Fundamentos de Programación
PEC5 - 20211

Fecha límite de entrega: **02/11/2021 a las 23:59**

Apellidos: _____
Nombre: _____

Entrega

La PEC deberá entregarse antes del **día 02 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 — 30%

- TD3 - Ser capaz de declarar y utilizar vectores de datos de forma correcta 25%
- **TD4 - Ser capaz de definir y utilizar tipos estructurados de datos de forma correcta 50%**
- Indicadores de E/S 15%:
  - 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 10 %:
  - 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 — 40%

- DA1 - Ser capaz de construir expresiones lógicas correctas y compactas para tratar la información a partir de datos de forma correcta 20%
- DA2 - Ser capaz de construir estructuras alternativas compactas y funcionales, y diseñar árboles de decisión 25%
- DA3- Ser capaz de construir estructuras iterativas de forma correcta y óptima 25%
- 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 20%
- 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: **obtener el mejor centro de trabajo colaborativo partiendo de datos guardados en tuplas en base a información introducida por el usuario que interacciona con la aplicación.**

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.

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

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

{Exercise 2.1}
{...}
```

```
algorithm UOCoworking

    var
        city: string;
        maxPrice: real;
        bestCenter: tCoworkingCenter;

        {Exercise 2.1}
        {...}
    end var

    {Exercise 2.2}
    {Data input Center 1}
    {...}
    {Data input Center 1}
    {...}
    {Data input Center 1}
    {...}

    {Data input}
    writeString("MAX PRICE? [EUR] (A REAL) >>");
    maxPrice := readReal();

    writeString("CITY? (A STRING) >>");
    city := readString()

    {Exercise 2.3}
    {Data processing}
    {...}

    {Data output}
    {Exercise 2.4}
    writeString("RESULTS");
    {...}

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.

2. Diseño algorítmico

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

Para la lectura del resto de datos, se presupondrá que el usuario respetará el tipo de datos, rango de valores esperado o conjunto de valores posibles, y por lo tanto no será necesario realizar ninguna comprobación al respecto.

Para guardar la información, deberán declararse las variables, constantes y tipos de datos que se consideren necesarios, pudiéndose combinar con las que ya están declaradas en el algoritmo parcial del enunciado.

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

En lenguaje algorítmico, los tipos de datos enumerados no tienen una correspondencia numérica, y por este motivo no pueden leerse 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);
{...}
```

2.1. Definición de tipos de datos. Definir el tipo de datos estructurado *tCoworkingCenter* que representa la información de un centro de trabajo colaborativo. Los datos del centro que se desean guardar son los siguientes:

Variable	Descripción	Tipo / validación
name	Nombre del centro	Valor de tipo *string*
city	Localidad donde se ubica el centro	Valor de tipo *string*
category	Categoría del centro	Valor de tipo entero, que sigue el siguiente convenio (no se trata de un enumerado): - 1: Categoría 1 - BASIC - 2: Categoría 2 - STANDARD - 3: Categoría 3 - PREMIUM
centerType	Tipo del centro	Valor de tipo *tCoworkingType*
price	Precio del alquiler mensual de una sala de trabajo [eur]	Valor de tipo *real*
distanceFromCityCenter	Distancia hasta el centro de la localidad [km]	Valor de tipo *real*

A continuación, **declarar** dentro del algoritmo tres variables de tipo *tCoworkingCenter* .

2.2. Lectura de datos. Leer por el canal estándar de entrada los datos correspondientes a tres centros de trabajo colaborativo y guardarlos en tres variables del tipo declarado en el anterior apartado (*tCoworkingCenter*), seguidamente realizar la lectura de los siguientes datos adicionales necesarios para la aplicación:

- El precio máximo que se quiere pagar por alquiler de un centro, puede tener decimales.
- El nombre de la ciudad donde deberá estar localizado el centro.

Esta información será necesaria para el procesamiento de datos del siguiente apartado.

Este apartado se encuentra desarrollado parcialmente en el algoritmo del enunciado.

2.3. Procesamiento de datos. Hacer el siguiente cálculo:

- Buscar cuál de los tres centros leídos anteriormente está en la ciudad indicada, tiene el mejor precio y éste no supera el máximo introducido. En caso de empate entre dos o más centros, prevalece el centro introducido en primer lugar.

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.

2.4. Salida de datos. Mostrar por el canal estándar de salida el centro obtenido en el apartado 2.3. En caso de no encontrarse ningún centro, mostrar el siguiente mensaje:

writeString("THERE IS NO CENTER WITH DESIRED CONDITIONS");

2.5. Escalabilidad del algoritmo. Explica con tus propias palabras cómo se podría enfocar este ejercicio para poder dar de alta de manera interactiva un elevado número de centros, siempre manteniendo la funcionalidad solicitada en los apartados 2.2, 2.3 y 2.4, teniendo en cuenta que la búsqueda del centro será sobre aquellos introducidos . No hay que hacer el diseño, simplemente tenéis que razonar la propuesta.

3. 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.
- Tened en cuenta que en lenguaje C no se puede utilizar el operador = para asignar el contenido de una tupla a otra, para esto hay que hacer la asignación de valores campo a campo. En cambio, la codificación algorítmica sí que permite la asignación directa entre tuplas mediante el operador := . En la XWIKI correspondiente a esta PEC5 podéis encontrar más información al respecto.

4. 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.