Manul Tecnico

Código fuente de

{Software de Registros de Vacunación Covid-19}

Librerías utilizadas

Introducción

Bienvenido al manual técnico del software de registros de vacunación Covid-19. Este manual proporciona información detallada sobre el funcionamiento y el uso del software, diseñado para facilitar el registro y seguimiento de la vacunación contra el Covid-19.

El software de registros de vacunación Covid-19 es una herramienta poderosa que permite a los profesionales de la salud y al personal encargado de la gestión de la vacunación mantener un registro preciso y actualizado de las personas vacunadas. Además, brinda funcionalidades para agilizar el proceso de programación de citas, seguimiento de dosis administradas, generación de informes y estadísticas, y gestión eficiente de los suministros de vacunas.

Este manual proporciona instrucciones sobre la instalación, configuración y uso del software, así como soluciones a problemas comunes. ¡Esperamos que esta guía sea de utilidad para optimizar el proceso de vacunación y garantizar un registro completo y seguro de las vacunas administradas!

```
#include <iostream>
#include <conio.h>
#include <stdlib.h>
#include <string.h>
#include <fstream>
#include <windows.h>
```

Cada una de las librerías implementadas en el código tiene un propósito específico y han sido incluidas en el código para proporcionar las funcionalidades necesarias en términos de entrada y salida, manipulación de archivos, manipulación de cadenas y operaciones específicas del entorno Windows. Cada una de ellas desempeña un papel importante en la implementación y funcionalidaddel programa.

<iostream>: Esta librería proporciona las funciones básicas de entrada y salida estándar en C++. Permite mostrar mensajes en la consola y leer datos del usuario a través de la entrada estándar.

<conio.h>: Esta librería contiene funciones para realizar operaciones relacionadas con la consola, como leer teclas presionadas y controlar el cursor en la pantalla..

<stdlib.h>: Esta librería proporciona funciones y tipos de datos para realizar operaciones estándar en C++, como el manejo de memoria dinámica, generación de números aleatorios, entre otros.

<string.h>: Esta librería proporciona funciones para manipular cadenas de caracteres en C++. liza para operaciones como copiar, concatenar y comparar cadenas.

<fstream>: Esta librería permite la manipulación de archivos en C++. Proporciona clases y funcionespara leer y escribir en archivos, lo cual es utilizado en el código para leer datos de un archivo de texto.

<windows.h>: Esta librería es específica para sistemas operativos Windows y proporciona funciones para realizar operaciones relacionadas con el entorno de Windows, como el control del cursor, el cambio de colores de texto, entre otros. Su uso en el código sugiere que el programa fue diseñado para ser ejecutado en un entorno Windows

Estructura utilizadas

```
struct Persona {

    string nombre;
    string dpi;
    string departamento;
    string municipio;
    string fechal;
    string fecha2;
    string fecha3;
    string ubicacion;


};

struct Nodo {
    Persona dato;
    Nodo* izquierdo;
    Nodo* derecho;
    Nodo* padre;
};
```

se utilizan dos estructuras: struct Persona y struct Nodo La estructura struct Persona se utiliza para representar a una persona con diferentes atributos, como su nombre, DPI (Documento Personal de Identificación),

departamento, municipio, fechas y ubicación. Cada instancia de esta estructura contendrá la información específica de una persona en particular.
La estructura struct Nodo se utiliza para crear nodos en una estructura de datos de árbol. Cada nodo tiene un dato de tipo Persona que almacena la información de una persona, y también tiene punteros a otros nodos, como izquierdo, derecho y padre. Estos punteros se utilizan para establecer las relaciones entre los nodos y construir la estructura de un árbol.
El uso de estas estructuras en conjunto sugiere que el código está implementando una estructura de árbol donde cada nodo contiene información sobre una persona y está vinculado a otros nodos.

