Recursividade

Recursividade

Em uma definição recursiva um item é definido em termos

de si mesmo, ou seja, o item que está sendo definido aparece como parte da definição;

Em todas as funções recursivas existe:

- – Caso base (um ou mais) cujo resultado é imediatamente conhecido.
- – Passo recursivo em que se tenta resolver um sub-problema do problema inicial.

Recursividade

- As funções recursivas são em sua maioria soluções mais elegantes e simples, se comparadas a funções tradicionais ou iterativas, já que executam tarefas repetitivas sem utilizar nenhuma estrutura de repetição, como *for* ou *while*. Porém essa elegância e simplicidade têm um preço que requer muita atenção em sua implementação.

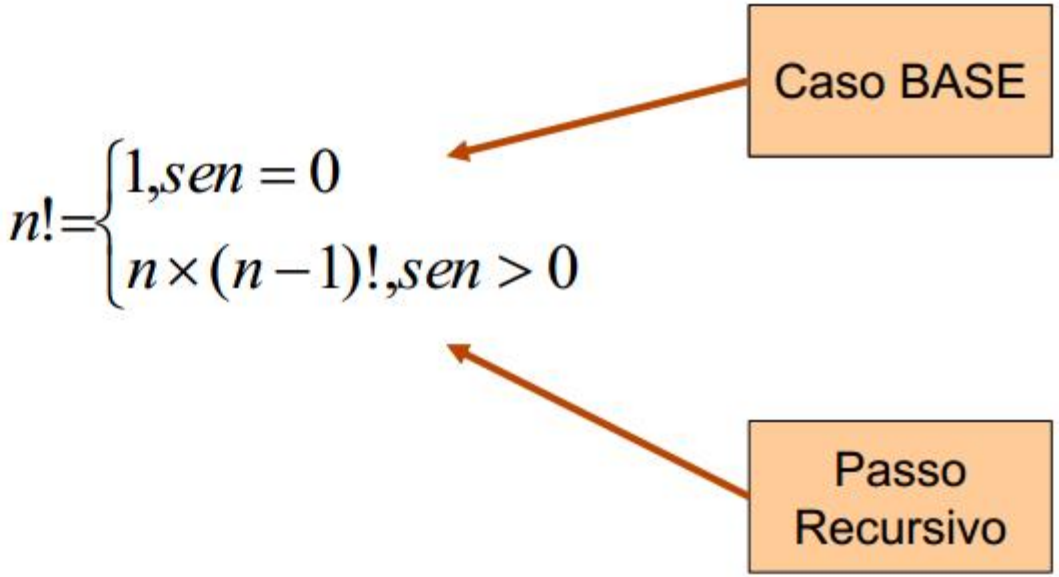
Recursividade

- Trazendo a recursividade para o nosso cotidiano um ótimo exemplo está na ação de contar um saco de moedas, onde a cada ato de retirar uma moeda do saco precisa-se “contar dinheiro” que corresponde a identificar qual é o valor da moeda e somá-la à quantia que ainda está no saco.

- Para identificar a quantia que ainda está no saco basta chamar a mesma função “contar dinheiro” novamente, porém dessa vez já considerando que essa moeda não está mais lá. E este processo de retirar uma moeda, identificar seu valor e somar com o restante do saco se repete até que o saco esteja vazio, quando atingiremos o ponto de parada e a função retornará o valor zero, indicando que não há mais moedas no saco.

Definições Recursivas

- **Exemplo:** o fatorial de um número



Definições Recursivas

- **Exercício:** forneça a definição recursiva para a operação de potenciação

$$x^n = \begin{cases} 1, \text{ se } n = 0 \\ x \times x^{(n-1)}, \text{ se } n > 0 \end{cases}$$

Caso BASE

Passo Recursivo

Funções Recursivas

- Definição:
  - Uma função recursiva é aquela que faz uma chamada para si mesma. Essa chamada pode ser:
    - direta: uma função ***A*** chama a ela própria
    - indireta: função ***A*** chama uma função ***B*** que, por sua vez, chama ***A***

```
/* Recursao direta */
void func_rec(int n)
{
   ...
   func_rec(n-1);
   ...
}
```

Funções Recursivas

- Exemplo: função recursiva para cálculo de fatorial

$$n! = \begin{cases} 1, \text{ se } n = 0 \\ n \times (n-1)!, \text{ se } n > 0 \end{cases}$$

```
/* Função recursiva para cálculo do fatorial */
int fat (int n)
{
   if (n==0)
      return 1;
   else
      return n*fat(n-1);
}
```

