UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Informática II
Semestre 2025 – 1
Desafío II. 12 de Mayo del 2025

Desafío II: Mercado de estadías hogareñas UdeAStay

Uno de los objetivos de programar consiste en modelar una situación del mundo real, delimitada por un contexto. Al representar en el programa las entidades y acciones de interés, se pueden estudiar o automatizar las dinámicas propias del contexto respectivo.

Objetivos

- Desarrollar la capacidad de análisis y solución de problemas en los estudiantes, enfrentándolos a problemáticas de la vida cotidiana.
- Verificar si el estudiante adquirió las destrezas y conocimientos fundamentales de la programación orientada a objetos en C++: abstracción, encapsulación, relaciones, diseño de diagrama de clases, funciones amigas, sobrecarga y uso de plantillas.

Si usted ha llevado un proceso disciplinado de aprendizaje a lo largo del semestre, esta es una oportunidad de demostrarlo y podrá plantear una solución satisfactoria. En caso contrario, podrá identificar sus debilidades y tomar medidas a fin de poder abordar situaciones similares a futuro.

Trate de valorar la verdadera complejidad del problema planteado, no se rinda antes de intentarlo o de plantear los posibles escenarios de solución. Se dará cuenta que si bien, al principio le puede parecer difícil; ya ha tenido la oportunidad de enfrentarse a problemas similares. Si se toma el tiempo adecuado para analizar, el proceso de codificación será eficaz y no le tomará mucho tiempo.

Esperamos que disfrute del desafío propuesto. Lea primero todo el documento antes de comenzar y asegúrese de entender muy bien las instrucciones antes de desarrollar esta actividad evaluativa.

Fue revisado por los profesores Aníbal Guerra y Augusto Salazar.

Introducción

Una estadía hogareña (en inglés, homestay) es una experiencia que permite rentar a visitantes foráneos, un alojamiento perteneciente a un particular. Es una forma de hospitalidad y alojamiento orientada a brindar una experiencia de conocimiento cercano de la cultura local. El objetivo de este desafío es desarrollar un sistema

para la administración de un mercado de estadías hogareñas, utilizando POO. El sistema debe permitir la gestión eficiente de los alojamientos, las reservaciones, los anfitriones y los huéspedes, entre otros.

Figura 1. Vista panorámica del problado, uno de los principales sectores turísticos de la ciudad de Medellín (Tomado de: https://es.m.wikipedia.org/wiki/Archivo:El_Poblado_Medell.jpg)

UdeAStay aspira a posicionarse como una compañía líder, ayudando a promover la actividad turística y a regular el comercio de alojamientos en el departamento de Antioquia. En el largo plazo, se espera consolidar su operación a través de una plataforma en línea que ofrezca su servicio a nivel nacional.

Cada alojamiento posee: un nombre, un código identificador, un anfitrión responsable, un departamento y municipio al que pertenece, un tipo (casa o apartamento), una dirección, un precio por noche, un conjunto de amenidades (ascensor, piscina, aire acondicionado, caja fuerte, parqueadero, patio, etc. Adicionalmente, para cada alojamiento se conocen las fechas futuras (hasta por un año) en las que se encuentra reservado por algún huésped. Para cada reservación es necesario conocer: fecha de entrada, duración (en cantidad de noches), código de la reservación, código del alojamiento, documento del huésped, método de pago (PSE, TCrédito), fecha del pago, y monto. Adicionalmente, el huésped puede incluir anotaciones en su reservación, por lo que debe disponer de espacio para escribir sus inquietudes al anfitrión, con hasta 1000 caracteres.

De los anfitriones se conoce su número de documento, su antigüedad en la plataforma (en meses), su puntuación (de 0 a 5.0) y los respectivos alojamientos que administra. De los huéspedes se conoce su número de documento, su antigüedad en la plataforma, su puntuación (de 0 a 5.0) y la información de sus reservas; en caso de que posea alguna. Un anfitrión puede manejar uno o muchos alojamientos, mientras que un huésped puede estar registrado en la plataforma teniendo cero o más reservaciones, siempre y cuando sus reservaciones no coincidan en ninguna fecha.

Funcionalidades esenciales

I - Carga / actualización de los datos: Se requiere desarrollar los algoritmos requeridos a fin de leer y actualizar los datos en el almacenamiento permanente. Esta funcionalidad no debe aparecer visible en el menú presentado a los usuarios.

