

Fundamentos de Programación

PEC6 - 20191

Fecha límite de entrega: 11/11/2019

Estudiante

Apellidos:

Nombre:

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 hoteles por el canal estándar de salida de forma ordenada. Es decir

Si $h1 \leq h2$ mostrar los datos del hotel h1 primero

En caso contrario mostrar los datos del hotel h2 primero

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

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)

Datos de entrada	
Nombre variable	Valor entrada
...	...

Salida
...

b2)

Datos de entrada	
Nombre variable	Valor entrada
...	...

Salida
...

b3)

Datos de entrada	
Nombre variable	Valor entrada
...	...

Salida
...

b4)

Datos de entrada	
Nombre variable	Valor entrada
...	...

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 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.