



Informe

Sistema de Reservas de Vuelos de Dragón

Integrantes:

- Matias Arenas
- Saúl Arnulfo Muñoz Pedreros

| | |
|---|-----------|
| Enunciado..... | 3 |
| Problemática..... | 3 |
| Necesitamos..... | 3 |
| Descripción..... | 3 |
| Realizar Reserva..... | 3 |
| Cancelar Reserva..... | 4 |
| Buscar Reserva..... | 4 |
| Mostrar Reservas Por Destino..... | 4 |
| Mostrar Reservas Ordenadas..... | 4 |
| Consideraciones de Envío..... | 4 |
| Introducción..... | 5 |
| Metodología..... | 5 |
| Resultados..... | 6 |
| Funciones Usadas..... | 7 |
| Función principal..... | 7 |
| Función del menú..... | 7 |
| Función de inserción..... | 7 |
| Función de eliminación..... | 8 |
| Función de búsqueda binaria..... | 8 |
| Función de búsqueda por destino..... | 9 |
| Función de recorrido en orden..... | 9 |
| Función de inicialización..... | 10 |
| Función de valor mínimo..... | 10 |
| Función de liberación de memoria del árbol..... | 10 |
| Definición del nodo..... | 10 |
| Análisis..... | 11 |
| Conclusiones..... | 12 |
| Recomendaciones..... | 12 |

Enunciado

Sistema de Reservas de Vuelos de Dragón

Problemática

Aventurero, tenemos un problema muy grande, los viajes a otros reinos se han complicado, pues tenemos mucho desorden, ayúdanos con esas maquinitas inventadas para poder organizarnos.

Necesitamos

- Realizar reservas de vuelos
- Cancelar reservas
- Buscar en los registros de reservas
- Mostrar reservas por destino
- Mostrar reservas ordenadas

Descripción

Debes desarrollar un sistema de reservas de vuelos utilizando un árbol binario de búsqueda. Cada reserva de vuelo estará representada por un número de reserva (entero), el nombre del aventurero (cadena de caracteres) y el destino del vuelo (cadena de caracteres). El sistema debe permitir las siguientes operaciones.

Realizar Reserva

Esta función permite realizar una nueva reserva de vuelo. El aventurero proporcionará el número de reserva, el nombre del pasajero y el destino del vuelo. La reserva se insertará en el árbol binario de búsqueda según el número de reserva.

Cancelar Reserva

Esta función permitirá cancelar una reserva existente. El aventurero proporcionará el número de reserva y el sistema deberá eliminarla del árbol binario de búsqueda.

Buscar Reserva

Esta función buscará una reserva específica en el sistema. El usuario proporcionará el número de reserva y el sistema deberá mostrar el nombre del aventurero y el destino correspondiente, si la reserva existe.

Mostrar Reservas Por Destino

Esta función mostrará todas las reservas de vuelo para un destino específico. El aventurero proporcionará el destino y el sistema deberá mostrar el número de reserva y el nombre del aventurero de todas las reservas que coincidan con ese destino.

Mostrar Reservas Ordenadas

Esta función mostrará todas las reservas de vuelo ordenadas por número de reserva en orden ascendente. El sistema debe proporcionar un menú de opciones para que el aventurero pueda interactuar con él. Debes implementar todas las funciones necesarias para realizar las operaciones descritas anteriormente.

Consideraciones de Envío

- Se puede realizar en dúos
- Medio de Entrega: subir en Moodle UBB los códigos fuentes C y el informe, compactado en un archivo llamado con los apellidos de cada uno de los integrantes.

Formato Informe: formato pdf que contenga, para cada problema:

- Enunciado de Problema
- Un análisis simple de la solución planteada, indicando la(s) estructura (s) utilizada (s).
- Casos de prueba (valores de entrada y captura de imágenes de la ejecución para las salidas asociadas).
- Conclusión y comentarios al trabajo efectuado.
- Recuerda gestionar correctamente la memoria y liberarla cuando sea necesario.

Introducción

Esta es una implementación en lenguaje C de una estructura de datos de árbol binario de búsqueda para una aplicación de gestión de reservas. El programa permite a los usuarios ingresar nuevas reservas, cancelar reservas existentes, buscar reservas por número de reserva o destino, y mostrar todas las reservas almacenadas en el árbol. El código incluye funciones para inicializar un nuevo nodo, insertar un nodo en el árbol, eliminar un nodo del árbol, y buscar un nodo en el árbol por número de reserva o destino. También se incluye una función para liberar la memoria utilizada por el árbol.

