Prácticas de Programación
PR3 - 20211

Fecha límite de entrega: **19 / 12 / 2021**

Formato y fecha de entrega

La práctica debe entregarse antes del día **19 de diciembre de 2021** a las 23:59.

Es necesario entregar un fichero en formato **ZIP**, que contenga una carpeta UOC20211 con el directorio principal de vuestro proyecto, siguiendo la estructura de carpetas y nombres de ficheros especificados en el enunciado de la práctica. No debe contener ningún fichero ZIP en su interior:

- Un fichero **README.txt** con el siguiente formato (ver ejemplo):

```
Formato:
Correo electrónico UOC
Apellidos, Nombre
Sistema operativo utilizado

Ejemplo:
estudiante1@uoc.edu
Apellido1 Apellido2, Nombre
Windows 10
```

- Los ficheros de prueba sin modificaciones.
- Los ficheros *.c y *.h resultantes de los ejercicios realizados.
- Los ficheros .workspace y .project que definen el espacio de trabajo y los proyectos de Codelite.
- Todos los ficheros deben estar dentro de las carpetas correctas (src, test, …).

La entrega debe realizarse en el apartado de entregas de EC del aula de teoría antes de la fecha límite de la entrega.

El incumplimiento del formato de entrega especificado anteriormente puede suponer un suspenso de la práctica.

Objetivos

- Saber interpretar y seguir el código de terceras personas.
- Saber compilar proyectos de código organizados en carpetas y librerías.
- Saber implementar un proyecto de código a partir de su especificación.

Criterios de corrección:

Cada ejercicio tiene asociada su puntuación sobre el total de la actividad. Se valorará tanto que las respuestas sean correctas como que también sean completas.

- No seguir el **formato de entrega**, tanto por lo que se refiere al **tipo y nombre de los ficheros** como al contenido solicitado, comportará una **penalización importante** o la cualificación con una **D de la actividad**.
- El código entregado **debe compilar para ser evaluado**. Si compila, se valorará:
    - Que **funcionen** tal como se describe en el enunciado.
    - Que obtenga el **resultado esperado** dadas unas condiciones y datos de entrada diseñadas (pruebas proporcionadas).
    - Que se respeten los **criterios de estilo y** que el código esté **debidamente comentado**. Se valorará especialmente el uso de comentarios en inglés.
    - Que les **estructuras** utilizadas sean las correctas.
    - Que se **separe correctamente la declaración e implementación** de las acciones y funciones, utilizando los ficheros correctos.
    - El **grado de optimización** en tiempo y recursos utilizados en la solución entregada.
    - Que se realice una **gestión de memoria** adecuada, liberando la memoria cuando sea necesario.

Aviso

Aprovechamos para recordar que **está totalmente prohibido copiar en las Prácticas** de la asignatura. Se entiende que puede haber un trabajo o comunicación entre los estudiantes durante la realización de la actividad, pero la entrega de esta debe que ser individual y diferenciada del resto. Las entregas serán analizadas con **herramientas de detección de plagio.**

Así pues, las entregas que contengan alguna parte idéntica respecto a entregas de otros estudiantes serán consideradas copias y todos los implicados (sin que sea relevante el vínculo existente entre ellos) suspenderán la actividad entregada.

Guía citación: https://biblioteca.uoc.edu/es/contenidos/Como-citar/index.html
Monográfico sobre plagio:
http://biblioteca.uoc.edu/es/biblioguias/biblioguia/Plagio-academico/

Observaciones

Esta PEC presenta el proyecto que se desarrollará durante las distintas actividades del semestre, que se ha simplificado y adaptado a las necesidades académicas.

En este documento se utilizan los siguientes símbolos para hacer referencia a los bloques de diseño y programación:

 Indica que el código mostrado es en **lenguaje** algorítmico.

 Indica que el código mostrado es en **lenguaje** C.

 Muestra la ejecución de un programa en **lenguaje** C.

