Universitat Oberta de Catalunya uoc.edu

Fundamentos de Programación
PEC6 - 20182

Fecha límite de entrega: 08/04/2019

Estudiante
Apellidos: Giménez Gorrís
Nombre: Álvaro

Objetivos

- Saber modularizar el código utilizando acciones y funciones
- Comprender la diferencia entre acción y función.
- Entender que es un parámetro actual y distinguirlo de un parámetro formal.

Formato y fecha de entrega

La PEC se debe entregar antes del día **8 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
- Un workspace de Codelite que contenga los archivos .c pedidos en el 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 la compañía UOCAirways, nos han pedido nuestra colaboración para crear un programa que les ayude a gestionar las peticiones simultáneas de aterrizaje de dos aviones. En los siguientes ejercicios, trabajaremos con funciones y acciones, así como tipos de datos estructurados, conjuntamente con la entrada y salida interactiva para gestionar los datos de los aviones.

Disponemos del siguiente algoritmo, en lenguaje algorítmico, a medio diseñar:

```
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
```

Ejercicio 1: Modularidad [50%]

Apartado a: [10%] Diseña la acción *planeRead* que devuelva un parámetro de tipo *tPlane* tras leer desde el canal estándar de entrada la información de un *avión.*

Nota: En lenguaje algorítmico disponemos de la función *readUtility* que podemos utilizar sin necesidad de implementar.

```
action planeRead (out plane: tPlane)

        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
```

Apartado b: [10%]Diseña la acción *planeWrite* que escriba por el canal estándar de salida la información de un avión. La acción debe tener un parámetro de entrada de tipo *tPlane.*

Nota: En lenguaje algorítmico disponemos de la acción *writeUtility* que podemos utilizar sin necesidad de implementar.

action planeWrite (**in** plane: tPlane)

writeString(“Plane id: ”);
writeInteger(“plane.id”);
writeString(“Plane model: ”);
writeString(“plane.model”);
writeString(“Plane fabrication year: ”);
writeInteger(“plane.year”);
writeString(“Plane utility: ”);
writeUtility(“plane.utility”);
writeString(“Plane weight: ”);
writeReal(“plane.weight”);
writeString(“Maximum Speed of the plane: ”);
writeReal(“plane.maxSpeed”);
writeString(“Maximum height of the plane: ”);
writeInteger(“plane.maxHeight”);
writeString(“Plane motors: ”);
writeInteger(“plane.motors”);
writeString(“Plane seats: ”);
writeInteger(“plane.seats”);
writeString(“Is the plane active?: ”);

end action

Apartado c: [10%]Diseña la acción *planeCopy* que copie los campos de un avión a otro avión. La acción debe tener un parámetro de entrada de tipo *tPlane* (que contiene el avión inicial) y un parámetro de salida de tipo *tPlane* (que contiene el avión donde van a parar los datos que se copian).

action planeCopy(**in** plane: tPlane, **out** plane: tPlane);

plane.id := plane.id;
plane.model := plane.model;
plane.year := plane.year;
plane.utility := plane.utility;
plane.weight := plane.weight;
plane.maxSpeed := plane.maxSpeed;
plane.maxHeight := plane.maxHeight;
plane.motors := plane.motors;
plane.seats := plane.seats;
plane.isActive := plane.isActive**;**

end action

Apartado d: [10%]Diseña la función *planeAnalise* que tiene dos parámetros de entrada *plane1* y *plane2* de tipo *tPlane.* La función devolverá un entero de valor 1 si el avión *plane1* es el más rápido y devolverá 2 si el más rápido es el avión *plane2.* En caso de que los dos aviones tengan la misma velocidad máxima, se devolverá un 1 si es *plane1* el que puede volar más alto y 2 en caso contrario. En caso de que sigan empatados, se devolverá un 0.

```
function planeAnalyze(in plane1: tPlane, in plane2: tPlane): integer

var

        result : integer;

end var

if  (plane1.maxSpeed > plane2.maxSpeed) then

        result := 1;

else
        if (plane1.maxSpeed < plane2.maxSpeed) then

                result := 2;
        else
                if (plane1.maxHeight > plane2.maxHeight) then

                        result := 1;
                else
                        if (plane1.maxHeight < plane2.maxHeight) then
                                result := 2;
                        else
                                result := 0;
                        end if
                end if
        end if
end if

return (result);

end function
```

