Universitat de Girona
**Escola Politècnica Superior**



# Exercici 1 de laboratori

## Estructures de Dades i Algorítmica

Curs 2025/26

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

**David Fíguls**
david.figuls@udg.edu

(PDF creat el 6 d'octubre de 2025)

# Índex

**1 Presentació** **2**

**2 Feina a fer** **2**
2.1 Operacions públiques de `Mobilitat` . . . 3
2.2 Detalls del disseny i implementació . . . 3
2.2.1 Estructures de dades . . . 4
2.2.2 Format per l'entrada de dades . . . 4

**3 Lliurament** **4**

**4 Dades** **5**

**5 Joc de proves i sortida** **5**
5.1 Fitxer de proves: `inMobilitatCurtE1.txt` . . . 5
5.2 Sortida esperada del fitxer de proves: `outMobilitatCurtE1.txt` . . . 6

**6 Com descompondre una línia d'un fitxer CSV** **8**

## 1 Presentació

Estem interessats en l'estudi de la mobilitat de l'àrea metropolitana de Barcelona. Tenim dades anònimes d'enquestes de mobilitat, amb informació dels desplaçaments que realitza cada persona enquestada, incloent comarca d'origen i destí, hora del dia, durada, distància, transports utilitzats, sexe, edat, estudis, entre d'altres.

El conjunt de dades està format per un fitxer amb unes 80 K files, una per cada trajecte de cada persona entrevistada. 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. La distància del trajecte està codificada amb un enter (de l'1 al 7) segons aquesta taula:

| Codi | Distància |
|---|---|
| 1 | Menys de 500 metres |
| 2 | De 500 a 2.000 metres |
| 3 | De 2.000 a 5.000 metres |
| 4 | De 5.000 a 10.000 metres |
| 5 | De 10.000 a 50.000 metres |
| 6 | De 50.000 a 100.000 metres |
| 7 | Més de 100.000 metres |

Taula 1: Taula de distàncies.

## 2 Feina a fer

La vostra feina consistirà en llegir les dades d'un fitxer CSV amb dades de les enquestes de dos anys (2022 - 2023) , i guardar la informació necessària en una estructura de dades que heu de dissenyar i implementar utilitzant diversos contenidors de la biblioteca STL. Un cop guardades, haureu d'implementar diferents consultes respectant els criteris d'eficiència que us detallem més endavant.

Les microdades estaran emmagatzemades en una classe anomenada `Mobilitat`, 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 `Mobilitat`[1] 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.

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

## 2.1 Operacions públiques de `Mobilitat`

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

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

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

Llegeix totes les files del fitxer CSV de la ruta indicada i va emmagatzemant els valorsPer tal de facilitar-vos la lectura de dades, a l'apartat Com descompondre una línia d'un fitxer CSV us donem instruccions i codi d'exemple sobre com fer-ho.

Tingueu present que la primera línia del fitxer conté els noms de les columnes i, per tant, no forma part de les dades. El camp d'hora d'inici pot ser >=24, significant que és una hora de l'endemà. Feu mòdul 24 per restringir-ho de 0..23.

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) `int nombreTransports(int distancia) const;`

Donat un codi de `distància` (valor d'1 a 7), obtenir nombre de tipus de trasports diferents (cotxe, moto, bus... corresponents al camp mitjaPrincipal del fitxer csv) que s'utilitzen per fer trajectes d'aquesta distància.

(c) `>CONTENIDOR< nombrePersonesPerTransport(int distancia) const;`

Donat un codi de `distància` (valor d'1 a 7), retorna un contenidor amb el nombre de persones (`int`) que utilitzen cada tipus de transport (`string`) per fer trajectes d'aquesta distància.

El resultat estarà ordenat per nombre de persones descendentment.

(d) `>CONTENIDOR< mesRapid() const;`

Retorna un contenidor que guarda, per cada distància, el tipus de trasport més ràpid i el temps promig corresponent.

## 2.2 Detalls del disseny i implementació

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

1. Us donem la definició bàsica de 4 classes que necessitareu. La classe `Mobilitat` agrupa totes les dades i implementa les operacions abans descrites.
2. Dissenyeu i implementeu totes les classes necessàries separant bé les definicions (fitxers `.h`) de les implementacions (fitxers `.cpp`).
3. El programa principal, que desplega el menú d'opcions, es dirà `main.cpp` i tindrà, entre altres coses, un objecte de la classe `Mobilitat`.