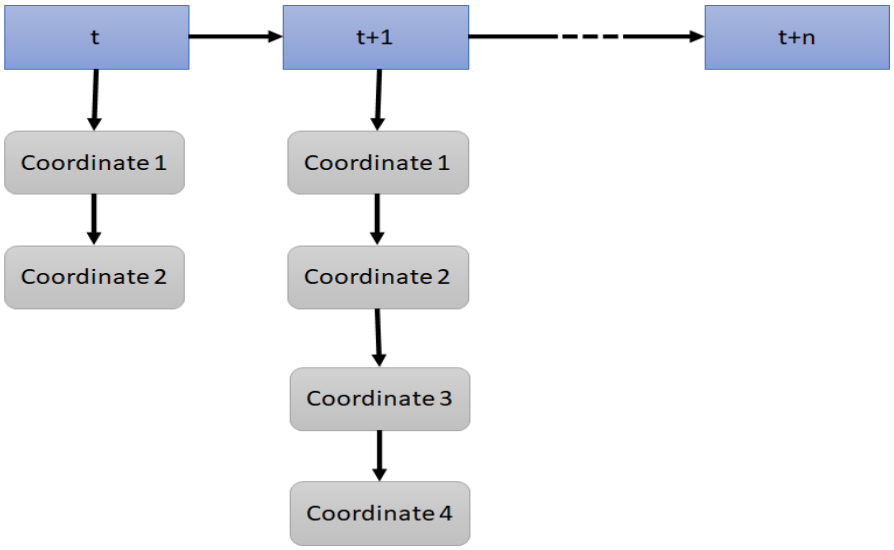
Enunciado

Siguiendo con la estructura implementada en la PR2:

```
type
      tCoordinateNode = record
            coordinate : tCoordinate;
            numPersons : integer;
            next: pointer to tCoordinateNode;
      end record

      tDateTimeNode = record
            dateTime : tDateTime;
            coordinatesList : pointer to tCoordinateNode;
            next: pointer to tDateTimeNode;
      end record
end type
```

Que guardaba para cada momento temporal **t** la lista de coordenadas que contenían personas:



En esta práctica vamos a añadir un poco más de complejidad a esta estructura y realizar el análisis de los contactos estrechos.

Entorno de pruebas

Para facilitar la verificación de vuestro código y su estructura antes de la entrega, hemos puesto a vuestra disposición un mecanismo de verificación. Se trata de un sistema basado en Linux que verifica vuestra entrega y os notifica los resultados.

Para utilizarlo:

1. Preparad el **fichero ZIP** siguiendo las instrucciones de entrega.
    a. La estructura es la misma que el código facilitado con el enunciado.
    b. El **README.txt** debe contener vuestros datos.
    c. No debe incluir las carpetas **bin**, **lib**, ni otras carpetas resultantes de la compilación.
2. Enviad un correo desde vuestra **cuenta institucional de la UOC** con las siguientes características:
    a. **Dirección**: pelp.inbox@gmail.com
    b. **Asunto**: 7923e89a-4140-459a-a7fb-3ed1e6b83464
    c. **Contenido**: Adjuntar el fichero ZIP
3. En **unas horas**, vais a recibir un correo con el resultado de la ejecución de vuestro código y el resultado de las pruebas.
    a. No es un sistema de entrega, tenéis que entregar igualmente vuestra práctica en el **apartado de EC del aula de teoría**.
    b. No es un sistema de compilación. Esta herramienta está pensada para que podáis validar que el código que os funciona en local va a funcionar en un entorno de prueba.
    c. **Hay un límite** de 2 entregas por día y un máximo de 10 entregas en total.
    d. No esperéis al último día, ya que quizás no tengáis los resultados antes de la fecha límite de entrega.
    e. El sistema no acepta entregas pasada la fecha límite de la actividad.

Nota: Este sistema está en fase de pruebas, si encontráis algún problema indicadlo a los profesores para que podamos corregir cualquier incidencia.

Ejercicio 1: Modificación de los tipos de datos [30%]

Queremos modificar el tipo **tCoordinateNode** para que además de contener el número de personas en esa coordenada, contenga una lista de los documentos de las personas que estaban en la coordenada en ese momento.

En los distintos apartados de este ejercicio se guía paso a paso los cambios necesarios.

