Fundamentos de Programación
PEC6 - 20211

Fecha límite de entrega: **08/11/2021 a las 23:59**

Apellidos: _____
Nombre: _____

Entrega

La PEC deberá entregarse antes del **día 08 del 11 a las 23:59**.

La entrega debe realizarse en el apartado de **entregas de EC** del Aula de teoría. Se corregirá **únicamente** la **última versión** entregada dentro del plazo establecido.

Se debe entregar un único archivo en **formato ZIP**, que contenga:

- Un documento, en **formato PDF**, con el diseño algorítmico. No es necesario incluir todo el enunciado, solo las respuestas.
- El ***workspace* de Codelite**, con el proyecto en C del Ejercicio 2, tal y como se explica en el apartado correspondiente de la *xWiki*.

No respetar el formato de entrega puede conllevar que la actividad no pueda ser corregida, y en cualquier caso **penalizará** su evaluación.

A continuación se citan algunos ejemplos de formatos de entrega incorrectos:

- Documentos en formato distinto a PDF (.docx, .odt, .rtf, .txt, .pages, etc.).
- Ficheros y carpetas sueltos del proyecto en C (.c, .h, .workspace, etc.).

Actualizaciones del enunciado

Cualquier **aclaración**, actualización o corrección de posibles errores del enunciado se publicará en los **tablones del aula de teoría**. Es importante tener en cuenta estas aclaraciones para resolver la actividad, ya que **en caso de discrepancia, tendrán preferencia sobre el enunciado original**.

Objetivos de aprendizaje

Los **objetivos de aprendizaje** de esta PEC son los que se indican a continuación. Estos objetivos de aprendizaje constituyen a su vez, los **indicadores** en los que se basará la corrección y evaluación de la actividad. Los objetivos de aprendizaje **en negrita** aparecen por primera vez en esta PEC.

Tratamiento de datos 20%

- TD4 - Ser capaz de definir y utilizar tipos estructurados de datos de forma correcta 40%
- Indicadores de E/S 40%:
  - TD7 - Ser capaz de leer datos por el canal estándar de forma correcta
  - TD8 - Ser capaz de mostrar datos por el canal estándar de forma correcta
- Resto de indicadores de tratamiento de datos 20%:
  - TD1 - Ser capaz de elegir los tipos básicos de datos de forma correcta
  - TD2 - Ser capaz de definir y usar los tipos enumerativos de forma correcta
  - TD5 - Ser capaz de declarar y usar variables de forma correcta
  - TD6 - Ser capaz de definir y usar constantes de forma correcta

Diseño algorítmico 50%

- DA1 - Ser capaz de construir expresiones lógicas correctas y compactas para tratar la información a partir de datos de forma correcta 15%
- DA2 - Ser capaz de construir estructuras alternativas compactas y funcionales, y diseñar árboles de decisión 15%
- DA4 - Ser capaz de diseñar un algoritmo funcional y sencillo que dé solución al problema planteado a través un flujo de ejecución de sentencias óptimo 10%
- **DA5 - Ser capaz de declarar y utilizar acciones y funciones de forma correcta 50%**
- DA7 - Ser capaz de utilizar la notación algorítmica de forma adecuada y aplicar las buenas prácticas en cuanto a la nomenclatura y el uso de comentario. 10%

Codificación 20%

- CO1 - Ser capaz de construir un programa en C plenamente funcional, partiendo de un algoritmo diseñado previamente adaptándolo a las particularidades del lenguaje C 40%
- CO3 - Ser capaz de seguir las normas de estilo establecidas para el lenguaje C 10%
- CO4 - Ser capaz de superar los juegos de pruebas y casos extremos con éxito 50%

Herramientas y entorno 10%

- EN1 - Ser capaz de construir un programa completo con la estructura esperada, libre de errores y warnings 100%

Enunciado

Siguiendo con la ayuda que proporcionamos a la compañía UOCoworking, nos han pedido nuestra colaboración para crear un programa que les ayude a gestionar los datos de sus centros de trabajo colaborativo y trabajadores. En este ejercicio trabajaremos con tipos de datos estructurados juntamente con la entrada y salida interactiva para gestionar los datos de las salas de trabajo de alquiler.

Para dar respuesta a esta petición, a lo largo de las distintas PEC iremos desarrollando una pequeña parte de la aplicación, que se encargará de la gestión de los centros de trabajo colaborativo, mediante la definición del modelo de datos y la implementación de distintas funcionalidades y algoritmos.

En las prácticas PR1 y PR2, se completará la aplicación con la gestión de los usuarios.

