Universitat de Girona
Escola Politècnica Superior

Pràctica 1: Estructures de Dades Juguem amb Ce Essa Ves? Oh, my God!!!

Estructures de Dades i Algorítmica

Curs 2024/25

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

David Fíguls
david.figuls@udg.edu

(PDF creat el 8 d'octubre de 2024)

Índex

1 Presentació

Els fitxers CSV (***C**omma-**S**eparated **V**alues*) són fitxers de text que emmagatzemen informació tabular en què cada línia (registre) representa una fila de la taula i, dins de cada línia, les columnes (camps) estan separades per un caràcter predefinit.

És una forma d'emmagatzemar taules molt senzilla i que, amb diverses variants, es fa servir des dels anys 70 del segle passat. El separador inicial pels camps era la coma (d'aquí ve del nom 😎), tot i que es pot fer servir qualsevol caràcter (trobem força fitxers CSV on el separador és, per exemple, el punt i coma).

Una altra cosa a tenir en compte és que, quan el caràcter separador pot formar part de les dades, podem posar el valor de la columna entre cometes dobles per evitar interpretar el caràcter com a separador. Per exemple, si volem representar que, amb el codi 545, tenim en Joan Puig que viu al Carrer del Puig, 50 de Girona, posaríem:

```
545,Joan,Puig,"Carrer del Puig, 50",Girona
```

en comptes de:

```
545,Joan,Puig,Carrer del Puig, 50,Girona
```

On el 50 es llegiria com un camp nou perquè la coma de després de `Puig` s'interpretaria com un marcador de separació de camp. Quan es fan servir `"`, aquestes no es consideren part de la dada. Si féssim servir el `;` com a separador, llavors no caldria:

```
545;Joan;Puig;Carrer del Puig, 50;Girona
```

Finalment, comentar que un fitxer CSV conté només dades: no hi ha cap informació sobre la codificació de caràcters usada (UTF8, ISO-8859, Windows-1252, etc.), ni sobre els tipus de les dades representades (text, números, dates, etc.), ni sobre el separador... Tot això ho hem de saber nosaltres en el moment de llegir el fitxer.

1.1 GOD

El projecte GOD (Girona Open Data) té com a objectiu posar a disposició de les persones i de les empreses, sense restriccions de drets d'autor, copyright, patents o altres mecanismes de control, dades de caràcter públic existents en els sistemes informàtics de l'Ajuntament de Girona.

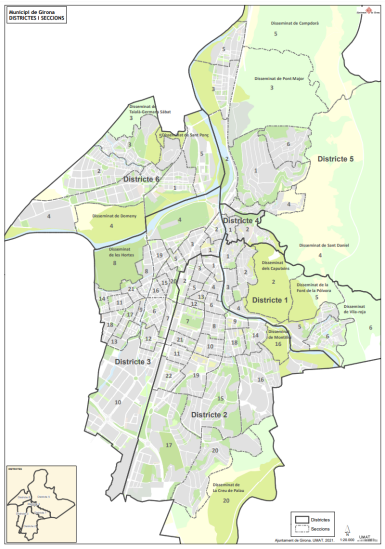
El GOD ofereix a la seva web[1] el detall del padró municipal dels darrers anys en diferents formats, entre ells CSV. El fitxer CSV del padró de l'ajuntament de Girona conté dades anònimes de cadascun dels habitants empadronats a Girona, amb informació del sexe, any naixement, estudis, nacionalitat i país/província/ciutat de naixement i de procedència.

El conjunt de dades està format per una taula amb unes 100 K files, una per habitant. La primera fila són les etiquetes de les columnes i la resta són els valors. El separador és el punt i coma i TOTS els camps estan tancats entre cometes.

[1] `https://www.girona.cat/opendata`

2 Feina a fer

La vostra feina consistirà en llegir les dades d'un fitxer CSV amb dades del padró de diferents anys, i classificar-les per any, districte i secció. Tota aquesta informació haurem d'emmagatzemar-la en diversos contenidors STL per tal de poder fer consultes diverses sobre les dades o executar algun procés sobre elles.