   Organitzeu el codi en accions i funcions. **No feu un main monolític**!
4. Heu de respectar fil per randa els noms i els paràmetres de les operacions que consten en aquest enunciat.
5. No oblideu les pre- i postcondicions. Per les classes poseu-les als fitxers `.h`, i pel `main.cpp` on definiu les accions i funcions.
6. 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.
7. L'entrada seguirà les pautes definides a la subsecció 2.2.2 i es farà tota des del programa principal amb l'excepció comentada anteriorment pel mètode de `Mobilitat` 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**.

### 2.2.1 Estructures de dades

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

- Les consultes (opcions b-d) no poden ser lineals sobre el nombre de persones.
- Les consultes (b-d) han de tenir un cost assimptòtic constant $O(1)$. Recordeu però, que un cost $O(N)$ on N sigui molt petita, es pot considerar $O(1)$.
- Les consultes (b-d) han de ser el més eficient possibles.
- 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.

### 2.2.2 Format per l'entrada de dades

El programa presentarà un menú per escollir quina operació es vol fer llegint un número entre 0 i 4 (0 per acabar). En funció de l'operació escollida es llegiran les dades necessàries, s'executarà l'operació interactuant amb `Mobilitat` 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.

**02-04** ... cadascun dels mètodes descrits anteriorment (b) a (d) 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. A l'apartat Joc de proves i sortida hi trobareu un exemple de joc de proves, i al Moodle en teniu un de més gran.

## 3 Lliurament

El dia de l'entrega haureu de tenir el vostre programa a `bas.udg.edu`, preparat per compilar i executar. Això inclou:

1. el codi font (només fitxers `.cpp` i `.h`, res de fitxers objecte ni executables)
2. els fitxers del joc de proves (veure Sec. 5),
3. el fitxer `llegeix.me` on expliqueu quin objectiu té cada fitxer del joc de proves (i qualsevol altre comentari que vulgueu fer sobre el vostre codi).

**IMPORTANT:** Cal que seguiu les instruccions sobre com lliurar les activitats de laboratori que teniu a Moodle. Assegureu-vos que ho feu com us demanem, sobretot les pre- i postcondicions tant a les classes com al `main`.

# 4 Dades

Us oferim dos fitxers CSV per fer proves. El primer és un subconjunt de les dades:

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

I el segon conté les respostes de totes les enquestes dels dos anys estudiats:

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

Tots els fitxers fan servir la `;` com a separador i fan servir cometes per delimitar les dades.

**No pugeu el directori de dades grans 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; per exemple:**
`/u/prof/dfiguls/Public/mobilitatLlarg.csv`.

# 5 Joc de proves i sortida

Cal que acompanyeu el vostre codi amb un joc de proves. Recordeu que aquest està format per un conjunt de fitxers tipus text, substituint l'entrada estàndard, ens permeten provar sistemàticament els nostres programes.

A continuació descrivim el joc de proves bàsic que compartim amb vosaltres. Vosaltres **l'heu de complementar amb altres fitxers** per assegurar que el vostre programa funciona en diferents interaccions amb l'usuari.

El joc de proves que compartim consta d'un fitxer per a cada fitxer de dades. Disposeu a Moodle dels dos fitxers del joc de proves (i les corresponents sortides esperades):

- `inMobilitatCurtE1.txt`, `outMobilitatCurtE1.txt` (vegeu contingut a Sec. 5.1 i 5.2, respectivament)
- `inMobilitatLlargE1.txt`, `outMobilitatLlargE1.txt` (que trobareu a Moodle)

Els fitxers `in***E1.txt` inclouen la ruta a `bas.udg.edu` dels fitxers `mobilitatCurt.csv` i `mobilitatLlarg.csv`. Si voleu executar aquests jocs de proves en local (no a `bas`), recordeu de canviar la ruta per la ruta al directori on els teniu guardats en el vostre ordinador.

Heu d'executar el vostre programa redireccionant l'entrada estàndard a cadascun dels fitxers d'entrada. És **molt important** que el contingut del fitxer correspongui exactament amb el que demana el programa. Si redireccioneu també la sortida del programa amb:

```
$ ./e1 < inMobilitatCurtE1.txt 1> meuOutMobilitatCurtE1.txt 2>/dev/null
```

es guardarà al fitxer `meuOutMobilitatCurtE1.txt` tot el que es mostraria per sortida estàndar (`cout`). Pot ser útil per comparar-ho (`$ diff meuOutMobilitatCurtE1.txt outMobilitatCurtE1.txt`) amb la sortida esperada que us donem.

