Programación de Fundamentos
PRÁCTICA 2. Semestre 20211.
Gestión de espacios de coworking

Formato y fecha de entrega

Se debe presentar la práctica antes del **28 de diciembre a las 23:59h.** Para la entrega deberá entregar un archivo en formato **ZIP** de nombre ***logincampus*_pr2** en minúsculas (donde *logincampus* es el nombre de usuario con el que hacéis login en el Campus). El ZIP debe contener:

- Workspace CodeLite entero, con todos los archivos que se piden.

Para reducir el tamaño de los archivos y evitar problemas de envío que se pueden dar al incluir ejecutables, es necesario eliminar lo que genera el compilador. Podéis utilizar la opción "Clean" del workspace o eliminarlos directamente (las subcarpetas Menú y Test son las que contienen todos los binarios que genera el compilador).

Es necesario realizar la entrega en el Registro de Evaluación Continua (REC) del aula de teoría.

Presentación

Esta práctica culmina el proyecto comenzado en la práctica 2. Hemos construido una aplicación para gestionar los espacios de co-trabajo y los co-trabajadores de una empresa de coworking y ahora nos piden gestionar las reservas de espacio. Trabajaremos con un TAD lista para manejar las reservas en sus diferentes estados y un TAD cola para representar las colas de entrada y salida de los co-trabajadores en los centros de co-trabajo.

Competencias

Transversales

- Capacidad de comunicación en lengua extranjera.

Específicas

- Capacidad de diseñar y construir aplicaciones informáticas mediante técnicas de desarrollo, integración y reutilización.
- Conocimientos básicos sobre el uso y la programación de los ordenadores, sistemas operativos, y programas informáticos con aplicación a la ingeniería.

Objetivos

- Analizar un enunciado y extraer los requerimientos tanto de tipos de datos como funcionales (algoritmos)
- Analizar, entender y modificar adecuadamente código existente
- Saber utilizar tipos de datos abstractos
- Saber utilizar punteros

Recursos

Para realizar esta actividad tenéis a vuestra disposición los siguientes recursos :

- Materiales en formato web de la asignatura
- Laboratorio de C

Criterios de valoración

Cada ejercicio tiene asociada la puntuación porcentual sobre el total de la actividad. Se valorará tanto la corrección de las respuestas como su compleción.

- **Los ejercicios en lenguaje C, deben compilarse para ser evaluados**. En tal caso, se valorará:
  - Que funcionen correctamente.
  - Que pasen los juegos de prueba
  - Que se respeten los criterios de estilo (ver la *Guía de estilo de programación en C* que tiene en la Wiki).
  - Que el código esté comentado (preferiblemente en inglés). En caso necesario (cuando el código no sea trivial) se valorará la presencia de comentarios explicativos.
  - Que las estructuras utilizadas sean las adecuadas.

Descripción del proyecto

En esta práctica gestionaremos los movimientos de entrada y salida de co-trabajadores en los centros de co-trabajo. Los centros, a diario, tendrán una cola de entrada y salida de co-trabajadores que habrá que ir gestionando, utilizando para tal fin, las listas de reservas pendientes, en curso y completadas.

Una reserva de un centro contendrá la siguiente información:

- Identificador del co-trabajador que realiza la reserva
- Número de personas asociadas a esta reserva
- Día de entrada y día de salida
- Relación de espacios asignados
- Equipamiento necesario (sólo espacio, espacio + telefonía, espacio + telefonía + internet o bien, espacio + telefonía + internet + proyector)
- Importe total de la estancia.

Por cada centro, guardaremos la relación de sus reservas mediante tres listas:

- pendingBookings. Es la lista de reservas que tienen como fecha de entrada una fecha futura en el tiempo. Es decir, pendientes de ser ejecutadas.
- currentBookings: Es la lista de reservas que tienen una fecha de entrada pasada en el tiempo pero una fecha de salida futura. Por tanto, son reservas en ejecución.
- completedBookings: Es la lista de reservas que tienen una fecha de salida pasada en el tiempo. Es decir, son reservas ya ejecutadas.

Los centros organizan sus procesos de checkin y checkout en dos horas diferenciadas. A las 13h se realizan los checkout's y a las 15h se realizan los checkin's. Este comportamiento se modelará mediante dos colas que representarán a los co-trabajadores que están pendientes de entrada y salida en el centro:

- checkinQueue. Es la cola de co-trabajadores pendientes de hacer el checkin.
- checkoutQueue. Es la cola de co-trabajadores pendientes del checkout.

