Anhanguera

Aqui o seu esforço ganha força.

Recursividade

Prof. Esp. Rodrigo Hentz

Definição

- Recursividade é o processo de se usar uma rotina ou conjunto de rotinas chamando a si mesma.
- Uma função é dita recursiva se invoca a si mesma direta ou indiretamente.

Fatorial de um número

- O fatorial de um número n é definido como o produto de todos os inteiros entre n e 1.
- Fatorial de 0 é definido como 1.
- Exemplo: fatorial de 5 = 5 * 4 * 3 * 2 * 1 = 120.

Anhanguera

Fatorial de um número

- Na matemática usamos o símbolo ! para indicar a função fatorial.
- Sua definição seria:

$$n! = 1 \text{ se } n == 0$$

$$n! = n * (n -1) * (n – 2) * ... * 1 \text{ se } n > 0$$

- Os três pontos são uma abreviação para os números compreendidos entre n – 3 e 2 multiplicados.

Anhanguera

Fatorial de um número

- Para cada número devemos criar uma fórmula para calcular seu fatorial?
- 0! = 1
- 1! = 1
- 2! = 2 * 1
- 3! = 3 * 2 * 1
- ...
- 10! = 10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1

Fatorial de um número

- A resposta é não! Podemos definir um algoritmo que aceite um inteiro como parâmetro e retorne o fatorial deste parâmetro.

Anhanguera

Fatorial de um número - Forma iterativa

- A forma iterativa requer a repetição explícita de um processo até que determinada condição seja satisfeita.

```
int fatorial(int numero)
{
  int t, f = 1;
  for (t = 1; t <= numero; t++) f = f * t;
  return f;
}
```

Anhanguera

Fatorial de um número

- Vamos olhar mais de perto a definição de n! que lista uma fórmula separada para cada valor de n. Pegamos como exemplo o fatorial de 4.

$$4! = 4 * 3 * 2 * 1$$

não seria igual a

$$4! = 4 * 3!$$

Anhanguera

Fatorial de um número

- Ou podemos usar uma notação matemática usada anteriormente teríamos:

$$n! = 1 \text{ se } n == 0$$

$$n! = n * (n – 1)! \text{ se } n > 0$$

Anhanguera

Fatorial de um número – Forma recursiva

```
int fatorial(int numero)
{
  int f;
  if (numero == 0) return 1;
  f = numero * fatorial(numero – 1);
  return f;
}
```

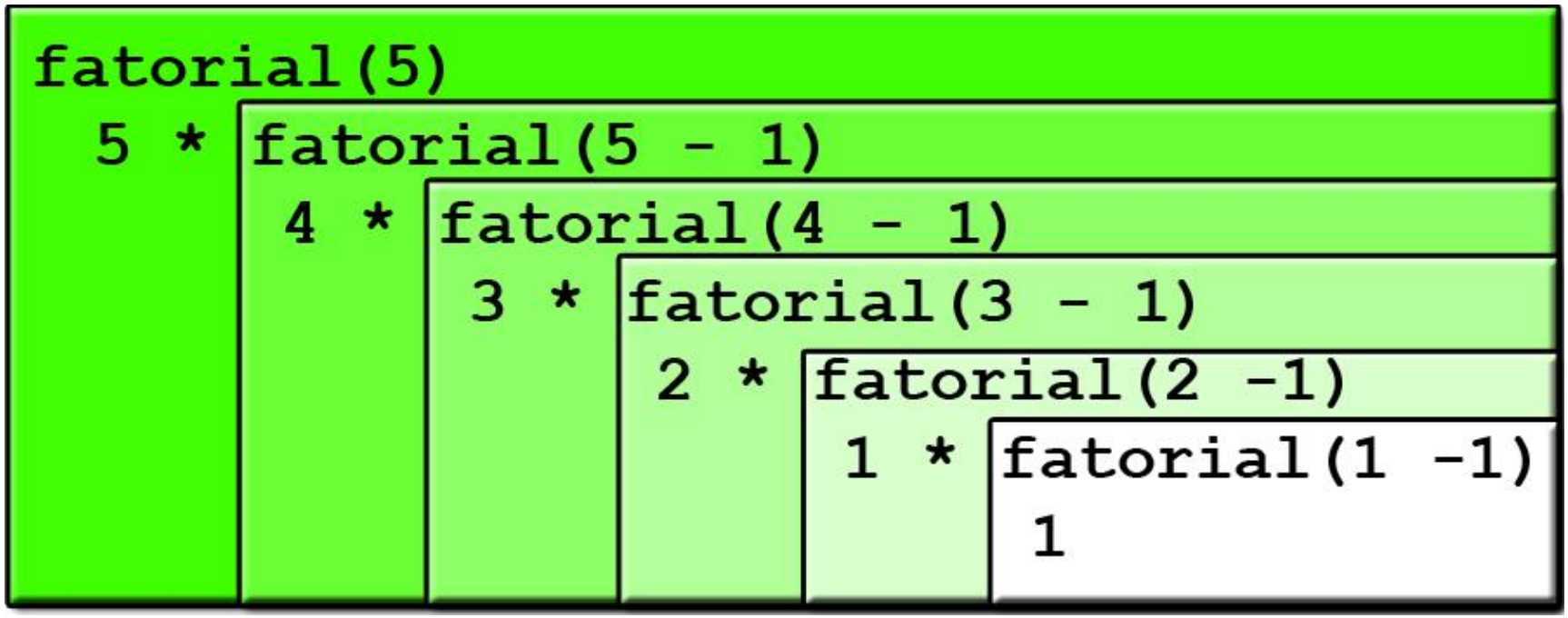
$$fatorial(n) = \begin{cases} 1 & \text{se } n = 0 \\ n \times fatorial(n-1) & \text{se } n > 0 \end{cases}$$

