


Recursão

Recursão, Arquivos de Cabeçalho, Argumentos na Main(), Enumeração.



Recursão

- Uma função é recursiva quando chama a si mesma de forma direta ou indireta.
- Exemplo:

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

Entrada	Saída	
Sum(1)	1	
Sum(2)	2 + sum(1)	2 + 1
Sum(3)	3 + sum(2)	3 + 2 + 1
Sum(4)	4 + sum(3)	4 + 3 + 2 + 1

Recursão

- Exemplo:

```
int fat(int n){  
    if (n<=1)  
        return 1;  
    else  
        return (n*fat(n-1));  
}
```

- A partir de um determinado valor de n , as saídas do programa podem fornecer valores errados. Qual o motivo?

Recursão

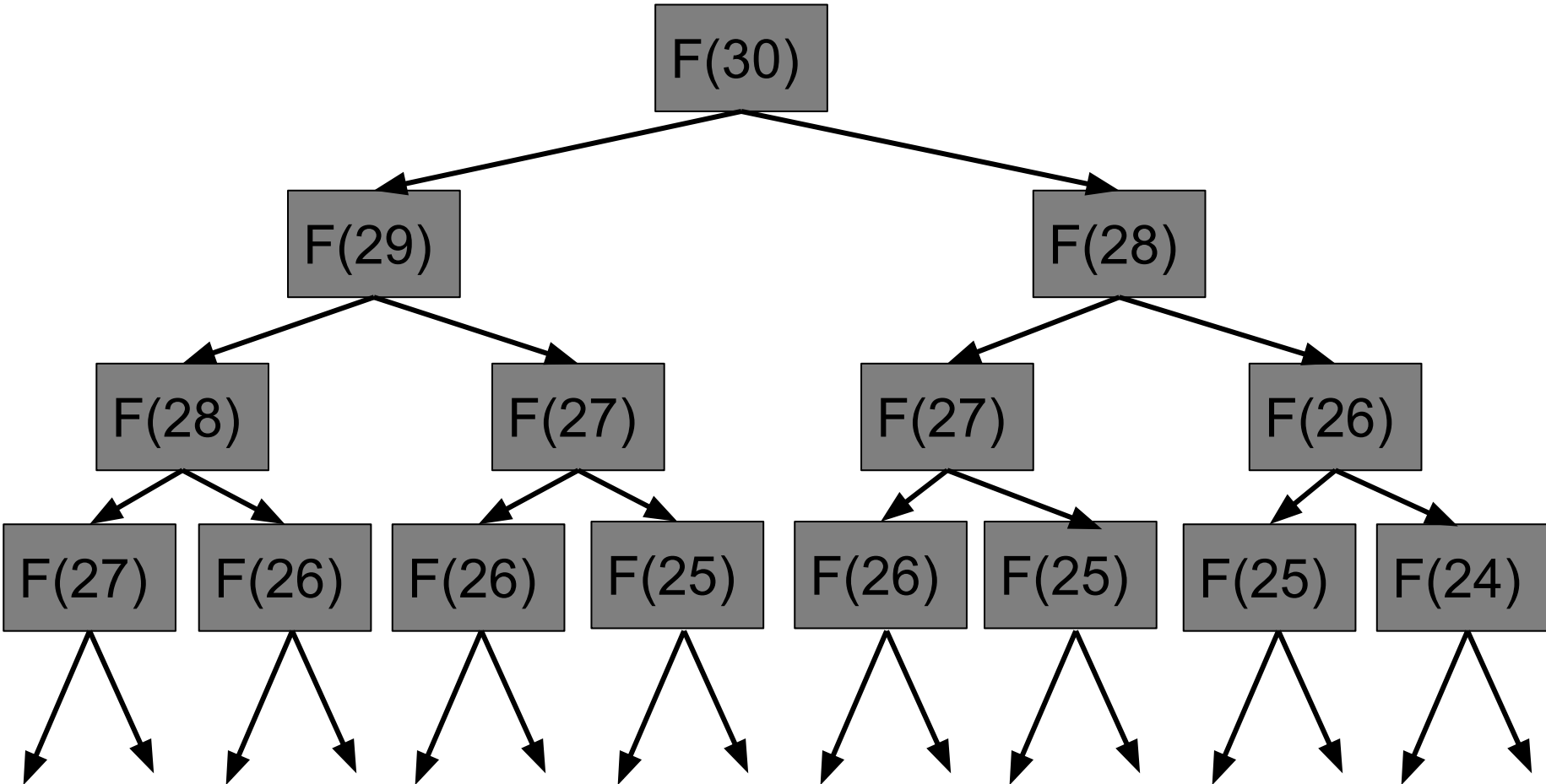
- Exemplo: Sequência de Fibonacci.

