“Análisis de Datos Meteorológicos de la Ciudad de Bogotá.

Programa meteorológico”

Bianca Ferrari Parra

Richard Navas

Laura Ladino

Introducción a la programación

Profesor:

Ingeniero: Johan Santiago Romero Duarte

Mayo 2024

Pontificia Universidad Javeriana (PUJ)

Facultad de Ingeniería

1. Introducción (objetivos del programa)

Para el presente proyecto sobre el "Análisis de Datos Meteorológicos de la Ciudad de Bogotá", se emplearon los principios fundamentales de la lógica e ingeniería de sistemas para abordar un problema informático vinculado a los registros meteorológicos de una localidad específica. Dicho esto, se desarrolló un sistema integral para analizar exhaustivamente la información presente en la base de datos meteorológicos de la ciudad de Bogotá. Para ello, se ha diseñado un sistema de autenticación y registro de usuarios, que posteriormente desemboca en un menú de navegación intuitivo, facilitando así el acceso a diversas funcionalidades del programa. Estas incluyen la consulta de temperaturas máximas y mínimas, el cálculo de promedios de temperatura, la visualización de porcentajes de humedad en función de otras condiciones, así como la evaluación de la probabilidad de incendios y tormentas. Además, se ha integrado la capacidad de insertar nuevos datos meteorológicos en la base de datos existente. Finalmente, las consultas mencionadas hechas por el usuario son registradas en un archivo txt, como salida, indicándole aquellas operaciones que realizo al estar dentro del programa.

Para la construcción de este programa, se utilizaron las plataformas de desarrollo de Visual Studio Code que proporciona un entorno de desarrollo, y Replit, una plataforma colaborativa que funciona directamente desde un navegador web, permitiendo a los usuarios escribir, ejecutar y compartir código de forma eficiente. Una vez completado el desarrollo del código en estas plataformas, se cargó el código en la plataforma de GitHub para su posterior visualización y gestión.

2. Descripción de las librerías utilizadas, y las funciones realizadas en el programa

Dentro de este programa se incluyeron tres librerías para su correcto funcionamiento, la iostream, la ftream y la sstream. La primera se considera como la librería estándar de entrada y salida en C++. Esta proporciona las funciones necesarias para interactuar con el usuario a través de la consola. Facilita la entrada de datos desde el teclado y la salida de resultados en la pantalla. Es crucial para la creación de aplicaciones interactivas, lo cual este proyecto requiere como objetivo principal. La segunda proporciona las herramientas necesarias para realizar operaciones de entrada y salida en archivos. Permite abrir, cerrar, leer, crear y escribir en archivos de texto u otros tipos de archivos (como binarios en este caso). Finalmente, la tercera, permite la manipulación de cadenas de texto (en nuestro caso provenientes de un archivo txt) como si fueran flujos de entrada y salida. Esta se usa para convertir valores de diferentes tipos (como enteros, flotantes o cadenas) en una representación de cadena de caracteres y viceversa.

Dicho esto, a continuación, se presenta las características operativas de cada función. Cómo comentario para aclarar, la gran mayoría de estas funciones no se declararon con parámetros, dado que todas son funciones independientes y aunque hagan parte de un mismo programa, podrían funcionar perfectamente solas (hay algunas excepciones que necesitan parámetro, más no cuentan con más de 1). Además de esto, se decidió declarar así, dado que en la gran mayoría de estas se abre el archivo de datos txt para realizar las operaciones.

- **Función crearBin:**
  - Se revisa si el archivo de los usuario.bin está creado
  - Si no, se crea usuario.bin con usuario y contraseña predeterminados “1” “1” para acceso fácil por medio de la función .write, el proceso detallado estará explicado en la función registrar

