Proyecto Algoritmos y Estructuras de Datos

Isabela Ruiz

RESUMEN

Diseñar e implementar una libreta de contactos en C++ que utilice eficientemente algoritmos y estructuras de datos, como árboles de búsqueda binaria, listas enlazadas, vectores y colas, para gestionar una lista de contactos personales, permitiendo a los usuarios realizar operaciones diversas de manera ordenada y eficaz.

ABSTRACT

This project focuses on the design and implementation of a contact book in C++, efficiently utilizing algorithms and data structures such as binary search trees, linked lists, vectors, and queues. The objective is to enable users to manage a personal contact list efficiently, with key operations like adding, deleting, and searching for contacts, along with advanced features such as visualizing frequent contacts and performing automatic backups. The modular and scalable approach is supported by an intuitive user interface and comprehensive testing to ensure the quality and functionality of the system.

PALABRAS CLAVE

C++, libreta de contactos, algoritmos, estructuras de datos, árbol de búsqueda binaria, listas enlazadas, vectores, colas, interfaz de usuario, copias de seguridad, contactos frecuentes.

KEY WORDS

C++, contact book, algorithms, data structures, binary search tree, linked lists, vectors, queues, user interface, backups, frequent contacts

INTRODUCCIÓN

En el marco del desarrollo de un proyecto centrado en Algoritmos y Estructuras de Datos, se plantea el diseño e implementación de una libreta de contactos en C++. El objetivo principal de esta libreta es permitir a los usuarios gestionar una lista de contactos personales de manera eficiente y organizada.

I. OBJETIVO GENERAL

Diseñar e implementar una libreta de contactos en C++ que utilice eficientemente algoritmos y estructuras de datos, como árboles de búsqueda binaria, listas enlazadas, vectores y colas, para gestionar una lista de contactos personales, permitiendo a los usuarios realizar operaciones diversas de manera ordenada y eficaz.

II. OBJETIVOS ESPECIFICOS

Desarrollar una interfaz de usuario amigable: Diseñar una interfaz intuitiva que facilite a los usuarios agregar, eliminar y buscar contactos, así como acceder a las funcionalidades de la libreta de manera clara.

Implementar la estructura de datos principal: Desarrollar un árbol de búsqueda binaria (BST) para mantener la lista de contactos ordenada alfabéticamente por nombre, gestionando duplicados mediante listas enlazadas.

Optimizar el acceso a la información de contactos: Utilizar un vector para mantener un índice rápido de los contactos, mejorando la eficiencia en la recuperación de información y optimizando las operaciones de búsqueda.

Establecer un sistema de copias de seguridad automatizadas: Implementar una cola para realizar copias de seguridad periódicas de la libreta en un archivo de texto, garantizando la seguridad y recuperación de datos en caso de fallos.

Incorporar una herramienta de análisis de contactos frecuentes: Desarrollar una funcionalidad adicional que permita visualizar los contactos más frecuentes, proporcionando a los usuarios información útil sobre sus interacciones y conexiones.

III. PROBLEMA

Gestión Ineficiente de Contactos Personales

En la actualidad, la gestión de contactos personales se ha vuelto cada vez más compleja debido a la diversificación de la información asociada a cada contacto y la creciente cantidad de interacciones sociales. La ausencia de herramientas eficientes para organizar y acceder a estos datos ha generado desafíos significativos en términos de usabilidad y pérdida potencial de información valiosa. En este contexto, surge la necesidad de abordar el problema de manera estructurada mediante el diseño e implementación de una libreta de contactos en C++.

IV. ALCANCE

El alcance de este proyecto abarcará el diseño e implementación de una libreta de contactos en C++ que

cumpla con los requisitos funcionales establecidos. La aplicación permitirá a los usuarios realizar operaciones clave, como agregar, eliminar y buscar contactos, además de acceder a funcionalidades avanzadas, como la visualización de contactos más frecuentes y la realización de copias de seguridad automáticas. La libreta estará basada en algoritmos y estructuras de datos, incorporando un árbol de búsqueda binaria para mantener la lista ordenada alfabéticamente, listas enlazadas para manejar duplicados y un vector para optimizar el acceso a la información. La implementación incluirá una interfaz de usuario intuitiva y se realizarán pruebas exhaustivas para garantizar la calidad y funcionalidad del sistema. Sin embargo, se establecerán límites en cuanto a la complejidad de las operaciones y la cantidad de contactos manejados para asegurar un rendimiento eficiente y una experiencia de usuario fluida. Este proyecto no contemplará integraciones externas ni la implementación de funciones avanzadas no especificadas en los requisitos funcionales.