Les microdades estaran emmagatzemades en una classe anomenada `Padro`, que tindrà una part pública formada pels mètodes que s'especificaran tot seguit. Serà feina vostra decidir si faran falta mètodes privats, altres classes i implementar-ho tot.

IMPORTANT: No podeu modificar la part pública de la classe `Padro`[2] i cal que respecteu fil per randa els noms i els tipus dels mètodes i paràmetres. Tot el que ha de fer el programa (`main`) es pot fer fent servir aquests mètodes públics.

Alguns dels paràmetres dels mètodes que heu d'implementar veureu que són d'uns tipus concrets que no estan definits. Sereu vosaltres els qui els haureu d'implementar.

2.1 Operacions públiques de `Padro`

IMPORTANT: Les pre- i postcondicions dels mètodes les haureu d'escriure vosaltres al `Padro.h` tenint en compte com implementeu la vostra classe.

Les operacions públiques que haurà de proporcionar la classe `Padro` són les següents:

(a) `int llegirDades(const string& path);`

Llegeix les files del fitxer CSV de la ruta indicada i va emmagatzemant els valors.

Tingueu present que la primera línia del fitxer conté els noms de les columnes i, per tant, no forma part de les dades. Les dades que poguessin estar carregades prèviament s'eliminen.

El mètode retorna el número de files totals llegides (que podrà ser 0 per indicar que no s'ha pogut llegir cap fila de dades pel motiu que sigui).

(b) `bool existeixAny(int any) const;`

Retorna un `bool` indicant si tenim dades d'aquest `any`.

(c) `map<int, long> obtenirNumHabitantsPerAny() const;`

Retorna un `map` indicant, per cada any, el nombre d'habitants.

(d) `vector<long> obtenirNumHabitantsPerDistricte(int any) const;`

Retorna un `vector` amb els habitants de cada districte l'any indicat.

(e) `map<int, long> obtenirNumHabitantsPerSeccio(int any, int districte) const;`

Retorna un `map<int, long>` amb els habitants de cada secció de l'any i districte indicats.

(f) `ResumEstudis resumEstudis() const;`

Retorna un `ResumEstudis` que contindrà, per cada any, tots els nivells d'estudis (identificats pel seu nom).

(g) `map<int,int> nombreEstudisDistricte(int disctricte) const;`

Retorna un `map<int,int>` que contindrà, per cada any, el nombre d'estudis diferents dels habitants del districte indicat.

(h) `ResumNivellEstudis resumNivellEstudis() const;`

Retorna un `ResumNivellEstudis` que contindrà, per cada any, el nivell d'estudis promig de cada disctricte, del districte 1 al 6. Calculem el promig d'estudis sumant el codi de l'estudi de tots els habitant del districte i dividint-lo pel nombre d'habitants del districte. Marcarem amb un + el districte més estudiós de cada any, i amb un - el menys.

[2] Si creieu haver detectat un error en la part pública, poseu-vos en contacte amb el vostre professor de laboratori i comenteu-l'hi.

(i) `ResumNacionalitats resumNacionalitats() const;`

Retorna un `ResumNacionalitats` que contindrà, per cada any, totes les nacionalitats (identificades pel seu nom) i el nombre d'habitants de cada nacionalitat. Les nacionalitats s'ordenaran pel nombre d'habitants en ordre descendent.

(j) `map<int,string> movimentsComunitat(int codiNacionalitat) const;`

Retorna un `Resum` que contindrà, per cada any, el nom del disctricte que concentra més habitats d'una determinada nacionalitat.

(k) `ResumEdat resumEdat() const;`

Retorna un `ResumEdat` que contindrà, per cada any, tots els districtes (identificats pel seu nom) i l'edat promig dels seus habitants. Els districtes s'ordenaran pel promig, en ordre ascendent.

(l) `map<int, string> movimentVells() const;`

Retorna un `map<int,string>` que contindrà, per cada any, el nom del districte més envellit.

