Universitat Oberta de Catalunya uoc.edu

Fonaments de Programació
PAC6 - 20182

Data límit de lliurament: 8/4/2019

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 d'un paràmetre formal.

Format i data de lliurament

Cal lliurar la PAC abans del dia **8 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
- Un workspace de Codelite que contingui els fitxers .c demanats 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 la companyia UOCAirways, ens han demanat la nostra col·laboració per a crear un programa que els ajudi a gestionar les peticions simultànies d'aterratge de dos avions. En els següents exercicis, treballarem amb funcions i accions, així com tipus de dades estructurats, conjuntament amb l'entrada i sortida interactiva per gestionar les dades dels avions.

Disposem del 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;
          weight: real;
          maxSpeed: real;
          maxHeight: integer;
          motors: integer;
          seats: integer;
          isActive: boolean;
     end record
end type

algorithm UOCAirways
var
    plane1:  tPlane;
    plane2:  tPlane;
end var

    {input information for plane 1}
    writeString("Identifier for plane 1 (integer): >>");
    plane1.id := readInteger();

    writeString("Model for plane 1 (string): >>");
    plane1.model := readString();

    writeString("Year for plane 1 (integer):  >>");
    plane1.year := readInteger();

    writeString("Utility for plane 1 (enter a number being 0=COMMERCIAL, 1=PRIVATE,
    2=GOVERNMENTAL, 3=MILITAR, 4=EXPERIMENTAL, 5=OTHERS: >>");
    plane1.utility := readUtility();

    writeString("Weight for plane 1 (Tons): >>");
    plane1.weight := readReal();

    writeString("Max speed for plane 1 (Km/h): >>");
    plane1.maxSpeed := readReal();

    writeString("Max height for plane 1 (metres): >>");
    plane1.maxHeight := readInteger();
```

```
    writeString("Number of motors for plane 1 (integer): >>");
    plane1.motors := readInteger();

    writeString("Number of seats for plane 1 (integer): >>");
    plane1.seats := readInteger();

    writeString("Is plane 1 active? (true/false): >>");
    plane1.isActive := readBoolean();

    {input information for plane 2}
    writeString("Identifier for plane 2 (integer): >>");
    plane2.id := readInteger();

    writeString("Model for plane 2 (string): >>");
    plane2.model := readString();

    writeString("Year for plane 2 (integer):  >>");
    plane2.year := readInteger();

    writeString("Utility for plane 2 (enter a number being 0=COMMERCIAL, 1=PRIVATE,
    2=GOVERNMENTAL, 3=MILITAR, 4=EXPERIMENTAL, 5=OTHERS: >>");
    plane2.utility := readUtility();

    writeString("Weight for plane 2 (Tons): >>");
    plane2.weight := readReal();

    writeString("Max speed for plane 2 (Km/h): >>");
    plane2.maxSpeed := readReal();

    writeString("Max height for plane 2 (metres): >>");
    plane2.maxHeight := readInteger();

    writeString("Number of motors for plane 2 (integer): >>");
    plane2.motors := readInteger();

    writeString("Number of seats for plane 2 (integer): >>");
    plane2.seats := readInteger();

    writeString("Is plane 2 active? (true/false): >>");
    plane2.isActive := readBoolean ();

{Algorithm to complete …}

end algorithm
```

Exercici 1: Modularitat [50%]

Apartat a: [10%] Dissenya l'acció *planeRead* que retorni un paràmetre de tipus *tPlane* després de llegir des del canal estàndard d'entrada la informació d'un avió.

Nota: En llenguatge algorítmic disposem de la funció *readUtility* que podem fer servir sense necessitat d'implementar.

