Fundamentos de Programación
PEC8 - 20211

Fecha límite de entrega: **22/11/2021 a las 23:59**

Apellidos: _____
Nombre: _____

Entrega

La PEC deberá entregarse como muy tarde el **día 22 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 tenéis 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 40%

- 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%
- 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 10%
- DA5 - Ser capaz de declarar y utilizar acciones y funciones de forma correcta 25%
- 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 25%

- 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 15%

- 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 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: **Leer y procesar los datos estructurados de centros de trabajo colaborativo, y obtener el mejor de ellos mediante el uso de acciones y funciones.**

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}

    MAX_CENTERS: integer = 100;  {max num. of coworks centers}

    {…}
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;
        percentOccupation: real;
    end record

    tCoworkingCentersTable = record
        centers: vector[MAX_CENTERS] of tCoworkingCenter;
        numCenters: integer;
    end record


end type

{Exercise 2.1}
{…}

{Exercise 2.2}
{…}

{Exercise 2.3}
{…}


algorithm UOCoworking
  var
     centersTable: tCoworkingCentersTable;
     filename: string;
     isRead: boolean;
    {…}
   end var

   {Exercise 2.4}
   {centersTable initialization}
   {…}

   {Load data from file}
   writeString("LOAD DATA FROM FILE. ENTER FILE NAME >>");
   filename := readString();
   {…}

   if isRead then
      writeString("CENTERS SUCCESSFULLY LOADED");

      {Data input}
      {…}

      {Exercise 2.5}
      writeString("RESULTS");
      {…}

   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 ha añadido el siguiente campo a la estructura tCoworkingCenter necesario para la implementación de los ejercicios a realizar en esta PEC.

Variable	Descripción	Tipo / validación
percentOccupation	Porcentaje de ocupación del centro.	Valor de tipo real

Se dispone de la siguiente acción *centersTableLoadDataFromFile*::

action *centersTableLoadDataFromFile(**out** centersTable: tCoworkingCentersTable, **in** filename: **string, out** isRead: **boolean**);*

Esta acción que lee datos del fichero *filename* y los carga en la tabla *centersTable*. Puede considerarse ya definida y no hace falta diseñarla. El resultado (*true, false*) de la operación de lectura se guarda dentro del parámetro de salida *isRead*, la lectura correcta del fichero el valor de *isRead* será *true*, en caso contrario será *false*.

2. Diseño algorítmico

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

Para el desarrollo, se deberá optar entre una acción o función. Así mismo, deberán definirse el número, tipo y clase de parámetros (entrada, salida, entrada/salida) que se consideren necesarios, y en su caso, el tipo de datos de retorno. Finalmente, 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.

2.1. Desarrollo de funciones/acciones. Desarrollar la función/acción *centersTableInitialize(...)* que inicializa una tabla de tipo *tCoworkingCentersTable.* Una tabla de centros inicializada es aquella no dispone de centros.

2.2. Desarrollo de funciones/acciones. Desarrollar la función/acción *centersTableSelect(...)* que, a partir de una tabla de centros tipo *tCoworkingCentersTable*, creará una nueva tabla con una selección de centros. Los criterios de selección de dichos centros vendrán parametrizados de la forma siguiente: ciudad y distancia óptima al centro de esta, precio óptimo, y puntuación mínima exigida de centro. Los centros seleccionados deberán estar localizados en la ciudad indicada y deberán tener una puntuación mínima

en base a su precio y distancia al centro de la ciudad, para calcular esta puntuación podéis hacer uso la función *pointsCenter(...)* de la PEC7 sin necesidad de volver a implementarla.

2.3. Desarrollo de funciones/acciones. Desarrollar la función/acción *centersAverageOccupation(...)* que dada una tabla de centros tipo *tCoworkingCentersTable* y una ciudad, devuelve la media aritmética global de ocupación de los centros de la ciudad indicada. Esta media es un porcentaje, que debe calcularse como el número total de números de salas de trabajo ocupadas en los centros de la ciudad indicada dividido por el número total de número de salas de trabajo disponibles en dichos centros.

Ejemplo de cómo se calcularía la media a partir de los siguientes datos:

Centro 1, 20 salas en total, 20% de ocupación.
Centro 2, 30 salas en total, 50% de ocupación.
Centro 3, 50 salas en total, 10% de ocupación.