(m) `pair<string,long> mesJoves(int anyInicial, int anyFinal) const;`

Retorna un `pair<string,long>` amb el nom del districte i l'increment d'habitants "joves" (entre 20 i 30 anys, ambdós inclosos) del districte on el nombre d'habitants "joves" ha augmentat més entre els anys inicial i final.

(n) `list<string> estudisEdat(int any, int districte, int edat, int codiNacionalitat) const;`

Retorna un `list<string>` amb els diferents estudis (nom) dels habitants de l'any, districte, edat i codiNacionalitat indicats.

2.2 Codi que us donem fet

Per tal de facilitar-vos la lectura de dades, us donem feta una funció que rep una línia d'un fitxer CSV i retorna, en un `vector<string>`, el valor de cadascuna de les columnes que la formen. Aquest codi es troba als fitxers `eines.h` i `eines.cpp` que trobareu a Moodle. A la pàgina 6 teniu el codi font i un exemple de com fer-ho servir.

2.3 Detalls del disseny i implementació

Per implementar tot el que es demana haureu de tenir en compte les coses següents:

1. Dissenyeu i implementeu totes les classes necessàries separant bé les definicions (fitxers `.h`) de les implementacions (fitxers `.cpp`).
2. No oblideu les pre- i postcondicions. Per les classes poseu-les als fitxers `.h`, i pel `main.cpp` on definiu les accions i funcions.
3. Ha d'existir una classe `Padro` que agrupi totes les dades i que defineixi i implementi les operacions abans descrites.
4. Heu de respectar fil per randa els noms i els paràmetres de les operacions que consten en aquest enunciat.
5. Heu de fer servir contenidors de l'`STL` (i algorismes genèrics si us resolen alguna cosa). Això implica no utilitzar arrays de C ni estructures de dades basades en punters.
6. El programa principal es dirà `main.cpp` i tindrà, entre altres coses, un objecte de la classe `Padro`.
7. L'entrada seguirà les pautes definides a la subsecció 2.3.2 i es farà tota des del programa principal amb l'excepció comentada anteriorment pel mètode de `Padro` que llegeix dels fitxers CSV.
8. El format de la sortida és lliure, tot i que és recomanable que sigui exactament com la sortida dels fitxers de proves que us proporcionem. Així podreu utilitzar qualsevol aplicació tipus `diff` per localitzar les diferències entre la vostra sortida i la sortida esperada que us proporcionem. Podeu separar la sortida pròpiament dita cap al `cout`, i els missatges d'informació per a l'usuari (per exemple, les opcions del menú, què s'espera que proporcioni a cada pas, etc.) cap al `cerr`.
9. Haureu de proporcionar un joc de proves complet, format per fitxers anomenats `t1.txt`, `t2.txt`, `t3.txt`, ..., `tn.txt`. **No provarem el programa introduint les dades des del teclat. Ho provarem només des dels jocs de proves, vostres i nostres**.

2.3.1 Estructures de dades

Heu de decidir quina estructura de dades utilitzar en base als següents punts:

- Les consultes de la (b) a la (e) (inclosa) han de tenir un cost assimptòtic constant $O(1)$. Recordeu però, un cost $O(N)$ on N sigui molt petita, es pot considerar $O(1)$.
- Les consultes sobre estudis i nacionalitats, de la (f) a la (j) (inclosa) han de ser el més eficient possibles.
- Les consultes sobre edats, de la (k) a la (n) (inclosa) no tenen cap restricció de cost.
- No podem definir atributs per guardar dades precalculades, tal com sumes d'imports, totals de línies... a no ser que realment ens suposi un estalvi de temps significatiu.
- Un cop satisfetes aquestes restriccions, cal triar les estructures de dades que minimitzin l'espai de memòria.

2.3.2 Format per l'entrada de dades

El programa presentarà un menú per escollir quina operació es vol fer llegint un número entre 0 i 14 (0 per acabar). En funció de l'operació escollida es llegiran les dades necessàries, s'executarà l'operació interactuant amb `Padro` via els seus mètodes, i es tornarà al menú per fer alguna altra operació o acabar. Les opcions són:

0 `Acabar`. No es llegeix cap dada. S'acaba el programa.

