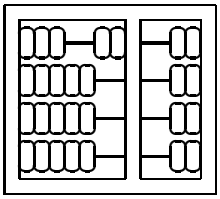

MC102 – ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES
Instituto de Computação — UNICAMP
1º Semestre de 2005
Turmas Q e T

Profa.: Amanda Meincke Melo
amanda.melo@ic.unicamp.br

Aula 24 - Introdução à Recursão

1 Objetivos

- Compreender recursão e conceitos subjacentes.
- Construir funções recursivas na linguagem C.

2 Motivação

- Um algoritmo computacional recursivo pode ser estruturado, analisado e provado, analogamente à demonstração de um teorema por indução (projeto por indução é um ferramental valioso para o desenvolvimento de algoritmos computacionais).
- Em geral, a solução recursiva de um problema necessita de menos variáveis e de menos instruções de controle que sua versão iterativa não recursiva correspondente.
- Programas recursivos são mais simples de escrever, analisar e entender.

3 Recursividade

Um objeto é dito recursivo (ou apresenta recursividade) se for definido em termos de si próprio.

A definição de objetos recursivos é comum na matemática. Um exemplo bastante conhecido é o da soma dos números inteiros no intervalo [m, n], onde m, n $\in \mathbf{Z}$ e m $\leq$ n.

3.1 $\sum_{k=m}^{n}$

3.1.1 Definição 1

$$\sum_{k=m}^{n} = \begin{cases} m & \text{se } n = m \text{ e} \\ \sum_{k=m}^{n-1} + n & \text{se } n > m. \end{cases}$$

3.1.2 Definição 2

$$\sum_{k=m}^{n} = \begin{cases} m & \text{se } n = m \text{ e} \\ m + \sum_{k=m+1}^{n} & \text{se } n > m. \end{cases}$$

3.1.3 Definição 3

$$\sum_{k=m}^{n} = \begin{cases} m & \text{se } n = m \text{ e} \\ \sum_{k=m}^{\lfloor (m+n)/2 \rfloor} + \sum_{k=\lfloor (m+n)/2 \rfloor + 1}^{n} & \text{se } n > m. \end{cases}$$

Também é possível desenvolver algoritmos computacionais recursivos, ou seja, que são definidos em termos deles mesmos.

3.2 Recursão

É o processo de resolução de um problema, reduzindo-o em um ou mais subproblemas com as seguintes características:

- São idênticos aos problemas originais;
- São mais simples de resolver.

Uma vez realizada a primeira subdivisão, a mesma técnica de decomposição é usada para dividir cada subproblema. Eventualmente, os subproblemas tornam-se tão simples que é possível resolvê-los sem efetuar novas subdivisões. A solução completa do problema original é obtida através da "montagem"das soluções componentes. Este processo de resolução de problemas está diretamente ligado ao conceito de indução matemática.

A **indução fraca** toma como hipótese a idéia de que a solução de um problema de tamanho t pode ser obtida a partir da solução de subproblemas de tamanho $t - 1$. Essa é a idéia subjacente às definições 1 e 2 para $\sum_{k=m}^{n}$. Já a **indução forte** adota a hipótese que a solução de um problema de tamanho t pode ser obtida a partir da solução de subproblemas de tamanho t', para todo $t' < t$. Essa é a idéia subjacente à definição 3 para $\sum_{k=m}^{n}$, abordagem para resolução de problemas também conhecida como **divisão e conquista**.

Os objetos de programação que iremos utilizar para desenvolver algoritmos recursivos serão os procedimentos e as funções.

4 Funções Recursivas

Uma rotina (função ou procedimento) é recursiva se ela chama a si mesma. Uma rotina recursiva pode ser de dois tipos:

- **Rotina recursiva direta**: aquela que, em sua descrição, faz uma chamada a si mesma;
- **Rotina recursiva indireta**: aquela que não faz uma chamada a si mesma em sua descrição, mas faz chamada a outras rotinas que podem vir a chamá-la.

4.1 Exemplos de Funções Recursivas Diretas

4.1.1 Cálculo do $\sum_{k=m}^{n}$

- Codificação da primeira definição

```
int Soma (int m, int n) {
   if (n == m)
      return (m);
   else
      return (Soma (m, n-1) + n);
}
```

- Codificação da segunda definição

```
int Soma (int m, int n) {
   if (n == m)
      return (m);
   else
      return (m + Soma (m+1, n));
}
```

- Codificação da terceira definição

```
int Soma (int m, int n) {
   if (n == m)
      return (m);
   else
      return (Soma (m, (m+n)/2) + Soma ((m+n)/2 + 1, n));
}
```

Um problema que pode ser resolvido de maneira recursiva também pode ser resolvido de maneira iterativa. A seguir é a apresentada a versão iterativa para a função $\sum_{k=m}^{n}$:

```
int Soma (int m, int n) {
   int soma = 0, i;

   for (i = m; i <= n; i++)
      soma += i;

   return (soma);
}
```

4.1.2 Cálculo do número de dígitos de um número inteiro

A função, a seguir, calcula o número de dígitos de um número inteiro de forma iterativa:

```
int numero_digitos (int n) {
   int cont = 1;

   while (abs (n) > 9) {
      n = n/10;
      cont ++;
   }

  return (cont);
}
```

Sua versão recursiva é mais simples e de fácil entendimento. Note que o número de dígitos representa o número de chamadas à função:

```
int numero_digitos (int n) {
   if (abs (n) <= 9)
      return (1);
   else
     return (1 + numero_digitos (n/10));
}
```

4.1.3 Cálculo de $n!$

A função fatorial pode ser definida da seguinte maneira:

$$n! = \begin{cases} 1 & \text{se } n = 0 \text{ e} \\ n.(n-1)! & \text{se } n > 0. \end{cases}$$

Sua codificação recursiva na linguagem C:

```
double Fatorial (double n) {
   if (n == 0)
      return (1);
   else
      return (n * Fatorial (n-1));
}
```

Sua codificação iterativa na linguagem C:

```
double Fatorial (double n) {
   int fatorial = 1, i;

   if (n == 0)
      fatorial = 1;
   else {
      for (i = n; i > 1; i--)
         fatorial *= i;
   }
   return (fatorial);
}
```

4.1.4 Cálculo de x^n

A função x^n, pode ser definida como:

$$x^n = \begin{cases} 1/x^n & \text{se } n < 0, \\ 1 & \text{se } n = 0 \text{ e} \\ x \times x^{n-1} & \text{se } n > 0. \end{cases}$$

Sua codificação recursiva na linguagem C:

```
float Potencia (float x, int n) {
   if (n == 0)
      return (1);
   else {
      if (n < 0)
         return (1/Potencia (x, abs(n)));
      else
         return (x * Potencia (x, n-1));
   }
}
```

Sua codificação iterativa na linguagem C:

```
float Potencia (float x, int n) {
   float potencia = 1, i;

   if (n < 0) {
      for (i = 1; i <= -n; i++)
         potencia *= 1/x;
   }
   else {
      if (n == 0)
         potencia = 1;
      else
         for (i = 1; i <= n; i++)
            potencia *= x;
   }

   return (potencia);
}
```

5 Referências

Rubio, J. A. "Recursividad en un Primer Curso de Computación", Departamento de Ciencias de la Computación, Universidad de Chile, Santiago, Chile.

Notas de Aula do **prof. Alexandre Falcão**, [online]: http://www.dcc.unicamp.br/ afalcao/mc102/

Notas de Aula do **prof. Flávio Keidi Miyazawa**.

Notas de Aula da **profa. M. Cecília F. Rubira**