## 5.1 Fitxer de proves: `inMobilitatCurtE1.txt`

```
01
/u/prof/dfiguls/Public/mobilitatCurt.csv
02
1
02
2
02
3
02
4
02
5
02
6
02
7
03
1
03
2
03
3
03
4
03
5
03
6
```

```
03
7
04
0
```

## 5.2 Sortida esperada del fitxer de proves: `outMobilitatCurtE1.txt`

```
**********
* 01: Llegir dades *
**********
Numero de linies: 348
**********
* 02: Nombre transports *
**********
Distància Menys de 500 metres ==> 3
**********
* 02: Nombre transports *
**********
Distància De 500 a 2.000 metres ==> 8
**********
* 02: Nombre transports *
**********
Distància De 2.000 a 5.000 metres ==> 13
**********
* 02: Nombre transports *
**********
Distància De 5.000 a 10.000 metres ==> 8
**********
* 02: Nombre transports *
**********
Distància De 10.000 a 50.000 metres ==> 6
**********
* 02: Nombre transports *
**********
Distància De 50.000 a 100.000 metres ==> 0
**********
* 02: Nombre transports *
**********
Distància Més de 100.000 metres ==> 0
**********
* 03: Nombre persones per transport *
**********
Distància Menys de 500 metres
0 : Peu >5 minuts => 27
1 : Peu <= 5 minuts => 25
2 : Cotxe com a conductora => 1
**********
* 03: Nombre persones per transport *
**********
Distància De 500 a 2.000 metres
0 : Peu >5 minuts => 30
1 : Metro => 9
2 : Autobús TMB => 6
3 : Bicicleta => 6
4 : Cotxe com a conductora => 4
5 : Moto com a conductora => 3
6 : Altres no motoritzats => 1
7 : Tramvia => 1
**********
* 03: Nombre persones per transport *
**********
Distància De 2.000 a 5.000 metres
0 : Metro => 13
1 : Autobús TMB => 8
2 : Cotxe com a conductora => 4
3 : Peu >5 minuts => 4
4 : FGC => 2
5 : Taxi => 2
6 : Bicicleta => 2
7 : Altres no motoritzats => 1
8 : Furgoneta => 1
9 : Tramvia => 1
10 : Moto com a conductora => 1
11 : Ciclomotor com a conductora => 1
12 : Ciclomotor com a acompanyant => 1
**********
* 03: Nombre persones per transport *
**********
Distància De 5.000 a 10.000 metres
0 : Metro => 6
1 : Altres autobús => 3
2 : Cotxe com a conductora => 3
3 : Autobús TMB => 2
4 : Cotxe com a acompanyant => 2
5 : Tramvia => 1
6 : Taxi => 1
7 : Moto com a conductora => 1
**********
* 03: Nombre persones per transport *
```

```
************************************
Distància De 10.000 a 50.000 metres
0 : Cotxe com a conductora => 3
1 : Cotxe com a acompanyant => 2
2 : FGC => 2
3 : Altres autobús => 2
4 : Moto com a conductora => 1
5 : Rodalies Renfe => 1
************************************
* 03: Nombre persones per transport *
************************************
Distància De 50.000 a 100.000 metres
************************************
* 03: Nombre persones per transport *
************************************
Distància Més de 100.000 metres
******************
* 04: Més ràpid *
******************
Menys de 500 metres : Peu <= 5 minuts => 3.89091
De 500 a 2.000 metres : Moto com a conductora => 8.75
De 2.000 a 5.000 metres : Altres no motoritzats => 6.5
De 5.000 a 10.000 metres : Taxi => 20
De 10.000 a 50.000 metres : Moto com a conductora => 30
De 50.000 a 100.000 metres : -- => 1.79769e+308
Més de 100.000 metres : -- => 1.79769e+308
```

# 6 Com descompondre una línia d'un fitxer CSV

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. El codi que us donem és: els fitxers `eines.h` i `eines.cpp`, i un `prova_tokens_csv.cpp` que mostra com fer-ho servir. Ho trobareu a Moodle.

## 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;
}
```