Universitat Oberta de Catalunya uoc.edu

Fundamentos de Programación
PEC5 - 20182

Fecha límite de entrega: 01/04/2019

Estudiante
Apellidos: Giménez Gorrís.
Nombre: Álvaro.

Objetivos

- Saber definir correctamente los tipos de datos estructurados.
- Aprender a utilizar la entrada y salida para modificar campos concretos de estructuras de datos sencillos.
- Saber utilizar la as funciones de conversión de tipos cuando sea necesario.

Formato y fecha de entrega

La PEC se ha de entregar antes del día **1 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 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 la compañía UOCAirways, nos han pedido nuestra colaboración para crear un programa que les ayude a gestionar las bases de datos de sus aviones. En los siguientes ejercicios, trabajaremos con tipos de datos estructurados, conjuntamente con la entrada y salida interactiva para gestionar los datos de los aviones.

Nota: veréis que los datos que se piden de los aviones pueden variar ligeramente respecto a PECs anteriores, según el objetivo del ejercicio.

Ejercicio 1: Tipos estructurados de datos [50%]

Apartado a: [20%] Definir en **lenguaje algorítmico** el tipo de datos estructurado *tPlane,* que representa la información de un avión. Los datos que se quieren guardar son las siguientes:

- *id.* Identificador del avión. Se define como un entero.
- *modelo.* Nombre del modelo del avión. Se define como un *string*, con una longitud máxima de 15 caracteres.
- *year.* Año de fabricación del avión. Se define como un entero.
- utility. Uso que se le da al avión, que puede ser COMMERCIAL, PRIVATE, GOVERNMENTAL, MILITAR, EXPERIMENTAL o OTHERS. Hay que definir previamente un tipo enumerado que contenga los valores posibles.
- *weight.* Un real que indica el peso del avión, en toneladas (sin pasaje, ni combustible, ni carga).
- *maxSpeed.* Un real que indica la velocidad máxima del avión (en Km / h)
- *maxHeight.* Un entero que indica la altura máxima a la que puede llegar el avión (en metros).
- motores. Un entero que indica el número de motores del avión.
- seats. Un entero que indica el número de asientos.
- isActive. Un booleano que indica si el avión está en activo o no.

Apartat b: [60%] Implementar en lenguaje algorítmico un algoritmo que permita dar de alta de manera interactiva dos nuevos aviones, y guardar cada uno de los aviones en una variable diferente. Para trabajar con estos dos aviones, no se deben utilizar vectores.

El algoritmo va pidiendo al usuario que introduzca el valor de cada campo del avión mediante la entrada y salida estándar (teclado / pantalla), y lo guarda en el campo correspondiente de la estructura de datos definida en el apartado anterior.
Previamente a la lectura de cada uno de los campos, se mostrará por el canal estándar de salida un mensaje informativo de ayuda que indique al usuario los valores posibles u otras informaciones de interés.

Para leer los datos del tipo *tUtility*, en lenguaje algorítmico disponemos de la función *readUtility().* Una vez se han leído todos los datos de cada avión, el algoritmo debe mostrar por pantalla:

- El identificador del avión que tiene un número mayor de asientos entre los aviones en activo y comerciales entrados. En caso de dos aviones activos y comerciales que tengan el mismo número de asientos, se mostrará por pantalla el identificador del primer avión que se ha entrado por el canal estándar. En caso de que ninguno de los dos aviones sea comercial y esté en activo, en lugar de identificador, se escribirá un mensaje diciendo que ninguno de los dos aviones entrados están en activo y son comerciales a la vez.
- El número total de asientos entre los aviones en activo y comerciales entrados. En caso de que ninguno de los dos aviones esté en activo y sea comercial escribirá el valor cero.

```
algorithm UOCAirways

type

    tUtility = {COMMERCIAL, PRIVATE, GOVERNAMENTAL, MILITAR,
    EXPERIMENTAL, OTHERS}

    tPlane = record

        id : integer;
        model : string;
        year : integer;
        utility : tUtility;
        weight : real;
        maxSpeed : real;
        maxHeight : integer;
        engines : integer;
        seats : integer;
        isActive : boolean;

    end record

end type

var

    tPlane1, tPlane2, temp : tPlane;

end var
```

