Universitat Oberta de Catalunya uoc.edu

Fundamentos de Programación

PEC8 - 20182

Fecha límite de entrega: 29/04/2019

Estudiante

Apellidos: Giménez Gorrís.

Nombre: Álvaro

Objetivos

- Comprender el tipo de datos tabla y saber definirlo correctamente.
- Implementar recorridos y búsquedas dentro de las tablas.
- Profundizar en la modularización del código utilizando acciones y funciones.
- Profundizar en el uso de parámetros de entrada, parámetros de salida y parámetros de entrada / salida.

Formato y fecha de entrega

La PEC se debe entregar antes del día **29 de abril de 2019 a las 23:59**.

Para la entrega se deberá entregar un archivo en formato ZIP, que contenga:

- Este documento con la respuesta del ejercicio 1 y el apartado b del ejercicio 2.
- El workspace de Codeliteque contenga **el proyecto y todas las carpetas creadas** al ejercicio 2a.

Hay que hacer la entrega en el apartado de entregas de EC del aula de teoría.

Enunciado

Siguiendo con la ayuda que proporcionamos a UOCAirways, la compañía nos ha pedido nuestra colaboración para crear un programa que les ayude a gestionar sus aviones. En esta PEC, trabajaremos con el tipo de datos tabla, conjuntamente con la entrada y salida interactiva. Disponemos del siguiente algoritmo, en lenguaje algorítmico, a medio diseñar:

```
type
   tPlaneUtility = {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
    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 este ejercicio de lenguaje algorítmico podéis utilizar las acciones *planeRead*, *planeWrite* y *planeCopy* sin necesidad de diseñarlas.

Ejercicio 1: Diseño en lenguaje algorítmico (50%)

Apartado a: [10%]Define el tipo de datos *tPlanesTable,* en forma de tabla, para guardar los datos de varios aviones. Los datos de cada avión se guardan dentro de los campos del tipo estructurado *tPlane.* El número máximo de aviones queda fijado por la constante *MAX_PLANES.*

```
type
    tPlaneTable = record
        planesTable : vector[MAX_PLANES] of tPlane;
    end record
end type
```

Apartado b: [10%]Define la acción *planesTableInitialize* que recibe como parámetro una tabla de tipo *tPlanesTable* e inicializa el número de elementos a cero. Para hacer lo que pide la acción elige cómo debe ser el parámetro (de entrada, de salida o de entrada / salida).

```
action planeInicialze(inout planesTable : tPlaneTable)

    i : integer;

    for i := 0 to i < MAX_PLANES do

        planesTable.planesTable[i].id := 0;
        planesTable.planesTable[i].model := 0;
        planesTable.planesTable[i].year := 0;
        planesTable.planesTable[i].utility := 0;
        planesTable.planesTable[i].percentOccupationFirstclass := 0;
        planesTable.planesTable[i].percentOccupationBusiness := 0;
        planesTable.planesTable[i].percentOccupationTurist := 0;
        planesTable.planesTable[i].isActive:= 0;

    end for

end action
```

Apartado c: [30%] Define la función *planesTableOccupation* que recibe, como parámetros, una tabla de tipo *tPlanesTable* y un entero *idClass.* Este parámetro *idClass* puede tomar los valores *CODE_FIRSTCLASS, CODE_BUSINESS* o *CODE_TURIST* y representa, respectivamente, cuál de los campos *percentOccupationFirstclass, percentOccupationBusiness o percentOccupationTurist* del tipo *tPlane* debemos considerar. La función debe devolver un real con la ocupación media de la clase indicada por *idClass* de los aviones de la tabla. Por ejemplo, si *idClass* toma el valor CODE_FIRSTCLASS y

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

la función devolverá el valor 50.0