Exercício - Fatorial de um número

Usando a função abaixo de forma recursiva solicite ao usuário um número inteiro e escreva o fatorial deste numero.

```
int fatorial(int numero)
{
  int f;
  if (numero == 0) return 1;
  f = numero * fatorial(numero – 1); return f;
}
```

Recursividade



Anhanguera

Multiplicação de números naturais

- Outro exemplo de recursividade é a multiplicação de números naturais.

a * b

- Qual seria a definição iterativa deste processo?
- O produto a * b pode ser definido como a somado a si mesmo b vezes.

6 * 3 = 6 + 6 + 6

Anhanguera

Multiplicação de números naturais

- Qual seria a função recursiva para a multiplicação de números naturais definida abaixo?

$$a * b = a \text{ se } b == 1$$

$$a * b = a * (b - 1) + a \text{ se } b > 1$$

Multiplicação de números naturais

```
int multiplicacao(int a, int b) {
        if (b == 1) { //condição de parada
            return a;
        } else {
            return multiplicacao(a, b - 1) + a;
        }
    }
```

Exercício

Definir a função para calcular a potência de um número dada a definição recursiva abaixo com expoente inteiro:

$$x^n = \begin{cases} 1 / x^{(-n)} & \text{se } n < 0 \\ 1 & \text{se } n = 0 \\ x * x\ (n - 1) & \text{se } n > 0 \end{cases}$$

Exercício

```
double potencia(double x, int n)
{
  if (n == 0) return 1;
  else if (n > 0) return x * potencia(x, n – 1);
  else return 1 / potencia(x, -n);
}
```

Sequencia de Fibonacci

- A sucessão de Fibonacci ou sequência de Fibonacci é uma sequência de números naturais, na qual os primeiros dois termos são 0 e 1, e cada termo subsequente corresponde à soma dos dois precedentes.
- A sequência tem o nome do matemático do século XIII Leonardo de Pisa, conhecido como Leonardo Fibonacci, e os termos da sequência são chamados números de Fibonacci.
- Os números de Fibonacci são, portanto, os números que compõem a seguinte sequência de números inteiros:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, …

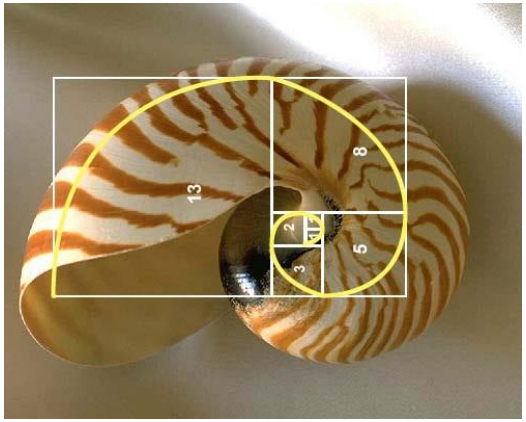
Sequencia de Fibonacci

- Em termos matemáticos, a sequência é definida recursivamente pela fórmula abaixo, sendo os dois primeiros termos F0= 0 e F1= 1.

$$F(n) = \begin{cases} 0, & \text{se } n = 0 \\ 1, & \text{se } n = 1 \\ F(n-1) + F(n-2) & \text{se } n > 1 \end{cases}$$

Sequencia de Fibonacci

- A sequência de Fibonacci tem aplicações na análise de mercados financeiros, na ciência da computação, física e na teoria dos jogos.
- Também aparece em configurações biológicas, como, por exemplo, na disposição dos galhos das árvores ou das folhas em uma haste, no arranjo do cone da alcachofra, do abacaxi, ou no desenrolar da samambaia.



Sequencia de Fibonacci

```
int fib(int n)
{
  int x, y;
  if (n <= 1) return n;
  x = fib(n – 1);
  y = fib(n - 2);
  return x + y;
}
```

Anhanguera

Vantagens e 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.

Finalização