Introdução à Ciência da Computação II

Recursão

Prof. Ricardo J. G. B. Campello

Agradecimentos

- Parte dos slides a seguir são adaptações dos originais em Pascal gentilmente cedidos pelo Prof. Rudinei Goularte

Sumário

- Recursão
  - Definição e Características
  - Tipos de Recursão
  - Exemplos e Exercícios

3

Recursão

- Um procedimento ou função é dito **recursivo** se este invoca (realiza uma chamada) a si mesmo
- Muitas definições computacionais são (ou podem ser) recursivas. Exemplo:
  - Sistemas Diretório-Arquivo consistem de um **diretório** cujo conteúdo consiste de arquivos e possivelmente outros **diretórios**
- Conceito está intimamente ligado com os conceitos matemáticos de relação de recorrência e indução
  - $a_n = 2 \times a_{n-1}$, $a_0 = 1 \rightarrow a_n = 2^n$

4

Recursão

- Exemplo (cálculo do fatorial):
  - Dado um número inteiro x não-negativo, define-se o **fatorial** de x como:
  - fat(x) = 1 se x = 0 ou x = 1
  - fat(x) = x * (x–1) * (x–2) * ... * 2 * 1

fat(x–1)

5

Recursão

- Em C, qualquer função pode ser chamada recursivamente
  - Exemplo: cálculo do fatorial

```
long int FAT(long int x) {
  if (x == 0 || x == 1) return 1;
  else return x*FAT(x–1); }
```

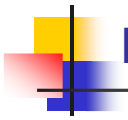
OU

```
long int FAT(long int x) {
   return (x <= 1 ? 1 : x*FAT(x–1) );
} /* Assume argumento x não negativo */
```

Projeto de Algoritmos por Recursão

- Exige **condição de parada** e **mudança de estado**
  - ou entra em ciclo infinito até estourar a pilha de recursão
  - Quais são elas no código recursivo do fatorial ???
- Vantagens
  - Muitos problemas são intrinsecamente recursivos
  - Código possivelmente mais claro (simples)
- Desvantagens
  - Velocidade (chamadas a módulos demandam tempo)
  - Uso adicional de memória (pilha de recursão)
  - Análise teórica possivelmente mais complexa

7



Projeto de Algoritmos por Recursão

- Exemplo / Exercício:
  - Desenvolva três funções distintas em C para calcular a relação de recorrência $a_n = 2 \times a_{n-1}$ ($a_0 = 1$), uma iterativa, uma recursiva e outra analítica (fórmula fechada $a_n = 2^n$)
  - As funções devem receber um inteiro não negativo n e retornar o termo a_n
  - Compare a solução recursiva com as demais sob o ponto de vista das suas possíveis desvantagens, mencionadas anteriormente

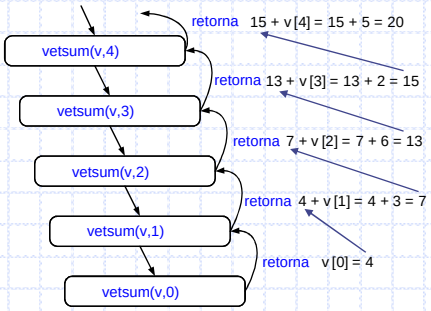
8

Traço de Recursão

```
float vetsum(float v[], int n){
  if (n == 0) return v[0];
  else return ( v[n] + vetsum(v, n–1) );
}
```

Traço de recursão:

Exemplo com v = [4 3 6 2 5]



9

Tipos de Recursão

- Alguns Tipos de Recursão:
  - **Recursão Linear**: Módulo com apenas uma chamada recursiva
  - **Recursão Binária**: Módulo com duas chamadas recursivas
  - **Recursão Múltipla**: Módulo com múltiplas chamadas recursivas
  - **Recursão Indireta**: Módulo A chama B que chama A
  - **Recursão de Cauda**:
    - Recursão linear cuja chamada recursiva é a *última operação* do módulo
    - Note que, tecnicamente, a recursão das funções **fat** e **vetsum** não são de cauda, mesmo estando na *última declaração* do módulo:
      - de fato, a última operação é uma multiplicação (fat) ou adição (vetsum)
      - no entanto, é usual chamar esse tipo de recursão como de cauda

10

Recursão

- Sempre existe versão iterativa de um programa recursivo
  - Conversão nem sempre é trivial
    - Normalmente simples para recursão do tipo cauda
- Exemplo 1: impressão de vetor

```
void printvet(float v[], int i, int f){
   if (i<=f){
        printf("%f ", v[i]);
        printvet(v,i+1,f); /* recursão de cauda */
   }
}
```

Recursivo

Iterativo

```
void printvet(float v[], int i, int f){
   int j;
   for (j=i; j<=f; j++)
      printf("%f ", v[j]);
}
```

Recursão

- Exemplo 2: inversão de vetor

```
void invvet(float v[], int i, int f){
   float aux;
   if (i<f){
        aux = v[i];
        v[i] = v[f];
        v[f] = aux;
        invvet(v,i+1,f-1);
   }
}
```

- Exercício:
  - A recursão acima é de cauda ?
  - Como é a versão iterativa dessa mesma função ?

Projeto de Algoritmos por Recursão

- Situações onde se recomenda usar recursão:
  - Quando o problema é intrinsecamente recursivo e sua solução como tal traz maior clareza

  E

  - Quando a alternativa recursiva não implica muita ineficiência em termos de tempo de execução e uso de memória (onde "muita" depende do problema e do usuário)

13

Projeto de Algoritmos por Recursão

- Situações onde se deve evitar recursão:
  - Quando existe um algoritmo iterativo igualmente claro e simples
    - P. ex. quando a recursão é de cauda
    - P. ex. quando uma única recursão já leva à condição de parada
  - Quando a recursão é ineficiente perante a uma versão iterativa
    - P. ex. quando muitos parâmetros são passados por valor
    - P. ex. quando pode haver sobrecarga da pilha de recursão
- Exemplo:
  - obtenção do n-ésimo valor da série de Fibonacci...

14

Exercícios Adicionais

- Qual o efeito final de inverter as linhas 3 e 4 do procedimento recursivo *printvet* visto anteriormente ?
- Escreva uma função recursiva que receba uma base real e um expoente inteiro e retorne o valor da base elevada ao expoente.
- É simples provar por indução matemática que a relação de recorrência $T_n = 2T_{n-1} + 1$ é, para $T_0 = 0$, equivalente a $T_n = 2^n - 1$. Escreva duas funções que recebam um inteiro n e retornem T_n, sendo uma recursiva e a outra não.
- Implemente uma função para calcular o n-ésimo elemento da seqüência de Fibonacci de forma recursiva e classifique a recursão segundo os tipos vistos em aula

15

Bibliografia

- Ziviani, N. *Projeto de Algoritmos*. Thomson, 2ª Edição, 2004.
- A. M. Tenembaum et al., *Data Structures Using C*, Prentice-Hall, 1990
- Schildt, H. "C Completo e Total", 3a. Edição, Pearson, 1997.

16