$$f(0)=0, f(1)=1, f(i+1)=f(i)+f(i-1), i=1,2,\dots$$

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

n	F(n)	Número de chamadas da função
0	1	1
1	1	1
2	1	3
...
23	28657	92735
24	46368	150049

Recursão



Recursão

- Recursão vs Iteração
 - Iteração usa estrutura de repetição
 - Recursão usa estrutura de seleção (chamadas de funções repetitivas).
 - Iteração usa violação da condição como critério de parada.
 - Recursão usa passo base como critério de parada.
 - Ambas podem ser executadas infinitamente, se o critério de parada não for satisfeito.
- Desvantagens da Recursão
 - Gera sobrecarga (overhead) com as chamadas de função, gerando gasto de tempo de processamento e espaço de memória.
 - Uma cópia da função (variáveis da função) é criada, consumindo memória.
 - Logo, a iteração tende a ser mais rápida por não fazer repetidas chamadas de funções.

Recursão

- Quando usar Recursão ou Iteração?

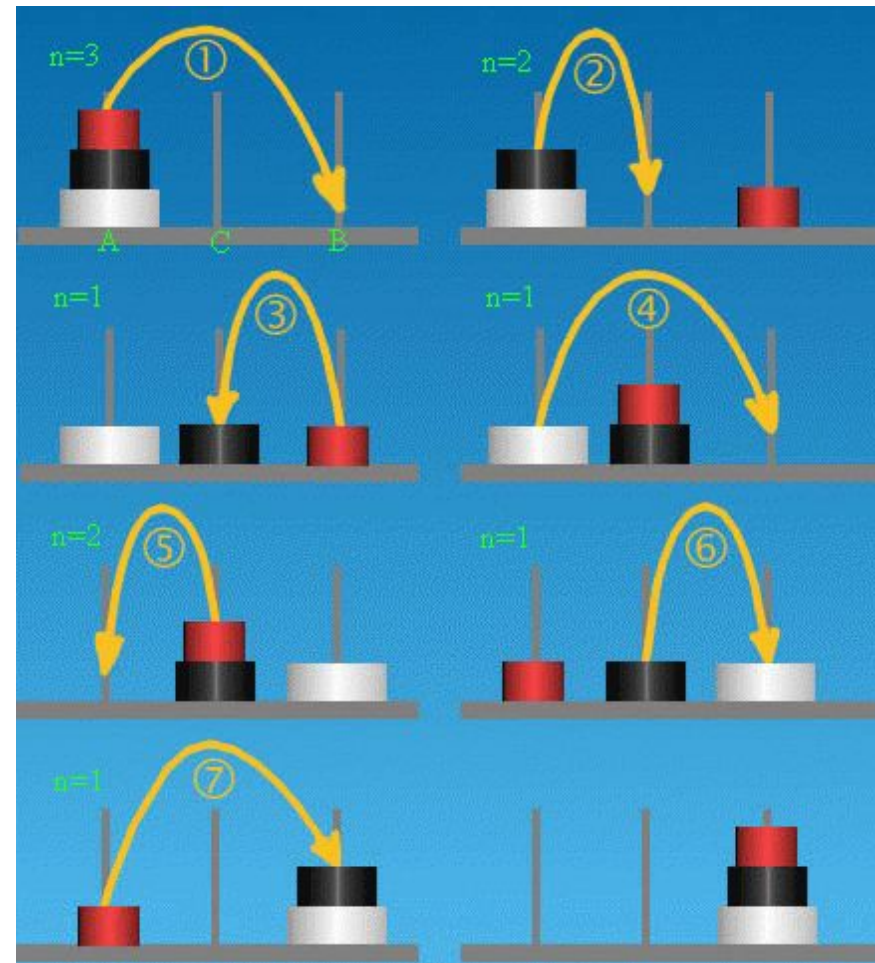
Recursão	Iteração
Estruturas condicionais	Estruturas de repetição
Repetição implícita	Repetição explícita
Caso base como critério de parada	Condição com critério de parada
Lento	Rápido
Solução simples	Solução complexa
Fácil manutenção	Difícil Manutenção

Arquivos de Cabeçalho

- Arquivos que compartilham declarações e definições podem ter essas informações centralizadas em arquivos de cabeçalho:

`<nome_arq.h>`

- Vamos considerar o desenvolvimento de um programa, composto por vários arquivos, para o problema das Torres de Hanoi.