Siempre se procesarán primero las salidas y después las entradas, por lo que no se empezarán a tratar los elementos de la cola de checkins hasta que la cola de checkouts esté vacía.

- Un checkin consistirá en comprobar que el cotrabajador tenía una reserva pendiente de entrada en la fecha actual y, si es así, asignarle los espacios y mover la reserva de la lista de reservas futuras a la de reservas en curso.
- Un checkout consistirá en comprobar que el co-trabajador tenía una reserva en curso que finaliza en la fecha actual y, en caso de que así sea, liberar el espacio (o espacios) y mover la reserva de la lista de reservas actuales a la de reservas completadas. Por último, se calculará el importe de la factura a pagar.

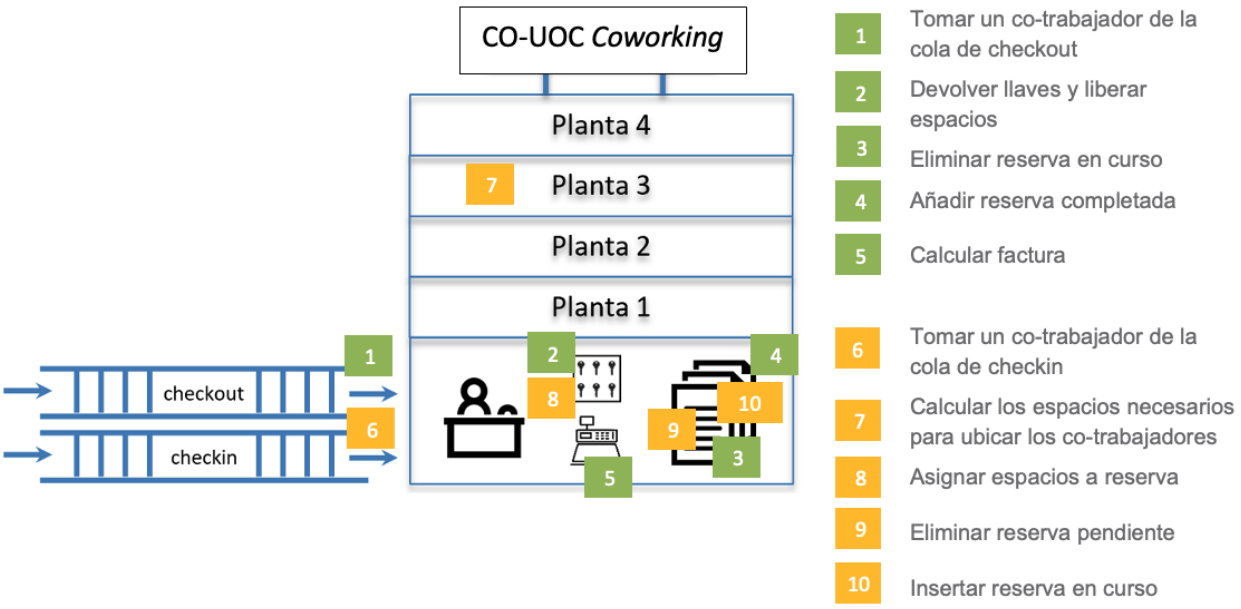
Consideraremos, en esta segunda práctica, que los centros gestionados por esta compañía de coworking tienen 40 espacios que siguen la misma distribución:

- Primera planta: 10 espacios de uso individual (numerados del 101 al 110).
- Segunda planta: 10 espacios de uso doble (numerados de 201 a 210).
- Tercera planta: 10 espacios de uso triple (numerados del 301 al 310).
- Cuarta planta: 10 espacios de uso cuádruple (numerados del 401 al 410).

El hecho de que haya el mismo número de espacios por planta aunque las superiores son de mayor superficie se debe a que los espacios de uso común de los centros se sitúan en las plantas más bajas del edificio (recepción, sala de actos, cocina, gimnasio, etc).

