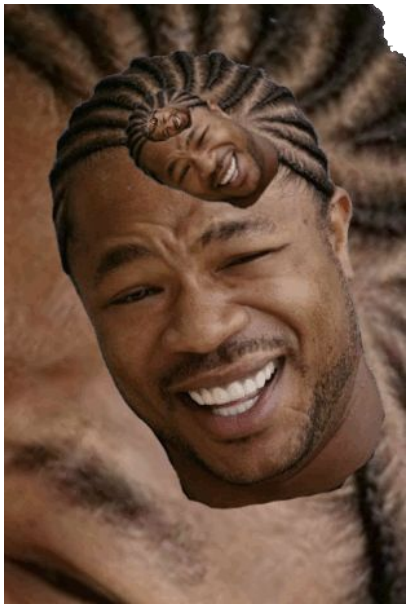
Aula 19 - Recursão

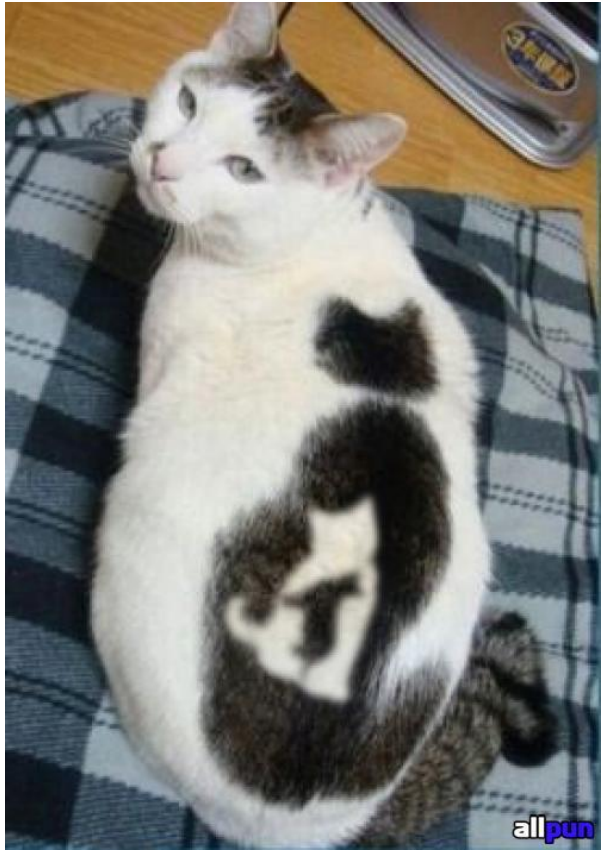
DJ Mp4

Roteiro da Aula

- Objetivos: apresentar o conceito de recursão e a modelagem de problemas com recursividade
- Fluxo:
  - O que é Recursão?
  - Forma Recursiva
  - Passos da recursão

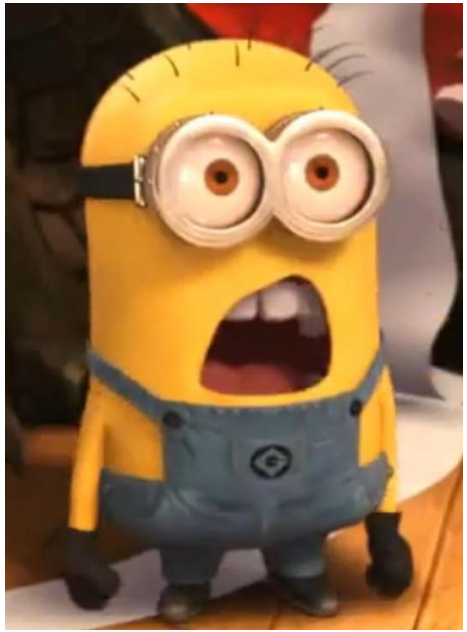


Inception!!!!!



Dividir para conquistar

- Muitas vezes, precisamos dividir um problema em subproblemas menores
  - Fica até mais fácil de resolver!
  - Exemplos:
    - Computação gráfica: BSP
    - Travessia em árvores



Recursão

- É um conceito geral (independe da computação)
  - Letras: GNU, Allegro (Allegro Low-Level Game Routines)
  - Matemática
  - Artes: música, pintura

- É a definição da solução de um problema em função de uma instância menor dele mesmo!

Exemplo de definição recursiva

- Fatorial de um número:
  - 5! = 5*4*3*2*1
  - 4! = 4*3*2*1
  - 5! = 5*4!

- Podemos dizer que fatorial(n) = n*fatorial(n-1)!