Apartado e: [40%]Modifica el algoritmo para que la lectura de la información de los aviones se haga a través de la acción *planeRead.* A continuación, completa el

algoritmo para que compruebe si los dos aviones entrados son militares. En caso afirmativo, hay que determinar cuál de los dos aviones puede volar más alto usando la función del apartado anterioryhayque mostrar la información de este avión por el canal estándar de salida (pantalla) con la acción *planeWrite.* Si los dos aviones tienen la misma velocidad, se mostrará por pantalla el avión que puede volar más alto. En caso de que continúe el empate, sólo se mostrará un mensaje diciendo que los dos aviones tienen características idénticas de velocidad y altura máxima. En el caso de que sólo uno de los dos aviones sea militar se escribirá un mensaje por pantalla diciendo cuál de los dos aviones es militar, se copiarán los datos del avión militar a una nueva variable de tipo *tPlane* haciendo uso de la acción *planeCopy* y se escribirá por pantalla la información de esta nueva variable. Finalmente, si ninguno de los dos aviones es militar, se escribirá por pantalla un mensaje diciendo que ninguno de los dos aviones es militar.

```
var
        onlyOnePlane : tPlane;
end var

writeString("Enter data of the firts plane: ");
planeRead(plane1);

writeString("Enter data of the second plane: ");
planeRead(plane2);

if (plane1.utility = 3 and plane2.utility = 3) then
        planeAnalyze(plane1, plane2);
        if (result = 1) then
                writeString("The military plane with highest speed or maximum
                height is the plane 1.");
                writeString("Characteristics of the plane 1: ");
                planeWrite(plane 1);
        else
                if (result = 2) then
                        writeString("The military plane with higuest speed or
                        máximum height is the plane 2.");
                        writeString("Characteristics of the plane 2: ");
```

```
                    planeWrite(plane 2);

            else

                    writeString("Both planes have the same characteristics.");

            end if
        end if

else
        if (plane1.utility = 3 or plane2.utility = 3) then

            if (plane1.utility = 3) then

                    writeString("The only military plane is the plane 1.");
                    planeCopy(plane1, onlyOnePlane);
                    planeWrite(onlyOnePlane);

            else

                    writeString("The only military plane is the plane 2.");
                    planeCopy(plane2, onlyOnePlane);
                    planeWrite(onlyOnePlane);

            end if

        else

            writeString("There are not military planes.");

        end if
end if
```

Apartado f: [20%] Explica cómo deberías modificar *planeAnalize* que se ha diseñado en el apartado *d* para que, dados dos parámetros de entrada *plane1* y *plane2* de tipo *tPlane,* devuelva un tipo *tPlane* que represente el avión que tiene velocidad máxima más elevada. En caso de que los dos aviones tengan la misma velocidad máxima, deberá devolver el avión que pueda volar más alto. En caso de que continúe el empate, podemos suponer que se devolverán los datos del primer avión. (Se pide que razones la respuesta. No es necesario que hagas el diseño)

Se debería de hacer una estructura alternativa que comparase el resultado de la función planeAnalyze, de manera que, si el resultado fuera 0 o 1, usara la acción planeCopy para guardar los datos del avión plane1 en la nueva variable creada en

el apartado anterior, mostrara un mensaje diciendo que el avión más rápido o que vuela más alto es el avión 1 y mostrara sus características por pantalla con la acción planeRead.

En caso contrario, mostraría un mensaje diciendo que el avión más rápido o que vuela más alto el el avión 2 y copiara sus características en la nueva variable mediante la acción planeCopy para finalemente mostrar sus características por pantalla mediante la acción planeRead.

Ejercicio 2: [50%]

Apartado a: [70%] Implementa en lenguaje C, el algoritmo del ejercicio anterior.

Nota: Recuerda que, en lenguaje C, hay que utilizar funciones específicas para poder copiar cadenas de caracteres. Concretamente, se puede utilizar el método *strcpy* de la librería *string.h,* que permite copiar dos strings.

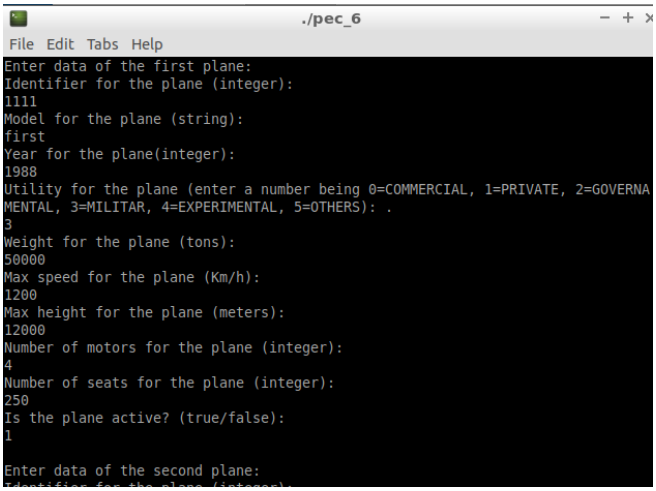
Apartado b: [30%] Como en las anteriores PEC se pide que muestres el funcionamiento del algoritmo **haciendo juegos de prueba.** Es decir que completes las siguientes tablas indicando, para unos valores de avionesque se han introducido en las diferentes estructuras, qué salida se espera en la ejecución del programa. Después comprueba que efectivamente el programa hace lo que se esperaba. Puedes añadir más filas en las tablas.