```
function planeTableOccupation(in planesTable : tPlaneTable, in idClass integer, in
planesNum : integer): real

i : integer;
j : integer;
av : real;
average : real;

j:= 0;

if (idClass = 1) then
        for i := 0 to i < planesNum do
                j := j + planesTable.PlanesTable[i].percentOccupationFirstclass;
        end for

        av := (integerToReal) j div (integerToReal) planesNum;
        writeString("The total average of the first class is: ");
        writeReal("av");
        average := av;

else
        if (idClass = 2) then
                for i := 0 to i < planesNum do
                        j := j + planesTable.PlanesTable[i].percentOccupationBusiness;
                end for

                av := (integerToReal) j div (integerToReal) planesNum;
                writeString("The total average of the first class is: ");
                writeReal("av");
                average := av;
        else
                for i := 0 to i < planesNum do
                        j := j + planesTable.PlanesTable[i].percentOccupationTurist;
                end for

                av := (integerToReal) j div (integerToReal) planesNum;
```

```
            writeString("The total average of the first class is: ");
            writeReal("av");
            average := av;
        end if
end if

return(average);

end function
```

Apartado d: [30%] Define la acción *planesTableSelectByOccupation,* que recibe los siguientes parámetros:

- *planesTable* es un parámetro de entrada del tipo *tPlanesTable.*
- *average* es un parámetro de entrada de tipo real.
- *idClass* es un parámetro de entrada de tipo entero.
- *selectTable* es un parámetro de salida del tipo *tPlanesTable.*

idClass tomarávalores *CODE_FIRSTCLASS, CODE_BUSINESS* o *CODE_TURIST* para representar, respectivamente, los campos *percentOccupationFirstclass, percentOccupationBusiness o percentOccupationTurist* del tipo *tPlane.* La acción analiza cada uno de los aviones de *planesTable* y comprueba si el campo representado por *idClass* tiene un valor de ocupación inferior al valor de media (que ha entrado a través de *average).* En caso afirmativo, desplaza el avión a la tabla de salida *selectTable.* Con los aviones que tengan un valor de ocupación igual o superior al valor de *average* no hay que hacer nada.

Nota: Recuerda que hay que inicializar la tabla de salida con la acción *planesTableInitialize* del apartado b.

```
action planesTableSelectByOccupation (in planesTable : tPlaneTable, in average : real, in idClass :
integer, out selectTable : tPlaneTable);
    f : integer;
    i : integer;
    h : real;
    av : real;
    c : integer;

    if (idClass = 1) then
        for i :=0 to i < f do
            f := f + 1;
            h := h + planesTable.planesTable[i].percentOccupationFirstclass;
            av := h div (integerToReal) f;

            if (av = average) then
                for i := 0 to i < f do
```

```
                    if(planesTable.planesTable[i].percentOccupationFirstclass < average)
                    then
                        c := c +1;
                    end if
                end for

                for i := 0 to i < f do
                    if(planesTable.planesTable[i].percentOccupationFirstclass < average)
                    then
                        selectTable.planesTable[i].id := planesTable.planesTable[i].id;
                    end if
                end for

                if (c = 1) then
                    writeString("There are 1 plane under de  average");
                    writeString("Id of the plane: ");
                else
                    writeString("There are" writeInteger("c") "planes under de
                    average");
                    writeString("Id of the planes: ");

                end if

                for i := 0 to I < f do
                    if(planesTable.planesTable[i].percentOccupationFirstclass < average)
                    then
                        writeString("Id: ");
                    end if
                end for
            end if

        end for

else
        if (idClass = 2) then
        for i :=0 to i < f do
            f := f + 1;
            h := h + planesTable.planesTable[i].percentOccupationBusiness;
            av := h div (integerToReal) f;

            if (av = average) then
                for i := 0 to i < f do
                    if(planesTable.planesTable[i].percentOccupationBusiness < average)
                    then
                        c := c +1;
                    end if
                end for

                for i := 0 to i < f do
                    if(planesTable.planesTable[i].percentOccupationBusiness < average)
                    then
                        selectTable.planesTable[i].id := planesTable.planesTable[i].id;
```

```
                end if
            end for

            if (c = 1) then
                writeString("There are 1 plane under de  average");
                writeString("Id of the plane: ");
            else
                writeString("There are" writeInteger("c") "planes under de
                average");
                writeString("Id of the planes: ");

            end if

            for i := 0 to I < f do
                if(planesTable.planesTable[i].percentOccupationBusiness < average)
                then
                    writeString("Id: ");
                end if
            end for
        end if

end for

else
for i :=0 to i < f do
        f := f + 1;
        h := h + planesTable.planesTable[i].percentOccupationTurist;
        av := h div (integerToReal) f;

        if (av = average) then
            for i := 0 to i < f do
                if(planesTable.planesTable[i].percentOccupationTurist < average)
                then
                    c := c +1;
                end if
            end for

            for i := 0 to i < f do
                if(planesTable.planesTable[i].percentOccupationTurist < average)
                then
                    selectTable.planesTable[i].id := planesTable.planesTable[i].id;
                end if
            end for

            if (c = 1) then
                writeString("There are 1 plane under de  average");
                writeString("Id of the plane: ");
            else
                writeString("There are" writeInteger("c") "planes under de
                average");
                writeString("Id of the planes: ");
```

