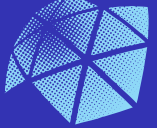
Laboratório de Algoritmos e Técnicas de Programação II

Aula 04 - Recursividade: fundamentação e como implementar

Álvaro Magri Nogueira da Cruz

Agenda



Introdução I

Definição

- Uma função recursiva é uma função que **chama a si mesma direta ou indiretamente**, por meio de outra função;
- Na verdade, a função sabe somente como resolver **os casos mais simples**, ou os chamados **casos básicos**.

Introdução II

O passo a passo

1. A função só sabe resolver o **caso mais simples**;
2. Se a função é chamada com um caso básico, ela simplesmente **retorna um resultado**;
3. Se uma função é chamada com um problema mais complexo, **ela divide o problema em duas partes**:
   - Uma parte que ela sabe como fazer;
   - Uma parte que ela não sabe como fazer.
4. Para tornar a recursão viável, **a segunda parte precisa ser** semelhante ao problema original, um pouco **mais simples**;
5. A etapa de recursão também inclui a palavra-chave *return*.

Exemplos I

Calculando fatoriais iterativamente

- **Cálculo de fatorial:** $n! = n * (n-1) * (n-2) \cdots * 1$
  - $1! = 1$ e $0! = 1$.
- E.g.: $5! = 5 * 4 * 3 * 2 * 1 = 120$.
- De forma iterativa temos:

```
fatorial = 1;
for(contador=numero; contador> 1; contador- -){
    fatorial = fatorial*contador;
}
```

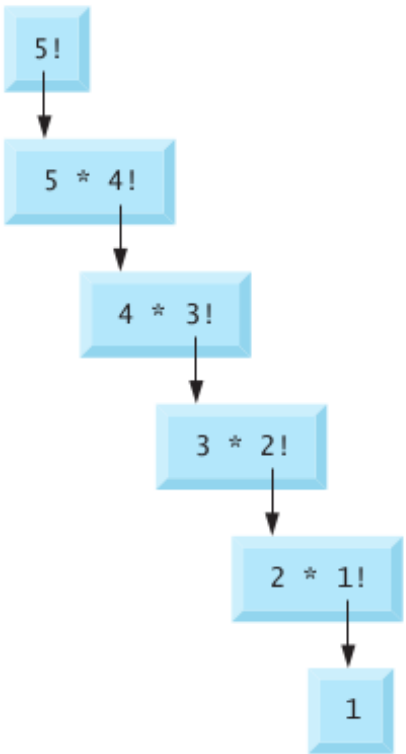
Exemplos II

Calculando fatoriais recursivamente

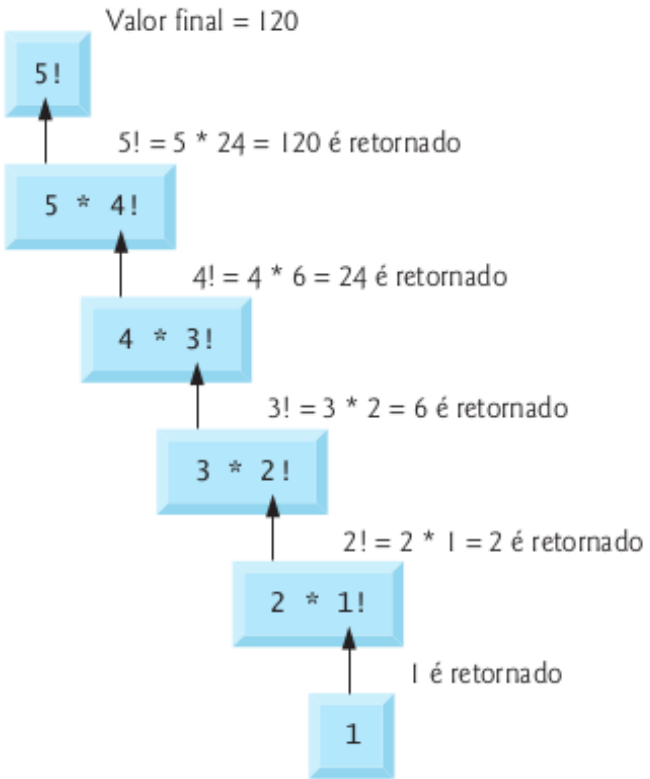
- Uma definição recursiva da função de fatorial é:
  - $n! = n * (n-1)!$.
- E.g.: $5! = 5 * 4!$:
  - $5! = 5 * 4 * 3 * 2 * 1$;
  - $5! = 5 * (4 * 3 * 2 * 1)$;
  - $5! = 5 * (4!)$.
- Observe a próxima Figura!

- Logo, a função deve retornar: **return** number * fatorial(number-1);

Exemplos III



(a) Sequência de chamadas recursivas.



(b) Valores retornados a partir de cada chamada recursiva.

Exemplos IV

Código iterativo - Fatorial

```c
#include <stdio.h>

long fatorial(long number);//prototipo

int main(){
    int i, entrada;//contador
    printf("Calcule o fatorial de: ");
    scanf("%d", &entrada);
    printf("\n%d\n", fatorial(entrada));
    return 0;
}

long fatorial(long number){
    long cont = number, resultado=1;
    for(cont = number; cont>=1; cont--){
        resultado = resultado * cont;
    }
    return resultado;
}
```