Arquivos de Cabeçalho

hanoi.h

```
#includes  
#defines  
....  
Lista de protótipos
```

main.c

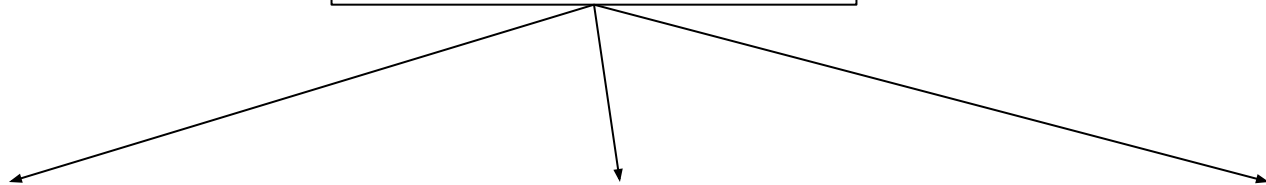
```
#include "hanoi.h"  
....
```

get.c

```
#include "hanoi.h"  
....
```

move.c

```
#include "hanoi.h"  
....
```



```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include <assert.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  extern int cnt;
6
7  int get_n(void);
8  void move(int n, char a, char b, char c);
9
```

```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include "hanoi.h"
2
3  int cnt = 0;
4
5  int main()
6  {
7      int n;
8      n=get_n();
9      assert(n>0);
10     move(n, 'A', 'B', 'C');
11     return 0;
12 }
13
```

```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include <assert.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  extern int cnt;
6
7  int get_n(void);
8  void move(int n, char a, char b, char c);
9  |
```

```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include "hanoi.h"
2
3  int get_n(void)
4  {
5      int n;
6
7      printf("%s",
8             "---\n"
9             "Torre de Hanoi:\n"
10            "\n"
11            "Há três torres: A, B e C.\n"
12            "\n"
13            "Os discos na torre A devem ser movidas para torre C. Apenas um\n"
14            "disco pode ser movido cada vez e a ordem em cada torre\n"
15            "precisa ser preservada a cada passo. Qualquer torre A, B,\n"
16            "ou C deve ser usada como torre intermediária na atribuição dos discos.\n"
17            "\n"
18            "O problema começa com n discos na torre A.\n"
19            "\n"
20            "Entrada n:");
21     if (scanf("%d",&n)!=1 || n < 1){
22         printf("\n Erro: Inteiro positivo não encontrado - tchau!!\n\n");
23         exit(1);
24     }
25     printf("\n");
26     return n;
27 }
28
```

```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include <assert.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  extern int cnt;
6
7  int get_n(void);
8  void move(int n, char a, char b, char c);
9
```

```
main.c ✕ hanoi.h ✕ get.c ✕ move.c ✕
1  #include "hanoi.h"
2
3  void move(int n, char a, char b, char c)
4  {
5      if(n==1){
6          ++cnt;
7          printf("%5d: %s%d%s%c%s%c.\n", cnt,
8              "Mover disco ", 1, " da torre ", a, " para torre ", c);
9      }
10     else{
11         move(n-1, a, c, b);
12         ++cnt;
13         printf("%5d: %s%d%s%c%s%c.\n", cnt,
14             "Mover disco ", n, " da torre ", a, " para torre ", c);
15         move(n-1, b, a, c);
16     }
17 }
18
```

Argumentos por Linha de Comando

- É possível “passar parâmetros” para iniciar a execução de um programa C.
- Esses parâmetros são chamados argumentos para a execução do programa.
- Quando a função main é chamada para iniciar a execução, ela é chamada com dois argumentos:

