Universitat Oberta de Catalunya uoc.edu

Fonaments de Programació

PAC7 - 20182

Data límit de lliurament: 15/04/2019

Estudiant

Cognoms: GALEANO SANCHO

Nom: DANIEL

Objectius

- Saber modularitzar el codi utilitzant accions i funcions.
- Comprendre la diferència entre acció i funció.
- Entendre que és un paràmetre actual i distingir-lo del paràmetre formal.
- Estructurar projectes en Codelite.

Format i data de lliurament

Cal lliurar la PAC abans del dia **15 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

En aquesta PAC continuarem treballant la modularitat dividint el codi en accions i funcions que ens permetin poder-lo reutilitzar. En aquest sentit, l'exercici de codificació no es limita a traduir a C l'exercici de llenguatge algorísmic. Es demana, a més, estructurar el projecte de Codelite en carpetes i fitxers.

Seguint amb l'ajuda que proporcionem a UOCAirways, la companyia ens ha demanat la nostra col·laboració per seguir treballant en el següent algorisme, en llenguatge algorísmic, a mig dissenyar.

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


algorithm UOCAirways

const
   NUM_CLASSES: integer = 3;
end const

{... algorithm to complete …}

end algorithm
```

Exercici 1: Disseny en llenguatge algorísmic (40%)

Nota: L'estructura de dades *tPlane* ha canviat lleugerament respecte la PAC anterior. Podem assumir, en aquest exercici de llenguatge algorísmic, que les accions *planeRead*, *planeWrite* i *planeCopy* de la PAC anterior ja han estat adaptades per funcionar amb la nova estructura del tipus *tPlane* i que les podem fer servir sense tornar a dissenyar.

Apartat a: [50%] Dissenya la funció *isFirstPlane* que té dos paràmetres *plane1, plane2* de tipus *tPlane* i retorna un booleà. La funció retorna *true* si el primer avió té prioritat per aterrar i *false* en cas contrari. El criteri per decidir la prioritat per aterrar s'estableix seguint el següent ordre segons els valors d':

1. Ocupació total (en %) de l'avió, entesa com a mitjana de l'ocupació de les tres classes (*percentOccupationFirstclass*, *percentOccupationBusiness*, *percentOccupationTurist*).
2. Ocupació (en %) de la primera classe (*percentOccupationFirstclass*).
3. Ocupació (en %) de la classe business (*percentOccupationBussines*).
4. Ocupació (en %) de la classe turista (*percentOccupationTurist*).

El funcionament és el següent: en primer lloc es compara l'ocupació total (en %) de *plane1* i *plane2*. Si l'ocupació total de *plane1* és superior a la de *plane2*, retorna *true*. Si l'ocupació total de *plane1* és inferior a la de *plane2*, retorna *false*. En cas d'empat, es passa a comparar l'ocupació (en %) de la primera classe (*percentOccupationFirstclass*), i així successivament. En cas que tots els camps comparats dels dos avions siguin iguals, la funció retornarà *true* (és a dir, donarà prioritat al primer avió).

```
function isFirstPlane(plane1: tPlane, plane2: tPlane): boolean

   var
      totalOccupation1, totalOccupation2: real;
      firstPlane: boolean;
   end var

   totalOccupation1 := (plane1.percentOccupationFirstclass  +
plane1.percentOccupationBusiness + plane1.percentOccupationTurist)/NUM_CLASSES;
   totalOccupation2 := (plane2.percentOccupationFirstclass  +
plane2.percentOccupationBusiness + plane2.percentOccupationTurist)/NUM_CLASSES;

   if totalOccupation1 > totalOccupation2 then
      firstPlane := true;
   else
      if totalOccupation1 < totalOccupation2 then
         firstPlane := false;
      else
         if plane1.percentOccupationFirstclass > plane2.percentOccupationFirstclass then
            firstPlane := true;
         else
            if plane1.percentOccupationFirstclass < plane2.percentOccupationFirstclass then
               firstPlane := false;
            else
               if plane1. percentOccupationBusiness > plane2. percentOccupationBusiness
then
                  firstPlane := true;
               else
                  if plane1. percentOccupationBusiness < plane2. percentOccupationBusiness
then
                     firstPlane := false;
                  else
                     if plane1. percentOccupationTurist > plane2. percentOccupationTurist
then
```