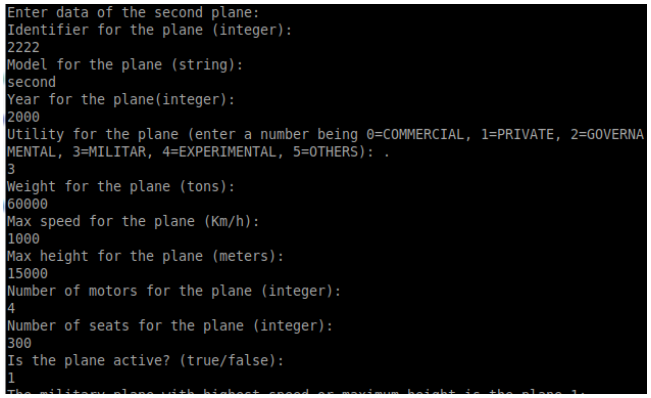
b1) Los dos aviones son militares:

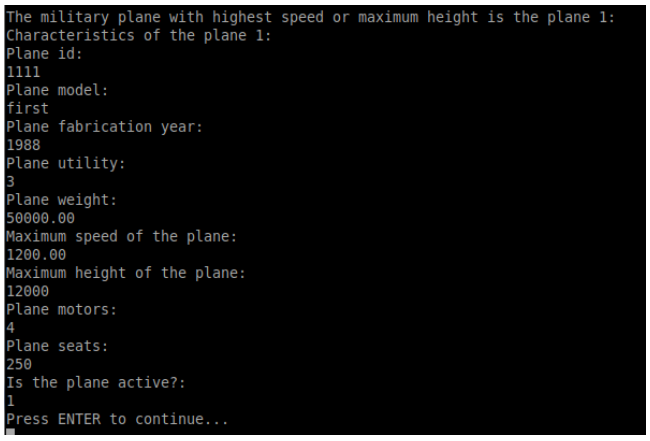
Datos de entrada	
Nombre variable	**Valor entrada**
Id1:	1111
Model1:	First
Year1:	1988
Utility1:	3
Weight1:	50000
maxSpeed1:	1200
maxHeight1:	12000
Motors1:	4
Seats1:	250
isActive1:	1
Id2:	2222
Model2:	Second
Year2:	2000
Utility2:	3
Weight2	60000
maxSpeed2:	1000

