Recursão

Prof. Angelo Loula
UEFS

Recursão

- Alguns problemas são definidos usando sua própria definição
- Tipicamente, cada passo do problema contém um passo menor do mesmo problema
- Exemplo:
  - Fatorial de 5 é igual a 5 vezes o fatorial de 4

Recursão

- se o problema pode ser resolvido facilmente,
  - resolva o problema;
- se o problema é grande,
  - elabore uma solução menor do problema,
  - relacione com o problema maior,
  - resolva o problema menor,
  - volte ao problema inicial.

Recursão

- Todo problema recursivo deve ter
  - Um caso base que pode ser resolvido diretamente;
  - Um passo recursivo que envolve um problema menor.
- Caso base interrompe a recursão
- Passos recursivos deve caminhar para caso base

$$f(x) = \begin{cases} 1, se\ x = 0 \\ x.f(x-1), se\ x > 0 \end{cases} \qquad f(x) = \begin{cases} 1, se\ x = 0 \\ x.f(x+1), se\ x > 0 \end{cases}$$

Recursão

- Exemplo:

Escreva uma função que recebe como parâmetro um inteiro positivo n e retorna a soma de todos os números inteiros entre 0 e n.

Recursão

- Solução Iterativa:

```
int Somatorio (int n) {
   int i, resp = 0;
   for (i = 0; i < n; i++){
     resp = resp + i;
   }
   return resp;
}
```


Recursão

- Solução Recursiva:

```
int Somatorio (int n) {
  if (n == 1){
     return n;
  } else {
     return (n + Somatorio (n - 1));
  }
}
```


Recursão

- Chamada inicial para a função recursiva :

```
int main () {
  int n;
  printf ("Qual o valor de n? ");
  scanf ("%d", &n);
  printf ("O somatorio de 0 a %d :", n);
  printf ("%d", Somatorio (n));
  return 0;
}
```

Recursão

- Chamadas recursivas :

```
O usuário digita 4

Somatorio(4) → 4 + Somatorio(3)
     Somatorio(3) → 3 + Somatorio(2)
          Somatorio(2) → 2 + Somatorio(1)
               Somatorio(1) → 1
          Somatorio(2) → 2 + 1 = 3
     Somatorio(3) → 3 + 3 = 6
Somatorio(4) → 4 + 6 = 10

O resultado é 10!
```


Recursão

- E o fatorial?

```
int Fatorial(int N){
  if( N == 0 ){
    return 1;
  } else {
    return(N * Fatorial(N -1));
  }
}
```


Recursão

- Série de Fibonacci

$$fibonacci(x) = \begin{cases} 0, se\ x = 0 \\ 1, se\ x = 1 \\ fibonacci(x-1) + fibonacci(x-2), se\ x > 1 \end{cases}$$

Recursão

- Fibonacci

```
int Fibonacci (int n) {
 if (n == 0 || n == 1){
  return n;
 } else {
  return (Fibonacci(n-1) + Fibonacci(n-2));
 }
}
```

Recursão

- É melhor não usar recursão:
  - *Quando as chamadas recursivas acontecem apenas no início ou apenas no fim da função*
    - Motivo: cada chamada recursiva reserva memória para as variáveis locais e parâmetros
    - Exemplos: as funções somatório e fatorial
  - *Quando o número de tarefas/cálculos repetidos é muito grande, a execução de um programa pode ficar inviável*
    - Motivo: em geral, cada chamada recursiva é independente da outra; se duas chamadas recursivas realizam os mesmos cálculos, esses cálculos serão repetidos
    - Exemplo: a função recursiva para a série de Fibonacci