Funciones principales

```
void menuABB();
void MENU_PRINCIPAL();
void encriptarCesar(Nodo* nodo);
void desencriptarCesar(Nodo* nodo);
Nodo* crearNodo(Persona n, Nodo* padre);
void insertar(Nodo*& nodo, Persona dato, int nivel);
void leer();
bool buscarPorDPI(Nodo* nodo, string dpi, Persona& persona_encontrada);
void modificarPersona(Nodo* nodo, string dpi);
void eliminar(Nodo*& arbol, string dpi);
void mostrarArbol(Nodo*& arbol, int auxY);
void mostrarPreorden(Nodo* nodo);
void mostrarPosorden(Nodo* nodo);
void mostrarInorden(Nodo* nodo);
```

Estas funciones desempeñan un papel fundamental en el programa al permitir la interacción con los árboles, realizar operaciones de búsqueda, inserción, eliminación y modificación de nodos, así como mostrar la estructura del árbol y encriptar/desencriptar los datos de las personas almacenadas.

void menuABB();: Esta función muestra el menú para realizar operaciones en un árbol de búsqueda binaria (ABB). Puede incluir opciones como insertar un nuevo nodo, buscar un nodo, eliminar un nodo, mostrar el árbol, etc

void MENU_PRINCIPAL();: Esta función muestra el menú principal del programa, que puede ofrecer opciones para trabajar con árboles de búsqueda binaria o árboles AVL. Es el punto de entrada para seleccionar qué tipo de árbol se desea utilizar.

void encriptarCesar(Nodo* nodo);: Esta función encripta los datos de una persona en el nodo utilizando el cifrado César. El cifrado César es una técnica de cifrado simple que desplaza cada letra en un número fijo de posiciones en el alfabeto.

void desencriptarCesar(Nodo* nodo);: Esta función realiza la operación inversa de la función encriptarCesar(). Desencripta los datos de una persona en el nodo utilizando el cifrado César para restaurar su forma original.

Nodo* crearNodo(Persona n, Nodo* padre);: Esta función crea un nuevo nodo en el árbol con losdatos de una persona específica. Recibe como argumentos la persona y un puntero al nodo padre delnuevo nodo.

void insertar(Nodo*& nodo, Persona dato, int nivel);: Esta función se encarga de insertar un nuevo nodo en el árbol. Recibe como argumentos un puntero al nodo raíz del árbol (o subárbol), los datos de la persona a insertar y el nivel actual del árbol. La inserción se realiza respetando las reglas del árbol de búsqueda binaria o del árbol AVL.

---

void leer();: Esta función se encarga de leer los datos de una persona desde la entrada estándar (porejemplo, el teclado) y crear un nuevo nodo en el árbol con esos datos.

bool buscarPorDPI(Nodo* nodo, string dpi, Persona& persona_encontrada);: Esta función buscaun nodo en el árbol que coincida con el DPI (Documento Personal de Identificación) especificado.
Recibe como argumentos el nodo raíz del árbol (o subárbol), el DPI a buscar y una referencia a una variable de tipo Persona en la que se almacenará la persona encontrada si se encuentra.

void modificarPersona(Nodo* nodo, string dpi);:
Esta función busca un nodo en el árbol quecoincida con el DPI especificado y permite modificar los datos de esa persona.

void eliminar(Nodo*& arbol, string dpi);: Esta función elimina un nodo del árbol que coincida con el DPI especificado. Recibe como argumentos un puntero al nodo raíz del árbol (o subárbol) y el DPI del nodo a eliminar.

void mostrarArbol(Nodo*& arbol, int auxY);: Esta función muestra visualmente la estructura del árbol en forma de árbol gráfico en la consola. Recibe como argumentos un puntero al nodo raíz del árbol (o subárbol) y un valor auxiliar para determinar la posición en el eje Y al mostrar el árbol.

void mostrarPreorden(Nodo* nodo);: Esta función muestra los datos de los nodos del árbol en unrecorrido preorden. El recorrido preorden visita primero el nodo raíz, luego el subárbol izquierdo y finalmente el subárbol derecho.

void mostrarPosorden(Nodo* nodo);: Esta función muestra los datos de los nodos del árbol en unrecorrido posorden. El recorrido posorden visita primero el subárbol izquierdo, luego el subárbol derecho y finalmente el nodo raíz.

void mostrarInorden(Nodo* nodo);: Esta función muestra los datos de los nodos del árbol en unrecorrido inorden. El recorrido inorden visita primero el subárbol

izquierdo, luego el nodo raíz y finalmente el subárbol derecho.