Forma Recursiva

- A recursão precisa de três elementos base:
  - Caso base ou condição de parada
  - Processamento a cada etapa
  - Chamada recursiva

Exemplo

```
int fatorial(int n){
    if(n == 1 || n == 0)
        return 1;
    else
        return n*fatorial(n-1);
}
```

```
int main(){
    int n;
    scanf("%d", &n);
    printf("%d\n", fatorial(n));
    return 0;
}
```

- Condição de Parada
- Processamento
- Chamada recursiva

Recursividade x Iteratividade

- Muitas vezes, a forma recursiva é mais natural para abordamos um problema
  - Série de Fibonacci

$$f(n) = \begin{cases} n = 0 & 0 \\ n = 1 & 1 \\ n > 1 & f(n-1) + f(n-2) \end{cases}$$

Fibonacci Iterativo

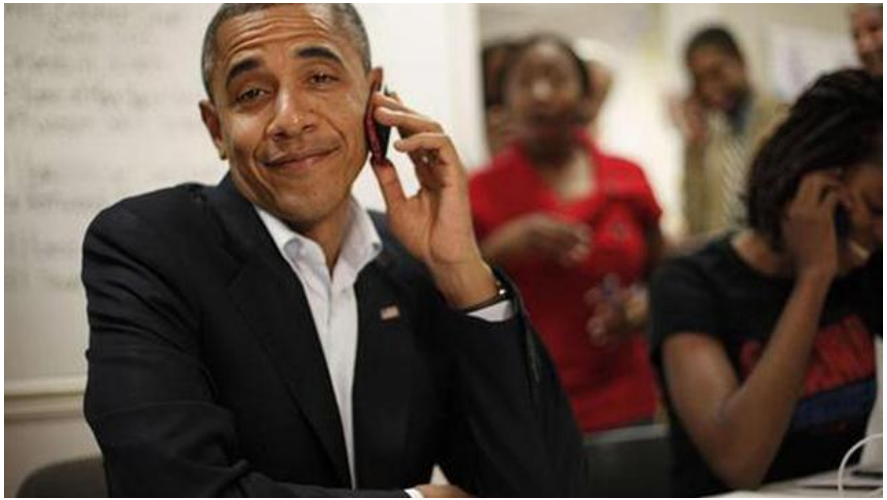
```
#include <stdio.h>
int main( void )
{
   int i, j, n, fib;
   scanf( "%i", &n );
   fib = 0;
   j = 1;
   for(i=1; i<n; i++)
   {
       j = fib + j;
       fib = j - fib;
   }
   printf(“%d\n”, fib);
   return 0;
}
```

Fibonacci Recursivo

```
int fib( int x )
{
   if( x <= 1 )
       return x;
   else
       return fib(x-1) + fib(x-2);
}
```

```
int main(){
       int n;
       scanf(“%d”, &n);

       printf(“%d\n”, fib(n));
       return 0;
}
```



Processamento

- O processamento de informações pode ocorrer:
  - antes da chamada recursiva (pré-processamento)
  - após a chamada recursiva (pós-processamento)

Exemplo - Conversão de Base

- Faça um programa que converta um número da base decimal para a base binária
  - Ex: 15 = 1111

Solução Recursiva

- Para n = 10
  - 10 / 2 = 5 (Resto 0)
  - 5 / 2 = 2 (Resto 1)
  - 2 /2 = 1 (Resto 0)
  - 1 / 2 = 0 (Resto 1)

- O valor binário é a ordem invertida dos restos: 1010

Solução Recursiva

- A divisão é um pré-processamento (você divide e então faz a chamada)

- A impressão é um pós-processamento (você só imprime o número após a chamada da função retornar)

Solução Recursiva

```
#include <stdio.h>

void binario(int n){
     if(n > 0){
          binario(n/2);
          printf("%d", n%2);
     }
}
```

```
int main(){
     int n;

     scanf("%d", &n)
     binario(n);
     printf("\n");

     return 0;
}
```

Explorando as possibilidades

- A recursão pode ser usada como procedimento

- Normalmente usamos recursão quando queremos explorar todas as possibilidades de um problema
  - Backtracking

Problema Troco

- Quantas formas diferentes de dar o troco?

Exemplo:

Compra 40 reais, Pagamento 50

- 1 nota de 10
- 2 notas de 5
- 5 notas de 2
- 1 nota de 5, 2 notas de 2 e uma moeda de 1
- 10 moedas de 1
- ….

Recursão Mútua ou Indireta

- Uma recursão pode ser montada usando mais de uma função

- Nesse caso, a recursão se dá quando um ciclo de chamadas é feito
  - Ex: função1 função2 função 1

Exemplo

```
int ehPar(int n){
    if(n == 0)
        return 1;
    else
        return ehImpar(n-1);
}
```

```
int ehImpar(int n){
    if(n == 0)
        return 0;
    else
        return ehPar(n-1);
}
```

Dúvidas?