{datos del primer avión}

writeString(“Enter the data of the first plane: ”);

writeString(“Enter the id of the first plane: ”);
writeString(“(Enter an integer)”);
tPlane1.id := **readInteger**();

writeString(“Enter the model of the first plane: ”);
writeString(“(Enter a string with maximum 15 characters)”);
tPlane1.model := **readString**();

writeString(“Enter the year of the first plane: ”);
writeString(“(Enter an integer)”);
tPlane1.year := **readInteger**();

writeString(“Enter the utility of the first plane: ”);
writeString(“(Enter 0 if is commercial, 1 if is private, 2 if is governamental, 3 if is militar, 4 if is experimental or 5 if is other)”);
tPlane1.utility := **readUtility**();

writeString(“Enter the weight of the first plane: ”);
writeString(“(Enter a real)”);
tPlane1.weight := **readReal**();

writeString(“Enter maximum speed of the first plane: ”);
writeString(“(Enter a real)”);
tPlane1.maxSpeed := **readReal**();

writeString(“Enter the máximum height of the first plane: ”);
writeString(“(Enter an integer)”);
tPlane1.maxHeight := **readInteger**();

writeString(“Enter the number of engines of the first plane: ”);
writeString(“(Enter an integer)”);
tPlane1.engines := **readInteger**();

writeString(“Enter the number of seats of the first plane: ”);
writeString(“(Enter an integer)”);
tPlane1.seats := **readInteger**();

writeString(“Enter if is active or not the first plane: ”);
writeString(“(Enter 0 if is false and 1 if is true)”);
tPlane1.isActive := **readBoolean**();

{datos del segundo avión}

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

writeString("Enter the id of the second plane: ");
writeString("(Enter an integer)");
tPlane2.id := readInteger();

writeString("Enter the model of the second plane: ");
writeString("(Enter a string with maximum 15 characters)");
tPlane2.model := readString();

writeString("Enter the year of the second plane: ");
writeString("(Enter an integer)");
tPlane2.year := readInteger();

writeString("Enter the utility of the second plane: ");
writeString("(Enter 0 if is commercial, 1 if is private, 2 if is governamental, 3 if is militar, 4 if
is experimental or 5 if is other)");
tPlane2.utility := readUtility();

writeString("Enter the weight of the second plane: ");
writeString("(Enter a real)");
tPlane2.weight := readReal();

writeString("Enter maximum speed of the second plane: ");
writeString("(Enter a real)");
tPlane2.maxSpeed := readReal();

writeString("Enter the maximum height of the second plane: ");
writeString("(Enter an integer)");
tPlane2.maxHeight := readInteger();

writeString("Enter the number of engines of the second plane: ");
writeString("(Enter an integer)");
tPlane2.engines := readInteger();

writeString("Enter the number of seats of the second plane: ");
writeString("(Enter an integer)");
tPlane2.seats := readInteger();

writeString("Enter if is active or not the second plane: ");
writeString("(Enter 0 if is false and 1 if is true)");
tPlane2.isActive := readBoolean();

if ((tPlane1.isActive = 1 and tPlane1.utility = 0) or (tPlane2.isActive = 1 and tPlane2.utility =
0) ) then
        if(tPlane1.seats = tPlane2.seats) then
                writeString("First and second planes have the same number of seats.");
```

```
        else
            if(tPlane1.seats > tPlane2.seats) then
                writeString("Id of the plane that have more number of seats are:
                ");
                writeInteger(tPlane1.id);
            else
                writeString("Id of the plane that have more number of seats are:
                ");
                writeInteger(tPlane2.id);
            end if
        end if

        writeString("Total number of seats of planes that are in active and commercial at
        the same time are: ");
        writeInteger(tPlane1.seats + tPlane2.seats);

else
        writeString("There are not planes that are in active and comercial at the same
        time.");
        writeString("Total number of seats of planes that are in active and commercial at
        the same time are: 0");