El resultado de la ocupación de cada centro sería:

En el centro 1 tenemos 4 salas ocupadas.
En el centro 2 tenemos 15 salas ocupadas.
En el centro 3 tenemos 5 salas ocupadas.

Para calcular la media de la ocupación en la ciudad lo haríamos así:
*centersAverageOccupation := integerToReal(100 * (4 + 15 + 5)) / integerToReal(20 + 30 + 50);*
Obteniendo como resultado final: *24.0 %*

2.4. Carga, lectura y procesamiento de datos. Leer del canal estándar de entrada de datos el nombre del fichero que contiene los datos de los centros de trabajo colaborativo necesarios para asignar valores a la tabla *centersTable* definida en el algoritmo del enunciado y previamente inicializada, para ello se hará uso de la acción predefinida *centersTableLoadDataFromFile(…),* indicando tras la llamada que la carga ha sido exitosa si el resultado de esta es correcta. En caso de error el algoritmo no continuará.

Tras una carga correcta de datos, leer por el canal estándar de entrada el nombre de ciudad, la distancia óptima al centro de esta, el precio óptimo, y la puntuación mínima. Estos valores se usarán para hacer una selección de centros de la tabla *centersTable* empleando acción/función *centersTableSelect(...)* desarrollada en el ejercicio 2.2.

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.

2.5. Salida de datos. Completar el algoritmo y mostrar por el canal estándar de salida los centros de trabajo colaborativo resultantes de la anterior selección. Primero se debe mostrar cuantos centros se han encontrado y a continuación mostrar los datos de dichos centros. En caso de no haberse recuperado ningún centro, mostrar un mensaje indicando este hecho.

Independientemente de los centros encontrados, mostrar por el canal estándar de salida un mensaje indicando cual es el porcentaje de ocupación global de los centros de la ciudad indicada, usando la acción/función *centersAverageOccupation(…)* desarrollada en el ejercicio 2.3.

Para mostrar los resultados se puede usar la acción writeCenter(...) sin necesidad de diseñarla.

3. Codificación en C

En este ejercicio hay que codificar en C el algoritmo anterior. Para esta PEC se proporciona el enunciado del programa con una parte del código ya implementado. Concretamente, hay que hacer lo siguiente:

1. Descomprimir el archivo que incluye el proyecto *Codelite*. El proyecto está estructurado en carpetas: carpeta *include* dónde está el archivo ***coworkingcenter.h*** y carpeta *src* donde están los archivos ***coworkingcenter.c*** y ***main.c***.
   La codificación debe seguir la estructura de carpetas y archivos proporcionados en el workspace así proporcionado y que se explica a continuación.
   También se pueden encontrar los ficheros *"centers1.txt", "centers2.txt"* y *"centers3.txt"* que se proporcionan para poder cargar datos en la tabla de centros de trabajo colaborativo..
2. Dentro del archivo ***coworkingcenter.h*** declarar las cabeceras de las acciones y funciones ***centersTableInitialize, centersTableSelect*** y ***centerAverageOccupation.***
3. Dentro del archivo ***coworkingcenter.c*** implementar las acciones y funciones ***centersTableInitialize, centersTableSelect*** y ***centerAverageOccupation.***
4. Dentro del fichero *main.c*
   a. se debe completar la declaración de variables para codificar ***main.c***
   b. se debe completar la codificación del ***main.c*** con los ejercicios 2.4, 2.5 y 2.6

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.
- Los tipos enumerados en C, tienen una correspondencia numérica. Por este motivo, podrán leerse y mostrarse como si fueran enteros, siguiendo las reglas establecidas y

siempre mediante la ayuda de una interfaz de usuario para interpretar los datos a leer/mostrar.

- En C no se deben copiar estructuras directamente. Por este motivo, en la codificación debéis usar la acción *copyCenter(...)* que ya viene dada en el código.
- En los archivos ***coworkingcenter.h*** y ***coworkingcenter.c*** también disponéis de las cabeceras e implementaciones de las acciones y funciones *writeCenter(…)*, *copyCenter(...), pointsCenter(...), getCenterObject(...), centersTableAddElement(...) y centersTableLoadDataFromFile(…).*

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.