```
                    end if

                    for i := 0 to l < f do
                        if(planesTable.planesTable[i].percentOccupationTurist < average)
                        then
                            writeString("Id: ");
                        end if
                    end for
                end if

            end for

            end if
        end if

end action
```

Apartado e: [20%] Completa el algoritmo que tenemos a medio diseñar haciendo lo siguiente:

a. Declara dos tablas de aviones *planesTable* y *selectTable* de tipo *tPlanesTable*
b. Inicializa las tablas de aviones a cero.
c. Solicita al usuario que introduzca un número de aviones a registrar.
d. Lee los datos de todos los aviones a registrar, guarda dentro de la tabla *planesTable* y actualiza el número de aviones de esta tabla.
e. Solicita al usuario que introduzca el valor de *idClass* que debe servir para determinar para qué clase se realizará el cálculo de la media de ocupación. Comprueba que el valor de *idClass* es correcto, es decir, que presenta los valores *CODE_FIRSTCLASS, CODE_BUSINESS* o *CODE_TURIST.* Si no es así, muestra un mensaje por pantalla avisando del error y vuelve a pedir un nuevo valor, hasta que sea correcto.
f. Calcula la media a través de la función *planesTableOccupation.*
g. Analiza y procesa los aviones de la tabla *planesTable* con la acción *planesTableSelectByOccupation* para tener los elementos seleccionados en la tabla *selectTable.*
h. Muestra por pantalla el número de aviones que hay en la tabla *selectTable.* En caso de que no esté vacía, muestra los campos de los aviones que contiene.

Por ejemplo, supongamos que tenemos la tabla *planesTable* con tres aviones que tienen identificador y porcentajes de ocupación siguientes:

```
plane1.id = 9;
...
plane1.percentOccupationFirstclass: = 100.0;
plane1.percentOccupationBusiness: = 100.0;
```

```
plane1.percentOccupationTurist: = 90.0;
...

plane2.id = 23;
...
plane2.percentOccupationFirstclass: = 50.0;
plane2.percentOccupationBusiness: = 80.0;
plane2.percentOccupationTurist: = 90.0;
...

plane3.id = 51;
...
plane3.percentOccupationFirstclass: = 0.0;
plane3.percentOccupationBusiness: = 90.0;
plane3.percentOccupationTurist: = 90.0;
...
```

Si hacemos el análisis de la primera clase, es decir, *idClass* igual a *CODE_FIRSTCLASS*, la tabla *selectTable* contendría sólo el avión con identificador 51, ya que la media de *percentOccupationFirstclass* es 50% y el avión con identificador 51 tiene un valor de *percentOccupationFirstclass* inferior (igual al 0%).

Si hacemos el análisis de la clase *business,* es decir, *idClass* igual a *CODE_BUSINESS*, la tabla *selectTable* contendría sólo el avión con identificador 23, ya que la media de *percentOccupationBusiness* es 90% y el avión con identificador 23 tiene un valor de *percentOccupationBusiness* inferior (igual al 80%).

Finalmente, si hacemos el análisis de la clase *turist,* es decir, *idClass* igual a *CODE_TURIST*, la tabla *selectTable* no contendría ningún avión ya que la media de *percentOccupationTurist* es 90% y no hay ningún avión con ocupación inferior.

