INF101 – Introdução à Programação II

**Roteiro de Prática: 11 de março de 2021**

## Introdução

O objetivo principal desta prática é familiarizar o estudante com a utilização da estrutura de dados *fila* que pode ser manipulada diretamente pela linguagem Python, usando lista e mantendo a política FIFO. Para ilustrar o uso de filas, vamos considerar o problema de simulação de filas de banco. Nesta aula, vamos considerar duas filas ao invés de apenas uma como visto em aula teórica. Para facilitar a entrada das operações a ser feitas com as filas, vamos considerar os seguintes códigos para as operações:

- A – atendimento da fila 1
- B – atendimento da fila 2
- F – chegada de um cliente na fila 1
- G – chegada de um cliente na fila 2
- S – saída da simulação

Outra modificação a ser feita também com relação ao que foi feito em aula teórica é poder trabalhar com várias operações digitadas uma única vez. Lá apenas uma operação é que era lida de cada vez, agora leia todas as operações como uma cadeia de caracteres (*string*). Por exemplo, a cadeia FFGBBBAS significa duas chegadas de novos clientes na fila 1 e uma chegada na fila 2, três atendimentos da fila 2 e um da fila 1, e, finalmente, a saída da simulação.

## Solução dada em aula teórica

Para facilitar, apresentamos abaixo a solução em Python que foi dada em aula teórica:

```
ultimo = 10
fila = list(range(1, ultimo+1))
while True:
   print("\nExistem %d clientes na fila." % len(fila))
   print("Fila atual:", fila)
   print("\nDigite F para adicionar um cliente ao fim da fila,")
   print("ou A para realizar o atendimento. S para sair.")
   operacao = input("Operação (F, A, ou S): ").upper()
   if operacao == "A":
      if len(fila) > 0:
         atendido = fila.pop(0)
         print("Cliente %d atendido." % atendido)
      else:
         print("Fila vazia! Ninguém para atender.")
   elif operacao == "F":
      ultimo = ultimo + 1 # incrementa o tíquete para novo cliente
      fila.append(ultimo)
   elif operacao == "S":
      break
   else:
      print("Operação inválida! Digite apenas F, A ou S.")
```

## Refinamento da solução

Nesta prática, vamos estruturar melhor a solução e considerar os acréscimos pedidos acima na Introdução. A leitura da cadeia de entrada com as operações a ser feitas nas filas será realizada pela função `leiaOperacoes()` que retornará a cadeia lida. A simulação em si será realizada pela função `simule(n, opers)` em que o parâmetro `n` conterá o número inicial de clientes que deverão ser distribuídos equitativamente pelas duas filas. Se `n` for ímpar, a primeira fila ficará com menos um cliente do que a segunda. O parâmetro `opers` conterá a cadeia de operações. **Sugestão:** Em Python, pode-se percorrer uma cadeia de caracteres usando o comando `for`. Por exemplo, podemos iterar por cada operação armazenada na cadeia `opers` da seguinte maneira:

```
for operacao in opers:
   ...
```

Portanto podemos usar este `for` ao invés do comando `while True` acima.

## Instruções

1. Abra o IDLE e crie um novo arquivo fonte denominado `p06.py`. Não se esqueça de salvá-lo de tempos em tempos, porque pode ocorrer falha de energia elétrica durante a aula prática.
2. Digite os comentários obrigatórios (nome, matrícula, data e uma breve descrição sobre o que o programa faz).
3. Estruture seu programa em três funções: `main()`, `leiaOperacoes()` e `simule(n, opers)`.
4. A função `main()` deve ser bem simples: emita uma mensagem sobre o programa. Veja no exemplo de teste abaixo qual mensagem deverá ser exatamente. Em seguida, peça ao usuário digitar quantos clientes serão alocados inicialmente nas filas. Depois chame a função `leiaOperacoes()` e, finalmente, a função `simule(n, opers)` em que `n` é o número inicial de clientes e `opers` é a cadeia contendo os códigos das operações a ser feitas nas filas.
5. Implemente as funções `leiaOperacoes()` e `simule(n, opers)`, conforme o refinamento da solução descrito acima.
6. Não se esqueça de colocar, no fim de seu código fonte, uma chamada da função `main()` para começar a execução realmente do programa.
7. Se seu programa entrar em *laço infinito,* digite CTRL-C na janela do *Python Shell* para interromper a execução do programa.
8. Teste seu programa com várias sequências de entrada e vários números de clientes iniciais. Não fique satisfeito com o teste abaixo apenas!

## Exemplo de teste

```
Simulação de duas filas de banco
Quantos clientes serão inicialmente? 11

Digite a sequência de operações a ser feitas:
F para adicionar um cliente na fila 1
G para adicionar um cliente na fila 2
A para atender cliente na fila 1
B para atender cliente na fila 2
S para sair da simulação
aBfGdbS

Existem 5 clientes na fila 1.
```

```
Fila 1 atual: [1, 2, 3, 4, 5]
Existem 6 clientes na fila 2.
Fila 2 atual: [6, 7, 8, 9, 10, 11]

==> Operação: A
Cliente 1 atendido.

Existem 4 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5]
Existem 6 clientes na fila 2.
Fila 2 atual: [6, 7, 8, 9, 10, 11]

==> Operação: B
Cliente 6 atendido.

Existem 4 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5]
Existem 5 clientes na fila 2.
Fila 2 atual: [7, 8, 9, 10, 11]

==> Operação: F

Existem 5 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5, 12]
Existem 5 clientes na fila 2.
Fila 2 atual: [7, 8, 9, 10, 11]

==> Operação: G

Existem 5 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5, 12]
Existem 6 clientes na fila 2.
Fila 2 atual: [7, 8, 9, 10, 11, 13]

==> Operação: D
Operação inválida! Digite apenas F, G, A, B ou S.

Existem 5 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5, 12]
Existem 6 clientes na fila 2.
Fila 2 atual: [7, 8, 9, 10, 11, 13]

==> Operação: B
Cliente 7 atendido.

Existem 5 clientes na fila 1.
Fila 1 atual: [2, 3, 4, 5, 12]
Existem 5 clientes na fila 2.
Fila 2 atual: [8, 9, 10, 11, 13]

==> Operação: S

Fim da simulação.
```

☞ Não se esqueça de preencher o cabeçalho do código fonte com seus dados, a data de hoje e uma breve descrição do programa.

Após certificar-se de que seu programa esteja correto, envie o arquivo do programa fonte (`p06.py`) através do sistema de entrega do LBI.