Caso BASE

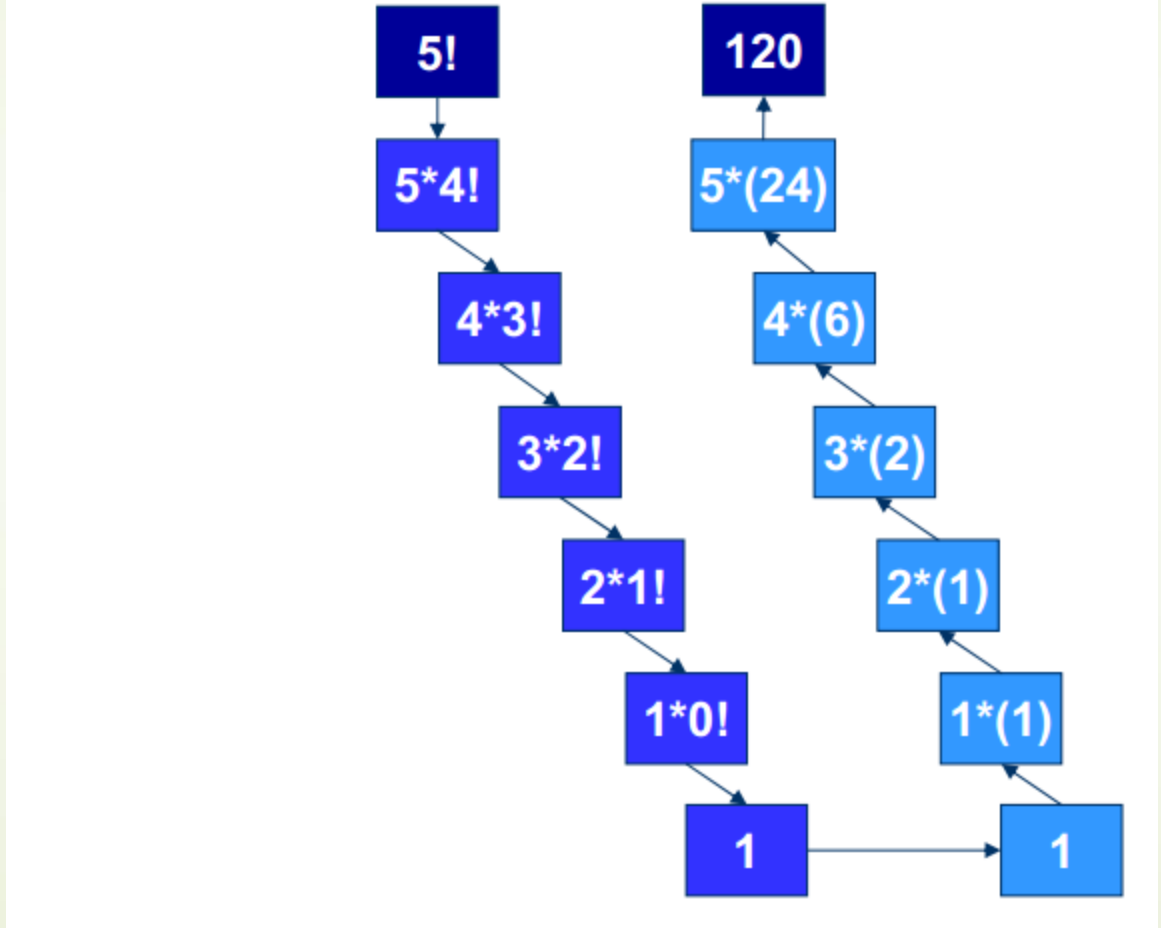
Passo Recursivo

$$fatorial(n) = \begin{cases} 1 & \text{se } n = 0 \\ n \times fatorial(n-1) & \text{se } n > 0 \end{cases}$$

A partir desta definição, também chamada relação de recorrência, calculamos $fatorial(3)$ da seguinte forma:

$$\begin{aligned} & fatorial(3) = 3 * fatorial(3-1) \\ & = 3 * fatorial(2) \\ & = 3 * 2 * fatorial(2-1) \\ & = 3 * 2 * fatorial(1) \\ & = 3 * 2 * 1 * fatorial(1-1) \\ & = 3 * 2 * 1 * fatorial(0) \\ & = 3 * 2 * 1 * 1 \\ & = 6 \end{aligned}$$

Chinês Fatorial



http://www.sssinformatica.com.br/prc/RecursividadeLinguagemC.pdf

Funções Recursivas

- Exemplo: função recursiva para cálculo de potenciação

```
/* Função recursiva para cálculo de potenciacao */
int pot (int x, int n)
{
   if (n==0)
      return 1;
   else
      return x*pot(x,n-1);
}
```

Caso BASE

Passo Recursivo

Fibonacci

```
1 - //Recursividade na Linguagem C
2 - //PRC - Prof. João
3 - //FIBONACCI
4 - #include "stdio.h"
5 - int fibonacci(int N) {
6 -     if( N == 1 )
7 -         return 1;
8 -     else
9 -         if( N == 2)
10 -            return 1;
11 -        else
12 -            return(fibonacci(N-1) + fibonacci(N-2));
13 - }
14 - main() {
15 -     int num, F;
16 -     printf("Entre com um número:    ");
17 -     scanf("%d", &num);
18 -     printf("A série de Fibonacci para %d elementos é:\n", num);
19 -     for(F=1;F <= num;F++) {
20 -         printf("%d, ", fibonacci(F));}
21 -         printf("\nOk");
22 - }
```

