Universitat de Girona
Escola Politècnica Superior

Pràctiques EDA

Curs 2022/23

GEINF- GDDV

Exercici de la sessió 1 de laboratori

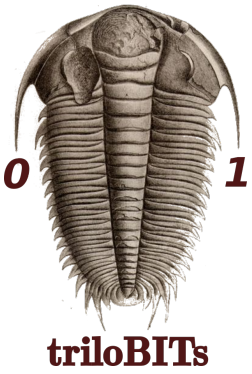
Universitat de Girona
Departament d'Informàtica, Matemàtica Aplicada i Estadística

(PDF generat el 4 d'octubre de 2022)

Corregit mètode string Estacio::codi() i explicació del mètode Estacio::lectures()

1 Presentació

triloBITs [1] és una empresa que ha rebut l'encàrrec del govern per muntar una app de seguiment del canvi climàtic. Com que tenen molts projectes en marxa necessitaran contractar gent per fer la feina i ens encarreguen una petita tasca com a primera prova per veure com ens defensem en C++ i STL.



Utilitzant tot el que s'ha explicat fins ara a classe (i altres coses que pugueu anar ampliant vosaltres) cal que feu un programa que llegeixi informació sobre comarques, estacions meteorològiques automàtiques (EMAs) i dades meteorològiques, ho vagi emmagatzemant en contenidors STL i tot seguit realitzi algunes consultes sobre aquestes dades.

Per fer les proves fareu servir les dades de la Xarxa d'Estacions Meteorològiques Automàtiques tot i que per facilitar-vos la feina hem tractat una mica aquestes dades per tal que us siguin de més bon llegir.

2 Què ha de fer el programa

2.1 Introducció de les dades:

Les dades les obtindreu des de l'entrada estàndard (`cin`) i anirem llegint els elements (magnituds meteorològiques, comarques, estacions meteorològiques automàtiques i lectures de dades meteorològiques) i les anireu afegint a la nostra estructura de dades. En aquesta part us heu de limitar a llegir les dades i a desar-les a l'estructura de dades; no es pot fer cap càlcul.

1. `Variables meteorològiques.` Llegireu:
   - `codi` (int) Codi de la variable.
   - `nom` (string) Descripció de la variable. (temperatura, pressió, humitat...)
   - `unitat` (string) Unitat de la variable (graus, bars,...).
2. `Comarques` Llegireu:
   - `codi` (int) Codi de la comarca.
   - `nom` (string) Nom de la comarca.
3. `Estacions Meteorològiques Automàtiques (EMA)` Llegireu:
   - `codi` (string) Codi de la EMA
   - `altitud` (double) Altitud on està situada
   - `latitud` (double)
   - `longitud` (double)
   - `nom` (string) Nom de la EMA
   - `municipi` (string) Municipi on està la EMA
   - `comarca` (enter) Codi de la comarca
4. `Lectures de dades meteo` Llegireu
   - `estacio` (string) Codi de la EMA
   - `variable` (iny) Codi de la variable meteo llegida
   - `data` (int) Data de la lectura en format `YYYYMMDD` (quatre primers dígits són l'any, els dos següents el mes i els dos últims el dia)
   - `hora` (int) Hora de la lectura en format `HHMMSS` (dos primers dígits l'hora en format 24 hores, els dos següents els minuts i els dos últims els segons)
   - `lectura` (double) el valor llegit

[1] El fundador de l'empresa és un aficionat als fòsils d'artròpodes

2.2 Processat de les dades.

Un cop llegides i emmagatzemades totes les dades fareu el següent:

1. **Mostrar la informació llegida**. Mostrareu, en un format similar a l'exemple que teniu al final del document, el següent:
   - Totes les variables. Una per línia i mostrant el codi, el nom i la seva unitat
   - Totes les comarques. Una per línia, mostrant el codi i el nom.
   - Totes les EMA. Per cada una mostrarem nom, codi entre parèntesi, municipi, nom de la comarca entre parèntesi, altitud, latitud i longitud i quantes lectures existeixen per ella.
2. **Lectures per EMA concretes**: Es demanarà un codi d'EMA i es mostraran les seves dades (en el mateix format que en l'apartat anterior però sense posar el total de lectures) i tot seguit s'aniran mostrant totes les lectures per aquella estació (una lectura per línia). Primer es mostrarà el nom de la variable i tot seguit totes les lectures que hi hagi per aquella variable (mostrar el dia, l'hora, el valor llegit i les unitats corresponents). Al final de tot es mostrarà el número de lectures mostrades (vegeu l'exemple del final del document.)

