Informe Desafío 2
Andrés Felipe Sepúlveda R.
71.363.392

a) Análisis del problema y consideraciones para la alternativa de solución propuesta.

El desafío 2 consiste en implementar una plataforma para reservas de alojamiento llamada UdeAStay, dicha implementación debe estar soportada en el paradigma de la POO expuesta en las clases del curso. Dentro de los aspectos clave a tener en cuenta están las características de los alojamientos, las cualidades de huéspedes y anfitriones y las condiciones para realizar las reservas.

- Alojamiento: debe poseer un nombre, un código, un anfitrión responsable, un departamento, un municipio, un tipo, una dirección, un precio por noche y unas amenidades. Dentro de las condiciones se debe conocer las fechas futuras de reserva.

- Reservación: fecha de entrada, duración, código, documento del huésped, método de pago, fecha de pago y monto.

- Anfitriones: documento, antigüedad, puntuación, alojamientos que administra.

- Huéspedes: documento, antigüedad, puntuación, reservas que posee.

Para presentar una solución posible a este desafío, se deben tener en cuenta la relación existente entre los cuatro grupos de datos presentados.

- ✓ Huésped – reservación: un huésped puede tener múltiples reservaciones y las reservaciones guardan el documento del huésped.
- ✓ Alojamiento – reservación: un alojamiento puede tener múltiples reservaciones y cada reservación contiene un código de alojamiento.
- ✓ Anfitrión – alojamiento: un anfitrión administra uno o varios alojamientos y cada alojamiento tiene un documento de anfitrión.

Para implementar la solución se van a definir 4 clases, las cuales coinciden con los grupos de datos mencionados anteriormente: Alojamiento, Reservación, Huésped y Anfitrión. Para el manejo de los archivos se dispondrán en el formato .txt con la información listada por líneas y separada por comas, con cada uno de los componentes funcionales requeridos.

Para la estructura general del código, de acuerdo al análisis preliminar realizado, se podrán utilizar unas funciones principales y otras auxiliares de modo que la modularidad sea el pilar del desarrollo.

Las funciones auxiliares serían:

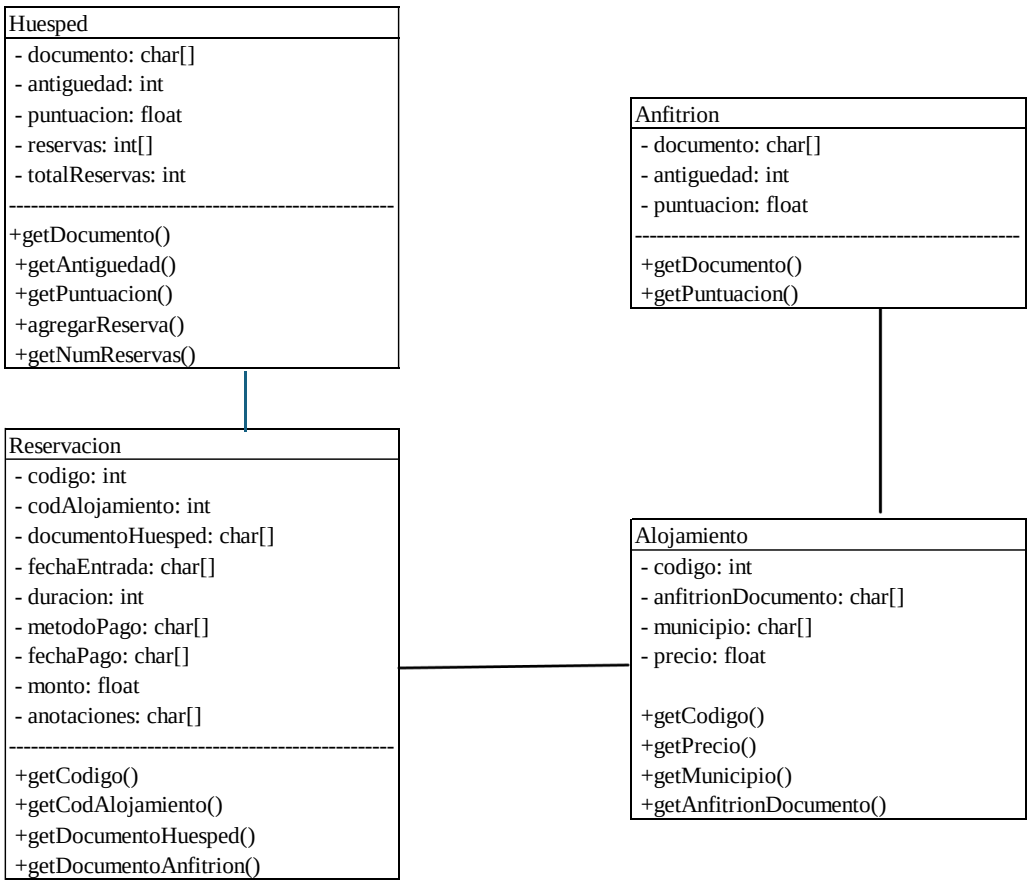
- Buscar huésped por documento.
- Buscar anfitrión por documento.
- Buscar alojamiento por código.

Las funciones principales serían:

- Mostrar el consumo.
- Anular reservación.
- Fecha anterior.
- Actualizar histórico.
- Funciones para cargar y guardar los archivos.

Con la estructura y las funciones mencionadas se puede plantear una solución al problema planteado cumpliendo con los requisitos mínimos.

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.

De manera inicial voy a plantear las funciones que se indican, esto estará sujeto a cambios a medida que avance con la solución del problema, teniendo en cuenta la complejidad que se vaya encontrando y las pruebas realizadas.

1. buscarHuespedPorDocumento / buscarAnfitrionPorDocumento / buscarAlojamientoPorCodigo

Propósito: Buscar un elemento (huésped, anfitrión o alojamiento) en su arreglo respectivo.

• Recorren un arreglo lineal (búsqueda secuencial).

• Comparan el valor buscado (documento o código).

• Devuelven un puntero al objeto encontrado o nullptr.

2. Reservacion::getDocumentoAnfitrion()

Propósito: Obtener el documento del anfitrión asociado a una reservación.

• A partir del código de alojamiento guardado en la reservación, busca el alojamiento en el arreglo global.

• Si lo encuentra, devuelve el documento del anfitrión asociado a ese alojamiento.

3. anularReservacion

Propósito: Permitir la anulación (eliminación) de una reserva si el usuario es el huésped o el anfitrión correspondiente.

• Recorre las reservaciones buscando la que coincide con el código dado.

• Verifica si el documento ingresado corresponde al huésped o al anfitrión del alojamiento reservado.

• Si tiene permisos, elimina la reservación desplazando los elementos restantes en el arreglo.

4. actualizarHistorico

Propósito: Mover al archivo histórico aquellas reservaciones cuya fecha de entrada ya pasó.

• Recorre el arreglo de reservaciones.

• Compara la fecha de entrada de cada reserva con la fecha actual (hoy).

• Si es anterior, escribe la información de esa reserva en el archivo historico_reservas.txt y la elimina del arreglo de reservaciones.

• Las reservas restantes se conservan en memoria.

5. mostrarConsumo

Propósito: Mostrar métricas sobre el uso de recursos del sistema.

• Imprime cuántas iteraciones se hicieron (útil para evaluar eficiencia).

• Calcula y muestra el uso de memoria basado en el tamaño de los objetos cargados.