Proyecto Programación

Cristian Camilo Muñetones

David Santiago Sotelo

David Antonio Pedraza

Johan Santiago Romero Duarte

22/05/24

Este código C++ es una solución integral desarrollada por el IDEAM (Instituto de Hidrología, Meteorología y Estudios Ambientales de Colombia) para abordar la complejidad del procesamiento y análisis de datos meteorológicos. Su función abarca desde la lectura inicial de datos hasta la presentación de resultados de forma precisa y comprensible, proporcionando herramientas para interpretar y comprender los fenómenos meteorológicos.

Comenzando con la lectura de los datos, el programa extrae los datos meteorológicos de un archivo de texto personalizado llamado "datos_metereologicos.txt". Estos datos se organizan y almacenan en una serie de estructuras de datos para contener información específica de cada registro, como el día, la temperatura, la humedad, la velocidad del viento y otros parámetros relevantes.

Una vez ingresados los datos, el programa ofrece una variedad de opciones para análisis y procesamiento. Desde la identificación de intensidades de calor hasta el cálculo de promedios mensuales de temperatura y humedad, el código proporciona herramientas importantes para comprender los patrones climáticos y evaluar los peligros potenciales de posibles incendios o huracanes.

Para garantizar el almacenamiento y el control de acceso, se ha desarrollado un sistema de registro y acceso de usuarios, cuyos datos se almacenan en un archivo binario independiente. Esto significa que solo los usuarios autorizados pueden acceder y utilizar las capacidades del programa, lo cual es fundamental para mantener la integridad y la confidencialidad de los datos confidenciales.

Los resultados del análisis se muestran claramente y son accesibles tanto en la consola como en un archivo de texto llamado "salidaProyecto.txt". Esto se hace utilizando la biblioteca iomanip para formatear la salida de una manera legible y comprensible, lo que facilita la interpretación de la información y la toma de decisiones informadas.

En arquitectura de código, se han utilizado estructuras de mensajes para organizar y procesar la información meteorológica. Estas estructuras incluyen una estructura general para mostrar datos diarios, así como otras estructuras para gestionar fechas e informes anuales. Además, se han desarrollado varias funciones auxiliares para simplificar el proceso, como ver fechas dentro de un rango o identificar años extremos en un pronóstico meteorológico.

En resumen, este código C++ proporciona una solución completa y sofisticada para procesar y analizar datos meteorológicos, integrando aspectos clave como la lectura, el almacenamiento, la visualización, el control del usuario y la visualización de resultados. Con estas características, el programa se posiciona como una herramienta esencial para el IDEAM en su misión de monitorear y prevenir desastres climáticos, permitiendo una comprensión más profunda y una mejor respuesta ante eventos climáticos extremos.

Posteriormente las funciones Este código está escrito en C++ y tiene varias funciones que realizan distintas tareas relacionadas con el manejo de datos meteorológicos. A continuación, explicaré cada función:

1. `registrarUsuario(string user, string password)`: Esta función se encarga de registrar un nuevo usuario en un archivo binario llamado `usuarios.bin`. Recibe como parámetros el nombre de usuario y la contraseña, y los escribe en el archivo.

2. `iniciarSesion(string user, string password)`: Esta función verifica si el nombre de usuario y la contraseña proporcionados coinciden con los almacenados en el archivo `usuarios.bin`. Retorna `true` si las credenciales son correctas, y `false` en caso contrario.

3. `leerArchivo(condicionMeteorologica DatosMet[], int& cantidadDatos)`: Esta función lee los datos meteorológicos desde un archivo de texto llamado `datos_meteorologicos.txt` y los almacena en un arreglo de estructuras `condicionMeteorologica`. Además, actualiza la variable `cantidadDatos` con la cantidad de registros leídos.

4. `escribirArchivo(condicionMeteorologica DatosMet[], int cantidadDatos)`: Esta función escribe los datos meteorológicos almacenados en el arreglo `DatosMet` en el archivo `datos_meteorologicos.txt`.

5. `ordenarPorFecha(condicionMeteorologica DatosMet[], int cantidadDatos)`: Esta función ordena los datos meteorológicos en el arreglo `DatosMet` según la fecha, utilizando el algoritmo de ordenamiento burbuja.

