Sobre Números Perfeitos

Mário Leite

…

"*Número Perfeito é aquele cuja soma de seus divisores (ele próprio excluído) é igual a ele mesmo*". Por exemplo, o número **496** é perfeito; seus divisores são: 1, 2, 4, 8, 16, 31, 62, 124, 248 e 496. Assim, **496 = 1 + 2 + 4 + 8 + 16 + 31 + 62 + 124 + 248**. A literatura mostra muitos artigos sobre este tipo de número, que começou por Aristóles, afirmando o seguinte: "s*e tantos números quantos se queira começando a partir da unidade forem dispostos continuamente numa proporção duplicada até que a soma de todos resulte num número primo, e se a soma multiplicada pelo último origina algum número, então o produto será um número perfeito*". Uma afirmação muito rebuscada e meio estranha, já que inclui números primos. Mas, objetivamente, do ponto de vista matemático, isto pode ser colocado na fórmula $\mathbf{2^n - 1}$ sendo **n** um número primo. Então, seguindo o raciocínio de Aristóteles, o produto $\mathbf{2^{(n-1)}(2^n - 1)}$ resultaria num número perfeito o que, na verdade, vale apenas para **n=2,3,5,7** pois, para o quinto valor de **n**, que é **11**, esta fórmula não funciona; mas, a afirmação de que "*se a soma dos divisores de um número for igual ao próprio número (ele próprio excluído...) continua válida*. O program "**MostraNumerosPerf**", codificado em **C** e em **Python** mostram a relação dos **oito** primeiros números perfeitos; e porque apenas oito!? Os números perfeitos crescem assustadoramente a partir do oitavo numero, além de serem muito raros. Para se ter uma ideia, o trigésimo número pergfeito é **2658455991569831744645692615953842176**. Por isto limitei a quantidade em oito pois, a partir deste o tempo de processanento para detectar número perfeito pode levar horas, mesmo utilizando um *hardware* bem potente e uma linguagem de programação bem rápida. Por isto, em vez de usar o critério de verificar a soma dos diviores, optei por utilizar a fórmula de Euclides-Euler do tipo $\mathbf{2^{(n-1)}*(2^n - 1)}$ onde **n** é um número primo e com valores prestabelecidos.

Obs: Note que há uma divergência sobre o valor do oitavo número perfeito no código em C e Python; com certeza deve ser problema de arredondamento no código em **C**, pois o *link* abaixo mostra um valor coincidende com a saída do código em Python. Também, cabe outro esclarecimento quanto ao fato de o programa só mostrar os oito primeiros números perfeitos: o nono número perfeito tem 37 dígitos, e mesmo que o Python possa computar números extremamente grandes utilizando biblioteca específica, ou usando formatação apropriada no código em C, o objetivo aqui foi apenas o de mostrar a lógica do programa para pesquisa de números perfietos.

https://rce.casadasciencias.org/rceapp/art/2020/056/ (acesso em 27/02/2022 – 11:37)

```c
//NumerosPerfeitos.C
//Mostra os n primeiros números perfeitos desejados pelo usuário, usando o método
//de Aristóteles-Euler.
//--------------------------------------------------------------------------------
#include "stdio.h"
#include "stdlib.h"
#include "conio.h"
#include "math.h"
int main() {
   int j, qteNum;
   int MAX = 8;  //limita quantidade de números gerados devido ao longo processamento
   double n,numPerf;
   double VetPrimos[8] = {2, 3, 5, 7, 13, 17, 19, 31};  //cria vetor de primos
   printf("Quanto números perfeitos você deseja? [máximo 8]: ");
   scanf("%d", &qteNum);
   printf("\n");
   while ((qteNum<1) || (qteNum>MAX)) {
      printf("Quanto números perfeitos você deseja? [máximo 8]: ");
      scanf("%i",&qteNum);
   }
   printf("\n");
  printf("%s%i%s", " Os ", qteNum," primeiros numeros Perfeitos.");
  printf("\n");
   for(j=0; j<qteNum; j++) {
      n = VetPrimos[j];
      numPerf = (pow(2,(n-1)) * (pow(2,n) - 1));  //fórmula de Euclides-Euler
      printf(" %.0f\n", numPerf);  // mostra o Número Perfeito
  }
  getch();
  return 0;
}
```