```
action planeRead(out plane: tPlane)

   {input information for the plane}
   writeString("Identifier for the plane (integer): >>");
   plane.id := readInteger();

   writeString("Model for the plane (string): >>");
   plane.model := readString();

   writeString("Year for the plane (integer):  >>");
   plane.year := readInteger();

   writeString("Utility for the plane (enter a number being 0=COMMERCIAL, 1=PRIVATE,
2=GOVERNMENTAL, 3=MILITAR, 4=EXPERIMENTAL, 5=OTHERS): >>");
   plane.utility := readUtility();

   writeString("Weight for the plane (Tons): >>");
   plane.weight := readReal();

   writeString("Max speed for the plane (Km/h): >>");
   plane.maxSpeed := readReal();

   writeString("Max height for the plane (metres): >>");
   plane.maxHeight := readInteger();

   writeString("Number of motors for the plane (integer): >>");
   plane.motors := readInteger();

   writeString("Number of seats for the plane (integer): >>");
   plane.seats := readInteger();

   writeString("Is the plane active? (true/false): >>");
   plane.isActive := readBoolean();

end action
```

Apartat b: [10%] Dissenya l'acció *planeWrite* que escrigui pel canal estàndard de sortida la informació d'un avió. L'acció ha de tenir un paràmetre d'entrada de tipus *tPlane.*

Nota: En llenguatge algorítmic disposem de l'acció *writeUtility* que podem fer servir sense necessitat d'implementar.

```
action planeWrite(in plane: tPlane)

   {output information for the plane}
   writeString("Identifier: ");
   writeInteger(plane.id);

   writeString("Model: ");
   writeString(plane.model);

   writeString("Year:  ");
   writeInteger(plane.year);

   writeString("Utility (0=COMMERCIAL, 1=PRIVATE, 2=GOVERNMENTAL, 3=MILITAR,
4=EXPERIMENTAL, 5=OTHERS): ");
   writeUtility(plane.utility);

   writeString("Weight (Tons): ");
   writeReal(plane.weight);

   writeString("Max speed (Km/h): ");
   writeReal(plane.maxSpeed);

   writeString("Max height (metres): ");
   writeInteger(plane.maxHeight);

   writeString("Number of motors: ");
   writeInteger(plane.motors);

   writeString("Number of seats: ");
   writeInteger(plane.seats);

   writeString("Is the plane active?: ");
   writeBoolean(plane.isActive);

end action
```

Apartat c: [10%] Dissenya l'acció *planeCopy* que copiï els camps d'un avió a un altre avió. L'acció ha de tenir un paràmetre d'entrada de tipus *tPlane* (que conté l'avió inicial) i un paràmetre de sortida de tipus *tPlane* (que conté l'avió on van a parar les dades que es copien).

```
action planeCopy(in planeIn: tPlane, out planeOut: tPlane)

   planeOut.id := planeIn.id;
   planeOut.model := planeIn.model;
   planeOut.year := planeIn.year;
   planeOut.utility := planeIn.utility;
   planeOut.weight := planeIn.weight;
   planeOut.maxSpeed := planeIn.maxSpeed;
   planeOut.maxHeight := planeIn.maxHeight;
   planeOut.motors := planeIn.motors;
   planeOut.seats := planeIn.seats;
   planeOut.isActive := planeIn.isActive;

end action
```

Apartat d: [10%] Dissenya la funció *planeAnalise* que té dos paràmetres d'entrada *plane1* i *plane2* de tipus *tPlane*. La funció retornarà un enter de valor 1 si l'avió *plane1* és el més ràpid i retornarà 2 si el més ràpid és l'avió *plane2*. En cas que els dos avions tinguin la mateixa velocitat màxima, es retornarà un 1 si és *plane1* el que pot volar més alt i 2 en cas contrari. En cas que continuin empatats, es retornarà un 0.