end if

end algorithm
```

Apartado c: [20%] Explica cómo se podría enfocar este ejercicio para poder dar de alta de manera interactiva un elevado número de aviones. No hay que hacer el diseño, simplemente tenéis que razonar la propuesta.

Para que el algoritmo nos fuera útil para un número elevado de aviones, tendríamos que utilizar parte del algoritmo de la anterior Pec.

En primer lugar, habría que declara un vector que tendría como valor una constante cuyo valor sería el máximo declarado por el programa, que en la anterior Pec era 10.

A continuación, el programa pediría al usuario que introdujera la cantidad de aviones que desea guardar y comprobaría que dicho valor entra en los valores predeterminados, declarando un ciclo for que pidiera la introducción de los datos de cada avión desde 1 hasta el número introducido por el usuario como en la Pec anterior.

Para poder mostrar el número de asientos del total de aviones que son de tipo comercial y estén activos, declararía un ciclo for que iría desde 1 hasta el número de aviones introducido por el usuario. En dicho ciclo for, introduciría un ciclo if que compararía si el avión del actual ciclo for está activo y es comercial. En caso afirmativo, la cantidad de

asientos de dicho vector se sumaría a una nueva variable. Por ejemplo, si la nueva variable fuera a, el programa declararía que a = a + los asientos de dicho vector. De esta forma, se sumarían todos los asientos que cumplieran dicha condición.

A la finalización de dicho ciclo for, mostraría el valor de dicha variable (a) por el canal estándar de salida.

Para poder mostrar la id del avión que más asientos tiene en esa condición, añadiría al anterior ciclo for las siguientes características:

En primer lugar compararía si el ciclo for actual (por ejemplo: i), es **diferente** del número de aviones que ha introducido el usuario.

En caso afirmativo, compararía el avión del ciclo for actual mas 1 (i+1) con el número de aviones que quiere introducir el usuario. Si el resultado de la comparación es menor o igual que el número de aviones introducido por el usuario, haría una tercera comparación en la que compararía si el ciclo actual (i) es mayor que el ciclo siguiente (i+1). En caso afirmativo, guardaría el valor de i en una nueva variable (por ejemplo: i = n).
De esta manera, se compararan uno a uno todos los números menos el último. Para solucionar esto, en caso de que la comparación de i es diferente del número de aviones introducido por el usuario sea falsa, el programa compararía la variable declarada anteriormente (n) con el número actual del ciclo for, que en este caso sería el último del vector. En caso de que el último valor del vector sea mayor que n, su valor se guardaría en la variable n. De esta manera quedarían comparados todos los aviones entre si y quedaría guardado el valor del que tuviera el mayor número de asientos en la variable n.

A continuación del ciclo for imprimiría si fuera necesario la id del avión de la posición del vector de la variable declarada anteriormente (n).

De esta forma, el algoritmo cumpliría con todas las necesidades pedidas y sería totalmente adaptable a un número de aviones aleatorio, con la única condición de que habría que establecer un número máximo de aviones como en la Pec anterior.

Ejercicio 2: Programación en C [50%]

Apartado a: [70%] Implementad, en **lenguaje C**, el algoritmo del ejercicio anterior. Recordad que en lenguaje C es necesario definir previamente la longitud máxima de las cadenas de caracteres, típicamente mediante una constante.

Apartado b: [30%] Como en las anteriores PEC se pide que mostréis el funcionamiento del algoritmo **haciendo juegos de prueba.** Es decir que completéis las siguientes tablas indicando, para unos valores de aviones que se han entrado en las diferentes estructuras, qué salida se espera en la ejecución del programa. Después debéis comprobar que efectivamente el programa hace lo que esperaba. Podéis añadir más filas a las tablas.

b1) Alguno de los dos aviones introducidos está en activo y es comercial:

Datos de entrada	
Nombre variable	**Valor entrada**
Id 1	1111
Model 1	first
Year 1	1988
Utility 1	0
Weight 1	50000
maxSpeed 1	1500
maxHeight 1	12000
Engines 1	4
Seats 1	100
isActive 1	1
Id 2	2222
Model 2	Second
Year 2	1988
Utility 2	2
Weight 2	60000
maxSpeed 2	1600
maxHeight 2	14000
Engines 2	4
Seats 2	200
Isactive 2	0

Datos de salida

b2) Los dos aviones introducidos están en activo y son comerciales, y tienen el mismo número de asientos:

Datos de entrada	
Nombre variable	**Valor entrada**
Id 1	1111
Model 1	first
Year 1	1988
Utility 1	0
Weight 1	50000
maxSpeed 1	1500
maxHeight 1	12000
Engines 1	4
Seats 1	200
isActive 1	1
Id 2	2222
Model 2	Second
Year 2	1988

Utility 2	0
Weight 2	60000
maxSpeed 2	1600
maxHeight 2	14000
Engines 2	4
Seats 2	100
Isactive 2	1

Datos de salida

b3) Ninguno de los dos aviones introducidos está en activo y es comercial a la vez.

Datos de entrada	
Nombre variable	**Valor entrada**
Id 1	1111
Model 1	first
Year 1	1988
Utility 1	3
Weight 1	50000
maxSpeed 1	1500
maxHeight 1	12000
Engines 1	4
Seats 1	200
isActive 1	0
Id 2	2222
Model 2	Second
Year 2	1988
Utility 2	2
Weight 2	60000
maxSpeed 2	1600
maxHeight 2	14000
Engines 2	4
Seats 2	100
Isactive 2	0

Dades de sortida

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 se razone correctamente la respuesta del apartado c del ejercicio 1.

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 Guía de estilo de programación en C que tenéis en la xWiki de contenido.
- Que se declaren los tipos adecuados según el tipo de datos que representa.