Para simplificar el problema, en el proceso de checkin, se asignarán espacios del tamaño que convenga para dar cabida a todos los co-trabajadores incluidos en una misma reserva. Es decir, los espacios no están pre-asignados sino que se asignan el mismo día de la entrada en función de la disponibilidad de espacios de las diferentes tipologías. Durante los días que dure la reserva de espacios, los co-trabajadores sí que podrán utilizar los mismos espacios que les fueron asignados en el momento de su entrada, pudiendo entrar y salir del centro con total libertad y utilizar los recursos del centro las 24h del día mientras su reserva sea vigente.

En el momento de entrar, a cada co-trabajador se le entregan unas llaves, que pueden ser: las llaves del candado de seguridad antirrobo para los portátiles (en el caso de un puesto de trabajo individual en espacio compartido), las llaves de acceso al espacio (si son espacios de dos o más co-trabajadores), las claves wifi (en caso de necesitar internet), etc. Estas llaves deben ser devueltas en el momento que finalice el período de reserva.



En el esquema anterior se pueden apreciar los pasos a seguir en los procesos de *checkin* (amarillo) y *checkout* (verde).

Estructuración del código

Junto con el enunciado se os facilita un nuevo workspace Codelite basado en la solución de la práctica 1, pero con algunos cambios y evoluciones. Este workspace será la base para esta segunda práctica.

Veréis que han aparecido nuevos archivos: **movements.c y .h** (que recoge las operaciones sobre las reservas), **list.c y .h** (que recoge las operaciones sobre el TAD lista) y, por último, **queue.c y .h** (que recoge las operaciones sobre el TAD cola). Además, se han introducido los nuevos tipos de datos relacionados en **data.h**:

- **tBooking**, que representa una reserva de un co-trabajador en un centro.
- **tBookingList**, para representar una lista de reservas.
- **tCoworkerQueue**, que representa una cola de co-trabajadores que esperan para entrar o salir del centro.
- **tMovement**, que representa a los movimientos de un día en un centro.
- **tMovementTable**, que contiene varios elementos de tipo tMovement.
- El tipo **tCoworkingCenter** incorpora también una tabla bidimensional que representa la distribución y ocupación de espacios dentro del centro (campos *layout* y *occupiedSpaces*).
- También se incorporan otros tipos auxiliares de soporte a los anteriores.

Aparte del código fuente añadido y/o modificado, veréis que aparece también un nuevo archivo de datos: el archivo de movimientos (movements.txt). Este archivo contiene la persistencia de los objetos de tipo *tMovement*.

Tests

El proyecto CodeLite se presenta, como en la Práctica 1, con dos modos de ejecución. Sin embargo, el modo Menu no incorpora ninguna entrada nueva que os permita probar interactivamente la gestión de las reservas. Esto es así porque se pretende que ejecutéis y probéis vuestro programa en modo Test.

En modo Test podréis ir verificando el funcionamiento de vuestro código a medida que completéis los ejercicios. Tened en cuenta que el resultado puede ser correcto aunque el código no esté bien (es decir, el resultado no os da una corrección definitiva) y que el total de tests superados no equivale a la nota de la práctica ya que el número de test por apartado no se corresponde con su peso en la calificación.

Enunciado

Ejercicio 1: Checkin de co-trabajadores en un centro [50%]

El procesamiento de los movimientos en un centro se realiza en base a dos procesos diferenciados: la entrada de cotrabajadores (checkin) y la salida de cotrabajadores (checkout). Esto queda de manifiesto en la acción processMovement (api.c) donde se realizan llamadas a los dos procesos de checkin y checkout. Primero siempre se llama el checkout (porque permite liberar espacios que pueden servir a los nuevos co-trabajadores) y, después, el checkin.

En este ejercicio nos centraremos en el proceso de checkin. A tal efecto, se pide que completéis estos dos apartados:

a) [15%] Implementad la función ***findCoworkerBooking*** (api.c) que, dados una lista de reservas y un identificador de co-trabajador, busca en la lista la primera reserva de la lista hecha por el co-trabajador indicado. En caso de encontrarse la reserva, se devolverá el índice que ocupa la reserva en la lista de reservas. En caso de que no se encuentre, se devolverá un valor NO_BOOKING.

b) [35%] Implementad la acción ***processCheckins*** (api.c) que permita seguir las acciones marcadas en el itinerario amarillo en el esquema anteriormente presentado. Se deben ir desencolando los co-trabajadores de la cola e ir procesando individualmente su checkin. Por cada co-trabajador, se buscará su reserva en la lista de reservas pendientes (pendingBookings) haciendo uso de la función del apartado anterior. Una vez localizada la reserva, se procesará sólo en caso de que la fecha de entrada de la reserva coincida con la actual (que se recibe por parámetro en la acción). El procesamiento de la reserva consistirá en eliminarla de la lista de reservas en pendientes e insertarla al principio de la lista de reservas en curso. Será necesario, además, calcular y asignar cuáles (y cuántos espacios) serán necesarios para alojar a los cotrabajadores incluidos en la reserva.

