

# Fonaments de Programació

## PAC9 - 20182

Data límit de lliurament: 20/05/2019

Estudiant

Cognoms: **GALEANO SANCHO**

Nom: **DANIEL**

### Objectius

- Comprendre que són els tipus abstractes de dades (TAD) i saber definir-los correctament.
- Saber identificar el TAD més adient per a cada problema i justificar-ne la seva elecció.

### Format i data de lliurament

Cal lliurar la PAC abans del dia **20 de maig de 2019 a les 23:59**.

Per al lliurament caldrà que entregueu un fitxer que contingui:

- Aquest document amb la resposta dels diferents exercicis plantejats.

Cal fer el lliurament a l'apartat de lliuraments d'AC de l'aula de teoria.

# Enunciat

Seguint amb l'ajuda que proporcionem a UOCAirways, la companyia ens ha demanat la nostra col·laboració. Ara, per a realitzar una anàlisi respecte els tipus de dades més adients per a gestionar aspectes de la compra de bitllets, així com de l'embarcament dels passatgers.

Per això, en aquesta PAC, ens centrarem en els aspectes d'anàlisi dels problemes plantejats, deixant de banda els aspectes de codificació i implementació. Per aquest motiu, cal respondre els exercicis d'aquesta PAC només en llenguatge algorísmic.

## Exercici 1: (20%)

L'aerolinia ofereix a la seva pàgina web l'opció de comprar bitllets dels seus vols. S'ofereixen bitllets per un vol durant uns dies fins que s'esgota el termini establert per la companyia o, també, fins que s'assoleix un determinat % de la capacitat de l'avió. Aquest % acostuma a ser un valor per sobre de 100, és a dir, que la companyia pot arribar a posar a la venda més bitllets del compte (*overbooking*). Aquesta és una pràctica comercial habitual que serveix per intentar garantir un alt % d'ocupació real del vol i assegurar així la rendibilitat. Hi ha també la possibilitat de que un passatger pugui cancel·lar el seu vol, i per tant, deixaria el seu lloc lliure. Sense necessitat de dissenyar la solució, indiqueu quin TAD proposaríeu per anar registrant les vendes de bitllets que la companyia fa d'un determinat vol. Justifiqueu la vostra resposta en base a les operacions de compra i cancel·lació.

Podríem utilitzar el TAD pila (*stack*) però, quan hi hagués una cancel·lació, hauríem d'utilitzar una pila auxiliar per tal d'identificar la persona que ha realitzat la cancel·lació. Per tant, és més intel·ligent utilitzar el TAD llista (*list*).

Quan un passatger realitza una compra, s'utilitzarà la funció *insert*. Com que en un TAD llista, es pot afegir un element a qualsevol posició de la llista, podríem afegir-lo de manera que la llista estigués ordenada per algun ordre en particular, per exemple, el NIF.

Per la cancel·lació hauríem de cercar el passatger dins de la llista. Seria útil realitzar aquesta cerca per NIF. Amb la funció *get*, obtindríem la informació del passatger i el *index* on es troba a la llista i, amb la funció *delete*, esborraríem el registre del passatger.

El nombre d'elements de la llista, *nelem*, correspon al nombre de vendes de la companyia en aquest vol. Si volguéssim les vendes totals, hauríem d'implementar un comptador dins de la funció *delete*, per tal d'anar sumant una unitat cada cop que es crida a aquesta funció.

## Exercici 2: (20%)

Quan s'acosta el dia del vol, s'obre la possibilitat de fer un procés de *check-in* a tots aquells viatgers que han comprat el bitllet. Aquest *check-in* és el que realment els garanteix un seient a l'avió. El resultat d'un *check-in* és l'emissió d'una targeta d'embarcament, que és el document que permet accedir dins de l'avió i on s'indica l'hora exacta i la porta d'embarcament assignades per al vol, així com la fila i el seient assignats. Ens interessa definir una estructura de dades, no necessàriament un TAD, per a relacionar, individualment, un passatger amb la seva targeta d'embarcament. Si anomenem *tCheckin* a aquesta estructura de dades, indica quina elecció faries per a la mateixa i quina informació creus que hauria de contenir.

Per emmagatzemar aquesta informació, faria servir una taula. Aquesta taula té tants elements com targetes d'embarcament diferents (*numBoardingPass*) fins a un màxim (*MAX\_BOARDING\_PASS*) donat pel màxim nombre de targetes d'embarcament permeses per aquest vol (tenint en compte el *overbooking* també).

Els elements seran de tipus *tBoardingPass* que és qui conté la informació important del vol:

- *id*: nombre enter que identifica de manera unívoca a la targeta d'embarcament. Podria ser el NIF o un número incremental, per exemple.
- *timeFlight*: nombre enter amb format HHMM, on HH és la hora local en format 24h i MM són els minuts.
- *gate*: cadena de caràcters per indicar la porta d'embarcament, típicament una lletra i un número, per exemple, A22.
- *row*: nombre enter que indica el número de la fila on està localitzat el seient.
- *seat*: caràcter que indica de quin seient es tracta, per exemple, F.