```
function planeAnalise(plane1: tPlane, plane2: tPlane): integer
   var
      res: integer;
   end var

   if plane1.maxSpeed = plane2.maxSpeed then
      if plane1.maxHeight = plane2.maxHeight then
         res := 0;
      else
         if plane1.maxHeight > plane2.maxHeight then
            res := 1;
         else
            res := 2;
         end if
      end if
   else
      if plane1.maxSpeed > plane2.maxSpeed then
         res := 1;
      else
         res := 2;
      end if
   end if
   return(res);
end function
```

Apartat e: [40%] Modifica l'algorisme per tal que la lectura de la informació dels avions es faci a través de l'acció *planeRead.* A continuació, completa l'algorisme per tal que comprovi si els dos avions entrats són militars. En cas afirmatiu, cal determinar quin dels dos avions pot volar més alt fent servir la funció de l'apartat anterior i cal mostrar la informació d'aquest avió pel canal estàndard de sortida (pantalla) amb l'acció *planeWrite*. Si els dos avions tenen la mateixa velocitat, es mostrarà per pantalla l'avió que pot volar més alt. En cas que continuï l'empat, només es mostrarà un missatge dient que els dos avions tenen característiques idèntiques de velocitat i alçada màxima. En el cas que només un dels dos avions sigui militar s'escriurà un missatge per pantalla dient quin dels dos avions és militar, es copiaran les dades de l'avió militar a una nova variable de tipus *tPlane* fent ús de l'acció *planeCopy* i s'escriurà per pantalla la informació d'aquesta nova variable. Finalment, si cap dels dos avions és militar, s'escriurà per pantalla un missatge dient que cap dels dos avions és militar.

```
type
   tUtility = {COMMERCIAL, PRIVATE, GOVERNMENTAL, MILITAR, EXPERIMENTAL,
OTHERS}
   tPlane = record
      id: integer;
      model: string;
      year: integer;
      utility: tUtility;
      weight: real;
      maxSpeed: real;
      maxHeight: integer;
      motors: integer;
      seats: integer;
      isActive: boolean;
      end record
end type

algorithm UOCAirways
var
   plane1:  tPlane;
   plane2:  tPlane;
   planeMilitar: tPlane;
end var

{input information for plane 1}
planeRead(plane1);

{input information for plane 2}
planeRead(plane2);

if plane1.utility = MILITAR then
   if plane2.utility = MILITAR then
      if planeAnalise = 1 then
         planeWrite(plane1);
      else
```

```
            if planeAnalise = 2 then
               planeWrite(plane2);
            else
               writeString("Both planes have the same maximum speed and height");
            end if
         end if
      else
         writeString("Plane 1 is militar");
         planeCopy(plane1,planeMilitar);
         planeWrite(planeMilitar);
      end if
   else
      if plane2.utility = MILITAR then
         writeString("Plane 2 is militar");
         planeCopy(plane2,planeMilitar);
         planeWrite(planeMilitar);
      else
         writeString("None of the planes are militar");
      end if
   end if

end algorithm
```

Apartat f: [20%] Explica com hauries de modificar *planeAnalize* que s'ha dissenyat a l'apartat *d* per tal que, donats dos paràmetres d'entrada *plane1* i *plane2* de tipus *tPlane,* retorni un tipus *tPlane* que representi l'avió que té velocitat màxima més elevada. En cas que els dos avions tinguin la mateixa velocitat màxima, s'haurà de retornar l'avió que pugui volar més alt. En cas que continuï l'empat, podem suposar que es retornaran les dades del primer avió. (Es demana que raoneu la resposta. No cal que feu el disseny)

Com ha de retornar un tipus *tPlane*, la capçalera seria la següent:

function planeAnalise(plane1: tPlane, plane2: tPlane): tPlane;

El contingut de la funció seria el mateix que a l'apartat d, però ara la variable entera *res* serà una variable de tipus *tPlane* i ens servirà per copiar l'avió guanyador. Per tant, haurem de substituir les expressions, com per exemple *res := 0;*, per la funció *planeCopy* de la següent manera:

planeCopy(plane1,res);

Exercici 2: [50%]

Apartat a: [70%] Implementeu en llenguatge C, l'algorisme de l'exercici anterior.

Nota: Recordeu que, en llenguatge C, cal utilitzar funcions específiques per poder copiar cadenes de caràcters. Concretament, podeu utilitzar el mètode *strcpy* de la llibreria *string.h*, que permet copiar dos strings.

