

## Recursividade ou Iteratividade? - Parte Final

Mário Leite

...

Apesar de algumas literaturas afirmarem que existem casos em que a *recursão* é a melhor solução (caso da *Função de Ackermann*) por outro lado, todas elas também concordam que, mesmo tornando o código mais compacto e mais elegante, as rotinas recursivas são bem mais lentas que as iterativas; às vezes muito, muito mais lentas, a ponto de inviabilizar o seu uso em programas comerciais. E para tirar dúvidas criei dois programas, ambos codificados em **C** e com o mesmo objetivo: “determinar o *n*-ésimo termo da Sequência de Fibonacci”: **ProgFibolte** que contém a função iterativa **FunFibolte()** e programa **ProgFiboRec** com a função recursiva **FunFiboRec()**, com o mesmo propósito: retornar esse elemento da sequência.

Os dois programas foram testados para encontrar os seguintes elementos dessa sequência: *décimo*, *vigésimo*, *trigésimo*, *quadragésimo*, *quinquagésimo*, *sexagésimo* e *centésimo*. Os resultados estão na **tabela 1**, incluindo os tempos médios gastos nos processamentos.

A **figura 1** mostra, em particular, o processamento com função recursiva para calcular o *sexagésimo* elemento, inclusive o tempo gasto para obter esse elemento.

Os resultados demonstram, claramente, que o processamento com função recursiva vai ficando cada vez mais lento à medida que aumenta o número de auto execuções, chegando a intervalos de tempo em que torna inviável sua utilização, como aconteceu para o centésimo elemento da sequência. Por outro lado, com o programa que utiliza função iterativa o processamento foi quase instantâneo (0 milissegundos), como pode ser visto na **figura 2**. Já com a função recursiva, para determinar o centésimo elemento, o programa ficou rodando por mais de doze horas seguidas; e não chegando a nenhum resultado, resolvi interromper a execução após **12h35min** de processamento contínuo, ficando evidente a sobrecarga na manutenção da pilha.

Portanto, ficou comprovado que mesmo tornando o código mais compacto e mais elegante (sobretudo nas estruturas em árvores) o uso de funções recursivas em programas profissionais deve ser muito bem planejado, com bastante critério e em processamentos que não exijam um número muito grande de auto execuções.

Para esclarecimentos: O processamento foi executado no seguinte sistema computacional:

- Sistema. Operacional: Windows 10 Pro
- Codificação: IDE Dev-C++ - V 5.11
- Sistema *desktop* (processador: Intel Core i3-7100, 3.91Ghz)
- Memória cache: 4 Mb
- RAM: 4 Gb
- HD de 500 Gb (40% espaço livre)

Portanto, pode ser que em situações diferentes os resultados sejam diferentes; mas, com a recursão sendo mais lenta que a iteração.

---

Elemento	Valor	Tempo com Iteração	Tempo com Recursão
10	55	0 ms	0 ms
20	6765	0 ms	0 ms
30	832040	0 ms	0 ms
40	102334155	0 ms	438 ms
50	12.586.269.025	0 ms	54.460 ms
60	1.548.008.755.920	0 ms	6.680.110 ms (1h51m20s)
100	354.224.848.179.262.000.000	0 ms	Acima de 12 horas

**Tabela 1 - Resultados obtidos na determinação de sete elementos da Sequência de Fibonacci**

Elemento	Valor	Tempo com Iteração	Tempo com Recursão
10	55	0 ms	0 ms
20	6765	0 ms	0 ms
30	832040	0 ms	0 ms
40	102334155	0 ms	438 ms
50	12.586.269.025	0 ms	54.460 ms
60	1.548.008.755.920	0 ms	6.680.110 ms (1h51m20s)
100	354.224.848.179.262.000.000	0 ms	Acima de 12 horas

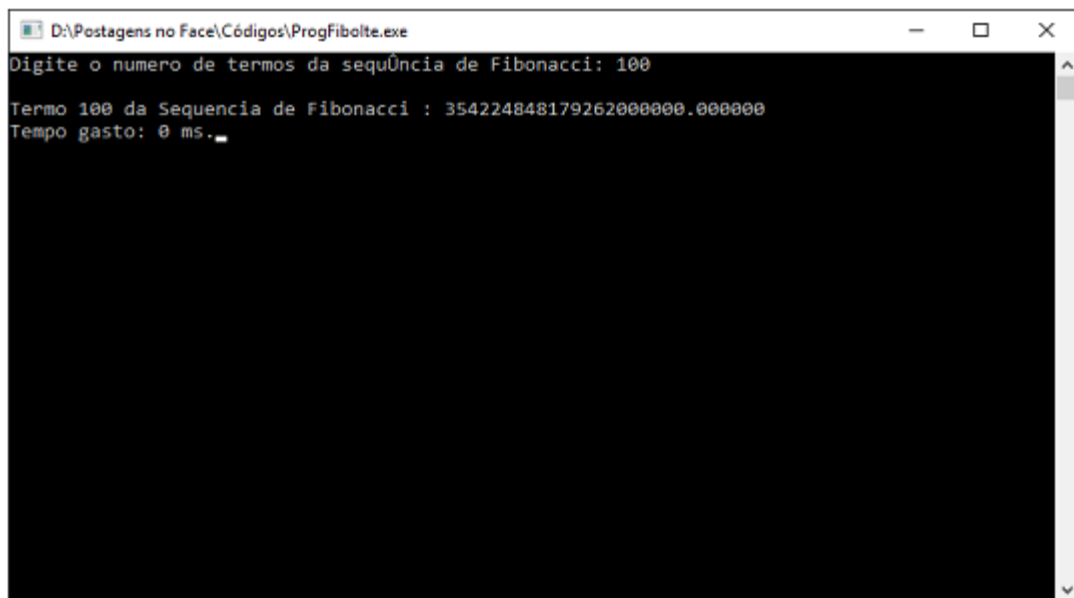
**Tabela 1 - Resultados obtidos na determinação de sete elementos da Sequência de Fibonacci**

```

D:\Postagens no Face\Códigos\ProgFiboRec.exe
Digite o numero de termos da sequôncia de Fibonacci: 60
Termo 60 da Sequencia de Fibonacci : 1548008755920.000000
Tempo gasto: 6.68011e+006 ms.