NOTAS:

- Para saber si un co-trabajador existe en la tabla de co-trabajadores puede buscar a partir de su identificador con la ayuda de la función *coworkerTableFind*.
- Tenéis disponibles operaciones para trabajar con listas de reservas en list.c.
- Podéis calcular los espacios necesarios para dar cobertura a la reserva haciendo una invocación a la acción *assignSpacesForBooking* (api.c).

Ejercicio 2: Checkout de co-trabajadores de un centro [50%]

Siguiendo el análisis del procesamiento de los movimientos de un centro de coworking, nos centramos ahora en el proceso de checkout. A tal efecto, se pide que resolváis los siguientes apartados:

a) [15%] Implementad la función ***calculatePrice*** (api.c) que, a partir de un centro, una reserva y una fecha actual, calcule el importe a pagar por la reserva. Para calcular el importe es necesario tener presente que se paga una cantidad por cada espacio reservado y otra por cada cotrabajador asignado dentro de estos espacios. Por lo que respecta al espacio, el centro marca un precio de referencia que corresponde al espacio doble (workplace por dos coworkers). A partir de ese precio, el espacio individual se calcula como un 0.75 del precio del doble. El espacio triple tiene un factor de 1.5 veces el precio del doble, mientras que en el caso del espacio cuádruple, el factor es de 1.75. Después, por cada cotrabajador, se paga según el equipamiento que necesita: 0€ adicionales si sólo necesita el espacio, 10€ si necesita espacio y telefonía, 25€ si necesita espacio, telefonía e internet y, por último, 40€ si necesita espacio, telefonía, internet y proyector. En caso de salidas posteriores a la fecha prevista, se carga un 20% adicional al total calculado (independientemente del tiempo excedido).

b) [35%] Implementad la acción ***processCheckouts*** (api.c) que permita seguir las acciones seguidas con itinerario verde según el esquema anteriormente presentado. Se irán desencolando los co-trabajadores y tratándolos en el orden en que se encuentran en la cola. Por cada cotrabajador, se buscará su reserva en la lista de reservas en curso (currentBookings) haciendo uso de la función del apartado a) del ejercicio 1. Una vez localizada la reserva, se procesará sólo en caso de que la fecha de salida de la reserva coincida con la actual (que se recibe por parámetro en la acción). El procesamiento de la reserva consistirá en eliminarla de la lista de reservas en curso e insertarla al principio de la lista de reservas completadas. Será necesario además calcular el precio de la estancia y liberar los espacios ocupados (campos price y assignedSpaces de tBooking).

NOTAS:

- Para saber si un co-trabajador existe en la tabla de co-trabajadores podeis buscar a partir de su id con la ayuda de la función coworkerTableFind.
- Tenéis disponibles operaciones para trabajar con listas de reservas en list.c.
- Podéis calcular el precio de una estancia con ayuda de la función calculatePrice, y la liberación de espacios mediante freeSpacesOfBooking.
- Para calcular el número de días que hay entre dos fechas (necesario para resolver el primer apartado), podéis utilizar el cálculo que hace la función de la página siguiente.

Observad que, para poder realizar estos cálculos, deberéis hacer un include <time.h> al principio del archivo.

```
int numberOfDays(tDate d1, tDate d2)

{

    double seconds;

    struct tm t1= { 0, 0, 0, 0, 0, 0, 0, 0, 0};

    struct tm t2= { 0, 0, 0, 0, 0, 0, 0, 0, 0};

    t1.tm_mday= d1.day;

    t1.tm_mon= d1.month-1;

    t1.tm_year= d1.year-1900;

    t2.tm_mday= d2.day;

    t2.tm_mon= d2.month-1;

    t2.tm_year= d2.year-1900;

    seconds= fabs( difftime( mktime(&t1), mktime(&t2) ) );

    return (int)(seconds / 86400.0);

}
```