Aplicación a desarrollar en esta PEC: **Comparar si dos centros que se encuentran en una misma ciudad cumplen determinadas características, y obtener el mejor de ellos partiendo de datos guardados en tuplas.**

El desarrollo de la aplicación en esta PEC tiene **cuatro partes**:

1. Interpretación de un algoritmo
2. Diseño algorítmico
3. Codificación en C
4. Prueba del programa en C

1. Interpretación de un algoritmo

Leer e interpretar el algoritmo desarrollado parcialmente que se expone a continuación.

```
const
    CATEGORY1: integer = 1;                 {Category 1 id}
    CATEGORY2: integer = 2;                 {Category 2 id}
    CATEGORY3: integer = 3;                 {Category 3 id}
end const

type
    tCoworkingType = {STARTUPS, FREELANCERS, RURAL, SPECIALIZED, GENERALIST}

    tCoworkingCenter = record
        name: string;
        city: string;
        category: integer;
```

```
        centerType: tCoworkingType;
        numSpaces: integer;
        price: real;
        distanceFromCityCenter: real;
        hasMeetingRooms: boolean;
        hasAuditorium: boolean;
    end record

end type

{Exercise 2.1}
{…}

{Exercise 2.2}
{…}

{Exercise 2.3}
{…}

{Exercise 2.4}
{…}

algorithm UOCoworking
    var
        center1: tCoworkingCenter;
        center2: tCoworkingCenter;
        price: real;
        distance: real;
       {…}
    end var

    {Exercise 2.5}
    {Data input Center 1}
    writeString("ENTER DATA FOR CENTER 1");
    {…}

    {Data input Center 2}
    writeString("ENTER DATA FOR CENTER 2");
    {…}

    {Data input aceptable price}
    writeString("ACCEPTABLE PRICE [EUR]? >>");
    price := readReal();

    {Data input aceptable distance}
    writeString("ACCEPTABLE DISTANCE FROM CITY CENTER [KM]? >>");
    distance := readReal();
    {…}

    writeString("RESULTS");

    if {…} then

        {…}

        if {…} then
            writeString("CENTER 1 SUITS YOU BETTER, AND THE DATA ARE:");
            writeCenter(center1);
            writeString ("THE SECOND BEST CENTER IS CENTER 2 AND THE DATA ARE:");
            writeCenter(center2);
        else
            writeString("CENTER 2 SUITS YOU BETTER, AND THE DATA ARE:");
            writeCenter(center2);
            writeString("THE SECOND BEST CENTER IS CENTER 1 AND THE DATA ARE:");
            writeCenter(center1);
        end if
    else
```

```
        writeString("CENTERS CAN'T BE COMPARED");
    end if

end algorithm
```

La estructura del algoritmo, así como los tipos de datos, constantes y variables que ya están declaradas servirán de base para el diseño algorítmico posterior.

Se han añadido los siguientes campos a la estructura tCoworkingCenter necesarios para la implementación de los ejercicios a realizar en esta PEC.

Variable	Descripción	Tipo / validación
numSpaces	Número de salas de trabajo disponibles en el centro para su alquiler.	Valor de tipo entero
hasMeetingRooms	El centro tiene sala de reuniones.	Valor de tipo booleano
hasAuditorium	El centro tiene un auditorio.	Valor de tipo booleano

2. Diseño algorítmico

Diseñar un algoritmo que incluya las funcionalidades que se detallan a continuación.

2.1. Desarrollo de funciones/acciones. Desarrollar la acción *readCenter(...)* que reciba como parámetro de salida una variable de nombre *center,* de tipo *tCoworkingCenter* sin inicializar, y devuelva la misma variable con los campos informados con los datos leídos por el canal estándar de entrada. A continuación se da la descripción de los parámetros.

Parámetro	Tipo	Clase	Descripción
center	tCoworkingCenter	Salida	Salida - los campos de la variable *center* se han inicializado con los datos leídos por el canal estándar de entrada.

La variable center representa los datos básicos de un centro de trabajo colaborativo, definida mediante el tipo estructurado tCoworkingCenter. Por ello, para leer los datos del centro, hay que leer todos los campos que forman parte de la variable.

En lenguaje algorítmico, los tipos de datos enumerados no tienen una correspondencia numérica, y por este motivo no pueden leerse y mostrarse como si fueran enteros. En su lugar, en esta PEC podemos usar la/s siguiente/es función/es, que podemos considerar definida/s "adhoc":

```
{...}
writeString("CENTER TYPE ? (...)");
centerType := readCenterType();
{...}
writeString("CENTER TYPE (...):");
writeCenterType(centerType);
{...}
```

Hay que evitar siempre que sea posible el uso de valores numéricos directos en el algoritmo, y utilizar en su lugar constantes previamente definidas.

2.2. Desarrollo de funciones/acciones. Desarrollar la función/acción *writeCenter (...)*. Esta recibe como parámetro de entrada una variable de nombre *center,* de tipo *tCoworkingCenter* ya inicializado y muestra por el canal estándar de salida los campos del centro. A continuación se da la descripción del parámetro.

