ESTRUTURA DE DADOS

PROF. NILTON

Aula de hoje

- Apresentação da disciplina;
- Metodologia de avaliação;
- Revisão de conceitos básicos;
- Introdução às estruturas de dados;
- Recursividade;

Apresentação da disciplina
Objetivos Gerais

Aprofundar conhecimentos sobre criação e manipulação de tipos abstratos de dados: listas, pilhas, filas e árvores

Apresentação da disciplina
Objetivos específicos

Criar, manipular e aplicar, por meio de uma linguagem de programação, os tipos abstratos de dados: listas, pilhas, filas e árvores.

Apresentação da disciplina
Ementa

Revisão dos conceitos básicos de tipos abstratos de dados. Pilhas, filas, alocação dinâmica, recursividade, listas encadeadas, tabelas de espalhamento e árvores. Aplicações das estruturas de dados em problemas computacionais.

Apresentação da disciplina

Bibliografia básica

ASCENCIO, A. F. G. Estruturas de dados. São Paulo: Pearson Brasil, 2011.

EDELWEISS, N; GALANTE, R. Estruturas de dados. V 18. Porto Alegre: Bookman, 2009.

PEREIRA, S. L. Estruturas de dados fundamentais – Conceitos e Aplicações. São Paulo: Érica, 2009.

Bibliografia complementar

KOFFMANN, E. B. Objetos, abstração, estrutura de dados e projeto. Rio de Janeiro: LTC, 2008.

Revisão

Tipos de dados (conjunto de valores que a variável pode assumir)

- int, float, double, char
  - Qualificadores short, long e unsigned

Estruturas simples

- Vetor unidimensional
- Vetor bidimensional ou multidimensional

Funções

Estrutura de Dados

Uma estrutura de dados é um modo particular de armazenamento e organização de dados em um computador de modo que possam ser usados eficientemente, facilitando sua busca e modificação.

Exemplos:

- Vetores
- registros
- listas encadeadas
- pilhas
- filas
- árvores
- grafos

Recursividade

Um módulo pode ser chamado ou ativado por qualquer outro módulo. Quando um módulo faz uma chamada ou ativa a si mesmo, dizemos que fez uma chamada recursiva.

Para entender a utilidade da recursão, podemos recorrer a um princípio importante da matemática discreta, q equações de recorrência. As funções de recorrência são funções expressas em função de si mesmas, ou seja, a resolução da função implica uma resolução anterior da função para outro valor. Para que essa solução tenha uma solução final, é necessário que haja uma base, um valor para o qual a resolução da função não seja recorrente/recursiva. Caso contrário a função original jamais se resolverá.

Recursividade

Exemplo com uma função simples (F) e uma função recorrente (R):

F(x) = 10 * x + 2

R(x) = 10 * R(x-1) + 2, para x > 1, e R(x) = 1 para x = 1

Recursividade

Resolução da função simples com os valores 1,2,3,4

F(x) = 10 * x + 2

F(1) = 10 * 1 + 2 = 12

F(2) = 10 * 2 + 2 = 22

F(3) = 10 * 3 + 2 = 32

F(4) = 10 * 4 + 2 = 42

Recursividade

R(x) = 10 * R(x-1) + 2

R(1) = 1 (definição ou base)

R(2) = 10 * R(1) + 2 = 10 * 1 + 2 = 12

R(3) = 10 * R(2) + 2 = 10 * 12 + 2 = 122

R(4) = 10 * R(3) + 2 = 10 * 122 + 2 = 1222

Recursividade

Como vimos, a resolução da função recorrente R para valores maiores que 1 só é possível após a resolução da função anterior. Para o exemplo anterior, utilizamos valores consecutivos, por isso pudemos utilizar para R(n) o resultado de R(n-1). Vejamos como seria a resolução de R(4) se antes não tivéssemos resolvido a equação anterior:

R(4) = 10 * R(3) + 2

= 10 * (10 * R(2) + 2) + 2

= 10 * (10 * (10 * R(1) + 2) + 2) + 2

= 10 * (10 * 1 + 2) + 2) + 2

= 10 * (10 * (12) + 2) + 2

= 10 * (122) + 2

= 1222

Fatorial

Por definição:

$n! = n * (n-1)!$, para $n > 0$;

$0! = 1$

De forma iterativa teríamos:

$5! = 5 * 4 * 3 * 2 * 1$

$5! = 120$

Fatorial

Podemos resolver esse problema da seguinte forma:

```
f = 1;
for (int i = n; i > 0; i--){
        f *= i;
}
```

Fatorial Recursivo

Por definição:

