

**Licenciatura em Informática de Gestão**

**1º Ano – Pós-Laboral**

**FASE 1 do trabalho de Algoritmos e Estruturas de Dados  
Maio 2018**

**SIMPAR**

**Simulação de passageiros em Partida Aérea**

# **ÍNDICE**

[ÍNDICE 2](#_Toc505105017)

[INTRODUÇÃO 3](#_Toc505105018)

[PROBLEMAS E OBJECTIVOS A ATINGIR 3](#_Toc505105019)

[DESCRIÇÃO DAS ESTRUTURAS DE DADOS 4](#_Toc505105020)

[DESCRIÇÃO SUMÁRIA DO CÓDIGO 5](#_Toc505105023)

[MANUAL DO UTILIZADOR 6](#_Toc505105024)

[CONCLUSÕES 9](#_Toc505105025)

[DIFICULDADES 9](#_Toc505105026)

[METAS ATINGIDAS 9](#_Toc505105027)

[TAREFAS DA EQUIPA Error! Bookmark not defined.](#_Toc505105028)

[APÊNDICES 10](#_Toc505105029)

[CÓDIGO FONTE 10](#_Toc505105030)

# **INTRODUÇÃO**

## **PROBLEMAS E OBJECTIVOS A ATINGIR**

Com o objectivo de resolver o problema colocado neste trabalho tentamos inicialmente colocar em práctica os conhecimentos, conceitos e competências adquiridos no âmbito desta cadeira nas aulas teoricas e práticas.

Aquando da interpretação do enunciado, é imperativo identificar os parâmetros pelos quais este se rege, estabelecendo desde cedo tarefas e compreender o objectivo final da Simulação. Vários destes parâmetros serão definidos aleatoriamente, outros serão estabelecidos pelo utilizador final.

O principal objectivo a que este grupo se propôs foi o de conseguir finalizar a simulçação em código Phyton, através do *interface Spyder*, com todas as funções e correspondendo a todos os requisitos que constam do enunciado.

# **DESCRIÇÃO DAS ESTRUTURAS DE DADOS**

Optámos pela criação de duas classes:

* class **Passageiro**: Descreve um passageiro, número de bagagens, instante em que foi colocado na queue.

* class **Balcao**: Descreve um balcão e a respectiva fila de passageiros.

Nesta classe estão definidas várias funções, para inicializar o balcão, aumular o tempo de atendimento do passageiro. No fim, demonstra o total de passageiros atendidos por balcão, o tempo médio de atendimento e o número médio de bagagens por passageiros.

# **DESCRIÇÃO SUMÁRIA DO CÓDIGO**

O código que consta do nosso projecto,foi elaborado com recurso a Classes, como tal criámos duas Classes – *Passageiro* e *Balcao*.

Tem como objectivo simular o atendimento de passageiros num aeroporto. Começa por solicitar ao utilizador os dados para a simulaçao:

- Numero máximo de passageiros a atender

- Numero máximo e bagagens por passageiro

- Numero de balcões de atendimento

- Numero de cilcos de tempo a executar (enquanto atende e recebe passageiros nas filas dos balcões).

O balcões são gerados, sendo guardado o seu numero e o numero de bagaens que atende por ciclo (feito de forma aleatória).

A colocação de passageiros na fila dos balções, pode ser feita numa fase inicial, consoante escolha do utilizador, utilzando para isso uma determinada percentagem que já estará em fila quando começar o atendimento (como as filas estão vazias, optou-se por colocar lá os passageiros de maneira distribuida uniformemente).

Após este passo iniciamos os clicos de tempo (cada ciclo de tempo percorre todos os balcões, tando para colocar passageiros na fila como para atende-los).

Colocamos passageiros na fila de espera (caso existam), segundo uma lógica de procurar as filas fazias ou com menos pessoas e escolher uma delas.

Uma vez iniciado o atendimento, o passageiro é atendido e retirado da fila, se o seu numero de bagagens for menor ou igual ao numero de bagagens que o balcão, ficando para o(s) ciclo(s) seguintes, se for caso disso.

Se todos os passageiros forem atendidos antes dos ciclos chegarem ao fim, não se excutam mais ciclos, não se aceitam mais passageiros nas filas e fazemos o output das estatisticas pedidas.

