Fundamentos de Programación
PEC3 - 20211

Fecha límite de entrega: **11/10/2021 a las 23:59**

Apellidos: _____
Nombre: _____

Entrega

La PEC deberá entregarse antes del **día 11 de octubre 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 forma correcta. 25%**
- TD7 - Ser capaz de leer datos por el canal estándar de forma correcta. 25%
- TD8 - Ser capaz de mostrar datos por el canal estándar de forma correcta. 25%
- Resto de indicadores de tratamiento de datos 25 %:
    - 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. 50%**
- 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%
- DA8 - Ser capaz de utilizar la notación algorítmica de forma adecuada y aplicar las buenas prácticas en cuanto a la nomenclatura y el uso de comentarios. 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 30%
- CO2 - Ser capaz de cumplir con los requisitos formales de entrada y salida de datos en C. 10%
- 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

La compañía *UOCoworking* nos ha encargado el desarrollo de una aplicación para gestionar una red de centro de trabajo cooperativo. En concreto, deberemos gestionar los centros de trabajo colaborativo y los trabajadores.

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, completaremos la aplicación con la gestión de los trabajadores.

Aplicación a desarrollar en esta PEC: **Obtener el mejor centro de trabajo colaborativo partiendo de datos guardados en vectores.**

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
    CENTER1: integer = 1;                   {CENTER 1 ID}
    CENTER2: integer = 2;                   {CENTER 2 ID}
    CENTER3: integer = 3;                   {CENTER 3 ID}

    MAX_COWORKCENTERS: integer = 3;         {Number of cowork centers}

    MIN_PRICE: real = 0.0;                  {Min. price}
    MIN_DISCOUNT: real = 0.0;               {Max. discount}
    MAX_DISCOUNT: real = 50.0;              {Max. discount}
end const

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

algorithm UOCoworking
    {Variable definition}
    var
        {Exercise 2.1}
        {...}

        {Exercise 2.2}
        {...}
```

```
        numCoworkers: integer;          {num. of coworkers, they want to rent spaces}
        bestCoworkCenter: integer;      {best coworking center id}
        numSpaces: integer;
    end var

    {Data input}
    {Exercise 2.3}
    {...}

    [Data validation}
    {...}

    writeString("COWORKERS? (AN INTEGER) >>");
     numCoworkers := readInteger();              {num. of coworkers, they want to rent
spaces}

    {Data processing}
    {Exercise 2.4}
    {...}

    {Data output}
    {Exercise 2.5}
    writeString("RESULTS");
    writeString("THE CHEAPEST WORK CENTER IS:");
    writeInteger(bestCoworkCenter);
    {...}

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.

2.1. Entrada de datos. Declarar **un vector** de **tres posiciones** para guardar el siguiente dato correspondiente a **tres centros de trabajo colaborativo** distintos (el dato correspondiente a cada centro, en una posición distinta del vector):

Variable	Descripción	Tipo / validación
price	Precio del alquiler mensual de una sala de trabajo en el centro [€]	Valor de tipo **real**

2.2. Entrada de datos. Declarar **un vector** de **tres posiciones** para guardar el siguiente dato correspondiente a **tres centros de trabajo colaborativo** distintos (el dato correspondiente a cada centro, en una posición distinta del vector):

Variable	Descripción	Tipo / validación
discount	Descuento sobre el precio del alquiler mensual de una sala de trabajo en el centro [%]	Valor de tipo **real**

2.3. Entrada de datos. Leer por el canal estándar de entrada, los siguientes datos correspondientes a **tres centros de trabajo colaborativo** y almacenarlos en los respectivos **vectores** declarados en el apartado anterior:

Dato	Descripción	Tipo / validación
price	Precio del alquiler mensual de una sala de trabajo en el centro [€]	Valor de tipo real
discount	Descuento sobre el precio del alquiler mensual de una sala de trabajo en el centro [%]	Valor de tipo real

Para la lectura y validación de esta información, se aplicarán las siguientes reglas:

- El precio debe ser un número **real positivo** (mayor que cero). En caso contrario, deberá mostrarse un mensaje de aviso y **finalizar el algoritmo**.
- El descuento debe ser un real **superior al 0% e inferior o igual al 50%**. En caso contrario, deberá mostrarse un mensaje de aviso y **finalizar el algoritmo**.

El resto de datos del centro de trabajo no son necesarios para el desarrollo de la aplicación en esta PEC.

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.

2.4. Procesamiento de datos. Calcular los siguientes datos, mediante la combinación de expresiones y estructuras algorítmicas, basándose en la información introducida por el usuario:

- El precio **total** del alquiler mensual de cada uno de los centros de trabajo, en función del número de trabajadores, el precio de alquiler de una sala de trabajo en cada centro y el descuento aplicable. Para ello, se pueden adaptar las expresiones

de la PEC2 (recordar que las salas de trabajo tienen una capacidad máxima de dos trabajadores).
- El centro de trabajo más económico. En caso de empate entre dos o más centros, prevalece el centro introducido en primer lugar.

Se podrán definir las variables auxiliares y constantes que se consideren necesarias para obtener y guardar el resultado, escogiendo el tipo de datos más apropiado, pudiéndose combinar con el algoritmo parcial ya desarrollado en el 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.

2.5. Salida de datos. Completar el algoritmo, mostrando por el canal estándar de salida los resultados.

En caso que todos los datos introducidos sean correctos:
- Mostrar el identificador del centro de trabajo más económico.
- Mostrar el precio del alquiler mensual del centro de trabajo más económico.

Como se ha indicado en los apartados anteriores, en caso de que alguno de los datos introducidos no sea correcto:
- Mostrar un mensaje de aviso y finalizar el algoritmo.

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

3. Codificación en C

Codificar en C el algoritmo diseñado anteriormente.

El programa en C debe cumplir con las siguientes particularidades:

- 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.

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.*

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.

*En caso de discrepancia, los textos literales que aparecen en la herramienta ACME **tendrán preferencia** respecto a los textos literales que aparecen en este enunciado.*