2 caso base

Passo Recursivo

http://www.sssinformatica.com.br/prc/RecursividadeLinguagemC.pdf

Conta Dígitos

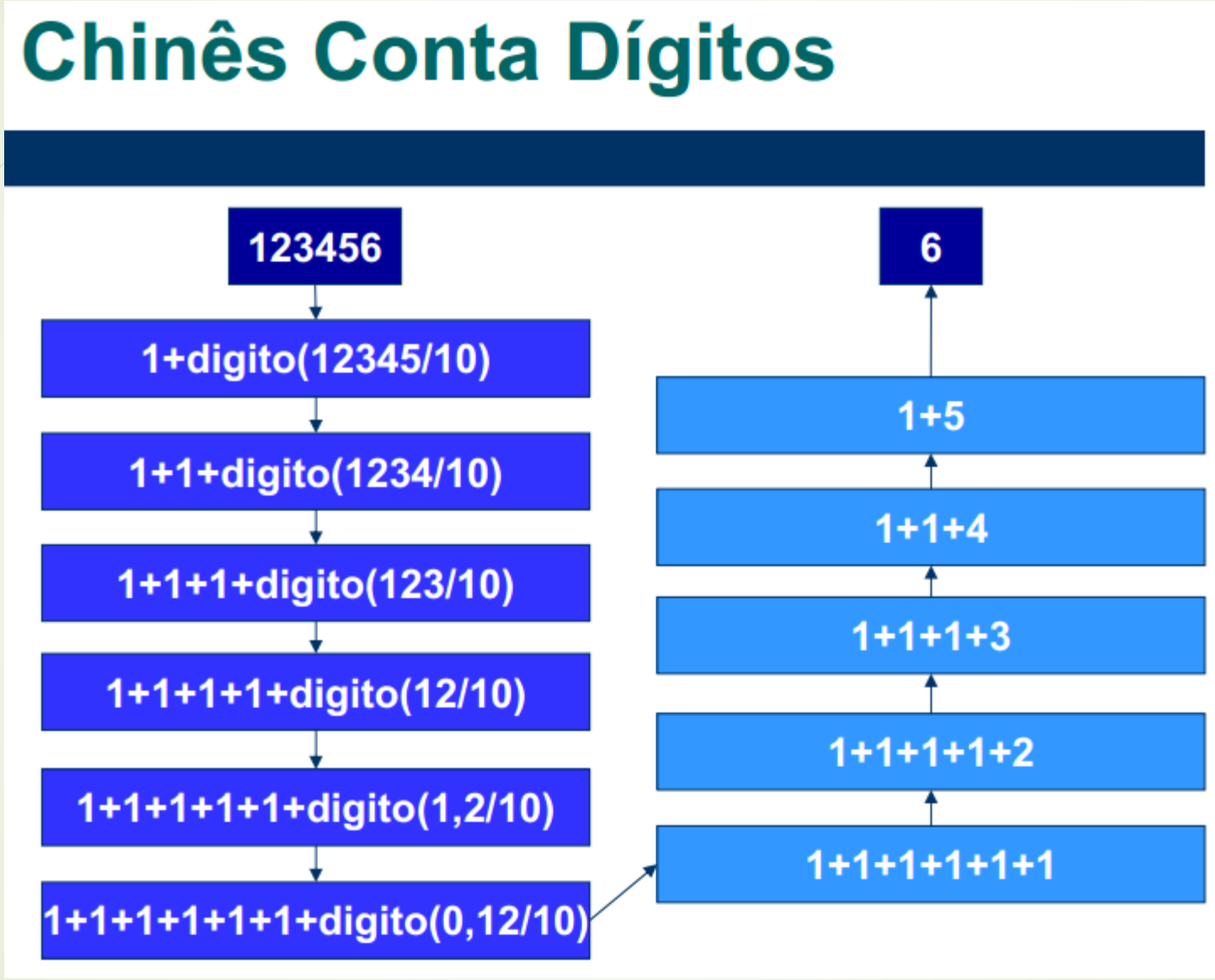
```
1 - //Recursividade na Linguagem C
2 - //PRC - Prof. João
3 - //CONTA DIGITOS
4 - #include “stdio.h”
5 - int digitos(int x) {
6 -     if( abs(x) < 10 )
7 -         return 1;
8 -     else
9 -         return(1 + digitos(x/10));
10 - }
11 - main() {
12 -     int num;
13 -     printf("Entre com um número: ");
14 -     scanf("%d", &num);
15 -     printf("O número de dígitos de %d é %d.", num, digitos(num));
16 - }
```

1 caso base

Passo Recursivo

http://www.sssinformatica.com.br/prc/RecursividadeLinguagemC.pdf

Chinês Conta Dígitos



http://www.sssinformatica.com.br/prc/RecursividadeLinguagemC.pdf

ANALISANDO RECURSIVIDADE

Vantagens X Desvantagens

- Um programa recursivo é mais elegante e menor que a sua versão iterativa, além de exibir com maior clareza o processo utilizado, desde que o problema ou os dados sejam naturalmente definidos através de recorrência.

- **Por outro lado**, um programa recursivo exige mais espaço de memória e é, na grande maioria dos casos, mais lento do que a versão iterativa.