20/20 : TB

CANNIZZARO.cpp


```
/**********************
Capteur.h
**********************/
#pragma once
#include <string>

using namespace std;

class Capteur
{
protected:
        static int nombreDeCapteurs;
        string type, noSerie;
public:
        Capteur(string type, string noSerie);
        string getType() { return type; }
        string getNoSerie() { return noSerie; }
        static int getNombreCapteurs() { return nombreDeCapteurs; }
        ~Capteur() { nombreDeCapteurs--; }
};



/**********************
CapteurTemperature.h
**********************/
#pragma once
#include "Capteur.h"
class CapteurTemperature :    public Capteur
{
protected:
        double temperatureMinAutorisee, temperatureMaxAutorisee;
public:
        CapteurTemperature(string numeroSerie, double min, double max);
        double getTemperature();
};



/**********************
CapteurTemperatureExterieure.h
**********************/
#pragma once
#include "CapteurTemperature.h"
class CapteurTemperatureExterieure :    public CapteurTemperature
{
public:
        CapteurTemperatureExterieure(string derniersChiffresNumeroSerie, double min, double max);
        bool verifieComformite();
};



/**********************
Capteur.cpp
**********************/
#include "Capteur.h"

using namespace std;

int Capteur::nombreDeCapteurs = 0;

Capteur::Capteur(string type, string noSerie) {
        this->type = type;
        this->noSerie = noSerie;
        nombreDeCapteurs++;
}

/**********************
CapteurTemperature.cpp
**********************/
#include "CapteurTemperature.h"

using namespace std;

CapteurTemperature::CapteurTemperature(string numeroSerie, double min, double max) : Capteur("TEMP", numeroSeri
e) {
        temperatureMinAutorisee = min;
        temperatureMaxAutorisee = max;
}

double CapteurTemperature::getTemperature() {
        double temp = rand();
        temp /= 0x7FFF;
        temp *= (temperatureMaxAutorisee - temperatureMinAutorisee);
        temp += temperatureMinAutorisee;
```

```
	return temp;
}

/**********************
CapteurTemperatureExterieure.cpp
**********************/
#include "CapteurTemperatureExterieure.h"

using namespace std;

CapteurTemperatureExterieure::CapteurTemperatureExterieure(string derniersChiffresNumeroSerie, double min, doubl
e max)
	: CapteurTemperature("33"+derniersChiffresNumeroSerie, min, max)
{

}

bool CapteurTemperatureExterieure::verifieComformite() {
	if (temperatureMinAutorisee < -50 && temperatureMaxAutorisee > 60) {
		return true;
	}
	else { return false; }
}

/**********************
TestCapteurs_A_Completer.cpp
**********************/
/*********************************************
Programme principal à compléter
Au final, l'exécution du programme doit
produire un résultat similaire à l'exécutable fourni
*********************************************/
#define _CRT_SECURE_NO_WARNINGS

#include <iostream> // bibliothèque de gestion des E/S
#include <conio.h>      // gestion de la console (ici _getch())
#include <windows.h>
#include <vector>

#include "CapteurTemperatureExterieure.h"

using namespace std; // utilisation de l'espace de nommage standard


/*===========================================
Fonction principale
===========================================*/
int main()
{
	// Prise en compte des accents
	// Il faudra choisir la police de caractères Consolas ou Lucida
	SetConsoleOutputCP(1252);

	//==== Déclaration d'éventuelles variables locales ====
	string verif;
	int nbrCapteurDeBase, nbrCapteurExterieur;

	// Affichage du nombre de capteurs
	cout << "ETAPE1 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 0)" << endl << endl;

	// Créer un tableau de 20 capteurs de température (modèles de base)
	// Ces  capteurs de temperature  ont des numéros de séries allant de "111101" à "111120"
	// de temperature mini -20.5 et temperature maxi +80.5
	CapteurTemperature* capteurDeBase[20];

	for (unsigned i = 0; i < 20; i++) {
		capteurDeBase[i] = new CapteurTemperature(to_string(111101 + i), -20.5, 80.5);
	}

	// Affichage du nombre de capteurs
	cout << "ETAPE2 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 20)" << endl << endl;

	// Afficher 8 mesures par capteurs (20 lignes de 8 valeurs)
	cout << "CAPTEURS DE TEMPERATURE (MODELE DE BASE) " << endl;
	cout.precision(4);      // 4 digits de précision
	for (unsigned i = 0; i < 20; i++) {
		cout << "Capteur de numero de serie : " << capteurDeBase[i]->getNoSerie() << endl;
		for (unsigned j = 0; j < 8; j++) {
			cout << capteurDeBase[i]->getTemperature() << "\t";
		}
		cout << "\n" << endl;
	}

	// Supprimer de la mémoire 10 capteurs
	for (unsigned i = 10; i < 20; i++) {
		delete capteurDeBase[i];
```

```
	}
	nbrCapteurDeBase = Capteur::getNombreCapteurs();
	// Affichage du nombre de capteurs
	cout << "ETAPE3 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 10)" << endl << endl;

	// Créer 12 capteurs de température extérieure dont les numéros de séries finissent par "5501" à "5512"
	// Les 11 premiers capteurs ont une temperature mini −85.5 et temperature maxi +300.5
	// le dernier capteur a une température mini de −45.5 et maxi de 60.6
	CapteurTemperatureExterieure* capteurExterieur[12];
	for (unsigned i = 0; i < 11; i++) {
		capteurExterieur[i] = new CapteurTemperatureExterieure(to_string(5501 + i), −85.5, 300.5);
	}
	capteurExterieur[11] = new CapteurTemperatureExterieure(to_string(5512), −45.5, 60.6);

	// Afficher 8 mesures par capteurs (12 lignes de 8 valeurs) et afficher pour chaque capteur s'il est con
forme ou non
	cout << "\n\nCAPTEURS DE TEMPERATURE (MODELE EXTERIEUR) " << endl;

	for (unsigned i = 0; i < 12; i++) {
		if (capteurExterieur[i]−>verifieComformite()) {
			verif = "CONFORME";
		}
		else {
			verif = "NON CONFORME";
		}
		cout << "Capteur de numero de serie : " << capteurExterieur[i]−>getNoSerie() << "\t" << verif << endl;
		for (unsigned j = 0; j < 8; j++) {
			cout << capteurExterieur[i]−>getTemperature() << "\t";
		}
		cout << "\n" << endl;
	}
	nbrCapteurExterieur = Capteur::getNombreCapteurs() − nbrCapteurDeBase;

	// Affichage du nombre de capteurs
	cout << "ETAPE4 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 22)" << endl << endl;

	// Supprimer tous les capteurs
	for (unsigned i = 0; i < nbrCapteurDeBase; i++) {
		delete capteurDeBase[i];
	}
	for (unsigned i = 0; i < nbrCapteurExterieur; i++) {
		delete capteurExterieur[i];
	}

	// Affichage du nombre de capteurs
	cout << "ETAPE5 : NOMBRE DE CAPTEURS = " << Capteur::getNombreCapteurs() << " (attendu 0)" << endl << endl;

	_getch(); // attente d'appui sur une touche
	return 0; // sortie du programme
}
```