V. Metodología

La implementación exitosa de la libreta de contactos en C++, centrada en algoritmos y estructuras de datos, requiere una metodología estructurada que aborde de manera efectiva los desafíos inherentes al manejo de información personal. En este contexto, se empleará una metodología que combine principios de desarrollo de software con un enfoque particular en eficiencia y organización.

Requisitos Funcionales

- **Agregar un Nuevo Contacto:** La libreta debe permitir la adición de nuevos contactos, incluyendo información como nombre, número de teléfono, redes sociales, un icono representativo y una dirección.
- **Eliminar un Contacto:** Facilitar la eliminación de un contacto existente a través del nombre.
- **Buscar un Contacto:** Proporcionar la capacidad de buscar un contacto por nombre y mostrar toda su información relevante.
- **Lista Ordenada:** Mostrar la lista completa de contactos ordenados alfabéticamente por nombre.
- **Contador de Contactos:** Informar la cantidad total de contactos presentes en la libreta.
- **Filtrar por Inicial:** Mostrar los contactos cuyos nombres comienzan con una letra específica.
- **Copia de Seguridad:** Implementar una cola para realizar copias de seguridad en un archivo de texto.

Estructuras de Datos Utilizadas

- **Árbol de Búsqueda Binaria (BST):** Mantener la lista de contactos ordenada alfabéticamente por nombre. Para manejar duplicados, se emplearán listas enlazadas.
- **Vector:** Utilizar un vector para mantener un índice rápido de los contactos, mejorando el acceso eficiente a la información.
- **Cola:** Para realizar copias de seguridad periódicas y garantizar la integridad de los datos.

Funcionalidades Adicionales

1. **Contactos Más Frecuentes:** Implementar una herramienta que permita visualizar los contactos más frecuentes.
2. **Mediciones de Calidad:** Realizar mediciones de calidad pertinentes al desarrollo, asegurando un rendimiento óptimo y la robustez del sistema.

VI. Analisis de Resultados

Operaciones Básicas: En el código, se implementa una clase ArbolBTS que representa un árbol binario de búsqueda (BST) para almacenar y gestionar contactos. Se realizan operaciones básicas como agregar (agregarContacto), eliminar (eliminarContacto), y buscar (buscarYmostrar) contactos en el árbol. Además, se muestra la lista de contactos ordenada alfabéticamente, se cuenta la cantidad total de contactos, y se muestran contactos cuyos nombres comienzan con una letra específica.

Persistencia de Datos: Se incorpora funcionalidad para realizar copias de seguridad (realizarCopiaSeguridad) y guardar la libreta de contactos en un archivo de texto (guardarEnArchivo). La implementación permite también obtener información sobre la última copia de seguridad realizada (obtenerUltimaCopiaSeguridad).

Frecuencia y Ordenamiento: Se proporciona la capacidad de mostrar contactos frecuentes (mostrarContactosFrecuentes) basados en la frecuencia de acceso. Además, se generan estructuras de datos como mapas y vectores para ordenar y almacenar contactos de manera eficiente.

VII. Conclusiones

La capacidad de realizar copias de seguridad y guardar información en archivos proporciona una manera robusta de preservar la información de contactos.

La aplicación de estructuras de datos adicionales, como listas enlazadas para gestionar contactos con el mismo nombre y mapas para mantener un orden

específico, mejora la versatilidad y eficiencia del sistema.

La introducción de estructuras de datos complementarias, como listas enlazadas y mapas, no solo enriquece la funcionalidad del sistema, sino que también optimiza el acceso y la manipulación de datos. La gestión eficiente de contactos con el mismo nombre a través de listas enlazadas y el mantenimiento de un orden específico mediante mapas contribuyen a una mayor versatilidad y eficiencia en la manipulación de la información de la libreta

No solo satisface las necesidades actuales de la aplicación de libreta de contactos, sino que también proporciona una base escalable y adaptable. La capacidad de incorporar fácilmente nuevas funcionalidades o ajustar la lógica de almacenamiento destaca la flexibilidad del diseño, permitiendo la expansión del sistema para abordar requisitos futuros o cambios en los requisitos del usuario.