3 Detalls de la implementació que heu de fer

Heu de respectar fil per randa els noms, tipus i paràmetres dels mètodes públics que consten en aquest enunciat així com usar els contenidors concrets que s'indiquen. Podeu afegir mètodes privats però **està prohibit** modificar la part pública de les classes.

Veureu que us donem unes indicacions sobre quins contenidors[2] cal fer servir però no us donem detalls sobre els atributs que han de tenir les classes: us indiquem només els mètodes que han de tenir i és feina vostra decidir quins atributs han de tenir.

1. Implementeu la classe `Variable` amb els següents mètodes:
   - `Variable()` constructor que deixarà codi a -1.0.
   - `Variable(int codi, const string &nom, const string &unitat);` un constructor amb els tres valors que defineixen una variable.
   - `int codi() const;` que retorna el codi de la variable.
   - `string nom() const;` que retorna el nom de la variable.
   - `string unitat() const;` que retorna la unitat de la variable.
2. Implementeu la classe `Comarca`. Una `comarca` tindrà un codi i un nom.
   - `Comarca()` constructor que deixarà codi a -1.0.
   - `Comarca(int codi, const string &nom);` un constructor amb els dos valors que defineixen una comarca.
   - `int codi() const;` que retorna el codi de la comarca.
   - `string nom() const;` que retorna el nom de la comarca.
3. Implementeu la classe `DataTemps` per emmagatzemar dies i hores concretes.
   - `DataTemps()` Constructor sense paràmetres (deixa la data a les 00:00:00 de l'1 de gener de 1970)
   - `DataTemps(int data, int temps);` Constructor que rep una data expressada en 8 dígits (YYYYMMDD) i un temps expressat en 6 dígits (HHMMSS). Per exemple les 15:32:18 del 7 de setembre de 2021 estaria representada pels enters `20210907` i `153218`.
   - Definiu i implementeu alguna cosa que permeti mostrar el contingut d'una data en el format `YYYY/MM/DD HH:MM:SS`. Us donem dues possibilitats i n'heu d'escollir una de les dues (la que vulgueu):

[2] l'elecció dels contenidors per aquesta activitat s'ha fet per tal que treballeu `vector` i `list` i no per motius d'eficiència.

(a) `friend ostream &operator<<(ostream &os, const DataTemps &d);` Que mostra per l'stream `os` el dia i hora en format `YYYY/MM/DD HH:MM:SS`.

(b) `string diaHora() const;` que retorna la data i hora en una string amb el format `YYYY/MM/DD HH:MM:SS`. Per aquesta conversió podeu fer servir un `stringstream` combinat amb `setw` (indicat quants caràcters ocuparà el següent element que mostrem) i `setfill` (indica quin caràcter farem servir per omplir els espais que puguin fer falta fins arribar als indicats pel `setw` previ). Per exemple:

```
#include <iostream>
#include <sstream>
#include <iomanip>
using namespace std;

int main(){
    int i=121, j=78;
    stringstream s;
    s<<setfill('0')<<setw(5)<<i<<"*"<<setw(3)<<j<<" = "<<i*j;
    cout <<s.str()<<endl;
}
```

mostrarà per la sortida estàndard el següent:

```
00121*078 = 9438
```

`00121` perquè la variable `i` diguem que ocupi 5 caràcters, `078` perquè la variable `j` reservem 3 caràcters i `9438` perquè per `i*j` no reservem res i ho mostra tal qual