Los programadores deben diseñar el formato de uno o más archivos destinados a contener la información de huéspedes, anfitriones, alojamientos y reservaciones (vigentes o históricas). Las reservaciones correspondientes a fechas pasadas deben quedar almacenadas de forma permanente (en el histórico).
El formato de los archivos debe presentarse y explicarse adecuadamente dentro del informe de documentación a entregar junto con el desafío. Esto se considera esencial dentro de la documentación.

Adicionalmente, presente un menú para acceder a funcionalidades que permitan:

II - Ingreso a la plataforma según el perfil (huésped o anfitrión).
Funcionalidad sencilla que permite iniciar sesión con las credenciales personales y mostrar el menú de funcionalidades disponibles según el rol. Los datos requeridos para verificar el inicio de sesión se recuperan del almacenamiento permanente. En este enunciado no se contempla implementar una funcionalidad de registro de usuarios.

III - Reservar alojamiento
Esta funcionalidad es exclusiva de los huéspedes.

A partir de una fecha, un municipio específico y una cantidad de noches, el programa debe mostrar al usuario un listado de los alojamientos disponibles y permitirle a continuación hacer la reservación de uno de ellos. El usuario puede elegir aplicar un filtro que corresponde a fijar el costo máximo por noche del alojamiento, establecer la puntuación mínima que debe tener el anfitrión, o ambas si así lo desea. Por ende, para concretar una reserva primero se realiza una búsqueda y luego el huésped selecciona entre los resultados que satisfagan el criterio.

Una segunda opción de búsqueda de alojamiento consiste en realizarlo directamente según el código del mismo.

Al finalizar la reserva, el programa debe mostrar comprobante de confirmación, indicando el código de la reserva (gestionada automáticamente), nombre del

usuario, código del alojamiento, la fecha de inicio y de finalización. Ambas fechas deben mostrarse en el formato: **nombreDía, día "de" nombreMes "del" año**.

Recuerde validar que el usuario no tenga otras reservas en ninguno de esos días y que el alojamiento esté disponible por la totalidad de la reservación.

IV - Anular reservación: Funcionalidad disponible tanto para el huésped como para el anfitrión asociado a una reservación. Elimina la reservación correspondiente al código indicado.

V. Consultar reservaciones anfitrión: Esta funcionalidad permite al anfitrión imprimir en pantalla todas las reservaciones activas para todos sus alojamientos, que correspondan a un rango de fechas específico.

VI. Actualizar histórico: Esta funcionalidad mueve todas las reservaciones de fechas previas a una "fecha de corte" al archivo histórico en el almacenamiento permanente, y actualiza las estructuras de datos para que permitan almacenar reservaciones en cualquier fecha de los próximos 12 meses

.

Respecto a esa "fecha de corte", puede asumir lo siguiente:

a) Se debe preguntar al usuario anfitrión.
b) No será menor que las fechas previamente existentes en el archivo histórico.
c) Se toma como base para establecer los próximos 12 meses que se habilitarán para reservaciones.

VII. Medición del consumo de recursos: Esta funcionalidad tiene propósitos académicos, a fin de soportar el desarrollo. De manera automática, se debe mostrar en pantalla dos valores a**l finalizar la ejecución de todas las demás funcionalidades del sistema:**

a. La cantidad de iteraciones requeridas (directa e indirectamente) para completar esa tarea específica.
b. El total de memoria que se está consumiendo en ese momento exacto, por conceptos de los objetos creados.

Requisitos del desarrollo

El cliente requiere un programa para la gestión del mercado de estadías hogareñas UdeAStay. Este debe permitir representar las estructuras de datos requeridas y ejecutar las funcionalidades especificadas en este enunciado.

Se recomienda delimitar el desarrollo a lo aquí descrito, ya que la dimensión del problema en un escenario totalmente real puede tener una complejidad superior al

tiempo estipulado para la entrega. En caso de duda sobre los requerimientos, consulte con el cliente.

De acuerdo con lo anterior, entre otras cosas, usted deberá:

1. **[10%]** Contextualice el problema, analícelo y diseñe el diagrama de clases correspondiente a su solución. Refleje adecuadamente las relaciones implícitas en la problemática usando la notación UML simplificada impartida en las clases teóricas.
   La entrega de este diagrama es obligatoria y sin ella no se dará lugar a la sustentación. Considere en su modelo: incorporar adecuadamente los constructores, constructores de copia, destructor, getters, setters, operaciones de despliegue, sobrecarga de métodos y de operadores necesarios.

