INF101 – Introdução à Programação II
Roteiro de Prática: 25 de março de 2021

Nome: _____ Matr.: _____

Introdução

Nesta prova, vamos resolver o problema de *adicionar um dia* a uma data qualquer do calendário gregoriano. Este é o calendário atual usado oficialmente pela maioria dos países. Sua vigência foi decretada pelo papa Gregório XIII (1502-1585), a partir do dia 15 de outubro de 1582, uma sexta-feira, que foi o dia seguinte a 4 de outubro do mesmo ano, uma quinta-feira. Essa eliminação de dez dias serviu para sanar anomalias do calendário juliano que considerava o tempo de translação da terra em torno do sol como sendo exatamente 365 dias e 6 horas. Isso, naquela época, já havia causado a antecipação de dez dias no início das estações do ano. O calendário gregoriano leva em conta o tempo médio de 365 dias, 5 horas, 49 minutos e 12 segundos para cada ano o que difere de apenas 27 segundos a mais em relação ao ano solar real.

Pois bem, vamos lá. Em princípio, é um problema de fácil solução pelo ser humano devido à necessidade constante de sabermos qual é a data de hoje pelo menos, a data de nosso aniversário e a do aniversário de nossos entes queridos, as datas dos feriados e dos eventos importantes etc. Abaixo apresentamos o algoritmo que vamos seguir:

```
Dados:
        d, m, a : inteiro;  // 1 ≤ d ≤ 31, 1 ≤ m ≤ 12, a > 1582

se d = "número de dias no mês m e ano a" então
        d ← 1;
        se m = 12 então
                m ← 1;
                a ← a + 1
        senão
                m ← m + 1
        fimse
senão
        d ← d + 1
fimse
// O resultado encontra-se na 3-tupla (dia, mês, ano).
```

Agora temos um problema delicado para resolver: como determinar o número de dias de cada mês? O calendário gregoriano define o seguinte:

Nº	Mês	Nº de Dias no Mês
1	Janeiro	31
2	Fevereiro	28 ou 29
3	Março	31

4	Abril	30
5	Maio	31
6	Junho	30
7	Julho	31
8	Agosto	31
9	Setembro	30
10	Outubro	31
11	Novembro	30
12	Dezembro	31

Resta somente a questão do mês de fevereiro que pode ter 28 ou 29 dias. São 28 dias nos anos normais e 29 nos anos bissextos. O que é ano bissexto? De acordo com o calendário gregoriano, é aquele que é múltiplo de 4, mas não de 100, ou aquele que é múltiplo de 400. Portanto tudo resolvido! A seguir, temos o algoritmo para a determinação do número de dias em cada mês m e ano a:

Dados:
    m : inteiro;   // o número do mês, conforme a tabela acima
    a : inteiro;   // o ano do mês m, a > 1582
    dias : inteiro; // o número de dias no mês m, ano a

escolha (m)
    caso 1: caso 3: caso 5: caso 7: caso 8: caso 10: caso 12:
        dias ← 31
    caso 2:
        se bissexto(a) então dias ← 29
        senão dias ← 28
        fimse
    caso 4: caso 6: caso 9: caso 11:
        dias ← 30
fimescolha

Infelizmente, Python, a nossa linguagem de implementação de programas, não possui o comando escolha (*select* ou *switch* em outras linguagens de programação). Por outro lado, podemos implementar uma solução muito eficiente em Python, usando uma lista para simular o comando escolha acima. Veja a solução abaixo por meio da função `num_dias_no_mes`:

```
def num_dias_no_mes(m, a):
    numDiasMes = [31, 29 if bissexto(a) else 28, 31, 30, 31, 30, 31, 31,
                  30, 31, 30, 31]
    return numDiasMes[m-1]
```

Instruções

1. Baixe do Sistema de Entrega de trabalhos o seguinte arquivo: `p08_gab1_esqueleto.py`.
2. Abra o IDLE e crie um novo arquivo fonte denominado `p08.py`. Copie para ele o conteúdo de `p08_gab1_esqueleto.py`. Não se esqueça de salvá-lo de tempos em tempos, porque pode ocorrer falha de energia elétrica durante a aula prática.
3. Preencha os comentários obrigatórios (nome, matrícula, data e uma breve descrição sobre o que o programa faz).
4. Estruture seu programa em quatro funções: `main()`, `bissexto(a)`, `num_dias_no_mes(m, a)`, `adicione_1dia(d, m, a)`.
5. A função `main` já está pronta (favor não mexer nela) e faz o seguinte: lê uma data válida no formato dia, mês, ano (na aula de hoje não vamos nos preocupar com a verificação da validade de uma data, porém pense no assunto); chama a função `adicione_1dia` passando a data lida como parâmetro; imprime a data de um dia adicionado; pergunta ao usuário se quer entrar com mais uma data; caso não queira, termina a execução do programa.
6. Implemente a função `bissexto(a)` com um parâmetro: o ano. A função deve retornar verdadeiro (`True` em Python), se o ano `a` for de fato bissexto; caso contrário, retorna falso (`False`).
7. Copie, para seu arquivo-fonte, a função `num_dias_no_mes(m, a)` como implementada acima. Caso você tenha uma solução melhor para a determinação do número de dias em um mês qualquer, você pode pôr a sua implementação. O que seria uma solução melhor? Seria aquela que ou tivesse menos linhas de código, ou o código fosse mais eficiente em execução do que o apresentado acima.
8. Implemente a função `adicione_1dia(d, m, a)` conforme o algoritmo dado acima na Introdução. Ela retorna a nova data.
9. Não se esqueça de chamar a função `main()` no final do seu código para desencadear todo o processo.
10. Teste seu programa com várias datas válidas. Ao se convencer de que o programa esteja correto, entregue o arquivo-fonte `p08.py` por meio do Sistema de Entrega do LBI.
11. Se seu programa entrar em *laço infinito,* digite CTRL-C na janela do *Shell*, para interromper a execução do programa. Após, conserte-o.

Exemplo de execução do programa

(**Obs.:** As linhas enfatizadas com fundo amarelo correspondem às entradas do usuário.)

```
Entre com uma data (dd mm aaaa):
25 3 2021
O dia seguinte é 26/03/2021.
Deseja continuar (s/n)? s

Entre com uma data (dd mm aaaa):
28 2 2021
O dia seguinte é 01/03/2021.
Deseja continuar (s/n)? s

Entre com uma data (dd mm aaaa):
31 12 1999
O dia seguinte é 01/01/2000.
Deseja continuar (s/n)? s

Entre com uma data (dd mm aaaa):
28 2 2000
O dia seguinte é 29/02/2000.
Deseja continuar (s/n)? s

Entre com uma data (dd mm aaaa):
28 2 2100
O dia seguinte é 01/03/2100.
Deseja continuar (s/n)? n
```