

# Fonaments de Programació

## PAC8 - 20182

Data límit de lliurament: 29/04/2019

Estudiant

Cognoms: **GALEANO SANCHO**

Nom: **DANIEL**

### Objectius

- Comprendre el tipus de dades taula i saber definir-lo correctament.
- Implementar recorreguts i cerques dins de les taules.
- Aprofundir en la modularització del codi utilitzant accions i funcions.
- Aprofundir en l'ús de paràmetres d'entrada, paràmetres de sortida i paràmetres d'entrada/sortida.

### Format i data de lliurament

Cal lliurar la PAC abans del dia **29 d'abril de 2019 a les 23:59**.

Per al lliurament caldrà que entregueu un fitxer en format ZIP, que contingui:

- Aquest document amb la resposta de l'exercici 1 i l'apartat b de l'exercici 2.
- El workspace de Codelite que contingui **el projecte i totes les carpetes creades** a l'exercici 2a.

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 per crear un programa que els ajudi a gestionar els seus avions. En aquesta PAC, treballarem amb el tipus de dades taula, conjuntament amb l'entrada i sortida interactiva. Disposem del següent algorisme, en llenguatge algorísmic, a mig dissenyar:

**type**

tPlaneUtility = {COMMERCIAL, PRIVATE, GOVERNMENTAL, MILITAR, EXPERIMENTAL, OTHERS}

tPlane = **record**

id: **integer**;

model: **string**;

year: **integer**;

utility: tUtility;

percentOccupationFirstclass: **real**;

percentOccupationBusiness: **real**;

percentOccupationTulist: **real**;

isActive:**boolean**;

**end record**

**end type**

**algorithm** UOCAirways

**const**

MAX\_PLANES: **integer** = 100;

CODE\_FIRSTCLASS: **integer** = 1;

CODE\_BUSINESS: **integer** = 2;

CODE\_TURIST: **integer** = 3;

**end const**

{... algorithm to complete ...}

**end algorithm**

**Nota:** En aquest exercici de llenguatge algorísmic podeu utilitzar les accions *planeRead*, *planeWrite* i *planeCopy* sense necessitat de dissenyar.

## Exercici 1: Disseny en llenguatge algorísmic (50%)

**Apartat a: [10%]** Definiu el tipus de dades *tPlanesTable*, en forma de taula, per desar les dades de diversos avions. Les dades de cada avió es guarden dins dels camps del tipus estructurat *tPlane*. El nombre màxim d'avions queda fixat per la constant *MAX\_PLANES*.

```
type
  tPlanesTable = record
    plane: vector[MAX_PLANES] of tPlane;
    numPlanes: integer;
  end record
end type
```

**Apartat b: [10%]** Definiu l'acció *planesTableInitialize* que rep com a paràmetre una taula de tipus *tPlanesTable* i n'inicialitza el número d'elements a zero. Per fer el que demana l'acció tria com ha de ser el paràmetre (d'entrada, de sortida o d'entrada/sortida).

```
action planesTableInitialize (out planeTable: tPlanesTable)
  planeTable.numPlanes := 0;
end action
```

**Apartat c: [30%]** Definiu la funció *planesTableOccupation* que rep, com a paràmetres, una taula de tipus *tPlanesTable* i un enter *idClass*. Aquest paràmetre *idClass* pot prendre els valors *CODE\_FIRSTCLASS*, *CODE\_BUSINESS* o *CODE\_TURIST* i representa, respectivament, quin dels camps *percentOccupationFirstclass*, *percentOccupationBusiness* o *percentOccupationTurist* del tipus *tPlane* hem de considerar. La funció ha de retornar un real amb l'ocupació mitjana de la classe indicada per *idClass* dels avions de la taula. Per exemple, si *idClass* pren el valor *CODE\_FIRSTCLASS* i

```
plane1.percentOccupationFirstclass := 100.0;
plane2.percentOccupationFirstclass := 50.0;
plane3.percentOccupationFirstclass := 0.0;
```

la funció retornarà el valor 50.0

```
function planesTableOccupation (planeTable: tPlanesTable, idClass: integer): real
  var
    aux: real;
    i: integer;
  end var

  aux := 0;
```

```

for i := 1 to planeTable.numPlanes do
  if idClass = CODE_FIRSTCLASS then
    aux := aux + planeTable.plane[i].percentOccupationFirstclass;
  else
    if idClass = CODE_BUSINESS then
      aux := aux + planeTable.plane[i].percentOccupationBusiness;
    else
      if idClass = CODE_TURIST then
        aux := aux + planeTable.plane[i].percentOccupationTurist;
      end if
    end if
  end if
end if
end for

aux := aux/planeTable.numPlanes;

return aux;

end function