2. **[20%]** Previo a la implementación, verifique el cumplimiento del requisito de eficiencia especificado en la sección 3 del apartado “Requisitos de la entrega” de este documento. Seleccione concienzudamente los tipos y estructuras de datos que le permitirán implementar las clases planteadas en su diagrama.

   Reduzca o elimine la redundancia innecesaria de datos y además analice el efecto de las estructuras de datos propuestas en la eficiencia de las instrucciones de cada funcionalidad. Este análisis debe reflejarlo en el informe y en una de las secciones del video.

3. **[70%]** Presente la implementación de su programa, cuya interacción se centra en un menú que permita acceder de manera independiente a las funcionalidades. Considere las restricciones implícitas en la lógica del problema. Por ejemplo, las reservaciones solo pueden crearse entre usuarios previamente existentes, entre otras. El código entregado debe coincidir en estructura y contenido con lo presentado en los apartados anteriores de esta sección.
   La ponderación de las funcionalidades se distribuye según su complejidad.

Formalidades de la entrega

A continuación, se describen los requisitos que se deben cumplir. El incumplimiento de cualquiera de ellos implicaría alguna penalidad o que su nota sea cero.

1. Genere un informe en donde se detalle el desarrollo del proyecto, explique entre otras cosas:
   a. Análisis del problema y consideraciones para la alternativa de solución propuesta.

   b. Diagrama de clases de la solución planteada. No debe ser un diagrama trivial que sólo incluya una o dos clases.
   c. Descripción en alto nivel la lógica de las tareas que usted definió para aquellos subprogramas cuya solución no sea trivial.
   d. Algoritmos implementados debidamente intra-documentados.
   e. Problemas de desarrollo que afrontó.
   f. Evolución de la solución y consideraciones para tener en cuenta en la implementación.

2. La solución debe ser implementada en lenguaje C++ y debe basarse en el paradigma de POO.

3. La implementación debe considerar el criterio de eficiencia y el buen diseño tanto de las estructuras de datos como de los módulos implementados. Considere además en este apartado la utilización de referencias vs la realización de copias innecesarias sobre datos estructurados.

4. La implementación no debe considerar el uso de la STL. Las estructuras de datos y códigos presentados deben ser creación propia y utilizar memoria dinámica. La modularidad de los componentes del sistema es obligatoria.

5. Se debe crear un repositorio público para cargar todos los archivos relacionados a la solución planteada (informe, código fuente y otros anexos). Cada estudiante debe desarrollar en su rama propia.

6. Una vez cumplida la fecha de entrega no se debe hacer modificación alguna al repositorio.

7. Se deben hacer *commits* de forma regular (al menos uno al día) de tal forma que se evidencie la evolución de la propuesta de solución y su implementación.

8. Se debe adjuntar un enlace de *youtube* a un video que debe incluir lo siguiente:
   a. Presentación de la solución planteada. Análisis realizado y explicación de la arquitectura del sistema (3 minutos máximo).
   b. Análisis de la solución con una perspectiva desde la eficiencia, tanto en el uso de memoria como en la ejecución de las funcionalidades requeridas (3 minutos máximo).
   c. Demostración de funcionamiento del sistema. Explicar cómo funciona: ejemplos demostrativos de su ejecución (3 minutos máximo).
   d. Explicación del código fuente. Justifique la elección de las variables y estructuras de control usadas, y destaque que ventaja ofrecen estos en comparación con otras opciones (5 minutos máximo).
   e. La duración total del video no debe exceder 14 minutos ni ser inferior a 5 minutos.

f. Asegúrese que el video tenga buen sonido y que se puedan visualizar bien los componentes presentados. Ambos miembros deben participar equitativamente en el video.

9. El plazo de entrega se divide en dos momentos:
    a. El día 17 de Mayo para adjuntar la evidencia del proceso de análisis y diseño de la solución.
    b. El día 23 de Mayo para adjuntar la evidencia del proceso de implementación.

10. Se deben adjuntar **dos enlaces**: uno al repositorio y otro al video, nada más.

11. Para la evaluación del desafío se realizará una sustentación oral en un horario concertado con el profesor. La asistencia a la sustentación es obligatoria para optar a calificación. Sólo pueden asistir estudiantes que no hayan cancelado.

12. Otros condicionantes esenciales para acceder a la sustentación:
    a. La documentación y diagrama de clases deben describir lo presentado en la estructura del proyecto.
    b. El manejo de las entradas y salidas requeridas por archivo.
    c. Documentación apropiada de los formatos de archivo y del resto del informe del desarrollo.