Fundamentos de Programación
PEC6 - 20191

Fecha límite de entrega: 11/11/2019

Estudiante
Apellidos: PABLO JOSÉ
Nombre: DELGADO FLORES

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 **11 de noviembre 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 UOCBookings, ahora solicitan nuestra colaboración para crear un programa que les ayude a gestionar algunos aspectos de sus hoteles.

Nota: En el algoritmo, todas las cadenas de caracteres (strings) deben ser sin espacios en blanco. En caso de querer poner un nombre compuesto, como por ejemplo New York, se deberá entrar New_York.

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

```
const
        MAX_BRAND: integer = 15;
        MAX_NAME: integer = 15;
        MAX_CITY: integer = 15;
end const

type
   tTypeHotel = {BUDGET, INN, RESORT, CONDO, LUXURY, COUNTRY}
      tHotel = record
        id: integer;
        brand:string;
        name:string;
        type: tTypeHotel;
        city: string;
        category:integer;
        priceDouble:real;
        distanceFromCityCenter: real;
        hasPool: boolean;
        hasGym: boolean;
        closeToSubway: boolean;
        percentOccupation: real;
    end record
end type


algorithm UOCBookings
var
   h1:  tHotel;
   h2:  tHotel;

end var

   {input data for hotel 1}

   ...

   {input data for hotel  2}
```

...

{Algorithm to complete …}

end algorithm

y disponemos de las siguientes acciones ya diseñadas que podéis usar para leer y escribir los datos de los hoteles.

```
action hotelRead(out h:tHotel);

action hotelWrite(in h:tHotel);
```

Ejercicio 1: Modularidad [50%]

Apartado a: [50%] Diseña la función *hotelCmp* que debe tener dos parámetros de entrada de tipo tHotel, h1 y h2, y que debe retornar

-1 si h1 < h2
0 si h1 = h2
1 si h1 > h2

El orden que debe tener en cuenta la función es el siguiente

- brand (orden alfabético)
- category (el de menos estrellas va primero)
- priceDouble (el más barato va primero)
- distanceFromCityCenter (el más cercano va primero)
- percentOccupation (la ocupación más baja va primero)
- closeToSubway (estar cerca del metro va primero)
- hasPool (no tener piscina va primero)
- hasGym (no tener gimnasio va primero)

Apartado b: [20%] Diseñar la función *hotelAcceptable* que retorna cierto si el hotel tiene piscina o gimnasio, está a menos de x Km del centro o tiene metro cerca y el precio es inferior a y €.

Es decir, lo que se pide es convertir la expresión del apartado b3 de la PEC2 en una función, que además de recibir un parámetro del tipo *tHotel*, reciba también dos parámetros *price* y *distance* de tipo *real,* que representen respectivamente el precio

deseado que se quiere pagar y la distancia máxima deseada del centro donde se quiere el hotel.

Apartado c: [30%] Completar el algoritmo de tal manera que

- lea los datos de los dos hoteles.
- Lea el precio máximo y la distancia máxima aceptables.
- Calcule si cada uno de los hoteles es aceptable, llamando a la función *hotelAcceptable* con los datos de cada uno de ellos.
- Si los dos hoteles son aceptables y están en la misma ciudad los compare utilizando la función *hotelCmp* y muestre los datos de los dos hoteles por el canal estándar de salida de tal manera que muestre el mayor primero, es decir,

  si h1 ≥ h2, mostrar primero los datos de h1 y después los de h2

  en caso contrario muestre primero los datos del hotel h2 y después los de h1.

.

  En caso contrario muestre un mensaje por el canal estándar de salida indicando que no se pueden comparar.