4. Implementeu la classe `Lectura` amb el següent:

   - `Lectura(const string &estacio, int variable, int data, int hora, double valor);` constructor amb les dades necessàries per crear una Lectura. El paràmetre `data` expresa un dia en el format `YYYYMMDD` i el paràmetre `hora` expresa una hora en el format `HHMMSS`
   - `string estacio() const;` retorna el codi de l'estació que ha fet la lectura
   - `int variable() const;` retorna el codi de la variable meteorològica
   - `DataTemps data() const;` retorna el dia i hora de la lectura. Haureu de definir la classe `DataTemps`.
   - `double valor() const;`

5. Implementeu la classe `Estacio` que contidrà les seves dades més les lectures que hi puguin haver. Aquestes lectures s'emmagatzemaran en un vector de llistes d'objectes de la classe `Lectura` de manera que a la posició `x` del vector hi tindrem la llista de lectures corresponents a la variable meteorològica amb codi `x`.

   - `Estacio()` constructor que deixarà codi a "*".
   - `Estacio(const string &codi, double altitud, double latitud, double longitud, const string &nom, const string &municipi, int comarca);` un constructor amb els set valors que defineixen una estació.
   - ~~`int`~~ `string codi() const;` que retorna el codi de l'estació.
   - `string nom() const;` que retorna el nom de l'estació.
   - `double altitud() const;` que retorna l'altitud de l'estació.
   - `double latitud() const;` que retorna la latitud.
   - `double longitud() const;` que retorna la longitud
   - `string municipi() const;` que retorna el nom del municipi on està l'estació.
   - `int comarca() const;` que retorna el codi de la comarca on està l'estació
   - `void afegeixLectura(const Lectura &l);` que afegeix `l` a la llista de `Lectura` que ocupa la posició del vector determinada pel codi de la variable meteorològica de `l`. **COMPTE:** Abans d'afegir-la cal comprovar la mida del vector i actuar en conseqüència.
   - `const vector<list<Lectura> > &lectures() const;` que retorna una ~~llista~~ un vector de llistes amb totes les lectures de l'estació.

6. Implementar la classe `Dades` on hi guardarem les variables meteorològiques (en un vector d'STL), les comarques (en una vector d'STL) i les estacions (en una llista d'STL). Tindrà els següents mètodes:

   - `void afegirVariable(const Variable &v);` Afegeix la variable meteorològica `v` al `vector` de variables. La variable `v` es posarà a la posició del vector coincidint amb el codi de `v`. **COMPTE:** Abans d'afegir-la cal comprovar la mida del vector i actuar en conseqüència.
   - `void afegirComarca(const Comarca &c);` Afegeix la comarca `c` al `vector` de comarques. La comarca `c` es posarà a la posició del vector coincidint amb el codi de `c`. **COMPTE:** Abans d'afegir-la cal comprovar la mida del vector i actuar en conseqüència.
   - `void afegirEstacio(const Estacio &e);` Afegeix l'estació `e` a la llista d'estacions **assegurant que la llista quedi ordenada per codi d'estació**.
   - `void afegirLectura(const Lectura &l);` Afegeix a l'estació identificada per `l.estacio()`. Per fer-ho cal utilitzar el mètode `afegeixLectura` de la classe `Estacio`.
   - `list <Variable> variables() const;` Retorna en un contenidor `list` totes les variables meteorològiques existents.**COMPTE:** la llista només contindrà els elements del vector de variables que tenen dades.
   - `list <Comarca> comarques() const;` Retorna en un contenidor `list` totes les comarques existents .**COMPTE:** la llista només contindrà els elements del vector de comarques que tenen dades.
   - `list <string> codisEstacions() const;` Retorna una llista amb els codis de totes les estacions existents. **COMPTE:** retornar una llista de codis; no una llista d'estacions.
   - `bool estacio(const string &codi, Estacio &e) const;` Si hi ha una estació amb el codi `codi` retorna `true` i al paràmetre `e` hi ha les dades de l'estació. Si no hi ha cap estació amb `codi` retorna `false` i el contingut d'`e` és indeterminat.
   - `bool comarca(int codi, Comarca &c) const;` Si hi ha una comarca amb el codi `codi` retorna `true` i al paràmetre `c` hi ha les dades de la comarca. Si no hi ha cap comarca amb `codi` retorna `false` i el contingut de `c` és indeterminat.
   - `bool variable(int codi, Variable &v) const;` Si hi ha una variable amb el codi `codi` retorna `true` i al paràmetre `v` hi ha les dades de la variable. Si no hi ha cap variable amb `codi` retorna `false` i el contingut de `v` és indeterminat.