`main(argc, argv)`

- `argc` (argument count): é o número de argumentos com que o programa foi chamado.
- `argv` (argument values): é um vetor de ponteiros do tipo char.

```
1  #include <stdio.h>
2
3  int main(int argc, char *argv[])
4  {
5      int i;
6      printf("argc = %d\n",argc);
7      for(i=0; i<argc; ++i){
8          printf("argv[%d] = %s\n", i, argv[i]);
9      }
10     return 0;
11 }
12
```



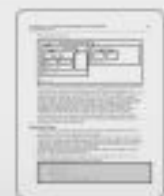
arg



arg.c



arg.o



Aula19.pdf

```
claudio@claudio-HP-ProBook-6460b: ~/Downloads/SSC0501
claudio@claudio-HP-ProBook-6460b:~/Downloads/SSC0501$ ./arg a is for apple
argc = 5
argv[0] = ./arg
argv[1] = a
argv[2] = is
argv[3] = for
argv[4] = apple
claudio@claudio-HP-ProBook-6460b:~/Downloads/SSC0501$
```

Enumeração

- Enumeração é um tipo definido na linguagem C que define um conjunto de constantes de enumeração inteiras representadas por identificadores.
- Palavra chave **enum** é utilizada para criar esses conjuntos:

enum dia {domingo, segunda, terça, quarta, quinta, sexta, sabado};

dia: tag para o conjunto

identificadores: domingo, segunda,....,sabado.

- As constantes de enumeração são identificadas por valores inteiros que começam por definição em 0

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6
7     enum dia {dom,
8     enum mes {jan=
9     enum fruta {pera=0, laranja=1, uva=4, banana=5};
10
11     enum dia diaSemana;
12     enum mes mesAtual;
13     enum fruta f1;
14     enum {cenoura=2, beterraba, batata} veg1, veg2;
15     enum fruta f2;
16
17
18     for(diaSemana=dom; diaSemana<=sab; diaSemana++){
19         printf("%d ", diaSemana);
20     }
21     printf("\n");
22
23     for(mesAtual=jan; mesAtual<=dez; mesAtual++){
24         printf("%d ", mesAtual);
25     }
26     printf("\n");
27     f1=pera; f2=uva;
28     printf("%d %d %d %d\n", f1, f1+1, f2, f2+1);
29     veg2=batata;
30     for(veg1=cenoura; veg1<=veg2; veg1++){
31         printf("%d ", veg1);
32     }
33
34     return 0;
35 }
```

```
0 1 2 3 4 5 6
1 2 3 4 5 6 7 8 9 10 11 12
8 9 4 5
2 3 4
Process returned 0 (0x0)   execution time : 0,002 s
Press ENTER to continue.
```



```

1  #include<stdio.h>
2  #include<stdlib.h>
3
4  enum meses{jan=1, fev, mar, abr, mai, jun,
5            jul, ago, set, out, nov, dez};
6  int main(void)
7  {
8      enum meses mes;
9
10     printf("Digite o numero do mes: ");
11     scanf("%d",&mes);
12     exibeMeses(mes);
13     exibeMeses(mes+1);
14     return 0;
15 }
16
17 void exibeMeses(enum meses m){
18     if((m >= jan) && (m <= dez)){
19         switch(m)
20         {
21             case jan:
22                 printf("%d - Janeiro",m);
23             break;
24             case fev:
25                 printf("%d - Fevereiro",m);
26             break;
27             case mar:
28                 printf("%d - Marco",m);
29             break;
30             case abr:
31                 printf("%d - Abril",m);
32             break;
33             case mai:
34                 printf("%d - Maio",m);
35             break;
36             case jun:
37                 printf("%d - Junho",m);
38             break;
39             case jul:
40                 printf("%d - Julho",m);
41             break;

```

```

/home/claudio
Digite o numero do mes: 4
4 - Abril
5 - Maio

```

```

40     printf("%d - Agosto",m);
41     break;
42     case ago:
43         printf("%d - Agosto",m);
44     break;
45     case set:
46         printf("%d - Setembro",m);
47     break;
48     case out:
49         printf("%d - Outubro",m);
50     break;
51     case nov:
52         printf("%d - Novembro",m);
53     break;
54     case dez:
55         printf("%d - Dezembro",m);
56     break;
57     }
58 }
59 else{
60     printf("INVALIDO!!!\n");
61 }
62 return 0;
63 }

```