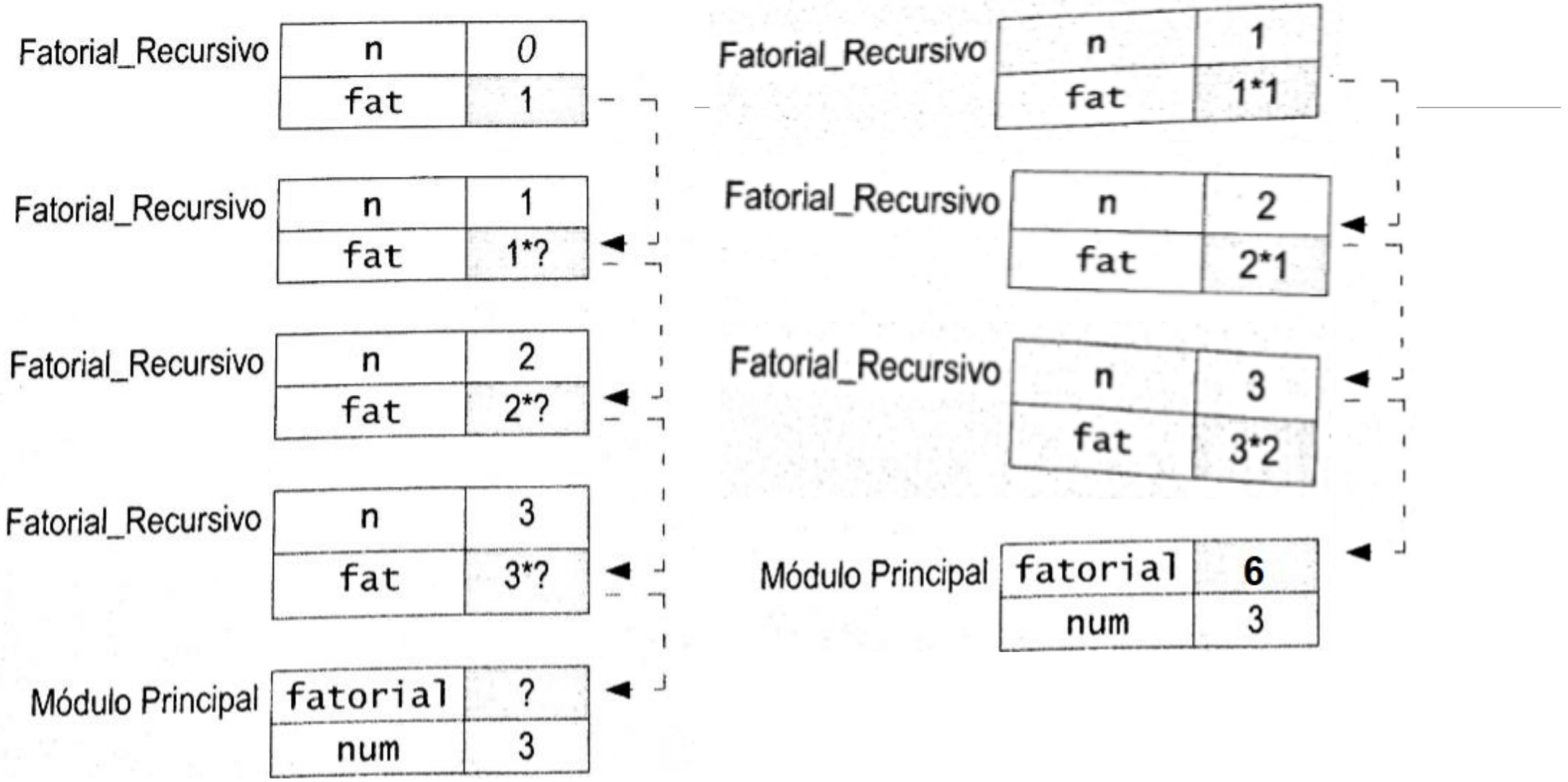
n! = n * (n-1)!, para n > 0;

0! = 1

Fatorial Recursivo

```
int Fatorial_Recursivo(int n) {
    int f;
    if ( n == 0) {
        f = 1;                                  // (base)
    }else{
         f = n * Fatorial_Recursivo(n - 1);     // função recorrente
     }
     return f;
}
```

Processo Fatorial Recursivo



Exercícios

1) Na matemática, a Sucessão de Fibonacci (também Sequência de Fibonacci), é uma sequência de números inteiros, começando normalmente por 0 e 1, na qual, cada termo subsequente corresponde à soma dos dois anteriores. Ex.: F(5) = 0, 1, 1, 2, 3, 5, 8

Dado o algoritmo a seguir:

fib(n) = fib(n-1) + fin(n-2) para n > 1

fib(0) = 0 e fib(1) = 1 (base)

Desenvolva um programa que calcule a serie Fibonacci de forma recursiva

Exercícios

2) Implemente um módulo recursivo que resolva a seguinte equação recorrente:

R(x) = 10 * R(x – 1) + 2 para x > 1, e R(x) = 1 para x = 1

3) Implemente um módulo recursivo que resolva a seguinte equação recorrente:

R(x) = 2 * R(x – 1) – 4 para x > 0, e R(0) = 2

4) Faça um algoritmo recursivo para o cálculo do somatório:

$$\sum_{i=1}^{n}(2 * r^2 + 2 * i + 8)$$

Obrigado!!!