```

**Apartat d: [30%]** Definiu l'acció *planesTableSelectByOccupation*, que rep els paràmetres següents:

- *planesTable* és un paràmetre d'entrada del tipus *tPlanesTable*.
- *average* és un paràmetre d'entrada de tipus real.
- *idClass* és un paràmetre d'entrada de tipus enter.
- *selectTable* és un paràmetre de sortida del tipus *tPlanesTable*.

*idClass* prendrà valors *CODE\_FIRSTCLASS*, *CODE\_BUSINESS* o *CODE\_TURIST* per representar, respectivament, els camps *percentOccupationFirstclass*, *percentOccupationBusiness* o *percentOccupationTurist* del tipus *tPlane*. L'acció analitza cada un dels avions de *planesTable* i comprova si el camp representat per *idClass* té un valor d'ocupació inferior al valor de mitjana (que ha entrat a través de *average*). En cas afirmatiu, desplaça l'avió a la taula de sortida *selectTable*. Amb els avions que tinguin un valor d'ocupació igual o superior al valor d'*average* no cal fer res.

**Nota:** Recordeu que cal inicialitzar la taula de sortida fent servir l'acció *planesTableInitialize* de l'apartat b.

```

action planesTableSelectByOccupation(in planesTable: tPlanesTable, in average: real, in
idClass: integer, out selectTable: tPlanesTable)
  var
    i: integer;
  end var

  planesTableInitialize(selectTable);

```

```

for i := 1 to planesTable.numPlanes do
  if idClass = CODE_FIRSTCLASS then
    if planesTable.plane[i].percentOccupationFirstclass < average then
      selectTable.numPlanes := selectTable.numPlanes + 1;
      planeCopy(planesTable.plane[i], selectTable.plane[selectTable.numPlanes]);
    end if
  else
    if idClass = CODE_BUSINESS then
      if planesTable.plane[i].percentOccupationBusiness < average then
        selectTable.numPlanes := selectTable.numPlanes + 1;
        planeCopy(planesTable.plane[i], selectTable.plane[selectTable.numPlanes]);
      end if
    else
      if idClass = CODE_TURIST then
        if planesTable.plane[i].percentOccupationTurist < average then
          selectTable.numPlanes := selectTable.numPlanes + 1;
          planeCopy(planesTable.plane[i], selectTable.plane[selectTable.numPlanes]);
        end if
      end if
    end if
  end if
end for
end action

```

Apartat e: [20%] Completeu l'algorisme que tenim a mig dissenyar fent el següent:

- Declareu dues taules d'avions *planesTable* i *selectTable* de tipus *tPlanesTable*
- Inicialitzeu les taules d'avions a zero.
- Sol·liciteu a l'usuari que introdueixi un nombre d'avions a registrar.
- Llegiu les dades de tots els avions a registrar, guardeu-los dins de la taula *planesTable* i actualitzeu el número d'avions d'aquesta taula.
- Sol·liciteu a l'usuari que introdueixi el valor de *idClass* que ha de servir per determinar per a quina classe es farà el càlcul de la mitjana d'ocupació. Comproveu que el valor d'*idClass* és correcte, és a dir, que presenta els valors *CODE\_FIRSTCLASS*, *CODE\_BUSINESS* o *CODE\_TURIST*. Si no és així, mostreu un missatge per pantalla avisant de l'error i torneu a demanar un nou valor, fins que sigui correcte.
- Calculeu la mitjana a través de la funció *planesTableOccupation*.
- Analitzeu i processeu els avions de la taula *planesTable* fent servir l'acció *planesTableSelectByOccupation* per tenir els elements seleccionats a la taula *selectTable*.
- Mostreu per pantalla el nombre d'avions que hi ha a la taula *selectTable*. En cas que no estigui buida, mostreu els camps dels avions que conté.