```
                    firstPlane := true;
                else
                    if plane1. percentOccupationTurist < plane2. percentOccupationTurist
then
                        firstPlane := false;
                    else
                        firstPlane := true;
                    end if
                end if
            end if
        end if
    end if
  end if

  return(firstPlane);

end function
```

Apartat b: [20%] Dissenya l'acció *planePriorized* que retorna en el paràmetre de sortida *planePriority* de tipus *tPlane* l'avió que té prioritat per aterrar. L'acció té a més dos paràmetres d'entrada *plane1 i plane2* de tipus *tPlane*. L'acció copia tots els camps de *plane1* o *plane2* a *planePriority*. El funcionament és el següent: copiarà tots els camps de *plane1* a *planePriority* en cas que sigui *plane1* el que tingui prioritat. En cas contrari, l'acció copiarà tots els camps de *plane2* a *planePriority.*

```
action planePriorized(in plane1: tPlane, in plane2: tPlane, out planePriority: tPlane)

  if isFirstPlane then
    planeCopy(plane1, planePriority);
  else
    planeCopy(plane2, planePriority);
  end if
end function
```

Apartat c: [20%] Completa l'algorisme que tenim a mig dissenyar per tal que llegeixi del canal estàndard d'entrada la informació de dos avions i la desi a les variables *plane1* i *plane2*. A continuació, l'algorisme ha de mostrar pel canal estàndard de sortida els camps de l'avió que té prioritat per aterrar, fent ús de les accions i funcions que hem dissenyat prèviament.

Nota: Hem d'assumir que els avions són comercials i estan en actiu. Per tant, l'algorisme no cal que ho comprovi i funcionarà independentment del valor dels camps *utility* i *isActive*.

A continuació trobareu exemples de la sortida de l'algorisme per a dos casos diferents d'entrada.

Exemple 1:

```
...
plane1. percentOccupationFirstclass := 70.5;
plane1. percentOccupationBusiness:= 79.5;
plane1. percentOccupationTurist:= 93.0;
...
plane2. percentOccupationFirstclass := 77.5;
plane2. percentOccupationBusiness:= 75.5;
plane2. percentOccupationTurist:= 87.0;
…
```

El resultat que es mostraria pel canal estàndard de sortida en aquest cas seria *plane1* ja que *plane1* té una ocupació total del 81% i *plane2* té una ocupació total del 80%.

Exemple 2:

```
...
plane1. percentOccupationFirstclass := 70.5;
plane1. percentOccupationBusiness:= 79.5;
plane1. percentOccupationTurist:= 90.0;
...
plane2. percentOccupationFirstclass := 83.5;
plane2. percentOccupationBusiness:= 69.5;
plane2. percentOccupationTurist:= 87.0;
…
```

En aquest cas, l'ocupació total dels dos avions és del 80%. No obstant es mostraria pel canal estàndard de sortida *plane2* ja que té prioritat davant de *plane1* degut al camp *percentOccupationFirstclass*.

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

```
algorithm UOCAirways

   const
      NUM_CLASSES: integer = 3;
   end const

   var
      priorPlane: tPlane;
   end var

   writeString("Plane 1: ");
   planeRead(plane1);

   writeString("Plane 2: ");
   planeRead(plane2);

   planePriorized(plane1, plane2, priorPlane);
   writeString("Priorized plane: ");
   planeWrite(priorPlane);

end algorithm
```

Apartat d: [10%] A l'apartat *c* hem assumit que els avions són comercials i estan en actiu. Des del punt de vista de la modularització, explica què caldria fer a algorisme anterior per tal que comprovés que el valor del camp *utility* fos COMMERCIAL i el valor del camp *isActive* fos *true*. No cal fer el disseny, només cal explicar-ho.

Podríem construir una funció de tipus *boolean* que tingués com paràmetre una variable de tipus *tPlane* i que ens retornés *true* si es compleix que *plane.utility* és igual a *COMMERCIAL* i, alhora, *plane.isActive* és igual a *true*.

La capçalera d'aquesta funció podria ser:

```
function isCommercialAndActive(plane: tPlane): boolean;
```

Exercici 2: Programació en C [60%]

Apartat a: [70%] En aquest exercici cal **codificar en llenguatge C** l'exercici 1 (apartats a, b i c) i, a més, estructurar el codi en carpetes. Concretament cal fer el següent:

1. Crea un nou projecte en Codelite anomenat *Planes*. Crea una carpeta *include* i una carpeta *src* en aquest projecte seguint les explicacions que podeu trobar a la unitat de la xWiki *Modularitat en Codelite* de l'assignatura.

2. Dins de la carpeta *include*, crea un nou fitxer anomenat **plane.h** que contingui la declaració del tipus estructurat *tPlane*, la del tipus enumerat *tUtility* i la del tipus *boolean*, així com les constants necessàries.
3. Copia les capçaleres de totes les accions/funcions que hagis de fer servir (***planeRead, planeWrite, planePriorized, isFisrtPlane***, etc.) al fitxer **plane.h**.
4. Dins de la carpeta *src*, crea un nou fitxer **plane.c**. Posa-hi el codi de les accions/funcions que hagis de fer servir (***planeRead, planeWrite, planePriorized, isFisrtPlane***, etc.)

   Nota: A l'exercici 1 hem assumit que les accions *planeRead*, *planeWrite i planeCopy* (de la PAC anterior) ja havien estat adaptades en llenguatge algorísmic. Ara, en aquest exercici de codificació en C, cal que les modifiquis per tal que funcionin correctament amb la nova estructura del tipus *tPlane.*

5. Codifica l'algorisme de l'exercici 1, apartat c, dins de la funció principal **main.c**
6. Comprova que compila i funciona correctament.

```c
/*
** File: main.c
** Author: Daniel Galeano Sancho
** Date: 10-04-2019
** Description: PAC7
*/

/* System header files */
#include <stdio.h>
#include "plane.h"

/* Main function */
int main(int argc, char **argv){

      /* Variables */
      tPlane plane1, plane2, priorPlane;

    /* Input information for plane 1 */
      printf("Plane 1:\n");
      planeRead(&plane1);
      printf("\n");

      /* Input information for plane 2 */
      printf("Plane 2:\n");
      planeRead(&plane2);
      printf("\n");

      /* Analysing priority for planes */
      planePriorized(plane1, plane2, &priorPlane);

      /* Showing priority plane */
      printf("Priorized plane: \n");
      planeWrite(priorPlane);

      return 0;
```

```
}


/*
** File: plane.c
** Author: Daniel Galeano Sancho
** Date: 10-04-2019
** Description: PAC7
*/

/* System header files */
#include <stdio.h>
#include <string.h>
#include "plane.h"

/* Read information of a plane */
void planeRead(tPlane *plane){

    printf("Identifier for the plane (integer): >> ");
    scanf("%d",&(plane->id));
    printf("Model for the plane (string): >> ");
      scanf("%s",plane->model);
    printf("Year for the plane (integer):  >> ");
    scanf("%d",&(plane->year));
    printf("Utility for the plane (enter a number being 0=COMMERCIAL,
1=PRIVATE, 2=GOVERNMENTAL, 3=MILITAR, 4=EXPERIMENTAL, 5=OTHERS): >> ");
    scanf("%u",&(plane->utility));
    printf("First class percent occupation for the plane: >> ");
    scanf("%f",&(plane->percentOccupationFirstclass));
    printf("Business class percent occupation for the plane: >> ");
    scanf("%f",&(plane->percentOccupationBusiness));
    printf("Tourist class percent occupation for the plane: >> ");
    scanf("%f",&(plane->percentOccupationTurist));
    printf("Is the plane active? (0 for FALSE, 1 for TRUE): >> ");
    scanf("%u",&(plane->isActive));