7. Escriure un programa principal que s'encarregui de l'entrada de dades i de les consultes que es demanen.

   - Tot (variables, comarques, estacions i lectures) s'anirà emmagatzemant en un objecte de la classe `Dades` que anireu omplint invocant els mètodes pertinents a mesura que aneu llegint les dades.
   - **Introducció de les dades:** Les dades a llegir les teniu indicades a l'apartat 2.1 on hi consta l'ordre en que cal llegir-les i el tipus de cada dada. Llegirem
     (a) **Variables meteorològiques**: anirem llegint fins que ens entri un -1 com a codi. Quan llegim un -1, passarem a llegir comarques.
     (b) **Comarques**: anirem llegint fins que ens entri un -1 com a codi. Quan llegim un -1, passarem a llegir estacions meteorològiques.
     (c) **Estacions meteorològiques**: anirem llegint fins que ens entri un * com a codi. Quan llegim un *, passarem a llegir les lectures.
     (d) **Lectures**: anirem llegint fins que ens arribi un * com a codi d'estació. Quan llegim un *, passarem a l'apartat de consultes.
   - **Processat de les dades:**
     (a) Mostrar la informació llegida (seguint el que s'indica a l'apartat 2.2)
     (b) Lectures per EMA concretes: es demanarà un codi d'una estació i, si existeix, es mostraran les dades tal i com s'indica a l'apartat 2.2). Si no existeix es mostrarà un missatge indicant-ho. Anirem repetint el procés fins que el codi llegit sigui un *.
   - **IMPORTANT:** Organitzeu el codi en accions i funcions. **No** feu un main monolític...

4 Joc de proves

Cal que acompanyeu el vostre codi del joc de proves que heu fet servir per anar-ho provant. Recordeu que el joc de proves està format per un conjunt de fitxers tipus text que serveixen per substituir l'entrada estàndard i que ens permeten poder anar provant sistemàticament els nostres programes.

Per provar el programa redireccionareu l'entrada estàndard a un d'aquests fitxers i és molt important que el que poseu en el fitxer correspongui exactament amb el que introduiríeu des de teclat per provar-ho.

Al Moodle hi trobareu els dos fitxers que teniu una mica explicats als annexos finals i a vosaltres us toca fer altres fitxers per assegurar que el vostre programa té en compte les diferents possibilitats.

5 Material a lliurar

Hi haurà dos lliuraments que cal fer en els terminis que consten al Moodle.

5.1 Provisional

Caldrà fer-lo abans de la propera sessió de laboratori i servirà per fer un seguiment del que aneu fent.

1. A Moodle: un .zip amb el codi font (només fitxers .cpp i .h; res de fitxers objecte ni executables)
2. Els fitxers que tingueu del joc de proves i un `llegeix.me` on expliqueu cada fitxer del joc de proves quin objectiu té (i qualsevol altre comentari que vulgueu fer sobre el vostre codi). El fitxer .zip no tindrà cap directori a dins.

5.2 Definitiu

1. A Moodle: la versió definitiva del que calia lliurar a la versió provisional.
2. A `bas.udg.edu`: A la carpeta `$HOME/eda/s01` hi deixareu el codi font, l'executable, els fitxers del joc de proves i el `llegeix.me`

IMPORTANT: Cal que seguiu les instruccions sobre com lliurar els exercicis de laboratori i pràctiques que teniu al Moodle. Assegureu-vos que ho feu com us demanem, sobre tot les pre i postcondicions tant a les classes com al main.