Parámetro	Tipo	Clase	Descripción
center	tCoworkingCenter	Entrada	Parámetro *center* inicializado a mostrar por el canal de salida estándar.

La variable center *representa los datos básicos de un centro de trabajo colaborativo, definida mediante el tipo estructurado tCoworkingCenter. Por ello, para mostrar los datos del centro, hay que mostrar todos los campos que forman parte de la variable.*

2.3. Desarrollo de funciones/acciones. Desarrollar la función/acción *isAcceptableCenter (…)*. Esta recibe como parámetros una variable de nombre *center,* de tipo *tCoworkingCenter* inicializada, y dos variables *price* y *distance* de tipo real que representan un precio y una distancia al centro de la ciudad aceptables. La función retorna verdadero si el centro tiene sala de reuniones o auditorio, y tanto su precio como

su distancia al centro de la ciudad se encuentran dentro de los valores aceptables introducidos en los parámetros *price* y *distance*.

Parámetro	Tipo	Clase	Descripción
center	tCoworkingCenter	Entrada	Centro a aplicar la función/acción
price	real	Entrada	Precio aceptable [EUR]
distance	real	Entrada	Distancia acceptable al centro de la ciudad [km]
?	booleano	Valor de retorno de la función/acción	*True* si el centro tiene sala de reuniones o auditorio cocina y el precio y la distancia es la aceptable *False* en caso contrario.

2.4. Desarrollo de funciones/acciones. Desarrollar la función/acción *bestCenter(...)*. Esta recibe como parámetros dos variables, *center1* y *center2,* de tipo *tCoworkingCenter* correspondientes a dos centros y los compara para ver cual es el mejor de ambos.

Para comparar los dos centros se seguirán los siguientes criterios **en el orden indicado**:

- El de mayor categoría es mejor.
- El más cercano al centro de la ciudad es mejor.
- Tener una sala de reuniones es mejor.

En caso de que haya coincidencia en el primer criterio (los dos centros tienen la misma categoría), se pasará al siguiente criterio, y así sucesivamente.

A continuación se da la descripción de los parámetros y el valor retornado.

Parámetro	Tipo	Clase	Descripción
center1	tCoworkingCenter	Entrada	Primer centro usado para la comparación
center2	tCoworkingCenter	Entrada	Segundo centro usado para la comparación
?	entero	Valor de retorno de la función	1 si *center1* es mejor que *center2* 0 si *center1* es igual que *center2*

```
                                                    -1 si center2 es mejor que
                                                    center1
```

2.5. Procesamiento y salida de datos. Completar el algoritmo y mostrar por el canal estándar de salida los resultados, de acuerdo con el siguiente flujo de ejecución:

- Leer, por el canal estándar de entrada, los datos de dos centros, el precio y distancia máxima aceptables.
- Calcular si cada uno de los centros son aceptables usando la función *isAcceptableCenter (…)*. Si los dos centros son aceptables y están en la misma ciudad, compararlos utilizando la función *bestCenter(...)* y mostrar los datos mediante la acción *writeCenter(...) de* forma ordenada, es decir:
    1. Si centro 1 es mejor o igual a centro 2, mostrar en primer lugar los datos del centro 1.
    2. En caso contrario, mostrar primeramente los datos del centro 2.

- En caso de que alguno de los centros no sea aceptable o que se encuentren en distintas ciudades, mostrar mensaje indicando que los centros no pueden ser comparables entre sí.

Se deberán definir las variables auxiliares y constantes que se consideren necesarias para guardar el resultado de cada una de las expresiones solicitadas, escogiendo el tipo de datos más apropiado.

2. Codificación en C

Codificar en C el algoritmo diseñado anteriormente.

El programa en C debe cumplir con las siguientes particularidades:

- Los campos *name* y *city* del tipo de datos estructurado deben tener una longitud máxima de 15 caracteres.
- Los números reales deben mostrarse con una precisión de dos decimales.

3. Prueba del programa en C

Ejecutar y superar los juegos de prueba automáticos disponibles en la herramienta ACME.

*El proceso de validación y corrección de la herramienta ACME se basa en una comprobación **literal** de la salida obtenida por el programa sometido a prueba, con los*

resultados esperados de los juegos de prueba introducidos previamente. Por ello, los textos de la interfaz de usuario deben ser exactamente idénticos a los esperados, ***podéis recuperar dichos textos entrando dentro de la herramienta****, están disponibles en el enunciado asociado a la PEC.*

Hay que tener en cuenta que el proceso de copiar y pegar textos entre distintos editores, entornos y herramientas puede generar caracteres ocultos que hagan que la comparación de los textos literales sea incorrecta a pesar de que los textos literales aparentemente sean idénticos.