- **Función registrar:**
  - Se le pide al usuario ingresar un usuario, este se guarda en el string usuario,
  - Se busca tamaño de este con la función .size(), se verifica que sea un entero con static_cast<int> y se guarda en la variable tam (tamaño).
  - En usuario.bin se escribe la variable tam y después de esta, la variable usuario.
  - El mismo proceso se repite con la contraseña.
- **Función iniciar sesión:**
  - Se le pide al usuario que ingrese un usuario.
  - Se itera en el archivo usuario.bin hasta encontrar el usuario dado, esto por medio de la función .read, la cual lee primero el entero representando el tamaño del string que le sigue, y luego utiliza ese tamaño para leer el string
  - Al encontrar una coincidencia, se le pide al usuario que ingrese una contraseña.
  - Se lee el string que le sigue al usuario encontrado en usuario.bin y se revisa si coincide con la contraseña dada y si concuerda, continua a la función main.
- **Función crearTxt:**
  - Se crea el archivo txt de salida con el nombre del usuario
  - Esta se llama en todas las funciones donde se realizan operaciones para que posteriormente en la salida, mediante la descripción salida <<, se escriban las opciones dadas por el usuario.
- **Función menú**
  - Para esta función, que se llama como un carácter (char) y se muestran las opciones las cuales podría realizar el usuario.
  - Luego el usuario inserta como opción aquella operación que desea realizar.
- **Función ordenar txt:**

- Se abre el archivo “datos_meteorologicos”
- Se salta la primera línea que incluye las unidades del txt
- Se lee línea por línea el txt mientras la fecha se almacena en un entero con el formato año mes día ej. (2020-01-01 pasa a ser 20200101) y un contador cuenta la cantidad de líneas del txt
- El programa compara la fecha almacenada con la última fecha leída, si la última fecha medida es menor, se almacena
- Si la fecha leída es mayor que la última fecha escrita, no se guarda en la fecha almacenada
- Al finalizar la leída, se escribe la fecha almacenada en datos_ordenados.txt y pasa a ser la última fecha escrita
- Se repite el programa la cantidad de líneas que el contador haya contado

Después de la función anteriormente descrita se declara un stuct de datos_meteorologicos para la posterior validación de la estructura al agregar nuevas líneas en el archivo de datos meteorológicos txt. Dicho struct contiene con exactitud cada parte de la línea de texto que necesitaría para cumplir en su totalidad las condiciones.

- **Función validar_estructura:**
  - Esta función se declara cómo booleano, y se pide cómo parámetro la línea (nueva línea) que introduce el usuario para verificar si su estructura está escrita correctamente.
  - Para esta función se utiliza la funcionalidad del istringstream con nombre de ss. Su función es dividir toda la cadena de caracteres introducida por el usuario, lee los datos y los identifica separándolos por su tipo y espacios. A

otras palabras si es una cadena de caracteres, este la modifica para que sean datos distintos, ya sea enteros, flotantes o nuevamente una cadena de caracteres.

- Gracias a este y utilizando el struct de los datos meteorológicos se verifica el orden al igual que los condicionales para verificar que todos los datos insertados sean acertados.

- **Función agregar_lineas_archivo:**
  - Esta función ya explícitamente deja agregar las líneas al archivo de texto, pues en esta se abre el archivo de datos meteorológicos en modo de escritura (ios::app) para agregar al final del mismo los nuevos datos, sin perderse los anteriores.
  - Se declara un ciclo para introducir la nueva línea
  - Se declara un condicional y en caso de que el usuario ponga Fin, FIN o fin, no se introduzcan más datos. De lo contrario se pueden seguir poniendo nuevos datos.
  - Se declara un segundo condicional que llama a la función de validar_estructra, en caso de que esta sea falsa, debe volver a introducir de forma correcta la línea.
  - Se declara un tercer condicional para indicar que, si ya existe la fecha, se indica al usuario que vuelva a indicar otra que no exista dentro del archivo txt.
  - Si la anterior condición arroja falso, se escribe la nueva línea (después de haber verificado si la escritura es correcta).
  - Se llama a la función de ordenar el txt, así las nuevas líneas introducidas se ordenan en orden ascendente.

- Finalmente se cierra el archivo y se muestra al usuario que fueron adheridos los nuevos datos.

