**Alumnos :**

**Paredes raul 86618**

**agustin xxx**

**Objetivos**

Ejercitar los conceptos básicos de programación C++ vistos en las primeras clases. Familiarizarse con algunas de las herramientas de software que usamos en el curso y en los siguientes trabajos prácticos, implementando un programa que resuelva el problema que presentaremos más abajo.

Introducción

En este trabajo se procesara la medición de sensores de temperatura para su lectura, para ello las mediciones se pueden tomar de un solo sensor o de varios, en ambos casos, lo que se busca obtener la temperatura mínima, la máxima , el promedio y la cantidad de muestras que se toman para la medición.

**Desarrollo**

Para el desarrollo del mismo, lo que se va a tener en cuenta es que la lectura de las mediciones es proveniente de una base de datos, con un formato de la siguiente forma:

sensor1, sensor2, sensor3

1, 2, 3

4, 5, 6

7, 8, 9

Por lo que, la entrada deberá ser capaz de procesar y diferenciar, los espacios, las caracteres de los números y los datos de mediciones en blanco, ya que un espacio en blanco de alguna medición no influye en el promedio.

Las consultas que se van a realizar sobe la base de datos pueden provenir de un archivo o ser ingresados desde del teclado, en ambos casos el formato con se ingresan de la siguiente forma:

Sensor1, 2, 8

Sensor3, 1, 3

Con esta información el diseño del programa deberá calcular los promedios correspondientes.

**Implementación de clases**

Para este diseño, se va a implementar una clase llamada NetSensor, la cual almacena la información de las mediciones de la base de datos en una matriz, en la cual la primera fila contiene los nombres de los sensores y el resto de las filas los datos de las mediciones.

Como no se sabe la cantidad de sensores que se van a almacenar, ni tampoco la cantidad de datos a promediar, lo que se eligió fue un diseño utilizando memoria dinámica de la cantidad de sensores de entrada y la cantidad de datos de los mismos.

Para procesar las consultas las consultas, se implementó la clase Sensor, que almacena la consulta proveniente de la entrada, como se realiza una consulta por cada ingreso, se utilizó memoria estática.

Atributos de la clase NetSensor

Esta clase tiene como atributos privados un vector de string “name” que almacena los nombres de los sensores provenientes de la base de datos.

matriz float “\*\*data” que almecena los datos de las mediciones , en ambos casos, se mantiene el orden de como estaban representados en la base de datos

size\_t “row” que representa la cantidad de flas de la matriz .

size\_t “column” que representa la cantidad de columnas de la matriz.

**Métodos utilizado en las clase**

Para leer la base de datos, se utilizó el método readDataBase, la cual se encarga de almacenar la información proveniente de las mediciones con sus respectivos sensores, asignar los nombres , determinar el tamaño de row y column.

Clase Sensor

Para esta clase, los atributos que tiene como privado son el nombre del sensor de la conulta “name”, size\_t ranStart , ranEnd, que son el inicio y fin del rango de la consulta. Float “average,min,max” correspondientes a el promdio, minimo y maximo obtenidos de la consulta de entrada,

N, es la cantidad de elementos que se evaluan en el rango.

Metodos de la clase

queryTemp , es el encargo de determinar que tipo de consulta se va a realizar, si es una consulta entre todos los sensores o de un solo sensor.

get Average, obtiene el promedio, ya sea de un solo sensor o de todos

operator >> , esta sobrecargado el metodo de entrada para que todo el formato de la consulta sin errores.

Operator<<, esta sobrecargado el operador ya que pueden haber cuatro tipos de respuestas en la salida, BAD QUERY , NO DATA , UNKNOWN ID y la cuarta resuesta es los resultados obtenidos promedio, min ,max y cantidad de elementos

Clase cmdline.h

Esta calse se encarga de parsear los argumentos de entrada para la elección de los flujos de entrada y salida.

Para este caso se configuraron las siguintes opciones.

-d opcion obligatoria que corresponde a la base de datos

-i opcion de flujo de entrada, por defecto es cin

-o opción de flujo salida, si no se seleccona, tiene por defecto cout

Modo de compilación

Se utilizo el siguiente comando para la compilacion

g++ -Wall main.cpp cmdline.h Netsensor.cpp Netsensor.h Sensor.h Sensor.cpp

corridas de prueba

se realizaron las siguientes con los siguientes resultados

Entrada vacia

$ ./Tp0.out -d data.txt -i input.txt

No se obtiene nada a la salida

entrada con argumentos sin argumentos en linea de comandos

$ ./Tp0.out

Option -d is mandatory.

El programa no puede ejecutarse sin la opcion -d

Se prueba el programa con una un archivo de entrada que no existe por ejemplo “nofile.txt”

$./Tp0.out -d data.txt -i nofile.txt

cannot open nofile.txt.

Otra fomar es ejecutar el programa con una mala consulta , por ejemplo en

$cat input.txt

sensor1, r, u

$./Tp0.out -d data.txt -i input.txt

BAD QUERY

no tomara la opcion como valida y no se optine ninuguna salida

==2498== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

Si se quiere hacer una consulta con un rango no valido

$cat input.txt

sensor3, 100,200

$./Tp0.out -d data.txt -i input.txt

NO DATA

Consulta con un sensor que no se encuentra

$./Tp0.out .d data.txt -i input.txt

UNKNOW ID

Modo de prueba de fuga de memoria

Se utilizara valgrind para probar si existe fuga de memoria con la siguiente opcion

$ valgrind –leak-check=full ./TP0.out -d data.txt -i input.txt

==2498== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

Con lo cual no presenta fuga de memoria