número de colunas (`m`).

A função deve alocar uma matriz `n`$\times$`m` dinamicamente e preencher todos os campos com valor zero.

Importante: implemente a função `main` para mostrar um exemplo de uso da função criada anteriormente.

Aula de hoje

1. Exercícios da última aula
2. Matrizes dinâmicas (breve revisão)
3. Passando ponteiros por referência
4. Exemplos
5. Exercícios
6. Próximas aulas

Próxima aula

- Arquivos binários

- Revisão para Trabalho Prático
- Revisão para Prova 03

- **Prova 03**
- Entrega do **Trabalho Prático**

Perguntas?