Las siguientes funciones ya son parte de las operaciones requeridas por el programa con relación a los datos meteorológicos dados en el txt, aquí explicamos cada una:

- **Función mayorTemp:**
  - Se ingresa fecha1 y fecha2
  - Se abre datos_metrorologicos.txt
  - Se lee datos meteorológicos hasta encontrar fecha1
  - Se lee la temperatura línea por línea y se almacena si es mayor a la temperatura almacenada
  - Al encontrar fecha 2, leer su temperatura y compararla, se cierra el ciclo
  - Se escribe el resultado en salida.txt
- **Función menorTemp:**
  - Esta se estructura de la misma manera que la de mayorTemp pero en vez de almacenar si la temperatura es mayor, la almacena si es menor
- **Función promedioTempMes:**
  - Se ingresa un mes y un año
  - Se abre datos_metrorologicos.txt
  - Se lee datos meteorológicos hasta encontrar el mes y año
  - Se lee la temperatura de todas las líneas siguientes y se suma al total, esto se hace hasta que el mes cambie, por cada línea que se lee, un contador aumenta en 1
  - Cuando el mes cambia, se cierra el ciclo
  - Se divide la suma total entre el contador

- Se escribe el resultado en salida.txt

- **Función mayorPromedioHumedad:**
  - Se le pide al usuario que ingrese un año y un mes y se almacena en la variable mesBuscado
  - Se lee datos_meteorologicos.txt línea por línea hasta encontrar el mes y el año ingresados
  - Si cambia el mes, el ciclo se rompe
  - Si la condición meteorológica de la línea es nublada, se suma a humedadNublado y se suma 1 a contadorNublado, de igual manera si la condicion es solelada o lluviosa
  - Se divide el total de cada uno entre su contador respectivo
  - Se halla cuál es el mayor y se escribe en el txt
- **Función menorPromedioHumedad:**
  - Esta funcion tiene la misma estructura que la anterior, pero en vez de verificar el promedio del contador mayor, verifica en contador menor.
- **Función probabilidad_incendio:**
  - Se pide al usuario ingresar una fecha cualquiera, para mirar la probabilidad de incendio.
  - Se abre el archivo txt, y mediante un ciclo se lee la fecha ingresada mediante el istringstream ss por el usuario verificándose si existe en el archivo de datos meteorológicos. Si se verifica, posteriormente se separan mediante el ss los otros datos para calcular la probabilidad de incendio (que son la temperatura, velocidad del viento y la condición meteorológica.

- Posteriormente se declara el archivo de salida para que las siguientes operaciones se escriban en la salida txt.
- Se declaran los cuatro condicionales para verificar si la probabilidad es alta, media, baja o no aplica. Esta última es en caso de que los datos no estén dentro de los rangos de los condicionales dados.
- Finalmente se cierra el archivo de salida txt, donde se escribieron las operaciones realizadas por el usuario.

❖ **Función probabilidad_tormenta:**

- Esta función tiene la misma estructura que la de calcular la probabilidad de incendio, más se diferencian en los datos leídos para formar los condicionales, después de que se verifica que la fecha introducida por el usuario exista en el archivo de datos meteorológicos.
- Los datos en este caso son la humedad, la presión atmosférica y la condición meteorológica.

❖ **Función continuar_programa:**

- Esta función se declara cómo un booleano, para retornar como respuesta un verdadero o falso por medio de un bucle while indicando si el usuario aún desea realizar otra operación y mostrarle nuevamente el menú o finalizar con el programa (con un S/N).
- Dicha respuesta se recibe por medio de un getline (cin, respuesta).
- Si el usuario no marca ni S ni N, se le indica cómo error, dando la posibilidad de volver a introducir S o N.

❖ **Función main:**

- En esta función se llaman todas aquellas anteriores
- Se llama la función de crear el binario, y se da la bienvenida al usuario preguntando si desea iniciar sesión (1) o registrarse dentro del programa (2).
- Se declara un ciclo para verificar que decisión tomo el usuario, en caso de que no haya puesto ni 1 ni 2, se marca el error y se vuelve a pedir que ingrese la opción nuevamente.
- Si toma la decisión 1, se llama la función de iniciar sesión, de lo contrario se llama la función de registro y posteriormente la de iniciar sesión.
- A continuación, se crea el archivo txt de salida y se llama en la función el usuario introducido, para que este salga en el reporte de salida.
- Se declara la decisión como un char y se iguala a la función del menú. Luego se llama al cin.ignore(), el cual limpia el buffer, después de elegir la opción 1 o 2 para revisar el menú, para que continue el programa.
- Se declara un switch case para llamar cada función realizada con el char 1,2,3,4,5,6,7 o 8. En caso contrario se marca la opción no valida.
- Finalmente se declara un condicional de si desea continuar haciendo otra operación. Se llama la función de continuar programa y en caso de que salga falso, el programa se termina.