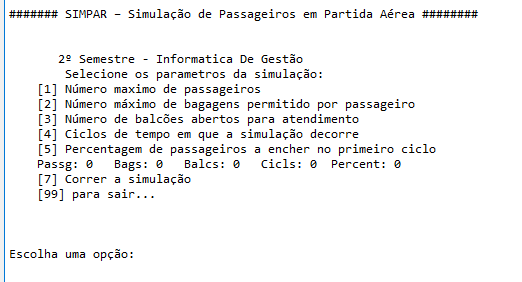
Chegando ao fim dos ciclos previstos, temos de verificar se existe algum balcão, ou balcões, com filas com passageiros por atender. Nesse caso fazemos os ciclos de tempo necessários para poder esvaziar (atender) as mesmas. De notar que não percorremos os balcões à procura de passageiros , e atendemos (isso criaria um numero de ciclos incorrecto), mas percorremos os balcões enquanto existem filas cheias.

No final são apresentados os outputs pedidos, explicados na proxima secção.

# **MANUAL DO UTILIZADOR**

O manual tem como principal objectivo ajudar o utilizador a navegar no programa SIMPAR. A opção **[99]** tem como objectivo sair do menu; e a tecla <ENTER> continuar.

No Menu Inicial de Chegada o utilizador escolhe uma das opções consoante o seu objectivo.



**[1 – 5]** – Cada número corresponde a um input dado pelo utilizador.

*[1] Número maximo de passageiros*

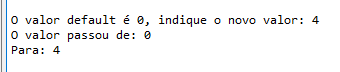
*[2] Número máximo de bagagens permitido por passageiro*

*[3] Número de balcões abertos para atendimento*

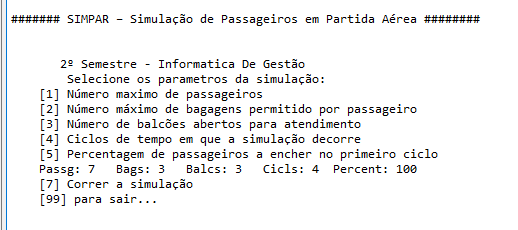
*[4] Ciclos de tempo em que a simulação decorre*

*[5] Percentagem de passageiros a encher no primeiro ciclo*

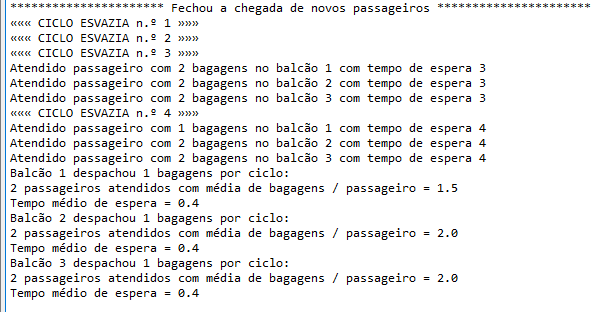
O programa reconhece o 0 (zero), como número pré-definido.



Depois de introduzidos os valores requeridos, o programa vai mostrando os mesmos entre as opções [5] e [7].



**[7] –** Para correr a Simulação com os dados inseridos previamente.



Durante a execução é mostrado:

««« CICLO n.º 1 »»» - Ciclo a ser executado

Balcão 1 tempo 0 : - [b:4 t:0] [b:1 t:0] [b:2 t:0] – A fila de espera do balcão, com b= bagagens do passageiro e t=ciclo em que foi gerado, neste caso 3 passageiros.

Atendido passageiro com 2 bagagens no balcão 2 com tempo de espera 1 – O atendimento realizado com a informação do numero de bagagens e tempo de espera 1 (ciclos de atendimento)

Balcão 1 despachou 1 bagagens por ciclo: - número de balcão e máximo de bagagens que o balcão despacha por ciclo

8 passageiros atendidos com média de bagagens / passageiro = 2.8 – Total de passageiros atendidos e a média de bagagens por passageiro

Tempo médio de espera = 0.8 – O tempo médio de espera

**[99] –** Para sair do programa GAU.



# **CONCLUSÕES**

## **DIFICULDADES**

Desde já uma grande dificuldade que existiu na execução deste trabalho foi conciliar as nossas vidas profissionais e pessoais, com a exigência que este trabalho nos colocou.

A questão dos ciclos foi considerada a mais desafiante. Quando o número de ciclos pré-determinado acaba, mas ainda existirem passageiros por atender causou-nos algum transtorno.

Sentimos algumas dificuldades na interpretação do enunciado e ao decidir o “aspecto final” do programa. Foi tomando diversas formas à medida que fomos progredindo com o mesmo.