01 `Llegir dades`. Es llegeix el fitxer CSV, s'emmagatzemen i es mostra quantes n'hi ha.

02-14 ... cadascun dels mètodes descrits anteriorment (b) a (n) en el mateix ordre.

Implementeu l'entrada tal i com es demana perquè, si no ho feu així, el vostre programa no funcionarà amb els nostres jocs de proves, que s'expliquen a la pàgina 8.

3 Instruccions de lliurament

Caldrà penjar a Moodle un fitxer comprimit **zip** o bé **tgz** amb un nom que segueixi el format

`CodiUsuari_CognomsNom_p1`

d'acord amb el document *"Instruccions per al lliurament de les activitats de laboratori"*. Aquest fitxer comprimit ha de contenir:

1. El codi font.
2. Els fitxers del joc de proves.
3. Un *script* de *bash* que permeti executar el vostre programa sobre el joc de proves (executant una prova darrere l'altra, sense preguntar res)[3].
4. Un fitxer `llegeix.me` amb aclariments que puguin facilitar la correcció. Sobretot, si hi ha alguna cosa que no funciona bé, expliqueu-ho en el `llegeix.me` en lloc de posar comentaris al lliurament de Moodle (que a vegades passen desapercebuts). Encara que funcioni tot bé, digueu-ho en aquest fitxer `llegeix.me`.
5. Una carpeta `html` amb la documentació generada amb doxygen[4].
6. Un document anomenat `informe.pdf` amb una explicació i justificació detallada de les estructures de dades utilitzades i dels fitxers del joc de proves, explicant què heu volgut provar amb cada un dels fitxers. Aquest informe haurà de tenir una portada amb el títol, el curs i el vostre nom. Es valorarà que estigui ben escrit, respectant la llengua que escolliu per fer l'informe.

A més del lliurament a Moodle, al directori ∼/eda/p1 del vostre compte de `bas.udg.edu` hi heu de posar, descomprimit, el mateix que hagueu penjat a Moodle, exceptuant la documentació (punts 5 i 6), que ocupa massa i ja la tindrem a Moodle. A `bas.udg.edu` també hi haurem de trobar l'executable.

Tots els fitxers de tipus text han d'estar en format UNIX.

Consulteu el document *"Instruccions per al lliurament de les activitats de laboratori"* per més detalls.

La data límit de lliurament està especificada a l'activitat corresponent de Moodle.

[3] En una propera sessió de laboratori us explicarem com fer un *script* de *bash*.

[4] En una propera sessió de laboratori us explicarem com utilitzar doxygen, `http://www.doxygen.nl/`

Annex: Codi per descompondre una línia d'un fitxer CSV

El codi que us donem per poder descompondre una línia d'un fitxer CSV és: els fitxers `eines.h` i `eines.cpp`, i un `prova_tokens_csv.cpp` que mostra com fer-ho servir.

El fitxer `eines.h`

```
#ifndef LECTURA_EINES_H
#define LECTURA_EINES_H

#include <string>
#include <vector>

using namespace std;

// pre: --
// post: excepcions: si cometes es cert, el primer caracter es " i no hi ha unes segones " que tanquin,
//       es genera una excepcio
// retorna: cadena entre posicio primer i seguent separador o final de linia. Si cometes es cert, quan el
//          token comenca per " busca la " que ho tanca i les elimina del token (si despres de les " inicials
//          es troba "", es a dir, dues " seguides, s'ignoren).
string token(const string& s, char separador, bool cometes, long& primer, long& ultim);

// pre: --
// post: --
// retorna: vector<string> amb tots els components d'una linia CSV basica (nomes tractant separadors). Un
//          component esta format per tots els caracters entre dos separadors, excepte el primer i l'ultim.
//          El primer component esta format per tots els caracters abans del primer separador. L'ultim
//          component esta format per tots els caracters despres de l'ultim separador.
vector<string> tokens(const string& s, char separador = ',', bool cometes = false);


#endif //LECTURA_EINES_H
```