SOLUCIÓN

```
const
      MAX_BRAND: integer = 15;
      MAX_NAME: integer = 15;
      MAX_CITY: integer = 15;
end const

type
    tTypeHotel = {BUDGET, INN, RESORT, CONDO, LUXURY, COUNTRY}
    tHotel = record
        id: integer;
        brand: string;
        name: string;
        type: tTypeHotel;
        city: string;
        category: integer;
        priceDouble: real;
        distanceFromCityCenter: real;
        hasPool: boolean;
        hasGym: boolean;
        closeToSubway: boolean;
        percentOccupation: real;
    end record
end type

function hotelCmp(h1:tHotel,h2:tHotel):integer
    var
        bestHotelF: integer;
    end var
    bestHotelF:=0;
    if (h1.brand<h2.brand) then
        bestHotelF:=-1;
    else
        if (h2.brand>h1.brand) then
            bestHotel:=1;
            else
                if (h1.category<h2.category) then
                    bestHotel:=-1;
                else
                    if (h2.category<h1.category) then
                        bestHotel:=1;
                    else
                        if (h1.priceDouble<h2.priceDouble) then
                            bestHotel:=-1;
                        else
                            if (h2.priceDouble<h1.priceDouble) then
                                bestHotel:=1;
                            else
                                if
(h1.distanceFromCityCenter<h2.distanceFromCityCenter) then
                                    bestHotel:=-1;
                                else
```

```
                                    if
(h2.distanceFromCityCenter<h1.distanceFromCityCenter) then
                                        bestHotel:=1;
                                    else
                                        if
(h1.percentOccupation<h2.percentOccupation) then
                                            bestHotel:=-1;
                                        else
                                            if
(h2.percentOccupation<h1.percentOccupation) then
                                                bestHotel:=1;
                                            else
                                                if
(h1.closeToSubway<h2.closeToSubway) then
                                                    bestHotel:=-1;
                                                else
                                                    if
(h2.closeToSubway<h1.closeToSubway) then
                                                        bestHotel:=1;
                                                    else
                                                        if
(h1.hasPool=false) then
                                                            bestHotel:=-1;
                                                        else
                                                            if
(h2.hasPool=false) then

bestHotel:=1;
                                                            else
                                                                if
(h1.hasGym=false) then

bestHotel:=-1;
                                                                else
                                                                    if
(h2.hasGym=false) then

bestHotel:=1;
                                                                    else

bestHotel:=0;
                                                                    end if
                                                                end if
                                                            end if
                                                        end if
                                                    end if
                                                end if
                                            end if
                                        end if
                                    end if
                                end if
                            end if
                        end if
                    end if
```

```
                end if
            end if
        end if
        return bestHotelF;
end function

function
hotelAcceptable(h:tHotel,acceptablePrice:float,acceptableDistance:float):b
oolean
        var
            hAccepF: boolean;
        end var
        if (((h.hasPool=true) or (h.hasGym=true)) and
((h.distanceFromCityCenter<acceptableDistance) or (h.closeToSubway=true))
and (h.priceDouble<acceptableDistance)) then
            hAccepF=true;
        else
            hAccepF=false;
        end if
        return hAccepF;
end function

algorithm UOCBookings
        var
            h1: tHotel;
            h2: tHotel;
            h: tHotel;
            bestHotel: integer;
            acceptablePrice: real;
            acceptableDistance: real;
            isH1Acceptable: boolean;
            isH2Acceptable: boolean;
        end var
        writeString(”ENTER DATA FOR HOTEL 1: ”);
        hotelRead(in h1:tHotel);
        writeString(”ENTER DATA FOR HOTEL 2: ”);
        hotelRead(in h2:tHotel);
        writeString(”WHAT’S THE MAXIMUM ACCEPTABLE PRICE? ”);
        acceptablePrice:=readReal();
        writeString(”WHAT’S THE MAXIMUM ACCEPTABLE DISTANCE FROM CITY CENTER?
”);
        acceptableDistance:=readReal();
        isH1Acceptable:=hotelAcceptable(h1, acceptablePrice,
acceptableDistance);
        isH2Acceptable:=hotelAcceptable(h2, acceptablePrice,
acceptableDistance);
        if ((isH1Acceptable=true) and (isH2Acceptable=true) and
(h1.city=h2.city)) then
            bestHotel:=hotelCmp(h1,h2);
            if (bestHotel=-1) then
                writeString(”THE HOTEL 1 SUITS YOU BETTER, AND THE DATA ARE”);
                hotelWrite(out h1:tHotel);
                writeString(”THE SECOND BEST HOTEL IS HOTEL 2 AND THE DATA
ARE”);
```

```
            hotelWrite(out h2:tHotel);
        else
            if (bestHotel=1) then
                writeString(”THE HOTEL 2 SUITS YOU BETTER, AND THE DATA
ARE”);
                hotelWrite(out h2:tHotel);
                writeString(”THE SECOND BEST HOTEL IS HOTEL 1 AND THE DATA
ARE”);
                hotelWrite(out h1:tHotel);
            else
                writeString(”THE HOTELS CAN’T BE COMPARED”);
            end if
        end if
        writeString(”THE HOTELS CAN’T BE COMPARED”);
    end if


end algorithm
```