Annex: Fitxer `testCurt.txt` pel joc de proves

El contingut del fitxer `testCurt.txt` és el següent:

```
1
Pressió atmosfèrica màxima
hPa
2
Pressió atmosfèrica mínima
hPa
32
Temperatura
ºC
34
Pressió atmosfèrica
hPa
35
Precipitació
mm
-1
26
Pallars Sobirà
25
Pallars Jussà
2
Alt Empordà
10
Baix Empordà
3
Alt Penedès
-1
Z1
2266
42.6469
0.98486
Bonaigua (2.266 m)
Alt Àneu
26
UE
4
42.0231
3.15719
Torroella de Montgrí
Torroella de Montgrí
10
Y5
152
42.2279
2.86295
Navata
Navata
2
*
Y5
32
20220301
000000
5.5
Y5
```

```
32
20220301
080000
9.9
Y5
35
20220301
160000
0
Z1
32
20220301
080000
1.6
Z1
32
20220301
160000
4.6
*
Z1
Y5
*
```

on hi ha 5 variables meteorològiques, 5 comarques, 3 estacions meteorològiques i 5 lectures de dades. Finalment hi ha els codis de dues estacions per consultar les seves lectures. Fixeu-vos amb els -1 i els * per indicar els canvis de blocs de dades.

si el programa es digués `s01` i executéssim des del *shell*: `./s01 < testPetit.txt` la sortida podria ser una cosa del tipus:

```
************************
* Variables conegudes *
************************
    1     Pressió atmosfèrica màxima (hPa)
    2     Pressió atmosfèrica mínima (hPa)
    32     Temperatura (°C)
    34     Pressió atmosfèrica (hPa)
    35     Precipitació (mm)
************************
* Comarques conegudes *
************************
    2: Alt Empordà
    3: Alt Penedès
    10: Baix Empordà
    25: Pallars Jussà
    26: Pallars Sobirà
************************
* Estacions conegudes *
************************
EMA Torroella de Montgrí (UE) municipi de Torroella de Montgrí (Baix Empordà)
    a 4 metres d'altitud i amb coordenades (42.0231, 3.15719)
    0 lectures disponibles

EMA Navata (Y5) municipi de Navata (Alt Empordà)
    a 152 metres d'altitud i amb coordenades (42.2279, 2.86295)
    3 lectures disponibles

EMA Bonaigua (2.266 m) (Z1) municipi de Alt Àneu (Pallars Sobirà)
    a 2266 metres d'altitud i amb coordenades (42.6469, 0.98486)
    2 lectures disponibles

********************
* Lecures per EMAs *
********************
Codi estació (* per acabar)? EMA Bonaigua (2.266 m) (Z1) municipi de Alt Àneu (Pallars Sobirà)
    a 2266 metres d'altitud i amb coordenades (42.6469, 0.98486)
    Temperatura
        2022/03/01 08:00:00: 1.6 °C
        2022/03/01 16:00:00: 4.6 °C
    2 lectures mostrades

Codi estació (* per acabar)? EMA Navata (Y5) municipi de Navata (Alt Empordà)
    a 152 metres d'altitud i amb coordenades (42.2279, 2.86295)
    Temperatura
        2022/03/01 00:00:00: 5.5 °C
        2022/03/01 08:00:00: 9.9 °C
    Precipitació
        2022/03/01 16:00:00: 0 mm
    3 lectures mostrades

Codi estació (* per acabar)?
```

Si l'execució que feu és `./s01 < testPetit.txt >testPetit.out` es generarà un fitxer `testPetit.out` amb tot el que es mostraria per pantalla. Pot ser útil per *debuggar*.

Annex: Fitxer `testLlarg.txt` pel joc de proves

Aquest fitxer no us el reproduim aquí per la seva extensió. Conté totes les variables (26), totes les comarques (41), totes les EMA (233) i unes quantes lectures de municipis de quatre comarques (855).