Per exemple, suposem que tenim la taula *planesTable* amb tres avions que tenen identificador i percentatges d'ocupació següents:

```
plane1.id=9;
...
plane1.percentOccupationFirstclass := 100.0;
plane1.percentOccupationBusiness:= 100.0;
plane1.percentOccupationTulist:= 90.0;
...
plane2.id=23;
...
plane2.percentOccupationFirstclass := 50.0;
plane2.percentOccupationBusiness:= 80.0;
plane2.percentOccupationTulist:= 90.0;
...
plane3.id=51;
...
plane3.percentOccupationFirstclass := 0.0;
plane3.percentOccupationBusiness:= 90.0;
plane3.percentOccupationTulist:= 90.0;
...
```

Si fem l'anàlisi de la primera classe, és a dir, *idClass* igual a *CODE\_FIRSTCLASS*, la taula *selectTable* contindria només l'avió amb identificador 51, ja que la mitjana de *percentOccupationFirstclass* és 50% i l'avió amb identificador 51 té un valor de *percentOccupationFirstclass* inferior (igual al 0%).

Si fem l'anàlisi de la classe *business*, és a dir, *idClass* igual a *CODE\_BUSINESS*, la taula *selectTable* contindria només l'avió amb identificador 23, ja que la mitjana de *percentOccupationBusiness* és 90% i l'avió amb identificador 23 té un valor de *percentOccupationBusiness* inferior (igual al 80%).

Finalment, si fem l'anàlisi de la classe *tulist*, és a dir, *idClass* igual a *CODE\_TURIST*, la taula *selectTable* no contindria cap avió ja que la mitjana de *percentOccupationTulist* és 90% i no hi cap avió amb ocupació inferior.

#### **type**

```
tPlaneUtility = {COMMERCIAL, PRIVATE, GOVERNMENTAL, MILITAR,
EXPERIMENTAL, OTHERS}
tPlane = record
id: integer;
model: string;
year: integer;
utility: tUtility;
percentOccupationFirstclass: real;
percentOccupationBusiness: real;
percentOccupationTulist: real;
isActive: boolean;
end record
```

**end type**

**algorithm** UOCAirways

**const**

MAX\_PLANES: **integer** = 100;  
CODE\_FIRSTCLASS: **integer** = 1;  
CODE\_BUSINESS: **integer** = 2;  
CODE\_TURIST: **integer** = 3;

**end const**

**var**

i, id: **integer**;  
planesTable, selectTable: tPlanesTable;  
avg: **real**;  
correct: **boolean**;

**end var**

correct := **false**;

planesTableInitialize(planesTable);

planesTableInitialize(selectTable);

writeString("Number of planes to enter: >>");

planesTable.numPlanes := readInteger();

**for** i := 1 **to** planesTable.numPlanes **do**

writeString("Plane number: ");

writeInteger(i);

planeRead(planesTable.plane[i]);

**end for**

**while not** correct **do**

writeString("Enter code of the class (1 for First Class, 2 for Business, 3 for Turist): >>");

id := readInteger();

**if** id = 1 **or** id = 2 **or** id = 3 **then**

found := **true**;

**else**

writeString("Incorrect code. Try again.");

**end if**

**end while**

avg := planesTableOccupation(planesTable, id);

planesTableSelectByOccupation(planesTable, avg, id, selectTable);

writeString("The number of planes below average is: " );

writeInteger(selectTable.numPlanes);

**if** selectTable.numPlanes ≠ 0 **then**

**for** i := 1 **to** selectTable.numPlanes **do**

planeWrite(selectTable.plane[i]);

**end for**

**end if**

**end algorithm**

## Exercici 2: Programació en C (50%)

Apartat a: [70%] En aquest exercici cal codificar en C l'exercici 1 seguint amb l'estructura de carpetes i fitxers. Concretament cal fer el següent:

1. Descomprimiu el fitxer *PlanesTable\_20182.zip* que inclou els fitxers del projecte. El projecte està estructurat en carpetes: carpeta *include* on hi ha el fitxer ***plane.h*** i carpeta *src* on hi ha els fitxers ***plane.c*** i ***main.c***.
2. Dins del fitxer ***plane.h*** definiu el tipus de dades *tPlanesTable* (exercici 1.a)
3. Dins del fitxer ***plane.h*** definiu les capçaleres de les accions/funcions *planesTableInitialize*, *planesTableOccupation*, *planesTableSelectByOccupation* (exercici 1.b, exercici 1.c i exercici 1.d).
4. Dins el fitxer ***plane.c*** implementeu l'acció *planesTableInitialize* (exercici 1.b)
5. Dins el fitxer ***plane.c***, implementeu la funció *planesTableOccupation* (exercici 1.c)
6. Dins del fitxer ***plane.c*** implementeu l'acció *planesTableSelectByOccupation* (exercici 1.d)
7. Dins del fitxer ***main.c***, implementeu els apartats a-h de l'exercici 1.e.
8. Comproveu el funcionament del programa, provant amb diferents valors d'ocupació pels avions i diferents valors d'*idClass*.

```
/*
** File: main.c
** Author: Daniel Galeano Sancho
** Date: 25-04-2019
** Description: PAC8
*/

#include <stdio.h>
#include "plane.h"
/*****
* Pac8/Pec8 code v.1
*****/

//main algorithm
int main(int argc, char **argv){
/*****
* Write main algorithm here
*****/

    int i, id;
    tPlanesTable planesTable, selectTable;
    double avg;
    boolean correct;
```



```

    correct = FALSE;

    planesTableInitialize(&planesTable);
    planesTableInitialize(&selectTable);

    printf("Number of planes to enter: >> ");
    scanf("%d",&(planesTable.numPlanes));

    for (i = 0;i<planesTable.numPlanes;i++){
        printf("\nPlane number: %d\n",i+1);
        planeRead(&(planesTable.plane[i]));
    }

    while (!correct){
        printf("\nEnter code of the class (1 for First Class, 2 for
Business, 3 for Turist): >> ");
        scanf("%d",&id);

        if ((id == 1)||(id == 2)||(id == 3)){
            correct = TRUE;
        }else{
            printf("Incorrect code. Try again.\n");
        }
    }

    avg = planesTableOccupation(planesTable, id);

    planesTableSelectByOccupation(planesTable, avg, id, &selectTable);

    printf("\nThe number of planes below average is:
%d\n\n",selectTable.numPlanes);

    if (selectTable.numPlanes != 0){
        for (i = 0;i<selectTable.numPlanes;i++){
            planeWrite(selectTable.plane[i]);
        }
    }

    return 0;
}

```

```

/*
** File: plane.c
** Author: Daniel Galeano Sancho
** Date: 25-04-2019
** Description: PAC8
*/

```

```

#include <stdio.h>
#include <string.h>
#include "plane.h"

```

```

void planeRead(tPlane *plane){

    printf("Identifier for plane (integer): >> ");
    scanf("%d", &plane->id);
}

```

```

    printf("Model for plane (string): >> ");
    scanf("%s", plane->model);

    printf("Year for plane (integer): >> ");
    scanf("%d", &plane->year);

    printf("Utility for plane (enter a number being 0=COMMERCIAL,
1=PRIVATE, 2=GOVERNMENTAL, 3=MILITAR, 4=EXPERIMENTAL, 5=OTHERS: >> ");
    scanf("%u", &plane->utility);

    printf("Percentage of ocupation in Firstclass for plane (real): >> ");
    scanf("%f", &plane->percentOccupationFirstclass);

    printf("Percentage of ocupation in Business for plane (real): >> ");
    scanf("%f", &plane->percentOccupationBusiness);

    printf("Percentage of ocupation in Turist for plane (real): >> ");
    scanf("%f", &plane->percentOccupationTurist);

    printf("Is plane active? (enter a number being 0=FALSE, 1=TRUE): >> ");
    scanf("%u", &plane->isActive);

}

void planeWrite(tPlane plane){

    printf("Plane id: %d\n", plane.id);
    printf("Plane model: %s\n", plane.model);
    printf("Plane year: %d\n", plane.year);
    printf("Plane utility: %u\n", plane.utility);
    printf("Percentage of ocupation in Firstclass: %.2f\n",
plane.percentOccupationFirstclass);
    printf("Percentage of ocupation in Business: %.2f\n",
plane.percentOccupationBusiness);
    printf("Percentage of ocupation in Turist: %.2f\n",
plane.percentOccupationTurist);
    printf("Is the plane active? %u\n", plane.isActive);
    printf("\n");
}

void planeCopy(tPlane planeSrc, tPlane *planeDst){

    planeDst->id = planeSrc.id;
    strcpy(planeDst->model, planeSrc.model);
    planeDst->year = planeSrc.year;
    planeDst->utility = planeSrc.utility;
    planeDst->percentOccupationFirstclass =
planeSrc.percentOccupationFirstclass;
    planeDst->percentOccupationBusiness =
planeSrc.percentOccupationBusiness;
    planeDst->percentOccupationTurist = planeSrc.percentOccupationTurist;
    planeDst->isActive = planeSrc.isActive;
}

/*****
* Write new functions here
*
*****/