```
/*
** File: main.c
** Author: Daniel Galeano Sancho
** Date: 03-04-2019
** Description: PAC6
*/

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

/* Constants */
#define MAX_LENGTH 15

/* 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 weight;
      float maxSpeed;
      int maxHeight;
      int motors;
      int seats;
      bool isActive;
}tPlane;

/* Exercise A. Action (void function) for reading information of a plane */
void planeRead(tPlane *plane){

    printf("Identifier for the plane (integer): >> ");
    scanf("%d",&(plane->id));
    printf("Model for the plane (string): >> ");
    //getchar();
      //gets(plane->model);
      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("Weight for the plane (Tons): >> ");
    scanf("%f",&(plane->weight));
    printf("Max speed for the plane (Km/h): >> ");
    scanf("%f",&(plane->maxSpeed));
    printf("Max height for the plane (metres): >> ");
```

```
    scanf("%d",&(plane->maxHeight));
    printf("Number of motors for the plane (integer): >> ");
    scanf("%d",&(plane->motors));
    printf("Number of seats for the plane (integer): >> ");
    scanf("%d",&(plane->seats));
    printf("Is the plane active? (0 for FALSE, 1 for TRUE): >> ");
    scanf("%u",&(plane->isActive));

}

/* Exercise B. Action (void function) for writing information of a plane */
void planeWrite(tPlane plane){

      printf("\nIdentifier: %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("Weight (Tons): %.2f\n",plane.weight);
      printf("Max speed (Km/h): %.2f\n",plane.maxSpeed);
      printf("Max height (metres): %d\n",plane.maxHeight);
      printf("Number of motors: %d\n",plane.motors);
      printf("Number of seats: %d\n",plane.seats);
      printf("Is the plane active? (0 for FALSE, 1 for TRUE):
%u\n",plane.isActive);
}

/* Exercise C. Action (void function) for copying information of a plane
into another plane */
void planeCopy(tPlane planeIn, tPlane *planeOut){

    planeOut->id = planeIn.id;
    strcpy(planeOut->model,planeIn.model);
    planeOut->year = planeIn.year;
    planeOut->utility = planeIn.utility;
    planeOut->weight = planeIn.weight;
    planeOut->maxSpeed = planeIn.maxSpeed;
    planeOut->maxHeight = planeIn.maxHeight;
    planeOut->motors = planeIn.motors;
    planeOut->seats = planeIn.seats;
    planeOut->isActive = planeIn.isActive;

}

/* Exercise D. Function for comparing features between two planes */
int planeAnalise(tPlane plane1, tPlane plane2){

    int res;

    if (plane1.maxSpeed == plane2.maxSpeed) {
        if (plane1.maxHeight == plane2.maxHeight) {
            res = 0;
        }else{
            if (plane1.maxHeight > plane2.maxHeight) {
                res = 1;
            }else{
                res = 2;
            }
        }
    }else{
        if (plane1.maxSpeed > plane2.maxSpeed) {
```

```
            res = 1;
        }else{
            res = 2;
        }
    }
    return res;
}

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

      /* Variables */
      tPlane plane1;
      tPlane plane2;
    tPlane planeMilitar;
      int analise;

    /* 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");

