Introdução à Recursão

Recursão

Prof. Lucas Boaventura
lucas.boaventura@unb.br

- **Recursão**
  - -Definição
  - -Em C
  - -Elementos da Recursão
  - -Algoritmo
  - -Funcionamento das Funções
  - -Exemplo: Divisão de Inteiros
  - -Vantagens
  - -Desvantagens
- **Dúvidas**
- **Exercícios**

Definição

• Recursão é uma instrução que tem o artifício de chamar a si mesma até que se chegue a algum resultado.

• A estrutura de recursão consiste em **caso base** e **hipótese de indução**.

• A operação fatorial é um exemplo de recursão em matemática:

n! = n x (n-1) x … x 1 = n.(n-1)!
0! = 1

Recursão

• Em C, recursões são funções que podem **chamar a si mesmas*,** podendo repetir instruções para que se chegue ao resultado esperado.

• Como na recursão comum, há um **caso base** e uma **hipótese de indução**.

• Se não for bem definido o caso base, um programa pode entrar em uma **recursão infinita**.

*Ou chamar uma função que depois chame primeira.

Recursão

.Exemplo:

```
int fat(int n) {
    int res;
    if(n > 0) {
        res = n * fat(n - 1);
    } else {
        res = 1;
    }
    return res;
}
```

A função fatorial calcula o fatorial de um dado número 'n'. Para isso, ele multiplica n pelo fatorial de n – 1, pois a hipótese de indução é que "n! = n.(n – 1)!". Se n for 0, o caso base, a função retornará 1.

Veja que:
0! = 1 => 1! = 1.0! = 1 => 2! = 2.1! = 2
3! = 3.2! = 6
4! = 4.3! = 24
5! = 5.4! = 120
6! = 6.5! = 720
...
E assim por diante.

Recursão

- A estrutura básica da recursão é:

- **Caso base:**
  - É a condição de parada da recursão.
  - Pode haver mais de um.
  - Ele não abre uma nova chamada da função.

- **Hipótese de Indução (ou Caso Recursivo):**
  - É o conjunto de instruções que a função exerce que incluem uma nova chamada da função.
  - Obrigatoriamente, há uma nova chamada recursiva.
  - É nisso que consiste a ideia de recursão.

Recursão

1. Receber um valor pra chamada.

2. Comparar entrada com o caso base.

3. Se for, o retorno será o valor definido para o caso.

4. Se não, aplicar a hipótese de indução, abrindo uma nova chamada para a função recursiva.

5. Esperar o retorno da nova chamada recursiva para definir o valor de retorno.

6. Retornar.

Recursão

• Funções recursivas, como quaisquer funções, tem suas próprias variáveis e parâmetros.

• A cada nova chamada recursiva, as variáveis são recriadas, e são independentes de suas versões em qualquer outra chamada da função.

• Portanto, cada chamada recursiva guarda seus próprios valores nas variáveis, isto se chama Estado.

• As únicas variáveis que podem ser acessadas em qualquer instância da recursão são as variáveis **static**.

Recursão - Divisão de Inteiros

- **Caso base: não é possível subtrair.**
  - Não é possível subtrair sem extrapolar o conjunto dos naturais.
  - Ou seja: dividendo < divisor.

- **Hipótese de Indução: subtrair os termos:**
  - dividendo >= divisor.
  - Devo adicionar 1 à contagem, e então devo somar isso à divisão do dividendo pelo divisor e o divisor.
  - (k >= a) => k/a = 1 + (k – a)/a.
  - Perceba que a divisão de inteiros k/a chama, recursivamente, a divisão de inteiros (k – a)/a.

Recursão - Divisão de Inteiros

- Exemplo:

```
int div(int dividendo, int divisor) {
   int quociente;

   if(dividendo < divisor) {
      quociente = 0;
   } else {
      dividendo -= divisor;
      quociente = 1 + div(dividendo,
divisor);
   }

   return quociente;
}
```

Recursão - Divisão de Inteiros

- Exemplo:

```
int quoc = div(12, 5);
```

dividendo = 12;
divisor = 5;
Como 12 >= 5, quociente será 1 + dividir(12 - 5 , 5)

dividendo = 7;
divisor = 5;
Como 7 >= 5, quociente será 1 + dividir(7 - 5 , 5)

dividendo = 2;
divisor = 5;
Como 2 < 5, quociente será 0

Quando a chamada recursiva acaba, voltamos a quem a chamou e temos
quociente = 1 + 0 = 1

Quando a chamada recursiva acaba, voltamos a quem a chamou e temos
quociente = 1 + 1 = 2

Recursão - Vantagens

• Há problemas que podem ser resolvidos de forma mais simples, recursivamente.

• Soluções recursivas são, em geral, mais elegantes que as soluções iterativas.

• Como C salva o Estado de execução de cada chamada recursiva, não há necessidade de se preocupar com salvar as variáveis antes de passar pra próxima chamada, como se precisaria ser feito em muitos casos iterativos.

Recursão - Desvantages

• Recursões são lentas, pois precisam recriar variáveis e parâmetros e reinicializar cada um.

• Em larga escala, podem consumir muita memória para guardar todos os estados de cada chamada recursiva.

• Definir uma condição inválida de parada pode criar uma recursão infinita. **Sempre se assegure de que a recursão atingirá o caso base.**

Dúvidas?

- lucas.boaventura@unb.br

Exercício 1

- Escreva um programa em C que lê do usuário uma frase que vá até o ponto final '.', e então imprima ela ao contrário.
- Obs.: O usuário pode entrar com um dígito por vez.

Ex:
Entrada:
Recursao e lindo.
Saída:
.odnil e oasruceR

Exercício 2

- **Faça um programa que receba um número N e calcule os 'N' termos da sequência de fibonacci e da sequência de golomb:**

Ex:
Entrada:
2
Saída:
0 1
1 2

Exercício 3

- **Defina um procedimento recursivo para, dado um número natural n na base decimal e uma outra base (entre 2 e 9) imprimi-lo nesta base .**

Ex:
Entrada (numero/base):
5 2
11 8
2013 5
Saída:
101
13
258