Metodología

La metodología usada en el desarrollo de este proyecto fue la “Metodología Agile”. Esta se basa en la colaboración y la adaptabilidad, se deben hacer pequeñas modificaciones funcionales en lugar de grandes entregas al final del proyecto. Esto permite responder rápidamente a los cambios y retroalimentación, y asegura que el proyecto se mantenga en el camino correcto. Para lograr un trabajo colaborativo, nos enfocamos en trabajar con la plataforma [GitHub](#), se creó un [Repositorio](#) y se seleccionó a los participantes del grupo como participantes del repositorio: [Matias Arenas](#) y [Saul Muñoz](#).

Resultados

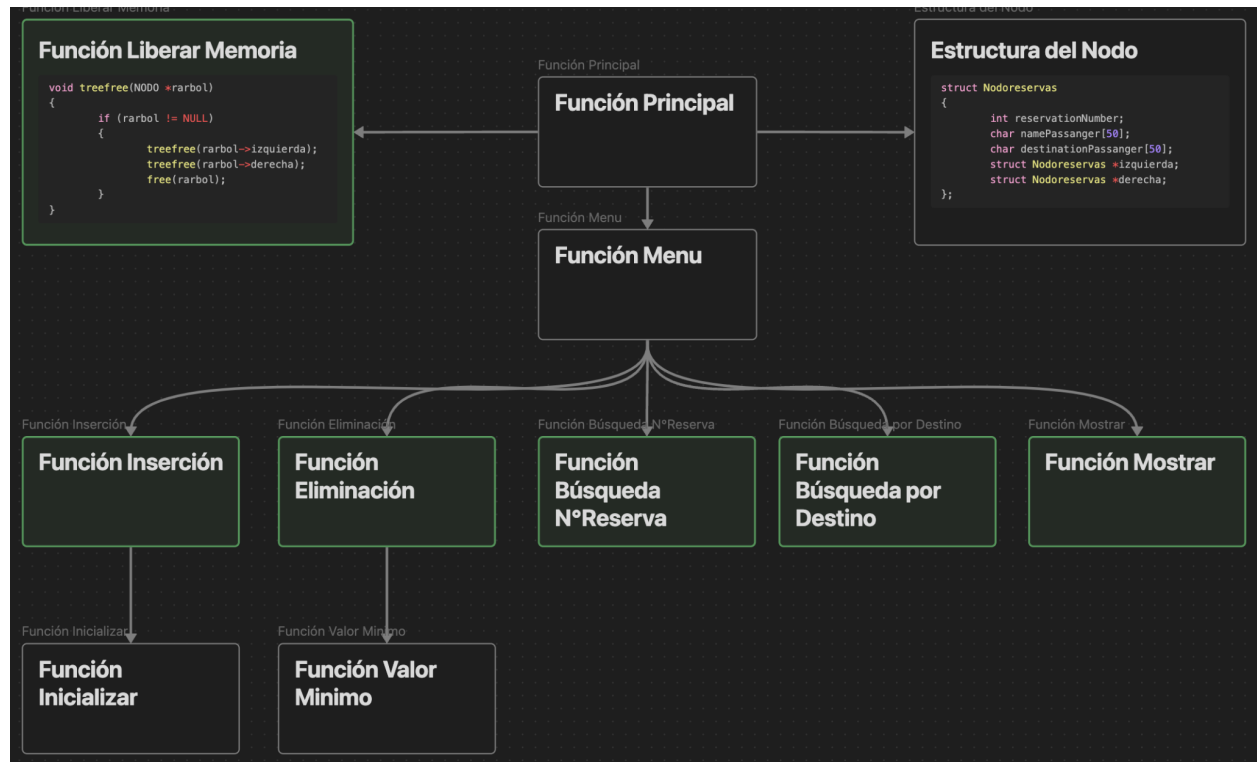


Imagen 1, Diagrama Código Reserva de Vuelos. Cuadros en verde: Función **Recursiva**, creada con Obsidian.

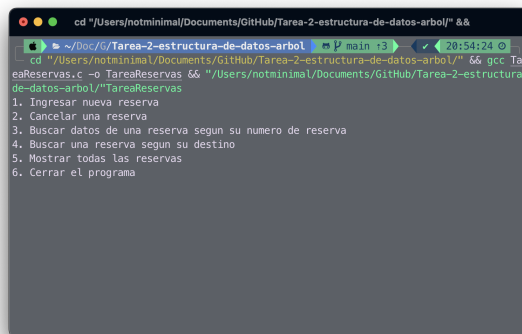
Funciones Usadas

Función principal

- Definir variables
- Recorrer las opciones del menú

Función del menú

- Mostrar las opciones del menú
- Obtener la entrada del usuario
- Validar la entrada del usuario