      /* Storing result of planeAnalise */
      analise = planeAnalise(plane1,plane2);

      if (plane1.utility == MILITAR) {
            if (plane2.utility == MILITAR) {
                  printf("Better plane: \n");
                  if (analise == 1) {
                        planeWrite(plane1);
                  }else{
                        if (analise == 2) {
                              planeWrite(plane2);
                        }else{
                              printf("Both planes have the same maximum
speed and height.\n");
                        }
                  }
            }else{
                  printf("Plane 1 is militar.\n");
                  planeCopy(plane1,&planeMilitar);
                  planeWrite(planeMilitar);
            }
      }else{
            if (plane2.utility == MILITAR) {
                  printf("Plane 2 is militar.\n");
                  planeCopy(plane2,&planeMilitar);
                  planeWrite(planeMilitar);
            }else{
                  printf("None of the planes are militar.\n");
            }
      }

      return 0;
}
```

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 les diferents estructures, quina sortida s'espera en l'execució del programa. Després comproveu que efectivament el programa fa el que esperàveu.

Podeu afegir més files a les taules.

b1) Els dos avions són militars:

Dades d'entrada	
Nom variable	**Valor entrada**
plane1.id	10
plane1.model	A320
plane1.year	2017
plane1.utility	3
plane1.weight	80
plane1.maxSpeed	910
plane1.maxHeight	11000
plane1.motors	4
plane1.seats	250
plane1.isActive	1
plane2.id	20
plane2.model	B777
plane2.year	2015
plane2.utility	3
plane2.weight	100
plane2.maxSpeed	910
plane2.maxHeight	12000
plane2.motors	6
plane2.seats	275
plane2.isActive	0

Sortida
Identifier: 20
Model: B777
Year: 2015
Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL, 3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): 3
Weight (Tons): 100.00
Max speed (Km/h): 910.00

Max height (metres): 12000
Number of motors: 6
Number of seats: 275
Is the plane active? (0 for FALSE, 1 for TRUE): 0

b2) Només un dels dos avions és militar:

Dades d'entrada	
Nom variable	**Valor entrada**
plane1.id	10
plane1.model	A320
plane1.year	2017
plane1.utility	3
plane1.weight	80
plane1.maxSpeed	910
plane1.maxHeight	11000
plane1.motors	4
plane1.seats	250
plane1.isActive	1
plane2.id	20
plane2.model	B777
plane2.year	2015
plane2.utility	1
plane2.weight	100
plane2.maxSpeed	910
plane2.maxHeight	12000
plane2.motors	6
plane2.seats	275
plane2.isActive	0

Sortida
Identifier: 10
Model: A320
Year: 2017
Utility (0 for COMMERCIAL, 1 for PRIVATE, 2 for GOVERNMENTAL, 3 for MILITAR, 4 for EXPERIMENTAL, 5 for OTHERS): 3
Weight (Tons): 80.00
Max speed (Km/h): 910.00
Max height (metres): 11000
Number of motors: 4

Number of seats: 250
Is the plane active? (0 for FALSE, 1 for TRUE): 1

b3) Cap dels dos avions és militar:

Dades d'entrada	
Nom variable	**Valor entrada**
plane1.id	10
plane1.model	A320
plane1.year	2017
plane1.utility	0
plane1.weight	80
plane1.maxSpeed	910
plane1.maxHeight	11000
plane1.motors	4
plane1.seats	250
plane1.isActive	1
plane2.id	20
plane2.model	B777
plane2.year	2015
plane2.utility	1
plane2.weight	100
plane2.maxSpeed	910
plane2.maxHeight	12000
plane2.motors	6
plane2.seats	275
plane2.isActive	0

Sortida
None of the planes are militar.

Criteris de correcció:

A l’exercici 1:

- Que se segueixi la notació algorísmica utilitzada a l’assignatura. Vegeu document Nomenclator a la xWiki de contingut.
- Que se segueixen les instruccions donades i l’algorisme respongui al problema plantejat.
- Que s’utilitzi correctament l’estructura alternativa i el tipus de dades estructurat.
- Que l'algorisme estigui modularitzat utilitzant accions i funcions
- Que es raoni correctament la resposta de l’apartat c de la primera pregunta.

A l’exercici 2:

- Que el programa s'adeqüi a les indicacions donades.
- Que el programa compila i funciona 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 es declarin els tipus adequats segons el tipus de dades que representa.