```
Algorithm UOCPLANES

planesTable : tPlaneTable;
selectTable : tPlaneTable;

planesNum : integer;
idClass : integer;
average : real;

writeString(“Enter how many planes do you want to introduce: ”);
readInteger(planesNum);

planesTableInicialize(planesTable);
planesTableInicialize(sselectTable);

for i := 0 to i < planesNum do
```

```
        planeRead(planesTable.planesTable[i]);
end for
writeString("Enter the code of the class that you want to analyze(First class = 1, Business =
2, Turist = 3)");
readInteger(idClass);

planesTableOccupation(planesTable, idClass, planesNum);

planesTableSelectByOccupation(planesTable, average, idClass, selectTable);

end algorithm UOCPLANES
```

Ejercicio 2: Programación en C (50%)

Apartado a: [70%] En este ejercicio hay que codificar en C del ejercicio 1 siguiendo con la estructura de carpetas y ficheros. Concretamente hay que hacer lo siguiente:

1. Descomprime el archivo *PlanesTable_20182.zip* que incluye los archivos del proyecto. El proyecto está estructurado en carpetas: carpeta *include* donde está el archivo ***plane.h***y carpeta *src* donde están los ficheros ***plane.c*** y ***main.c.***
2. Dentro del archivo ***plane.h*** define el tipo de datos *tPlanesTable* (ejercicio 1.a)
3. Dentro del archivo ***plane.h***define las cabeceras de las acciones / funciones *planesTableInitialize, planesTableOccupation, planesTableSelectByOccupation* (ejercicio 1.b, ejercicio 1.c y ejercicio 1. d).
4. Dentro del archivo ***plane.c*** implementa la acción *planesTableInitialize* (ejercicio 1.b)
5. Dentro del archivo ***plane.c*****,** implementa la función *planesTableOccupation* (ejercicio 1.c)
6. Dentro del archivo ***plane.c*** implementa la acción *planesTableSelectByOccupation* (ejercicio 1.d)
7. Dentro del fichero ***main.c*****,** implementa los apartados a-h del ejercicio 1.e.
8. Comprueba el funcionamiento del programa, probando con diferentes valores de ocupación para los aviones y diferentes valores de *idClass.*

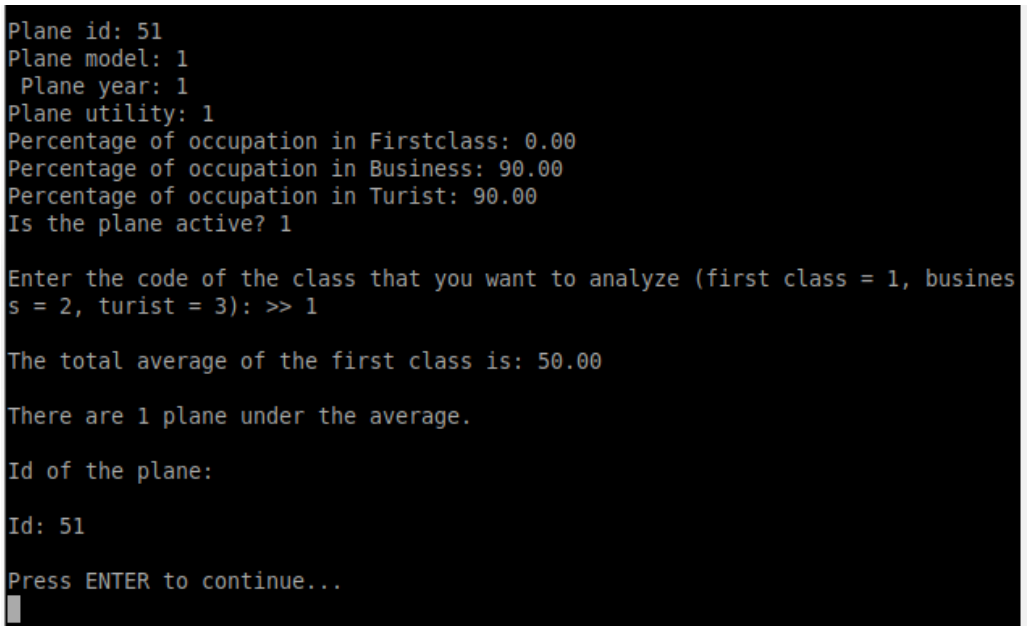
Apartado b: [30%]Como en las anteriores PEC se pide que muestres el funcionamiento del algoritmo **haciendo juegos de prueba.** Es decir que completes las siguientes tablas indicando, para unos valores de avionesque se han introducido en la tabla *planesTable,* qué salida se espera en la ejecución del programa.

b1) La tabla *selectTable* no está vacía (como en el caso *idClass=CODE_FIRSTCLASS* e *idClass=CODE_BUSINESS* del ejemplo):

caso 1:

Datos de entrada	
Nombre de la variable	**Valor de entrada**
Id plane1	9
percentOccupationFirstclass1	100
percentOccupationBusiness1	100
percentOccupationTurist1	90
Id plane2	23
percentOccupationFirstclass2	50
percentOccupationBusiness2	80
percentOccupationTurist2	90
Id plane3	51
percentOccupationFirstclass3	0
percentOccupationBusiness3	90
percentOccupationTurist3	90

Salida
Id : 51



caso 2:

Datos de entrada	
Nombre de la variable	**Valor de entrada**
Id plane1	9
percentOccupationFirstclass1	100
percentOccupationBusiness1	100
percentOccupationTurist1	90
Id plane2	23

percentOccupationFirstclass2	50
percentOccupationBusiness2	80
percentOccupationTurist2	90
Id plane3	51
percentOccupationFirstclass3	0
percentOccupationBusiness3	90
percentOccupationTurist3	90

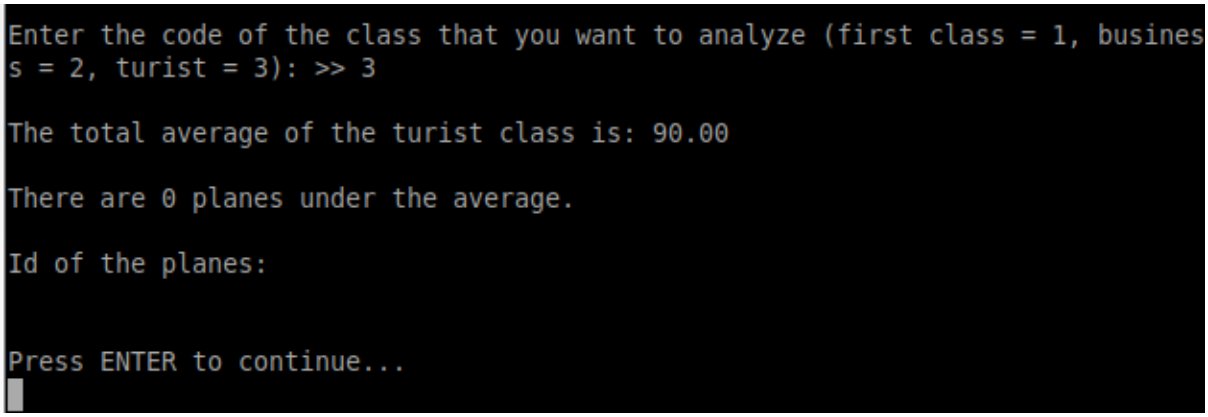
Salida
Id : 23