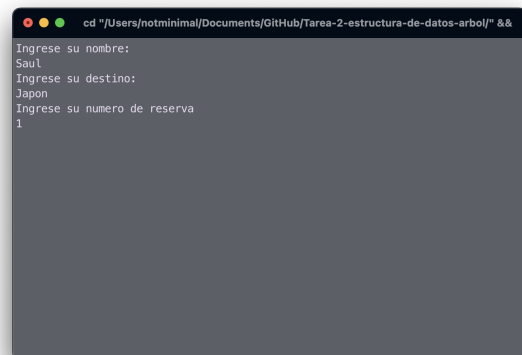


```
cd "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-de-datos-arbol/" &&
cd "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-de-datos-arbol/" && gcc Tar
eaReservas.c -o TareaReservas && "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-
de-datos-arbol/"TareaReservas
1. Ingresar nueva reserva
2. Cancelar una reserva
3. Buscar datos de una reserva segun su numero de reserva
4. Buscar una reserva segun su destino
5. Mostrar todas las reservas
6. Cerrar el programa
```

Imagen 2, Terminal iTerm.

Función de inserción

- Inicializar un nuevo nodo
- Recorrer el árbol para encontrar la posición correcta
- Insertar el nuevo nodo(Si existe, hacerlo de manera recursiva)



```
cd "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-de-datos-arbol/" &&
Ingrese su nombre:
Saul
Ingrese su destino:
Japon
Ingrese su numero de reserva
1
```

Imagen 3, Terminal iTerm.

Función de eliminación

- Recorrer el árbol para encontrar el nodo a eliminar
- Manejar diferentes casos para la eliminación del nodo



Imagen 4, Terminal iTerm.

Función de búsqueda binaria

- Recorrer el árbol para encontrar el nodo con el número de reserva correspondiente
- Devolver el resultado

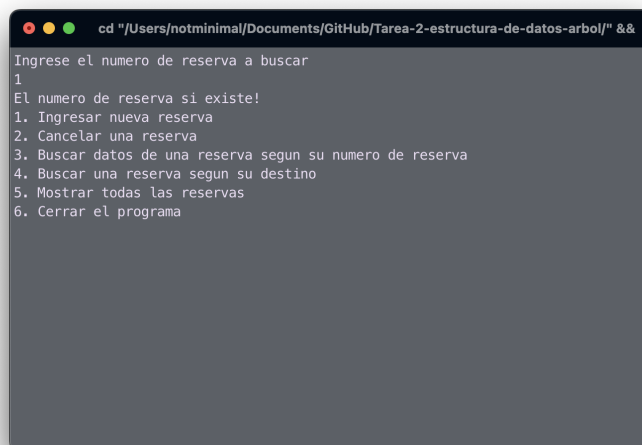
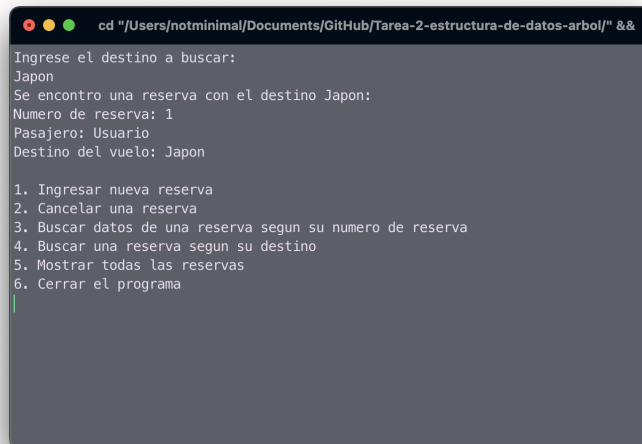


Imagen 4, Terminal iTerm.

Función de búsqueda por destino

- Recorrer el árbol para encontrar el nodo con el destino correspondiente y mostrarlo en esa misma instancia
- Devolver el resultado de la búsqueda



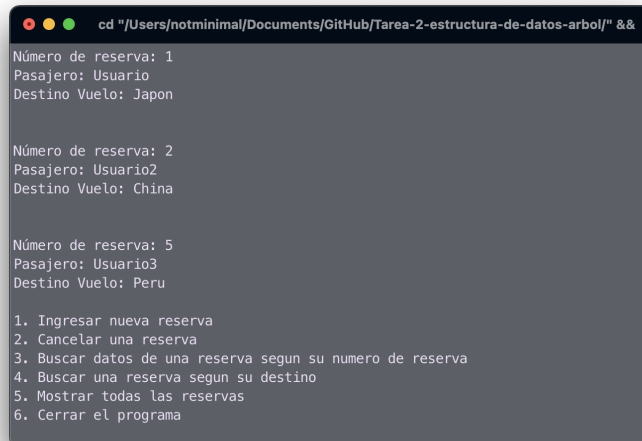
```
cd "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-de-datos-arbol/" &&
Ingrese el destino a buscar:
Japon
Se encontro una reserva con el destino Japon:
Numero de reserva: 1
Pasajero: Usuario
Destino del vuelo: Japon

1. Ingresar nueva reserva
2. Cancelar una reserva
3. Buscar datos de una reserva segun su numero de reserva
4. Buscar una reserva segun su destino
5. Mostrar todas las reservas
6. Cerrar el programa
```

Imagen 5, Terminal iTerm.