La definició del tipus d'estructura de dades quedaria de la següent manera:

**type**

**tCheckin = record**

    boardingPass: **vector**[MAX\_BOARDING\_PASS] **of** tBoardingPass;

    numBoardingPass: **integer**;

**end record**

**tBoardingPass = record**

    id: **integer**;

    timeFlight: **integer**;

    gate: **string**;

    row: **integer**;

    seat: **char**;

**end record**

**end type**

### Exercici 3: (25%)

Els usuaris que tenen una targeta d'embarcament amb un seient preferent tenen assignat un seient de les primeres files de l'avió. Seran els que tinguin un seient assignat entre les files 1 a 10. Se'ls donarà, a més, preferència a l'hora d'entrar i sortir de l'avió. La resta de passatgers amb targeta d'embarcament que no té seient preferent embarcaran per la porta del davant o del darrere de l'avió, atenent al seu número de seient: els usuaris que tenen targetes d'embarcament amb files compreses entre la 11 i la 20 embarquen per davant, mentre que els usuaris de les files compreses entre la 21 i la 40 embarquen per la porta del darrera. Els primers a pujar a l'avió són els passatgers preferents. Posteriorment, es dona pas simultani a la resta de passatgers perquè ocupen posicions diferents de l'avió i no es produirà cap col·lapse pel fet d'embarcar alhora.

Per últim, quan un passatger fa el *check-in* i es troba que ja s'han emès totes les targetes d'embarcament corresponent a tots els seients de l'avió, se li dona una targeta d'embarcament provisional on hi figura l'hora i la porta d'embarcament però sense fila ni seient assignats. D'aquesta forma, es pot dirigir a la porta d'embarcament i esperar que entri tothom. Si un cop es tanca l'embarcament hi ha encara seients lliures, els passatgers d'*overbooking* (els que tenen una targeta d'embarcament provisional) poden entrar a l'avió en l'ordre en què han recollit la seva targeta provisional.

Expliqueu com representaries les fileres d'embarcament segons el que s'ha exposat i trieu el TAD que millor modelitza el comportament desitjat per la companyia. Justifiqueu la resposta.

Volem representar fileres d'embarcament, que són fàcilment representables per el TAD cua (*queue*).

Concretament, necessitem quatre cues. Una de passatgers preferents (*preferentQueue*), que seuen de la fila 1 a la 10. Dos de passatgers no preferents (*noPreferentQueue20* i *noPreferentQueue40*), que seuen de la fila 11 a la 20 i de la fila 21 a la 40, respectivament. I, finalment, una última per passatgers amb targetes d'embarcament provisional (*overbookingQueue*). Aquesta definició la codifiquem de la següent manera;

```
preferentQueue      = queue(tBoardingPass);  
noPreferentQueue20 = queue(tBoardingPass);  
noPreferentQueue40 = queue(tBoardingPass);  
overbookingQueue   = queue(tBoardingPass);
```

Cada cop que arriba un passatger del vol a la porta d'embarcament, hem de comprovar la fila de la seva targeta d'embarcament i utilitzar la funció *enqueue* a la fila que pertorqui. Aquesta comprovació, si tenim en compte que la variable *checkin* és de tipus *tCheckin* i la variable *i* és de tipus *integer*, es pot codificar de la següent manera:

```

if checkin.boardingPass[i].row ≥ 1 and
  checkin.boardingPass[i].row ≤ 10 then
    enqueue(preferentQueue, checkin.boardingPass[i]);
else
  if checkin.boardingPass[i].row ≥ 11 and
    checkin.boardingPass[i].row ≤ 20 then
      enqueue(noPreferentQueue20, checkin.boardingPass[i]);
  else
    if checkin.boardingPass[i].row ≥ 21 and
      checkin.boardingPass[i].row ≤ 40 then
        enqueue(noPreferentQueue40, checkin.boardingPass[i]);
    else
      enqueue(overbookingQueue, checkin.boardingPass[i]);
    end if
  end if
end if