Exemplos V

Código recursivo - Fatorial

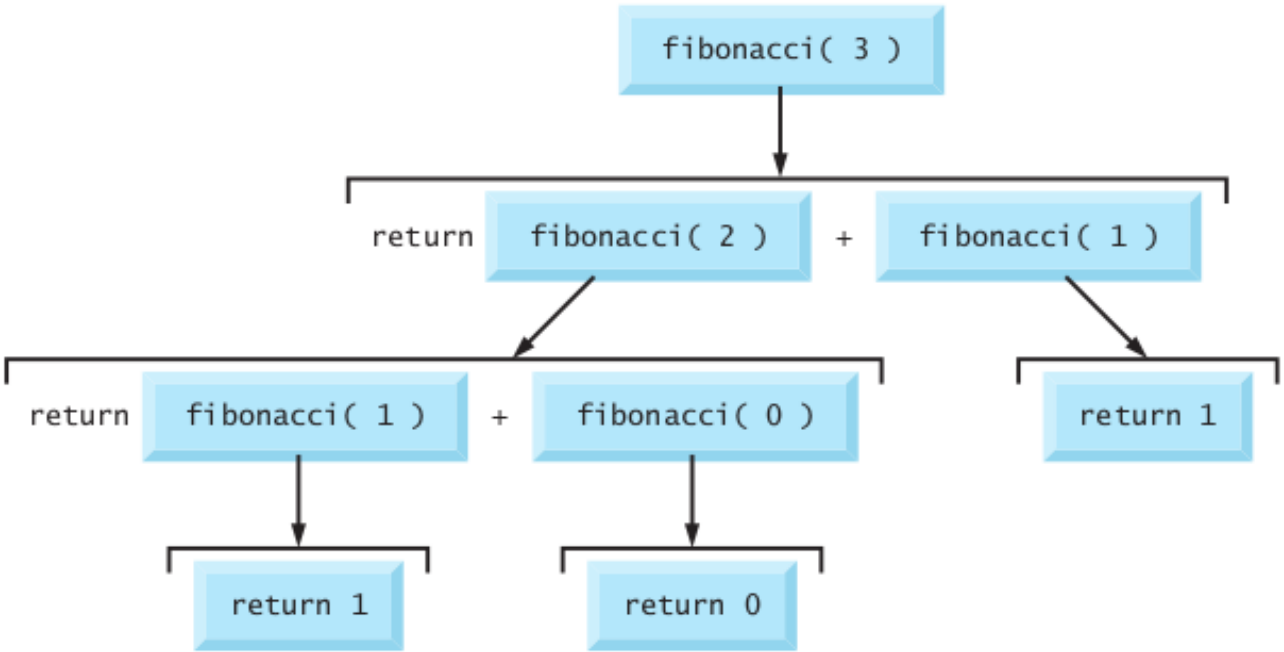
```c
#include <stdio.h>

long fatorial(long number);//prototipo

int main(){
    int i, entrada;//contador
    printf("Calcule o fatorial de: ");
    scanf("%d", &entrada);
    printf("\n%d\n", fatorial(entrada));
    return 0;
}

long fatorial(long number){
    if(number<=1){
        return 1;
    }
    else{
        return number*fatorial(number-1);
    }
}
```

Exemplos VI

A série de Fibonacci

- **0, 1, 1, 2, 3, 5, 8, 13, 21, ...**
- Começa com 0 e 1 e cada número de Fibonacci é a soma dos dois anteriores;
- Podemos definir recursivamente assim:
  - fibonacci(0) = 0;
  - fibonacci(1) = 1;
  - fibonacci(n) = fibonacci(n-1) + fibonacci(n-2))

Exemplos VII



Exemplos VIII

Código recursivo - Fibonacci

```c
#include <stdio.h>

int fibonacci(int number);//prototipo

int main(){
    int i, entrada;//contador
    printf("Digite um numero: ");
    scanf("%d", &entrada);
    printf("\nfibonacci(%d) = %d\n",entrada,fibonacci(
    entrada));
    return 0;
}

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

Exercícios

1. **Máximo divisor comum recursivo**. O máximo divisor comum dos inteiros x e y é o maior inteiro que divide x e y sem gerar resto. Escreva uma função recursiva mdc que retorne o máximo divisor comum de x e y . O mdc de x e y é definido recursivamente da seguinte forma: se y é igual a 0 , então mdc(x, y) é x ; caso contrário, mdc(x, y) é mdc(y, x % y) , onde % é o operador de módulo (ou resto da divisão).
2. É interessante observar a recursão "em ação". Modifique a função fatorial para imprimir sua variável local e o parâmetro de chamada recursiva. Para cada chamada recursiva, apresente as saídas em uma linha separada e acrescente um nível de recuo (TABULAÇÃO, o famoso "$\backslash t$"). Faça o máximo para tornar as saídas claras, interessantes e significativas. Seu objetivo aqui é projetar e implementar um formato de saída que ajude uma pessoa a entender melhor a recursão.

Referências I

1. **Deitel H. M., e Deitel P.J; "C: Como programar". 6.ed. Pearson Prentice Hall, 2011. 818p.**
2. Jean Paul Tremblay & Richard P. Bunt. "Ciência dos Computadores - Uma abordagem algorítmica". McGraw-Hill.
3. Jaime Evaristo. "Aprendendo a Programar / Programando em Turbo Pascal". Edufal - Editora da Univ. Federal de Alagoas. Maceió, 1996.
4. Harry Farrer et al. "Pascal Estruturado (da série "Programação Estruturada de Computadores")". Editora Guanabara Dois. Rio de Janeiro, 1985.
5. Stephen O'Brien. "Turbo Pascal 6 Completo e Total". Makron Books.

Referências II

6. Celes, W., Cerqueira, R., Rangel, J.L. "Introdução a Estrutura de Dados". Elsevier, 2004.
7. Feofiloff, P. "Algoritmos em Linguagem C". Elsevier, 2009. 208p.
8. Schildt, H. "C Completo e Total". 3ª ed. Pearson. 1996. 852p.