Función de recorrido en orden

- Recorrer el árbol en orden
- Imprimir la información del nodo



```
cd "/Users/notminimal/Documents/GitHub/Tarea-2-estructura-de-datos-arbol/" &&
Número de reserva: 1
Pasajero: Usuario
Destino Vuelo: Japon

Número de reserva: 2
Pasajero: Usuario2
Destino Vuelo: China

Número de reserva: 5
Pasajero: Usuario3
Destino Vuelo: Peru

1. Ingresar nueva reserva
2. Cancelar una reserva
3. Buscar datos de una reserva segun su numero de reserva
4. Buscar una reserva segun su destino
5. Mostrar todas las reservas
6. Cerrar el programa
```

Imagen 6, Terminal iTerm.

Función de inicialización

- Asignar memoria para un nuevo nodo
- Establecer los valores del nodo
- Devolver el nuevo nodo

Función de valor mínimo

- Recorrer el árbol para encontrar el nodo con el valor mínimo
- Devolver el resultado

Función de liberación de memoria del árbol

- Recorrer el árbol en postorden
- Liberar la memoria para cada nodo

Definición del nodo

- Definir la estructura para los nodos de reserva
- Declarar las variables para el número de reserva, el nombre del pasajero, el destino y los nodos hijos

Análisis

Además de ser un buen ejemplo de cómo utilizar una estructura de datos de árbol binario de búsqueda en la implementación de un sistema de reservas, en el código se aplicarán buenas prácticas de programación, como lo son un algoritmo de búsqueda binaria, liberación de memoria, etc.

Por ejemplo, se utilizan punteros para crear y manipular los nodos del árbol, lo que permite una gestión eficiente de la memoria y la eliminación de nodos. También se utilizan prototipos de funciones para declarar las funciones antes de su implementación, lo que ayuda a evitar errores de compilación y a mejorar la legibilidad del código.

El programa también incluye comentarios bien estructurados y descriptivos que explican qué hace cada parte del código, lo que facilita la comprensión y el mantenimiento del programa.

Otro aspecto destacable es que el programa incluye una función de liberación de memoria que se utiliza al final del programa para liberar la memoria utilizada por el árbol. Esto es importante para evitar pérdidas de memoria y garantizar que el programa se ejecute de manera eficiente.

En cuanto a las mejoras que se podrían realizar al programa, se podría ampliar la funcionalidad de búsqueda de reservas por destino para que devuelva todas las reservas que coincidan con el destino buscado en lugar de solo la primera que encuentre. También se podría agregar una función de edición de reservas existentes para permitir que los usuarios actualicen información de una reserva ya existente, como el nombre del pasajero o el destino del vuelo.

En definitiva, el código presentado es un buen ejemplo de cómo utilizar una estructura de datos en la implementación de un sistema de reservas y muestra buenas prácticas de programación en C.

Conclusiones

En conclusión, la implementación de un sistema de gestión de reservas de vuelos basado en árboles binarios de búsqueda resulta ser una solución práctica y eficiente. El código presentado demuestra una clara separación de funciones, lo que favorece la modularidad del sistema y permite su fácil comprensión y mantenimiento.

Entre las soluciones concretas que se presentan, destaca la inclusión de funciones que permiten agregar y cancelar reservas, buscar reservas por número o destino, y mostrar todas las reservas existentes. Además, se hace uso de memoria dinámica y técnicas de programación como la Divide & Conquer, lo que permite que el código sea eficiente y mantenible.

En cuanto a las prácticas usadas, se destaca el uso de comentarios a lo largo del código, lo que mejora su legibilidad y facilita su comprensión. También se hace uso de la biblioteca `string.h` para comparar y copiar strings, etc y `stdbool.h` para el manejo de valores booleanos, lo que permite una mayor claridad en el código.

En resumen, la implementación de un sistema de gestión de reservas de vuelos basado en árboles binarios es una solución efectiva y práctica para la gestión de datos en una empresa o institución. El código presentado demuestra una clara separación de funciones y el uso de técnicas de programación eficientes, lo que lo hace fácil de entender y mantener.

Recomendaciones

Para mejorar el código, se debe incluir comentarios más descriptivos en las funciones para que sea más fácil entender lo que está sucediendo. Además, sería conveniente agregar alguna validación de entrada de datos para evitar que el programa falle si se ingresan datos incorrectos o en un formato incorrecto.