maxHeight2:	15000
Motors2:	4
Seats2:	300
isActive2:	1

Salida
Characteristics of the plane 1



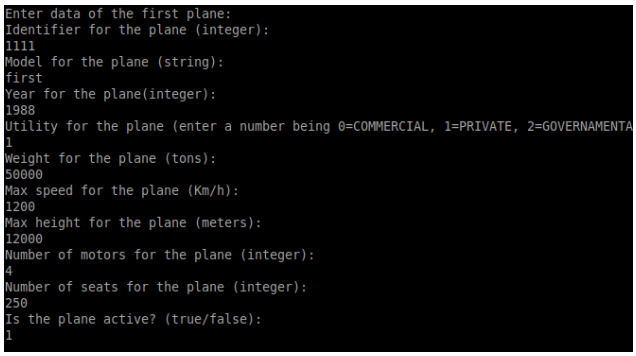


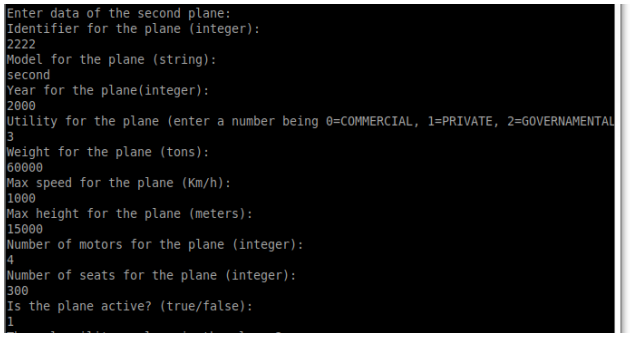


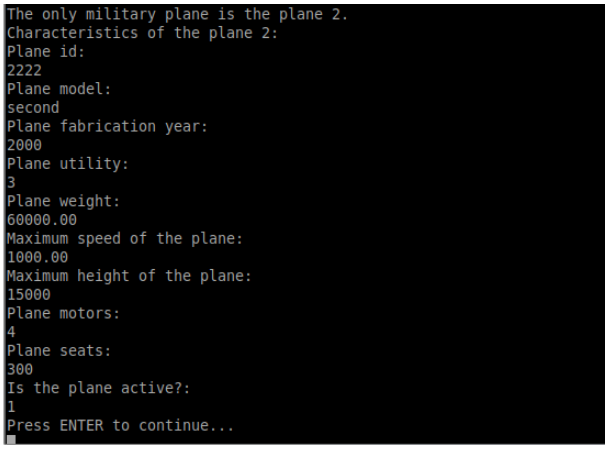
b2) Sólo uno de los dos aviones es militar:

Datos de entrada	
Nombre variable	**Valor entrada**
Id1:	1111
Model1:	First
Year1:	1988
Utility1:	1
Weight1:	50000
maxSpeed1:	1200
maxHeight1:	12000
Motors1:	4
Seats1:	250
isActive1:	1
Id2:	2222
Model2:	Second
Year2:	2000
Utility2:	3
Weight2	60000
maxSpeed2:	1000
maxHeight2:	15000
Motors2:	4
Seats2:	300
isActive2:	1

Salida
Characteristics of the plane 2



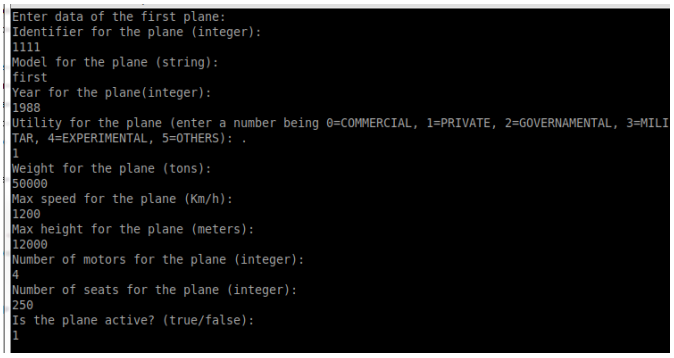


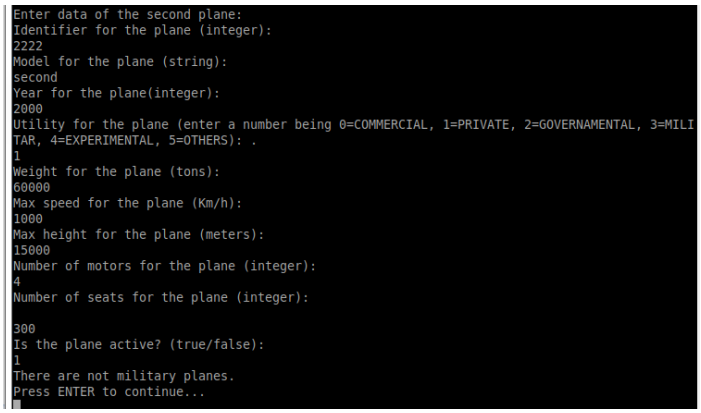


b3) Ninguno de los dos aviones es militar:

Datos de entrada	
Nombre variable	**Valor entrada**
Id1:	1111
Model1:	First
Year1:	1988
Utility1:	1
Weight1:	50000
maxSpeed1:	1200
maxHeight1:	12000
Motors1:	4
Seats1:	250
isActive1:	1
Id2:	2222
Model2:	Second
Year2:	2000
Utility2:	1000
Weight2	60000
maxSpeed2:	1
maxHeight2:	15000
Motors2:	4
Seats2:	300
isActive2:	1

Salida
There are not military planes





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 siguen las instrucciones dadas y el algoritmo responda al problema planteado.
- Que se utilice correctamente la estructura alternativa y el tipo de datos estructurado.
- Que el algoritmo esté modularizado utilizando acciones y funciones

- Que se razone correctamente la respuesta del apartado f de la primera pregunta.

En el ejercicio 2:

- Que el programa se adecue a las indicaciones dadas.
- Que el programa compila y funciona de acuerdo con lo que se pide.
- Que se respeten los criterios de estilo de programación C. Véase la Guía de estilo de programación en C que tiene en la Wiki de contenido.
- Que se declaren los tipos adecuados según el tipo de datos que representa.