```

**Figura 1 - Obtendo o sexagésimo elemento da Sequência de Fibonacci com recursão**



```
D:\Postagens no Face\Códigos\ProgFibolte.exe
Digite o numero de termos da sequ&eacutencia de Fibonacci: 100
Termo 100 da Sequencia de Fibonacci : 354224848179262000000.000000
Tempo gasto: 0 ms.█
```

Figura 2 - Obtendo o cent&easimoo elemento da Sequ&eacutencia de Fibonacci com itera&ccedil;o



```
D:\Postagens no Face\Códigos\ProgFiboRec.exe
Digite o numero de termos da sequ&eacutencia de Fibonacci: 60
Termo 60 da Sequencia de Fibonacci : 1548008755920.000000
Tempo gasto: 6.68011e+006 ms.█
```

Figura 1 - Obtendo o sexag&easimoo elemento da Sequ&eacutencia de Fibonacci com recurs&eacut;o

```
D:\Postagens no Face\Códigos\ProgFibolte.exe
Digite o numero de termos da sequência de Fibonacci: 100

Termo 100 da Sequencia de Fibonacci : 354224848179262000000.000000
Tempo gasto: 0 ms.
```

Figura 2 - Obtendo o centésimo elemento da Sequência de Fibonacci com iteração

```
[*] ProgFibolte.C
1  /*ProgFiboIte.C
2  Calcula o n-ésimo termo da Sequência de Fibonacci com função iterativa,
3  e o tempo gasto no processamento.
4  Autor: Mário Leite
5  */
6  #include <stdio.h>
7  #include <conio.h>
8  #include <stdlib.h>
9  #include <time.h>
10 //=====
11 //Define a função iterativa
12 double FunFiboIte(int x)
13 {
14     int j, F;
15     double Termo[1000]; //limita ao milésimo termo máximo
16     for(j=1; j<=x; j++) {
17         if((j==1) || (j==2))
18             Termo[j] = 1;
19         else
20             Termo[j] = Termo[j-1] + Termo[j-2];
21     }
22     return Termo[j-1];
23 } //FimFunção
24 //-----
25 int main() {
26     int n=0;
27     double F, Tempo;
28     while((n<3) || (n>1000)) {
29         printf("Digite o numero de termos da sequência de Fibonacci: ");
30         scanf("%i",&n);
31     }
32     clock_t Ticks[2];
33     Ticks[0] = clock(); //inicia a medição do tempo de processamento
34     F = FunFiboIte(n); //chama a função passando-lhe o parâmetro
35     Ticks[1] = clock(); //Inicia medição do tempo de processamento
36     Tempo = (Ticks[1] - Ticks[0]) * 1000.0 / CLOCKS_PER_SEC; //tempo em milissegundos
37     printf("\n");
38     printf("Termo %d da Sequencia de Fibonacci : %f", n, F);
39     printf("\n");
40     printf("Tempo gasto: %g ms.", Tempo);
41     getch();
42     return 0;
43 } //FimPrograma
```

[\*] ProgFiboRec.C

```
1  /*ProgFiboRec.C
2  Calcula o n-ésimo termo da Sequência de Fibonacci com função recursiva,
3  e o tempo gasto no processamento.
4  Autor: Mário Leite
5  */
6  #include <stdio.h>
7  #include <conio.h>
8  #include <stdlib.h>
9  #include <time.h>
10 //=====
11 //Define a função recursiva
12 double FunFiboRec(int x)
13 {
14     if((x<=1) || (x==2))
15         return 1;
16     else
17         return FunFiboRec(x-1) + FunFiboRec(x-2); //recursão
18 }//FimFunção
19 //-----
20 int main()
21 {
22     int n;
23     double F, Tempo;
24     n = 0;
25     while((n<3) || (n>1000)) { //limita ao milésimo termo máximo
26         printf("Digite o numero de termos da sequência de Fibonacci: ");
27         scanf("%i",&n);
28     }
29     clock_t Ticks[2];
30     Ticks[0] = clock(); //inicia a medição do tempo de processamento
31     F = FunFiboRec(n); //chama a função passando-lhe o parâmetro
32     Ticks[1] = clock(); //finaliza a medição do tempo de processamento
33     Tempo = (Ticks[1] - Ticks[0]) * 1000.0 / CLOCKS_PER_SEC; //tempo em milissegundos
34     printf("\n");
35     printf("Termo %d da Sequencia de Fibonacci : %f", n, F);
36     printf("\n");
37     printf("Tempo gasto: %g ms.", Tempo);
38     getch();
39     return 0;
40 }//FimPrograma//-----
```