```

Per últim, mostrem el comportament desitjat per la companyia en qüestió de com es buiden les cues. Tenint en compte que *now* és una variable que ens indica la hora actual en el format HHMM, que *FULL\_FLIGHT* és una constant igual al número màxim de passatgers que poden entrar a l'avió i que *passengers* és un comptador per tenir en compte quants passatgers han entrat a l'avió, ho codifiquem amb comentaris explicatius de la següent manera:

```

if now ≤ checkin.boardingPass[i].time then
  {if current time is before closing gates time, we have to empty queue with following conditions}

  if not emptyQueue(preferentQueue) then
    {while there is someone in the preferent queue, it is emptied before the others}

    dequeue(preferentQueue);
    passengers++;

  else
    {we empty both no-preferent queues because passengers enter through different doors}

    if not emptyQueue(noPreferentQueue20) then
      dequeue(noPreferentQueue20);
      passengers++;
    end if

    if not emptyQueue(noPreferentQueue40) then
      dequeue(noPreferentQueue40);
      passengers++;
    end if

  end if

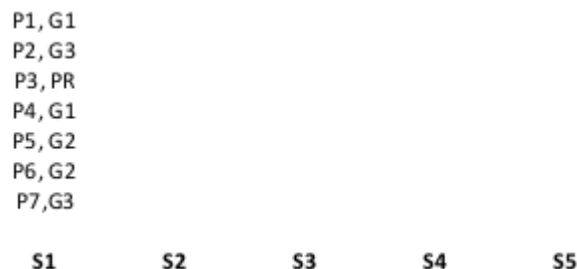
else
  {if current time is after closing gates time, it is time to empty overbooking queue, only if there
  is seats left in the flight}
  if passengers < FULL_FLIGHT then
    dequeue(overbookingQueue);
    passengers++;
  end if
end if

```

## Exercici 4: (35%)

Per tal de veure el que suposaria gestionar els passatgers que han fet el *check-in* per un vol mitjançant una estructura TAD de tipus pila, disposem del tipus pila de passatgers representat per *tStackCheckin* on cada passatger està representat pel tipus *tCheckin*. En aquest moment, no ens importa tant el contingut d'aquest tipus sinó el fet que conté la informació necessària de cada passatger. Entre aquesta informació, hi ha el grup d'embarcament, de forma simplificada, indicant la identificació del passatger (Px) i el grup d'embarcament: PR, G1, G2 i G3. Aquí PR representa el grup d'embarcament preferent; G1 representa el grup d'embarcament de les files compreses entre la 11 i la 20, que embarquen per davant; G2 representa el grup d'embarcament de les files compreses entre la 21 i la 40, que embarquen per darrera; G3 representa el grup d'embarcament en situació d'overbooking.

Disposem de 5 piles (Sx), on S1 conté tot el conjunt de passatgers i cada una de les altres piles (S2 a S5) està pensada per emmagatzemar els passatgers de cada un dels grups d'embarcament. Inicialment la disposició i contingut d'aquestes piles es tal com es mostra a la figura:



Associades al tipus *tStackCheckin* tenim les instruccions *push*, *top* i *pop* definides com:

```
action push(inout s: tStackCheckin, in p:tCheckin)
action pop(inout s: tStackCheckin)
function top(s: tStackCheckin):tCheckin
```

**Nota:** Per exemple, la següent seqüència d'instruccions mou el passatger de l'inici de la pila inicial (S1) i el situa com a primer lloc de la pila S4:

```
push(S4, top(S1));
pop(S1);
```



**Apartat 4.a (10%):** Dibuixeu la posició final de les piles de passatgers després de moure els passatgers executant el següent conjunt d'instruccions:

```
push(S3, top(S1));
pop(S1);
push(S5, top(S1));
pop(S1);
push(S2, top(S1));
pop(S1);
push(S3, top(S1));
pop(S1);
push(S4, top(S1));
pop(S1);
push(S4, top(S1));
pop(S1);
push(S5, top(S1));
pop(S1);
```

		P3, PR	P4, G1 P1, G1	P6, G2 P5, G2	P7, G3 P2, G3
<b>S1</b>	<b>S2</b>	<b>S3</b>	<b>S4</b>	<b>S5</b>	

**Apartat 4.b (25%):** Un cop reubicats els passatgers d'acord a la seva prioritat d'embarcament, resulta que degut a la condició de passatger freqüent dels passatgers P1 i P2, el primer ha estat millorat a passatger preferent, mentre que el segon ha estat millorat a passatger del grup G1. Es demana que escriviu el conjunt d'instruccions necessari per a generar la seqüència de moviments que ubica el passatger P1 a la pila S2 i el passatger P2 a la pila S3.

```
push(S1,top(S3));
pop(S3);
push(S2,top(S3));
pop(S3);
push(S1,top(S5));
pop(S5);
push(S3;top(S5));
pop(S5);
push(S5;top(S1));
pop(S1);
push(S3;top(S1));
pop(S1);
```

El resultat d'executar aquestes instruccions és, precisament, moure a la P1 al grup S2 i moure a la P2 al S3. Hem utilitzat la pila S1 com a pila auxiliar.

	P1, G1 P3, PR	P4, G1 P2, G3	P6, G2 P5, G2	P7, G3
<b>S1</b>	<b>S2</b>	<b>S3</b>	<b>S4</b>	<b>S5</b>

## Criteris de correcció:

- Que es segueixi la notació algorísmica utilitzada a l'assignatura. Vegeu document *Nomenclator* a la xWiki de contingut.
- Que es segueixin les instruccions donades i es justifiqui de manera adient la solució plantejada.