}

/* Write information of a plane */
void planeWrite(tPlane plane){

      printf("Identifier: %d\n",plane.id);
      printf("Model: %s\n",plane.model);
      printf("Year: %d\n",plane.year);
      printf("Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL,
3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): %u\n",plane.utility);
    printf("First class percent occupation:
%.2f\n",plane.percentOccupationFirstclass);
    printf("Business class percent occupation:
%.2f\n",plane.percentOccupationBusiness);
      printf("First class percent occupation:
%.2f\n",plane.percentOccupationTurist);
      printf("Is the plane active? (0 for FALSE, 1 for TRUE):
%u\n",plane.isActive);
}

/* Copy information of a plane, planeIn, into another plane, planeOut */
void planeCopy(tPlane planeIn, tPlane *planeOut){

    planeOut->id = planeIn.id;
    strcpy(planeOut->model,planeIn.model);
```

```
    planeOut->year = planeIn.year;
    planeOut->utility = planeIn.utility;
    planeOut->percentOccupationFirstclass =
planeIn.percentOccupationFirstclass;
    planeOut->percentOccupationBusiness =
planeIn.percentOccupationBusiness;
    planeOut->percentOccupationTurist = planeIn.percentOccupationTurist;
    planeOut->isActive = planeIn.isActive;

}

/* Exercise A */
bool isFirstPlane(tPlane plane1, tPlane plane2){

    float totalOccupation1, totalOccupation2;
    bool firstPlane;

    totalOccupation1 = (plane1.percentOccupationFirstclass  +
plane1.percentOccupationBusiness +
plane1.percentOccupationTurist)/NUM_CLASSES;
    totalOccupation2 = (plane2.percentOccupationFirstclass  +
plane2.percentOccupationBusiness +
plane2.percentOccupationTurist)/NUM_CLASSES;

    if (totalOccupation1 > totalOccupation2) {
        firstPlane = TRUE;
    }else{
        if (totalOccupation1 < totalOccupation2) {
            firstPlane = FALSE;
        }else{
            if (plane1.percentOccupationFirstclass >
plane2.percentOccupationFirstclass) {
                firstPlane = TRUE;
            }else{
                if (plane1.percentOccupationFirstclass <
plane2.percentOccupationFirstclass) {
                    firstPlane = FALSE;
                }else{
                    if (plane1. percentOccupationBusiness > plane2.
percentOccupationBusiness) {
                        firstPlane = TRUE;
                    }else{
                        if (plane1. percentOccupationBusiness < plane2.
percentOccupationBusiness) {
                            firstPlane = FALSE;
                        }else{
                            if (plane1. percentOccupationTurist > plane2.
percentOccupationTurist) {
                                firstPlane = TRUE;
                            }else{
                                if (plane1. percentOccupationTurist <
plane2. percentOccupationTurist) {
                                    firstPlane = FALSE;
                                }else{
                                    firstPlane = TRUE;
                                }
                            }
                        }
                    }
                }
            }
```

```
        }
    }
      return firstPlane;
}

/* Exercise B */
void planePriorized(tPlane plane1, tPlane plane2, tPlane *planePriority){

    if (isFirstPlane(plane1,plane2)) {
        planeCopy(plane1, planePriority);
    }else{
        planeCopy(plane2, planePriority);
    }
}

/*
** File: plane.h
** Author: Daniel Galeano Sancho
** Date: 10-04-2019
** Description: PAC7
*/

/* Constants */
#define MAX_LENGTH 15
#define NUM_CLASSES 3

/* User defined types */
typedef enum {COMMERCIAL, PRIVATE, GOVERNMENTAL, MILITAR, EXPERIMENTAL,
OTHERS} tUtility;
typedef enum {FALSE,TRUE} bool;
typedef struct{
      int id;
      char model[MAX_LENGTH];
      int year;
      tUtility utility;
      float percentOccupationFirstclass;
      float percentOccupationBusiness;
      float percentOccupationTurist;
      bool isActive;
}tPlane;