6. `esFechaEnRango(int anio, int mes, int dia, int anioInicio, int mesInicio, int diaInicio, int anioFin, int mesFin, int diaFin)`: Esta función verifica si una fecha específica está dentro de un rango de fechas dado. Retorna `true` si la fecha está dentro del rango, y `false` en caso contrario.

7. `examinarTemMax(condicionMeteorologica DatosMet[], int cantidadDatos, int anioInicio, int mesInicio, int diaInicio, int anioFin, int mesFin, int diaFin, int& anioMax, int& mesMax, int& diaMax)`: Esta función busca la temperatura máxima dentro de un rango de fechas especificado. Retorna la temperatura máxima y actualiza las variables `anioMax`, `mesMax` y `diaMax` con la fecha en la que se registró esa temperatura.

8. `examinarTempMin(condicionMeteorologica DatosMet[], int cantidadDatos, int anioInicio, int mesInicio, int diaInicio, int anioFin, int mesFin, int diaFin, int& anioMin, int& mesMin, int& diaMin)`: Esta función es similar a `examinarTemMax`, pero busca la temperatura mínima dentro de un rango de fechas especificado. Retorna la temperatura mínima y actualiza las variables `anioMin`, `mesMin` y `diaMin` con la fecha en la que se registró esa temperatura.

9. `promedioTemp(int cantidadDatos, condicionMeteorologica DatosMet[], int anio, int mes)`: Esta función calcula el promedio de temperatura para un mes y año específicos, utilizando los datos almacenados en el arreglo `DatosMet`.

10. `promHumedadMes(int cantidadDatos, condicionMeteorologica DatosMet[], const string& criterio, int anio, int mes, datosAnio arregloAnio[])`: Esta función calcula el promedio de humedad para un mes y año específicos, teniendo en cuenta la condición meteorológica. Además, determina la condición meteorológica con el mayor o menor porcentaje de humedad, según el criterio proporcionado (`"mayor"` o `"menor"`).

11. `encontrarMayorAnio(const condicionMeteorologica DatosMet[], int cantidadDatos)`: Esta función encuentra el año más reciente en los datos meteorológicos almacenados en el arreglo `DatosMet`.

12. `encontrarMenorAnio(const condicionMeteorologica DatosMet[], int cantidadDatos)`: Esta función encuentra el año más antiguo en los datos meteorológicos almacenados en el arreglo `DatosMet`.

13. `salidaPromedio(ofstream& salida, int cantidadDatos, condicionMeteorologica DatosMet[], const string& criterio, const string& salidaConsola)`: Esta función muestra los promedios de humedad y las condiciones meteorológicas correspondientes, ya sea en la consola o en un archivo de salida, según el criterio y la opción de salida proporcionados.

14. `probabilidadIncendio(ofstream& salida, int cantidadDatos, condicionMeteorologica DatosMet[], const string& salidaConsola, int dia, int mes, int anio)`: Esta función calcula la probabilidad de incendio para una fecha específica, basándose en la temperatura, la velocidad del viento y la condición meteorológica. Muestra los resultados en la consola o en un archivo de salida, según la opción de salida proporcionada.

15. `probabilidadTormenta(ofstream& salida, int cantidadDatos, condicionMeteorologica DatosMet[], const string& salidaConsola, int dia, int mes, int anio)`: Esta función calcula la probabilidad de tormenta para una fecha específica, basándose en la presión atmosférica, la humedad y la condición meteorológica. Muestra los resultados en la consola o en un archivo de salida, según la opción de salida proporcionada.

16. `main()`: Esta es la función principal del programa. Aquí se maneja el inicio de sesión, el registro de usuarios y el menú principal que permite al usuario seleccionar las diferentes opciones para analizar los datos meteorológicos.

En general, este código está diseñado para leer y procesar datos meteorológicos desde un archivo de texto, realizar cálculos y análisis basados en esos datos, y luego mostrar los resultados al usuario o escribirlos en un archivo de salida. Además, incluye funcionalidades para el manejo de usuarios, como el registro y el inicio de sesión.