**Procurando datas em um bilhão de decimais**

**Mário Leite**

**...**

Eu já havia apresentado uma solução para pesquisar e listar datas válidas num arquivo-texto de **PI** com um *milhão de decimais*, mostrando, inclusive, as posições das datas encontradas. O programa lia o arquivo e, encontrando uma data válida, guardava a posição de início da *string* dessa data e a inseria em uma lista. E, devido aos recursos oferecidos pelo Python, esta linguagem foi a escolhida para implementar o código-fonte. A execução do programa levou cerca de 2 segundos (num PC de 2 Ghz e 8 Gb de RAM), lendo esse arquivo de 977 Kb para listar todas as datas válidas encontradas no universo das decimais de PI. Empolgado com o projeto (e auxiliado pelo meu amigo ***Clésio Matias***), resolvi ir um pouco além, e foi criado o programa **“PesquisaDataPI.py”** que verifica se a data digitada (*dia*, *mês*, *ano*) está contida nas decimais de PI, num universo de 120 anos. Mas, neste caso, o conteúdo do arquivo lido é mil vezes maior que o anterior: PI com **um bilhão de decimais** (<https://stuff.mit.edu/afs/sipb/contrib/pi/pi-billion.txt>), cujo arquivo-texto tem 976,5 Mb: um monstro! Agora, em vez de mostrar todas as datas válidas do arquivo, é solicitada do usuário uma data (dia, mês, ano) que é pesquisada no arquivo; mais simples, porém, mais objetivo que no *site* do “Atractor”. O tempo de processamento leva um pouco mais de 3 minutos para procurar uma data válida; mas, vale a pena testar o programa!

**-------------------------------------------------------------------------**

Para adquirir o *pdf/book* de alguns livros meus sobre programação, entre em contado: **marleite@gmail com**

**-------------------------------------------------------------------------**

*'''  
PesquisaData.py  
Faz pesquisa de uma data válida num arquivo-texto de PI com um milhão de decimais  
Autor: Mário Leite & Clésio Matias*#------------------------------------------------------------------------------------  
endif = "endif"  
enddef = "enddef"  
endfor = "endfor"  
endwith = "endwith"  
#------------------------------------------------------------------------------------  
  
from datetime import date, datetime  
from os import read  
import time  
#------------------------------------------------------------------------------------  
  
#------ Escopo de funções -----------------------------------------------------------  
def AvaliaData(dia, mes, ano):  
 # Função que recebe uma cadeia de oito caracteres e verifica se é uma data válida  
 Dia = int(dia)  
 Mes = int(mes)  
 Ano = int(ano)  
 cond1 = (Dia >= 1 and Dia <= 31) and (Mes > 0 and Mes <= 12) \  
 and (Ano >= anoMin and Ano <= anoMax)  
 cond2 = True  
 cond3 = True  
 if (Mes == 2):  
 if ((Dia == 29) and (Mes == 2)): # verifica se o ano é bissexto  
 cond2 = ((Ano % 4 == 0) and (Ano % 100 != 0)) or (Ano % 400 == 0)  
 endif  
 if(Dia>29):  
 cond2 = False  
 endif  
 endif  
 if((Mes==4) or (Mes==6) or (Mes==9) or (Mes==11)) and (Dia>30):  
 cond3 = False  
 endif  
 cond = ((cond1) and (cond2) and (cond3))  
 if (cond):  
 return True  
 else:  
 return False  
 endif  
enddef  
  
#==== Programa principal ============================================================  
inicio = time.time() #liga o cronômetro  
anoMax = date.today().year # pega o ano atual como limite superior  
anoMin = anoMax - 120 # pega o ano do limite inferior (considerando 120 anos o máximo)  
ocorre = 0  
datasOrdenadas = {} # Dicionário que guarda as datas ordenadas  
dicDatas = {} # Dicionário que reserva as datas ocorridas e seus respectivos índices  
dia = input("Digite o dia: ")  
mes = input("Digite o mes: ")  
ano = input("Digite o ano: ")  
Data = dia + mes + ano  
  
vale = AvaliaData(dia, mes, ano)  
if(vale):  
 '''Abrindo o arquivo pi1Bi.txt de PI e inserindo-o numa string'''  
 with open('pi1Bi.txt', 'r') as pi:  
 # Aqui atribuimos o arquivo txt aberto à variável que será trabalhada  
 total = pi.read()  
 total = total.replace('.', '') # remove o ponto de PI  
 # Loop para percorrer toda a string, validando e mostrando as ocorrências de datas:  
 for j in range(len(total) - 7):  
 elemento = total[j:j + 8] # cria o elemento de 8 dígitos para ser avaliado  
 if(elemento == Data): # achou data válida  
 vale = AvaliaData(dia,mes,ano,) # chama função para validar a data  
 if (vale): # se a data é válida nas decimais  
 dicDatas[str(j)] = elemento[4:] + elemento[2:4] + elemento[0:2]  
 ocorre += 1 # incrementa as ocorrências  
 endif  
 endif  
 endfor  
 endwith  
endif  
  
#-------Mostra as ocorrências--------------------------------------------------------  
# Ordena as datas contidas no dicionário sem alterar seu índice nas decimais do PI  
if(vale):  
 for item in sorted(dicDatas, key=dicDatas.get, reverse=True):  
 datasOrdenadas[item] = dicDatas[item]  
 endfor  
  
 # Retorna datas ordenadas no formato DDMMYYYY e acrescenta a barra ('/')(DD/MM/YYYY)  
 for key, value in datasOrdenadas.items():  
 datasOrdenadas[key] = f'{value[6:]}/{value[4:6]}/{value[0:4]}'  
 endfor  
endif  
  
print()  
DataF = dia + "/" + mes + "/" + ano  
if(AvaliaData(dia,mes,ano)):  
 if(ocorre==1):  
 print(f'A data {DataF} foi encontrada na posição {datasOrdenadas[1]}.')  
 elif(ocorre>1):  
 print(f'A data {DataF} foi encontrada nas seguintes posições:')  
 for key, value in datasOrdenadas.items():  
 print(f'{key}')  
 endfor  
 else:  
 print(f'A data {DataF} não foi encontrada nas decimais de PI.')  
 endif  
else:  
 print(f'{DataF} não é uma data válida.')  
endif  
  
if(vale):  
 print()  
 print(f'Número de ocorrências de datas: {ocorre}')  
endif  
  
fim = time.time() #desliga o cronômetro  
tempo = round((fim - inicio),2) #tempo de processamento  
print()  
print(f'Tempo total de processamento: {tempo} seg')  
#==== Fim do Programa =================================================================