/* Read information of a plane */
void planeRead(tPlane *plane);

/* Write information of a plane */
void planeWrite(tPlane plane);

/* Copy information of a plane, planeIn, into another plane, planeOut */
void planeCopy(tPlane planeIn, tPlane *planeOut);

/* Analyse priority of plane1 and plane2. True: plane1>plane2, False:
plane1<plane2 */
bool isFirstPlane(tPlane plane1, tPlane plane2);

/* Returns a priorized plane between plane1 and plane2 */
```

```
void planePriorized(tPlane plane1, tPlane plane2, tPlane *planePriority);
```

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

Als següents jocs de proves, la sortida no depèn d'algunes variables de la tupla tPlane, per tant, s'ometen de les dades d'entrada. Concretament, s'ometen les variables *id*, *model*, *year*, *utility* i *isActive*.

b1) L'ocupació total (en %) d'un avió (entesa com a mitjana de l'ocupació de les tres classes) és superior a l'altre (com en el cas de l'exemple 1):

Dades d'entrada	
Nom de la variable	**Valor d'entrada**
plane1.percentOccupationFirstclass	75
plane1.percentOccupationBusiness	70
plane1.percentOccupationTurist	80
plane2.percentOccupationFirstclass	80
plane2.percentOccupationBusiness	75
plane2.percentOccupationTurist	75

Sortida
Priorized plane: Identifier: 2 Model: B122 Year: 1988 Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL, 3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): 0 First class percent occupation: 80.00 Business class percent occupation: 75.00 First class percent occupation: 75.00 Is the plane active? (0 for FALSE, 1 for TRUE): 1

b2) L'ocupació total (en %) dels dos avions (entesa com a mitjana de l'ocupació de les tres classes) és igual i tenen diferent ocupació en alguna de les altres classes (com en el cas de l'exemple 2):

Dades d'entrada	
Nom de la variable	**Valor d'entrada**
plane1.percentOccupationFirstclass	80
plane1.percentOccupationBusiness	70
plane1.percentOccupationTurist	75
plane2.percentOccupationFirstclass	75
plane2.percentOccupationBusiness	70
plane2.percentOccupationTurist	80

Sortida
Priorized plane: Identifier: 1 Model: A177 Year: 2008 Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL, 3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): 0 First class percent occupation: 80.00 Business class percent occupation: 70.00 First class percent occupation: 75.00 Is the plane active? (0 for FALSE, 1 for TRUE): 1

b3) Els dos avions tenen l'ocupació total igual i l'ocupació de les tres classes també igual:

Dades d'entrada	
Nom de la variable	**Valor d'entrada**
plane1.percentOccupationFirstclass	90
plane1.percentOccupationBusiness	75
plane1.percentOccupationTurist	80
plane2.percentOccupationFirstclass	90
plane2.percentOccupationBusiness	75
plane2.percentOccupationTurist	80

Sortida
Priorized plane: Identifier: 1 Model: A177 Year: 2008 Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL, 3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): 0 First class percent occupation: 90.00 Business class percent occupation: 75.00 First class percent occupation: 80.00 Is the plane active? (0 for FALSE, 1 for TRUE): 1

Criteris de correcció:

A l'exercici 1:

- Que segueix la notació algorísmica utilitzada a l'assignatura. Vegeu document *Nomenclator* a la xWiki de contingut.
- Que s'adequa a les instruccions donades i l'algorisme respongui al problema plantejat.
- Que es dissenya i es crida correctament les accions i funcions demanades.
- Que s'utilitzi correctament l'estructura alternativa i el tipus de dades estructurat.
- Que es raoni correctament la resposta de l'apartat d de la primera pregunta.

A l'exercici 2:

- Que el programa s'adequa les indicacions donades.
- Que el programa compila i funciona d'acord amb el que es demana.
- Que es respecten 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'implementa correctament la modularització del projecte, dividint el codi en carpetes i posant el que correspon a cada carpeta.