El fitxer `eines.cpp`

```
#include "eines.h"

vector<string> tokens(const string& s, char separador, bool cometes) {
   vector<string> resultat;
   if (!s.empty()) {
      long primer = 0, ultim = 0;
      while (ultim != string::npos) {
         resultat.push_back(token(s, separador, cometes, primer, ultim));
      }
   }
   return resultat;
}

string token(const string& s, char separador, bool cometes, long& primer, long& ultim) {
   string t;
   if (!cometes || s[primer] != '"') { // No volem tenir en compte les " o no comenca per "
      while (s[primer] == '␣' && primer < s.length()) { // ens mengem els espais inicials si no hi ha cometes
         primer++;
      }
      ultim = s.find(separador, primer);
      if (ultim == string::npos) {
         t = s.substr(primer);
      }
      else {
         t = s.substr(primer, ultim - primer);
         primer = ultim + 1; // ens mengem la ,
      }
   } else { // comenca per " i les volem tenir en compte com a delimitadors
      primer++;
      ultim = s.find('"', primer);
      while (s.length() > ultim + 1 && s[ultim + 1] == '"') {
         ultim = s.find('"', ultim + 2); // saltem "" (cometes dobles seguides)
      }
      if (ultim == string::npos) {
         throw ("cometes␣no␣tancades");
      }
      else {
         t = s.substr(primer, ultim - primer);
         primer = ultim + 2; // ens mengem els ",
         if (primer > s.length()) {
            ultim = string::npos; //era l'ultim token
         }
      }
   }
   return t;
}
```

El fitxer d'exemple `prova_tokens_csv.cpp`

És un programa principal que mostrar l'ús de les funcions dels fitxers anterior. Va llegint línia a línia el contingut d'un fitxer CSV i mostra primer tota la línia tal qual l'ha llegida, i a continuació mostra la descomposició de la línia que retorna la funció `tokens`, columna per columna.

Fixeu-vos que, quan invoquem `tokens`, li passem la línia que hem llegit, el caràcter que sabem que fa de separador (un `;`, en aquest cas) i un `false` per indicar que les dades no estan entre cometes.

Per fer-ho servir, cal que poseu un `#include "eines.h"` al `.cpp` on ho feu servir i que, després de llegir cada línia del fitxer CSV, invoqueu la funció `tokens`, que us retornarà un `vector<string>` amb el valor per separat de cada columna de la fila.

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include "eines.h"

using namespace std;

int main() {
   ifstream f;
   string linia;
   vector<string> items;
   string fitxer = "prova.csv";
   f.open(fitxer);
   if (!f.fail()) {
      getline(f, linia);
      while (!f.eof()) {
         items = tokens(linia, ',', false);
         cout << linia << endl;
         for (auto i : items) {
            cout << "\t[" << i << "]" << endl;
         }
         getline(f, linia);
      }
   }
   return 0;
}
```

Annex: Dades per treballar i exemple de fitxer de joc de proves

Us oferim dos fitxers CSV per fer proves. El primer és:

- `/u/prof/dfiguls/Public/padroCurt.csv`

I el segon és un fitxer CSV més grans que corresponen al padró real de l'ajuntament de Girona:

- `/u/prof/dfiguls/Public/padroLlarg.csv`

Els dos fitxers fan servir el `;` com a separador i fan servir cometes per delimitar les dades.

No pugeu el fitxer de dades gran a `bas.udg.edu` per fer les vostres proves perquè ocupa molt d'espai de disc i us podríeu quedar sense quota. Quan executeu el programa a bas entreu el fitxer indicat anteriorment amb el camí sencer:
`/u/prof/dfiguls/Public/padroLlarg.csv`.

A Moodle teniu un joc de proves bàsic amb dos fitxers, un pel fitxer `padroCurt.csv` i l'altre pel `padroLlarg.csv`. En tots dos casos disposeu del fitxer d'entrada i la sortida que es genera. Quan executeu aquests jocs de proves sense estar connectats a `bas` (en el vostre ordinador, a les aules...) recordeu de canviar el path del mètode (a) Llegir dades, per tal que vagi a buscar els fitxers CSV al directori on els teniu guardats en el vostre ordinador.