Se pide:

a) Modifica la definición del tipo de datos **tCoordinateNode** en el fichero ***datetime_node.h***, para añadir un vector con memoria dinámica que permita guardar los documentos de las personas que estaban en la coordenada.
b) Modifica el método ***coordinateNode_insert*** en el fichero ***datetime_node.c*** para que al añadir un nuevo nodo de tipo **tCoordinateNode**, se actualize la lista definida en el apartado anterior.
c) Modifica el método ***coordinateNode_free*** en el fichero ***datetime_node.c*** para que libere correctamente el nuevo contenido.
d) Implementa el método **api_getCoordinatePeopleCount** del fichero **api.c**, para que dada una estructura de datos **tApiData**, un momento temporal **tDateTime** y una coordenada **tCoordinate**, nos retorne el número de personas que hay en esa coordenada en ese momento temporal. Si no existe la coordenada, devolverá un cero.
e) Implementa el método **api_getCoordinatePeople** del fichero **api.c**, para que dada una estructura de datos **tApiData**, un momento temporal **tDateTime** y una coordenada **tCoordinate** nos retorne el vector de documentos de las personas en esa coordenada y momento temporal (definido en el apartado a de este ejercicio). Si esa coordenada no existe para ese momento temporal, retornará un valor NULL.

Nota: Podéis definir los métodos auxiliares que consideréis oportunos.

Ejercicio 2: Métodos de ordenación [20%]

En la PEC4 vimos algunos métodos de búsqueda, y en la teoría hemos visto que en general estos métodos requieren que los datos estén ordenados para poder ser aplicados.

Utilizando la definición de **quicksort** de los apuntes, se pide:

a) Implementa el método **sortPeople** del fichero ***datetime_node.c*** para que dado un nodo de tipo **tCoordinateNode** ordene las personas de ese nodo en orden alfabético (considerando los números como caracteres). Se debe utilizar el método de ordenación **quicksort**. Se recomienda definir los métodos adicionales que sean necesarios.

b) Modifica el método **api_getCoordinatePeople** del ejercicio anterior, para que devuelva la lista de personas ordenada.

Ejercicio 3: Métodos de búsqueda [20%]

En el ejercicio 3 de la PEC4 se pedía la implementación en lenguaje algorítmico del método ***getPersonIndex***, que realizaba una búsqueda binaria sobre una estructura de tipo **tPopulation**. En este ejercicio se pretende adaptar esa implementación para aplicar la búsqueda binaria sobre una lista de documentos, ahora utilizando lenguaje C.

Usando búsqueda binaria, implementa la función ***api_findCoordinatePerson*** del fichero **api.c**, para que dada una estructura de datos **tApiData**, un momento temporal **tDateTime,** una coordenada **tCoordinate** y un documento, nos indique si la persona con este documento estaba en esa coordenada en el momento temporal indicado (true) o no (false). Si no existe la coordenada, devolverá un valor falso (false).

Ejercicio 4: Contactos estrechos [30%]

Finalmente, queremos que la aplicación nos permita detectar los contactos estrechos y generar la información necesaria para avisarlos. Tal como se introdujo en la PEC1, los datos de salida del sistema van a tener el formato:

"email;when;where;detail"

- **email**: Hace referencia al correo electrónico del destinatario del mensaje.
- **when**: Fecha y hora en que se ha producido el contacto. El formato será "dd/MM/yyyy hh:mm" (dd: día con dos dígitos, MM: mes con dos dígitos, yyyy: año con cuatro dígitos, hh: hora con dos dígitos (00-23), mm: minutos con dos dígitos (00-59)).
- **where**: Coordenadas dónde se ha producido el contacto en el formato latitud,longitud.

Implementa el método **api_findContacts** del fichero ***api.c***, que dada una estructura de datos **tApiData**, un momento temporal inicial y uno de final de tipo **tDateTime,** y el documento de una persona, nos retorne en una estructura de tipo **tCSVData**, el listado de personas (en el formato anterior) que han sido contacto estrecho de la persona indicada, entre los momentos temporales indicados. Consideraremos que dos personas son contacto estrecho, si han estado en una misma coordenada **tCoordinate** en algún momento del rango temporal. Las entradas **tCSVEntry** del resultado tendrán el tipo (type) "CONTACT_MAIL". Los valores de retorno de esta función se detallan en la siguiente tabla:

E_SUCCESS	Operación ejecutada correctamente.
E_NOT_IMPLEMENTED	La funcionalidad aún no está implementada.
E_PERSON_NOT_FOUND	Se está preguntando por los contactos de una persona no existente.

Nota: Si una persona ha coincidido en más de una coordenada o momento, dentro del rango indicado, se generará un mensaje para cada coordenada y momento en que hayan coincidido.