Figura 1 - Código em C

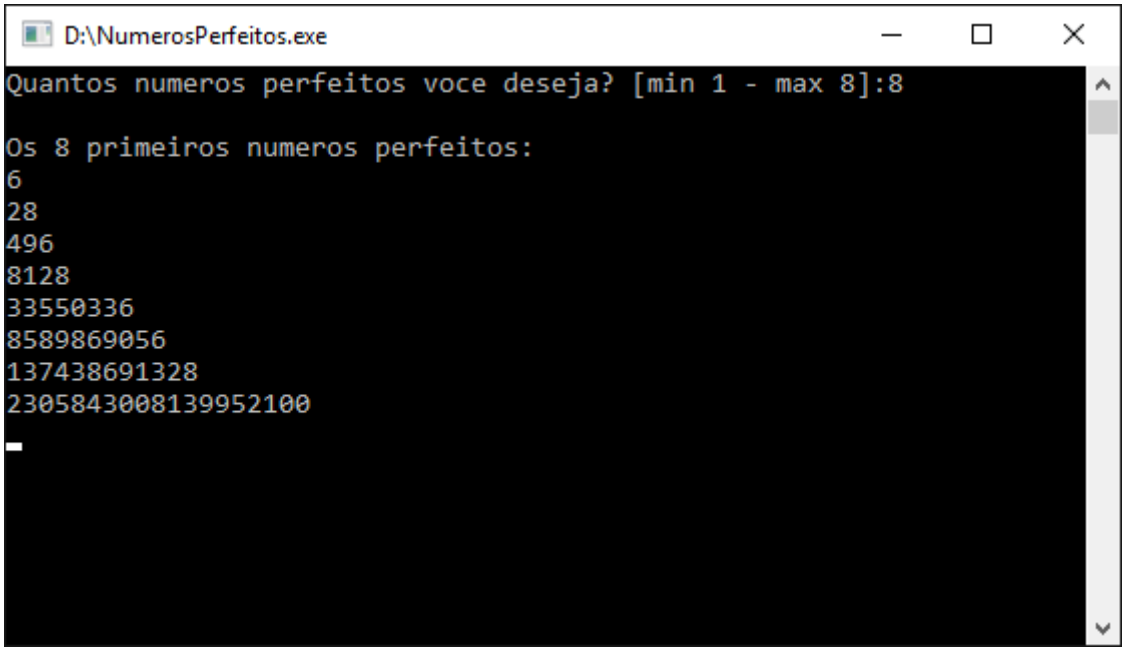

Figura 1.1 - Saída do programa codificado em C

```python
#NumerosPerfeitos.py"
#Mostra os oito primeiros Números Perfeitos
#Autor: Mário Leite
#-----------------------------------------------------------------------------------

#Define terminadores
endif = "endif"
endwhile = "endwhile"
endfor = "endfpr"

#Cria vetor de números primos para calcular número perfeito
MAX = 8
lstPrimos = [2,3,5,7,13,17,19,31]

qteNum = 0
while ((qteNum<1) or (qteNum>MAX)):
    qteNum = int(input("Quantos números perfeitos você deseja? [min 1 - max 8]: "))
endwhile

print("")
print("Os", qteNum, "primeiros números perfeitos:")
for j in range(0, (qteNum)):
   n = lstPrimos[j]
   numPerf = (2**(n-1)) * ((2**n - 1))  #fórmula de Euclides-Euler
   print(f'{numPerf}')  #mostra o Número Perfeito
endfor
```

Figura 2 - Código em Python

```
"D:\Cantinho da Programação\Códigos\Python\NumPerf\venv\Scripts\python.exe"
 "D:/Cantinho da Programação/Códigos/Python/NumPerf/NumerosPerfeitos.py"
Quantos números perfeitos você deseja? [min 1 – max 8]: 8


Os 8 primeiros números perfeitos:
6
28
496
8128
33550336
8589869056
137438691328
2305843008139952128

Process finished with exit code 0
```

Figura 2.1 - Saída do programa codificado em Python

https://rce.casadasciencias.org/rceapp/art/2020/056/

http://clubes.obmep.org.br/blog/numeros-especiais-numeros-perfeitos/