FUNCIONALIDAD DEL ARBOL AVL

Estructura utilizada AVL:

```
struct NodoAVL {
    string nombre;
    string dpi;
    string departamento;
    string municipio;
    string fecha1;
    string fecha2;
    string fecha3;
    string ubicacion;
    int altura;
    NodoAVL* izq;
    NodoAVL* der;
    NodoAVL* padre;
};
```

El código proporcionado implementa un árbol AVL en C++ y contiene funciones para realizar operaciones como inserción, búsqueda, eliminación y

recorrido del árbol. Además, incluye funciones para encriptar y desencriptar los campos "nombre", "departamento", "municipio" y "ubicación" de losnodos del árbol utilizando el cifrado César

Struct NodoAVL: Define la estructura de un nodo en el árbol AVL. Contiene varios campos, incluyendo los datos del nodo (nombre, dpi, departamento, municipio, fecha1, fecha2, fecha3, ubicación), un campo para mantener la altura del nodo, y punteros a los nodos izquierdo, derecho y padre.

Funciones principales del AVL

```
void encriptarCesarAVL(NodoAVL* nodoavl);
void desencriptarCesarAVL(NodoAVL* nodo);
void mostrarArbolavl(NodoAVL *&arbolavl, int auxY2);
int maximo(int a, int b);
int altura(NodoAVL* nodo);
int obtenerBalance(NodoAVL* nodo);
NodoAVL* rotacionDerecha(NodoAVL* y);
NodoAVL* rotacionIzquierda(NodoAVL* x);
NodoAVL* rotacionDerechaDerecha(NodoAVL* nodo);
NodoAVL* rotacionIzquierdaIzquierda(NodoAVL* nodo);
NodoAVL* insertarNodoAVL(NodoAVL* nodo, string dpi, string nombre, NodoAVL* padre = NULL);
bool buscarDatoAVL(NodoAVL* nodo, string dpi);
```

void encriptarCesarAVL(NodoAVL* nodoavl): Esta función realiza una encriptación César en los campos "nombre", "departamento", "municipio" y "ubicación" de un nodo AVL. Aplica el cifrado César para desplazar cada letra del campo tres posiciones hacia la derecha en el alfabeto.

void desencriptarCesarAVL(NodoAVL* nodo): Esta función realiza la desencriptación César en los campos "nombre", "departamento", "municipio" y "ubicación" de un nodo AVL. Aplica el cifrado César inverso para desplazar cada letra del campo tres posiciones hacia la izquierda en el alfabeto.

void mostrarArbolavl(NodoAVL *&arbolavl, int auxY2): Esta función muestra el contenido del árbol AVL en un formato visual. Imprime el DPI y el nombre de cada nodo en orden inorden.

int maximo(int a, int b): Esta función devuelve el máximo entre dos números enteros.

int altura(NodoAVL* nodo): Esta función devuelve la altura de un nodo en el árbol AVL.

int obtenerBalance(NodoAVL* nodo): Esta función devuelve el balance de un nodo en el árbol AVL, que es la diferencia entre la altura de su subárbol izquierdo y la altura de su subárbol derecho.

NodoAVL* rotacionDerecha(NodoAVL* y): Esta función realiza una rotación hacia la derecha en elárbol AVL. Recibe el nodo "y" como parámetro y realiza las operaciones necesarias para mantener el balance del árbol.

NodoAVL* rotacionIzquierda(NodoAVL* x): Esta función realiza una rotación hacia la izquierda en el árbol AVL. Recibe el nodo "x" como parámetro y realiza las operaciones necesarias para mantener el balance del árbol.

NodoAVL* rotacionDerechaDerecha(NodoAVL* nodo): Esta función realiza una doble rotación hacia la derecha en el árbol AVL. Recibe el nodo "nodo" como parámetro y realiza las operaciones necesarias para mantener el balance del árbol.

NodoAVL* rotacionIzquierdaIzquierda(NodoAVL* nodo): Esta función realiza una doble rotación hacia la izquierda en el árbol AVL. Recibe el nodo "nodo" como parámetro y realiza las operaciones necesarias para mantener el balance del árbol

NodoAVL* insertarNodoAVL(NodoAVL* nodo, string dpi, string nombre, NodoAVL* padre = NULL): Esta función inserta un nuevo nodo en el árbol AVL. Recibe el nodo raíz "nodo", el DPI y el nombre del nodo a insertar, y opcionalmente el nodo padre del nodo a insertar. Realiza las operaciones necesarias para mantener el balance del árbol después de la inserción.

bool buscarDatoAVL(NodoAVL* nodo, string dpi): Esta función busca un dato (DPI) en el árbol AVL. Recibe el nodo raíz ".