Recursividade

Prof. Tiago Massoni
Prof. Fernando Buarque

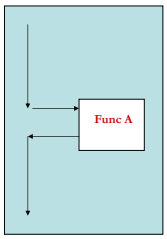
Engenharia da Computação

Poli - UPE

Funções regulares

- Elementos
  - declaração (inclui parâmetros formais),
  - chamada (inclui argumentos),
  - corpo e
  - retorno

Funções regulares

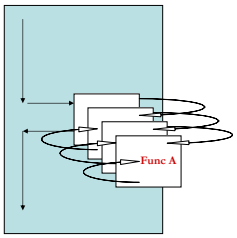


Funções recursivas

- Idênticas às regulares com uma diferença:
  - Chamada direta ou indireta à mesma função de dentro do seu corpo
  - Exemplos

a (parâmetros) {	a (parâmetros) {	b (parâmetros) {
...	...	...
a(argumentos);	b(argumentos);	a(argumentos);
} // direta	} // indireta	}

Funções recursivas



Funções definidas por elas mesmas

- Uma função pode ter uma definição indireta
- Exemplo
  - f(0)=0
  - $f(x)= 2f(x-1)+x^2$
- Como escreveríamos esta função em Java?

Recursão

- Definição especificada por um processo
  - Caso base
  - Chamada recursiva
- Exemplo: função fatorial

  0! = 1
  1! = 1
  2! = 2 * 1 = 2
  3! = 3 * 2 * 1 = 6
  ...
  n! = n * (n-1) * (n-2) ... * 1 se n>0
  ou ainda
  n! = n * (n-1)!

Recursão x Iteração: fatorial

```
prod =1;
for (x=n; x>0; x--) {
  prod *= x;
}
return (prod);
```

```
...
if(n==0)
   fact = 1;
else {
  fact = n*fat(x-1);
}
```


Caso base e terminação

- Quando definimos um método através dele, não estamos definindo uma instância particular do método através dele (chamadas diferentes)
- Caso base: define quando a função recursiva deve parar
  - Chamadas resolvidas sem recursão
  - Sempre deve haver progresso em direção ao caso base
  - Senão, círculo (estouro de pilha)

Exercício

- Escrever um programa em Java usando recursão que imprima a série de Fibonacci

  0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

  ou seja

  fib(0) = 0, fib(1) = 1, fib(2) = 1, ...

  i.e. fib(n) = n, se n==0 ou n==1

  fib(n) = fib(n-2) + fib(n-1), se n>= 2

Classe programa

```
public class Fibonacci {
   //metodos estaticos ...
   public static void main(String[] args){
   //codigo que chama os metodos estaticos
   ...
   }
}
```


Fibonacci iterativo

```
public static int fibonacciIterativo(int n){
  int soma = n;
  if (soma > 2) {
    int ant2=0;
    int ant1=1;
    soma = ant2+ant1;
    for (int i=3; i<=n; i++) {
      ant2=ant1;
      ant1=soma;
      soma=ant1+ant2;
    }
  }
  return soma;
}
```

Fibonacci recursivo

```
public static int fibonacciRecursivo(int n) {
  if (n <= 1)
    return n;
  else {
    return fibonacciRecursivo(n - 1) +
           fibonacciRecursivo(n - 2);
  }
}
```


Eficiência e simulação de recursividade

- A versão não-recursiva geralmente é mais eficiente (espaço e tempo)
  - tempo de entrar e sair do bloco recursivo
- Entretanto, algumas vezes a solução recursiva é o método mais lógico de resolver o problema
- Simulação de recursão pode ser feita utilizando armazenamento auxiliar (pilhas), porém é mais difícil e propensa a erros

Custo x benefício

- Eficiência de máquina x programador
- Frequência de processamento
- Complexidade do problema x solução
- Conclusão: uso depende do problema bem como do programador

Exercícios

- Método para imprimir um número inteiro não-negativo n, dado que só se pode imprimir um algarismo por vez
- Método para calcular o m.d.c de dois inteiros (dica: usar %)