```
Enter the code of the class that you want to analyze (first class = 1, busines
s = 2, turist = 3): >> 2

The total average of the business class is: 90.00

There are 1 plane under the average.

Id of the plane:

Id: 23

Press ENTER to continue...
```

b2) La tabla *selectTable* está vacía (como en el caso *idClass=CODE_TURIST* del ejemplo):

Datos de entrada	
Nombre de la variable	**Valor de entrada**
Id plane1	9
percentOccupationFirstclass1	100
percentOccupationBusiness1	100
percentOccupationTurist1	90
Id plane2	23
percentOccupationFirstclass2	50
percentOccupationBusiness2	80
percentOccupationTurist2	90
Id plane3	51
percentOccupationFirstclass3	0
percentOccupationBusiness3	90
percentOccupationTurist3	90

Salida



Criterios de corrección:

En el ejercicio 1:

- Que se siga la notación algorítmica utilizada en la asignatura. Véase documento *Nomenclator* la XWiki de contenido.
- Que se sigan las instrucciones dadas y el algoritmo responda al problema planteado.
- Que se diseñen y se llamen correctamente las acciones y funciones demandadas.

En el ejercicio 2:

- Que el programa se adecue a las indicaciones dadas.
- Que el programa compile y funcione de acuerdo con lo que se pide.
- Que se respeten los criterios de estilo de programación C. Véase la G*uía de estilo de programación en C* que tiene en la Wiki de contenido.
- Que se implemente correctamente la modularización del proyecto, dividiendo el código en carpetas y poniendo lo que corresponde a cada carpeta.