## **METAS ATINGIDAS**

Criação de um menu interactivo permitindo ao utilizador final inserir os dados que desejar. Após a inserção dos mesmos, é possivel calcular o tempo de espera por balcão consoante o número de bagagens; calcular o número de bagagens despachada por balcão por ciclo;

# **APÊNDICES**

## **CÓDIGO FONTE**

import math

from pythonds import Queue

from random import randint, choice

from shutil import get\_terminal\_size

# \*\*\*\*\*\*\*\*\*\*\* O código em baixo vai limpar o ecrã de forma a facilitar a leitura \*\*\*\*\*\*\*\*\*\* #

def limpa():

print("\n" \* get\_terminal\_size().lines, end="")

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Menu Inicial de Chegada \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* #

def menu():

limpa()

limpa()

print("\n")

print("####### SIMPAR – Simulação de Passageiros em Partida Aérea ########")

print("\n")

print(""" 2º Semestre - Informática de Gestão

Selecione os parâmetros da simulação:

[1] Número máximo de passageiros

[2] Número máximo de bagagens permitido por passageiro

[3] Número de balcões abertos para atendimento

[4] Ciclos de tempo em que a simulação decorre

[5] Percentagem de passageiros a encher no primeiro ciclo

Passageiros: {} Bagagens: {} Balcões: {} Ciclos: {} Percentagem: {}

[7] Correr a simulação

[99] para saír...""".format(passa, bag, balc, cicl, pench))

class Passageiro:

"""

Descreve um passageiro

"""

def \_\_init\_\_(self, bag\_pass, ciclo\_in):

"""

Inicializa um passageiro

:param bag\_pass: número de bagagens do passageiro

:param ciclo\_in: instante em que foi colocado na fila (número do ciclo da simulação)

"""

self.bag\_pass = bag\_pass

self.ciclo\_in = ciclo\_in

# self.atendidos = 0

def obtem\_bag\_pass(self):

"""

Devolve o valor de bag\_pass

:return: bag\_pass

"""

return self.bag\_pass

def obtem\_ciclo\_in(self):

"""

devolve o valor de ciclo\_in

:return: ciclo\_in

"""

return self.ciclo\_in

# def incr\_atendidos(self):

# """

# Incrementa em 1 o passt\_atend - total de passageiros atendidos

# :return: None

# """

#

# self.atendidos += 1

def \_\_str\_\_(self):

"""

Retorna o passageiro como uma string legivel para o utilizador

Output esperado:

[b:4 t:2]

:return: string

"""

return "[b:{} t:{}]".format(self.obtem\_bag\_pass(), self.obtem\_ciclo\_in())

class Balcao:

"""

Descreve um balcão e a respectiva fila de passageiros

"""

def \_\_init\_\_(self, n\_balcao, num\_bag):

"""

Inicializa um balcão com o número indicado

:param n\_balcao: número do balcão

:param num\_bag: o número máximo de bagagens permitido por passageiro

"""

self.n\_balcao = n\_balcao

self.fila = Queue()

self.inic\_atend = 0

self.passt\_atend = 0

self.numt\_bag = 0

self.tempt\_esp = 0

self.bag\_utemp = randint(1, num\_bag)

def obtem\_n\_balcao(self):

"""

Devolve o valor de n\_balcao

:return: n\_balcao

"""

return self.n\_balcao

def obtem\_fila(self):

"""

Devolve o valor da fila

:return: fila

"""

return self.fila

def muda\_inic\_atend(self, tempo\_atendimento):

"""

Acumula em inic\_atend o “valor” do tempo de atendimento do passageiro

:param tempo\_atendimento: tempo de atendimento

:return: None

"""

self.inic\_atend = tempo\_atendimento

def incr\_passt\_atend(self):

"""

Incrementa em 1 o passt\_atend - total de passageiros atendidos por este balcão

:return: None

"""

self.passt\_atend += 1

def muda\_numt\_bag(self, passageiro):

"""

Acumula em numt\_bag do balcão, o bag\_pass do passageiro quando este termina de ser atendido

:param passageiro: passageiro processado

:return: None

"""

self.numt\_bag += passageiro.obtem\_bag\_pass()

def muda\_tempt\_esp(self, tempo\_espera):

"""

Acumula em tempt\_esp o “t” tempo de espera do passageiro

:param tempo\_espera: Tempo de espera

:return: None

"""

self.tempt\_esp += tempo\_espera

def \_\_str\_\_(self):

"""

Retorna o balcão como uma string legível para o utilizador

Output esperado:

Quando tem passageiros na fila:

Balcão 2 tempo 2 : - [b:4 t:1] [b:2 t:2] -

Quando não tem passageiros na fila:

Balcão 0 tempo 1 : -

:return: string

"""