Ejercicio 2: [50%]

Apartado a: [80%] 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: [20%] Como en las anteriores PEC se pide que deis 4 juegos de prueba**.** Es decir que completéis las tablas siguientes indicando porque consideráis interesantes los datos utilizados para probar

b1)

En esta primera prueba voy introducir los datos de dos hoteles exactamente iguales, solo que en ciudades distintas para que no entre a hacer la función de **hotelCmp**.

Datos de entrada		
Hotel	**Nombre variable**	**Valor entrada**
1	id	111
	brand	VivaCadiz
	name	Caleta
	city	Cadiz
	category	4
	type	4

	PriceDouble	120
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
2	id	222
	brand	VivaHuelva
	name	Conquero
	city	Huelva
	category	4
	type	4
	PriceDouble	120
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
-	acceptablePrice	150
-	acceptableDistance	6

Salida
THE HOTELS CAN'T BE COMPARED

b2)

En la segunda prueba, voy a poner todos los datos iguales para que también me diga que no se pueden comparar. Va a entrar en las funciones **hotelCmp** y **hotelAcceptable** sin éxito.

Datos de entrada		
Hotel	**Nombre variable**	**Valor entrada**
1	id	111
	brand	VivaCadiz
	name	Caleta
	city	Cadiz
	category	4
	type	4

	PriceDouble	120
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
2	id	222
	brand	VivaCadiz
	name	PuertaMar
	city	Cadiz
	category	4
	type	4
	PriceDouble	120
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
-	acceptablePrice	150
-	acceptableDistance	6

Salida
THE HOTELS CAN'T BE COMPARED

b3)

En la tercera prueba, introduciré los datos para que el segundo hotel sea mejor que el primero y así se muestre en pantalla..

Datos de entrada		
Hotel	**Nombre variable**	**Valor entrada**
1	id	111
	brand	VivaCadiz
	name	Caleta
	city	Cadiz
	category	4
	type	4
	PriceDouble	120

	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
2	id	222
	brand	VivaCadiz
	name	PuertaMar
	city	Cadiz
	category	4
	type	4
	PriceDouble	100
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
-	acceptablePrice	150
-	acceptableDistance	6

Salida
THE HOTEL 2 SUITS YOU BETTER, AND THE DATA ARE {*DATOS_HOTEL2*} THE SECOND BEST HOTEL IS HOTEL 1 AND THE DATA ARE {*DATOS_HOTEL1*}

b4)

Por último, sacaré como mejor hotel el primero.

Datos de entrada		
Hotel	**Nombre variable**	**Valor entrada**
1	id	111
	brand	VivaCadiz
	name	Caleta
	city	Cadiz
	category	4
	type	4
	PriceDouble	120

	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
2	id	222
	brand	VivaCadiz
	name	PuertaMar
	city	Cadiz
	category	4
	type	4
	PriceDouble	145
	distanceFromCityCenter	5.2
	closeToSubway	0
	hasPool	1
	hasGym	1
	percentOccupation	25
-	acceptablePrice	150
-	acceptableDistance	6

Salida
THE HOTEL 1 SUITS YOU BETTER, AND THE DATA ARE {*DATOS_HOTEL1*} THE SECOND BEST HOTEL IS HOTEL 2 AND THE DATA ARE {*DATOS_HOTEL2*}

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

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.