```

```

void planesTableInitialize (tPlanesTable *planeTable){
    planeTable->numPlanes = 0;
}

double planesTableOccupation (tPlanesTable planeTable, int idClass){
    double aux;
    int i;

    aux = 0;

    for (i = 0; i < planeTable.numPlanes; i++){
        if (idClass == CODE_FIRSTCLASS){
            aux = aux +
(double)planeTable.plane[i].percentOccupationFirstclass;
        }else{
            if (idClass == CODE_BUSINESS){
                aux = aux +
(double)planeTable.plane[i].percentOccupationBusiness;
            }else{
                if (idClass == CODE_TURIST){
                    aux = aux +
(double)planeTable.plane[i].percentOccupationTurist;
                }
            }
        }
    }

    aux = aux/(double)planeTable.numPlanes;

    return aux;
}

void planesTableSelectByOccupation(tPlanesTable planesTable, double
average, int idClass, tPlanesTable *selectTable){
    int i;

    planesTableInitialize(selectTable);

    for (i = 0; i < planesTable.numPlanes; i++){
        if (idClass == CODE_FIRSTCLASS){
            if (planesTable.plane[i].percentOccupationFirstclass <
average){
                planeCopy(planesTable.plane[i], &selectTable-
>plane[selectTable->numPlanes]);
                selectTable->numPlanes = selectTable->numPlanes + 1;
            }
        }else{
            if (idClass == CODE_BUSINESS){
                if (planesTable.plane[i].percentOccupationBusiness <
average){
                    planeCopy(planesTable.plane[i], &selectTable-
>plane[selectTable->numPlanes]);
                    selectTable->numPlanes = selectTable->numPlanes + 1;
                }
            }else{
                if (idClass == CODE_TURIST){
                    if (planesTable.plane[i].percentOccupationTurist <
average){

```

```

        planeCopy(planesTable.plane[i], &selectTable->plane[selectTable->numPlanes]);
        selectTable->numPlanes = selectTable->numPlanes +
1;
    }
}

}

}

}

}

}

/*
** File: plane.h
** Author: Daniel Galeano Sancho
** Date: 25-04-2019
** Description: PAC8
*/

// define constants
#define MAX_CHAR 20+1
#define MAX_PLANES 50

/*****
*****
* Write constants here */
#define NUM_CLASSES 3
#define CODE_FIRSTCLASS 1
#define CODE_BUSINESS 2
#define CODE_TURIST 3

//define enumerated boolean type
typedef enum {FALSE, TRUE} boolean;

//define enumerated utility type
typedef enum {COMMERCIAL,PRIVATE,GOVERNMENTAL,MILITAR,EXPERIMENTAL,OTHERS}
tUtility;

//define tPlane type
typedef struct {
    int id;
    char model[MAX_CHAR];
    int year;
    tUtility utility;
    float percentOccupationFirstclass;
    float percentOccupationBusiness;
    float percentOccupationTurist;
    boolean isActive;
} tPlane;

/*****
*****
* Define table type here

*****
*****/
typedef struct {
    tPlane plane[MAX_PLANES];

```

```

        int numPlanes;
    } tPlanesTable;

/*****
*****
    * Write new functions headers here
*****
*****/
/* Action for table initialization */
void planesTableInitialize (tPlanesTable *planeTable);

/* Function for calculating average occupation */
double planesTableOccupation (tPlanesTable planeTable, int idClass);

/* Action for selecting planes whose occupation is below average */
void planesTableSelectByOccupation(tPlanesTable planesTable, double
average, int idClass, tPlanesTable *selectTable);

```

**Apartat b: [30%]** Com a les anteriors PAC es demana que mostreu el funcionament de l'algorisme **fent jocs de prova**. És a dir que completeu les taules següents indicant, per a uns valors d'avions que s'han introduït a la taula *planesTable*, quina sortida s'espera en l'execució del programa.