# Formata a lista de passageiros consoante as especificações

if self.fila.isEmpty():

str\_pass = "-"

else:

passageiros\_como\_str = [str(passageiro) for passageiro in self.fila.items]

str\_pass = "- {} - ".format(" ".join(passageiros\_como\_str))

return "Balcão {} tempo {} : {}".format(self.obtem\_n\_balcao(), self.tempt\_esp, str\_pass)

def mostra\_balcoes(balcoes):

"""

Mostra os detalhes dos balcoes

:param balcoes: Lista de balcões

:return: None

"""

for balcao in balcoes:

print(str(balcao))

def atende\_passageiros(tempo, balcoes):

"""

Atende passageiros nos balcões indicados

:param tempo: Ciclo de simulação

:param balcoes: Lista de balcões

:return: Passageiros colocados em fila

"""

atendidos = 0

for b in balcoes:

if b.obtem\_fila().isEmpty():

# Sem passageiros a processar

print('BALCÃO ' + str(b) + ' sem passageiros a processar')

b.muda\_inic\_atend(tempo)

continue

fila = b.obtem\_fila()

p = fila.items[-1] # Para ser Fifo, tem de ser desta forma porque Queue.enqueue() acrescenta no inicio da lista

tempo\_atendimento = tempo + b.inic\_atend

ut\_bag = math.ceil5(p.bag\_pass / b.bag\_utemp)

if ut\_bag < tempo\_atendimento:

tempo\_de\_espera = tempo - p.ciclo\_in

print("Atendido passageiro com {} bagagens no balcão {} com tempo de espera {}".format(

p.bag\_pass,

b.obtem\_n\_balcao(),

tempo\_de\_espera

)

)

b.muda\_inic\_atend(tempo + 1)

b.incr\_passt\_atend()

b.muda\_numt\_bag(p)

b.muda\_tempt\_esp(tempo\_de\_espera)

fila.items.remove(p)

atendidos += 1

return atendidos

def apresenta\_resultados(balcoes):

"""

Apresenta os resultados estatísticos finais

:param balcoes: Lista de balcões

:return: None

"""

for i in balcoes:

if i.passt\_atend > 0:

print("Balcão {} despachou {} bagagens por ciclo:".format(i.obtem\_n\_balcao(), i.bag\_utemp))

print(

"{} passageiros atendidos com média de bagagens / passageiro = {}".format(

i.passt\_atend,

round(i.numt\_bag / i.passt\_atend, 1)

)

)

print("Tempo médio de espera = {}".format(round(i.passt\_atend / i.inic\_atend, 1)))

else:

print("Balcão {} não atendeu passageiros".format(i.obtem\_n\_balcao()))

def simpar\_simula(num\_pass, num\_bag, num\_balcoes, ciclos, p\_enche):

"""

Corre uma simulação

:param num\_pass: o número de passageiros com bagagem previsto para este voo

:param num\_bag: o número máximo de bagagens permitido por passageiro

:param num\_balcoes: o número de balcões abertos para atendimento e despacho de bagagem

:param ciclos: os ciclos de tempo em que a simulação decorre.

:param p\_enche: % de passageiros a encher de arranque

:return: None

"""

atendidos = 0

total = num\_pass

balcoes = []

terco = ciclos / 3

for n\_balcao in range(1, num\_balcoes + 1): # gera balcões

balcoes.append(Balcao(n\_balcao, num\_bag))

# passageiros iniciais

enche = int((num\_pass \* p\_enche) / 100)

for i in range(0, enche):

for j in balcoes:

j.obtem\_fila().enqueue(Passageiro(randint(1, num\_bag), 0)) # aqui tempo é 0

num\_pass -= 1

# mostra\_balcoes(balcoes)

# Ocupar das filas

for ciclo in range(0, ciclos):

# Verifica se temos passageiros para criar

if num\_pass > 0:

for n\_balcao in range(1, num\_balcoes + 1): # aqui precorremos todos os balcões para colocar pessoas na fila

# Calcula a probabilidade de acrescentar passageiro

if ciclo <= terco:

probabilidade = 100

elif ciclo <= terco \* 2:

probabilidade = 80

else:

probabilidade = 60

temp = randint(0, 100)

#print('Terço ' + str(terco)+ ' Probabilidade '+str(probabilidade) +' temp '+ str(temp)) #só para perceber como está a funcionar a probabilidade

if probabilidade >= temp:

# Obtem tamanho da fila com menos passageiros

fila\_mais\_curta = min([balcao.obtem\_fila().size() for balcao in balcoes])

# Obtem apenas os balcões com o tamanha de fila mais curto

# (podem por exemplo existir vários balcões com 0 passageiros)

balcoes\_filas\_curtas = [balcao for balcao in balcoes if balcao.obtem\_fila().size() == fila\_mais\_curta]

# E escolhemos de forma aleatória qual usamos

balcao\_pretendido = choice(balcoes\_filas\_curtas)

# Cria passageiro

balcao\_pretendido.obtem\_fila().enqueue(Passageiro(randint(1, num\_bag), ciclo + 1))

num\_pass -= 1

#print('criei um passageiro no b ' + str (balcao\_pretendido)) #este print é de controle

print("««« CICLO n.º {} »»»".format(ciclo + 1))

atendidos = atendidos + atende\_passageiros(ciclo + 1, balcoes)

mostra\_balcoes(balcoes)

if atendidos >= total:

break

print('ATENDIDOS ' + str(atendidos) + ' total ' + str(total))

# Esvazear das filas

print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Fechou a chegada de novos passageiros \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')

conta = 0

#esvazia\_ciclo = 0

ciclo += 1 # novo ciclo

esvazia = True

while esvazia == True:

for balcao in balcoes: # vamos aos balcões ver se há filas de espera

# print('BALCOES '+str(balcao)+ 'estado da fila'+ str(balcao.obtem\_fila().isEmpty()))

if balcao.obtem\_fila().isEmpty() == False: # se a fila não estiver vazia

conta=conta+1 # conta é incrementado

if conta == 0: # Se não há filas cheias, sai

esvazia = False

else:

# esvazia\_ciclo += 1

ciclo += 1 # novo ciclo

print("««« CICLO ESVAZIA n.º {} »»»".format(ciclo ))

atende\_passageiros(ciclo, balcoes)

conta = 0 # Volta a zero para controlar o próximo ciclo

# Só para verificar

# for balcao in balcoes:

# print('BALCOES'+str(balcao)+ 'estado da fila'+ str(balcao.obtem\_fila().isEmpty()))

apresenta\_resultados(balcoes)

if \_\_name\_\_ == "\_\_main\_\_":

passa = 70

bag = 4

balc = 4

cicl = 10

pench = randint(0,100)

invalid = False # Inicialização da variável de verificação de erro na Escolha

while True:

menu() # Chamada do Menu

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False # Limpar a variável

try:

print("\n")

escolha = int(input("Escolha uma opção: "))

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

if escolha == 1:

limpa()

limpa()

# while True:

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False

try:

a = passa

aux = int(input("O valor default é " + str(a) + ", indique o novo valor: ")) # display do valor antigo

print("O valor passou de: " + str(a))

print("Para: " + str(aux))

if aux != a: # se diferente substitui

passa = aux

input()

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

elif escolha == 2:

limpa()

limpa()

# while True:

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False

try:

a = bag

aux = int(

input("O valor default é " + str(a) + ", indique o novo valor: ")) # display do valor antigo

print("O valor passou de: " + str(a))

print("Para: " + str(aux))

if aux != a: # se diferente substitui

bag = aux

input()

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

elif escolha == 3:

limpa()

limpa()

# while True:

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False

try:

a = balc

aux = int(

input("O valor default é " + str(a) + ", indique o novo valor: ")) # display do valor antigo

print("O valor passou de: " + str(a))

print("Para: " + str(aux))

if aux != a: # se diferente substitui

balc = aux

input()

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

elif escolha == 4:

limpa()

limpa()

# while True:

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False

try:

a = cicl

aux = int(

input("O valor default é " + str(a) + ", indique o novo valor: ")) # display do valor antigo

print("O valor passou de: " + str(a))

print("Para: " + str(aux))

if aux != a: # se diferente substitui

cicl = aux

input()

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

elif escolha == 5:

limpa()

limpa()

# while True:

if invalid: # Verificação se o utilizador escolheu uma opção incorrecta

print('A opção não é válida')

invalid = False

try:

a = pench

aux = int(

input("O valor default é " + str(a) + ", indique o novo valor: ")) # display do valor antigo

print("O valor passou de: " + str(a))

print("Para: " + str(aux))

if aux != a: # se diferente substitui

pench = aux

input()

except ValueError: # Se o Valor não for um inteiro estamos em estado de erro e tentamos novamente

invalid = True

continue # Volta ao início do ciclo While

elif escolha == 7:

simpar\_simula(passa, bag, balc, cicl, pench)

elif escolha == 99:

limpa()

limpa()

print("...adeus :( ")

break # Finalizar o programa

else:

invalid = True

continue # Volta ao início do ciclo While

input('Prima <ENTER> para continuar . . .')