Als següents jocs de proves, només es mostraran les variables que canvien d'un joc a un altre. Per a la resta de variables, sempre farem servir els valors mostrats a continuació.

Dades d'entrada	
Nom de la variable	Valor d'entrada
planesTable.numPlanes	3
planesTable.plane[0].id	10
planesTable.plane[0].model	A177
planesTable.plane[0].year	1988
planesTable.plane[0].utility	3
planesTable.plane[0].percentOccupationFirstclass	61.3
planesTable.plane[0].percentOccupationBusiness	75.8
planesTable.plane[0].percentOccupationTurist	95.3
planesTable.plane[0].isActive	0
planesTable.plane[1].id	20
planesTable.plane[1].model	B122
planesTable.plane[1].year	2004
planesTable.plane[1].utility	3
planesTable.plane[1].percentOccupationFirstclass	27.3
planesTable.plane[1].percentOccupationBusiness	40.1
planesTable.plane[1].percentOccupationTurist	95.3
planesTable.plane[1].isActive	0
planesTable.plane[2].id	30
planesTable.plane[2].model	B747
planesTable.plane[2].year	2009
planesTable.plane[2].utility	0

planesTable.plane[2].percentOccupationFirstclass	01.5
planesTable.plane[2].percentOccupationBusiness	03.6
planesTable.plane[2].percentOccupationTurist	95.3
planesTable.plane[2].isActive	1

b1) La taula *selectTable* no és buida (com en el cas *idClass=CODE\_FIRSTCLASS* i *idClass=CODE\_BUSINESS* de l'exemple):

Hem considerat dos casos en aquest apartat. El cas en que hi ha dos avions per sota de la mitjana (*idClass=1*) i el cas en que només hi ha un (*idClass=2*).

Dades d'entrada	
Nom de la variable	Valor d'entrada
idClass	1

Sortida
<p>The number of planes below average is: 2</p> <p>Plane id: 20  Plane model: B122  Plane year: 2004  Plane utility: 3  Percentage of occupation in Firstclass: 27.30  Percentage of occupation in Business: 40.10  Percentage of occupation in Turist: 95.30  Is the plane active? 0</p> <p>Plane id: 30  Plane model: B747  Plane year: 2009  Plane utility: 0  Percentage of occupation in Firstclass: 1.40  Percentage of occupation in Business: 3.60  Percentage of occupation in Turist: 95.30  Is the plane active? 1</p>

Dades d'entrada	
Nom de la variable	Valor d'entrada
idClass	2

Sortida
<p>The number of planes below average is: 1</p> <p>Plane id: 30  Plane model: B747  Plane year: 2009  Plane utility: 0  Percentage of occupation in Firstclass: 1.40  Percentage of occupation in Business: 3.60  Percentage of occupation in Turist: 95.30  Is the plane active? 1</p>

b2) La taula *selectTable* és buida (com en el cas *idClass=CODE\_TURIST* de l'exemple):

Dades d'entrada	
Nom de la variable	Valor d'entrada
idClass	3

Sortida
The number of planes below average is: 0

## Criteris de correcció:

### A l'exercici 1:

- 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 l'algorisme respongui al problema plantejat.
- Que es dissenyin i es cridin correctament les accions i funcions demanades.

### A l'exercici 2:

- Que el programa s'adeqüi a les indicacions donades.
- Que el programa compili i funcioni d'acord amb el que es demana.

- Que es respectin els criteris d'estil de programació C. Vegeu la *Guia d'estil de programació en C* que teniu a la Wiki de contingut.
- Que s'implementi correctament la modularització del projecte, dividint el codi en carpetes i posant el que correspon a cada carpeta.