Problema 3.2: Els fitxers

Josep M. Ribó

19 de desembre de 2008

1 Objectius

- Treballar amb fitxers
- Treballar amb fitxers de text
- Treballar amb fitxers seqüencials (usant la jerarquia de classes SeqFile)
- Treballar amb fitxers directes

Per fer aquest problema, cal que us llegiu:
- L'apartat 2.6 dels apunts.

2 Un fitxer de persones amb format *pseudo*-xml

2.1 XML

L'xml és un llenguatge de marques (semblant a l'html). Hi ha, però, diferències fonamentals entre l'html i l'xml. Veiem-ne algunes:

- L'html s'usa per introduir elements de format en un document (ex: negretes, línies en blanc, taules, enumeracions, etc.). Per contra, l'xml s'usa com a llenguatge per descriure dades, independentment del format en què aquelles dades s'acabaran presentant.

  Per exemple, un document xml pot descriure vàries persones. La primera s'anomena Pep, té 23 anys i com a nif, 23456789-A; mentre que la segona s'anomena Maria, t 30 anys i com a nif 11111111-R...

- Mentre que les marques de l'html són fixades (e.g., `<b>`, per la negreta, `<it>`, per la cursiva ...), en el cas d'xml, cada tipus de document (que descriu un tipus específic de dades) té el seu propi conjunt de marques.

  Per exemple, un document xml que descrigui persones podria tenir marques com ara `<person>`, `<dni>`, `<name>`...

  Per la seva banda, un document xml que descrigui assignatures podria tenir marques com ara `<subject>`, `<title>`, `<credits>`, `<course>`...

  L'estructura d'un document xml es defineix mitjançant els anomenats *DTDs* o també amb els *schemas*.

2.2 Un exemple de document XML

A continuació us presentem un document *aproximadament xml* per descriure una col.lecció de persones. El document no és exactament xml perquè no està basat en cap DTD o schema que el descrigui.

```
<persons>

  <person>
    <name> anna </name>
    <nif> 12345678A </nif>
    <age> 23 </age>
  </person>

  <person>
    <name> pep </name>
    <nif> 11111111C </nif>
    <age> 29 </age>
  </person>

  <person>
    <name> carles </name>
    <nif> 12121212X </nif>
    <age> 26 </age>
  </person>

</persons>
```

Notem que hi ha un parell de marques `<persons> ... </persons>` que encerclen tot el document.

Cada persona específica ve descrita per les marques `<person>...</person>`. I cada propietat de la persona ve encerclada per una marca diferent (e.g., `<name> ...</name>`).

Notem també que un fitxer xml **és un fitxer de text**.

3 Lectura del fitxer de persones

3.1 Decripció del fitxer

Per llegir un fitxer xml existeixen *parsers* (analitzadors sintàctics de xml) que ens permeten obtenir ràpidament els seus elements. Nosaltres, però, ens limitarem a fer un analitzador de fitxers de persones amb el format descrit anteriorment i amb la precondició que els fitxers que llegirem **no estaran mal formats**. Un exemple de fitxer mal format seria:

```
  <person>
    </name> anna </name>
    <age> 23 </age>
    <nif> 12345678A
  <person>
</persons>
```

Aquest fitxer està mal format perquè:

- No s'inicia amb la marca `persons`
- La propietat `name` s'inicia amb `/name`

- La propietat `nif` no té marca d'acabament (`/nif`)
- La propietat `age` no està ordenada adequadament (ha d'aparèixer la darrera).
- La persona no acaba correctament (`person` en lloc de `/person`).

Peró, com hem dit, per això no ens haurem de preocupar. Això sí, permetrem que hi hagi tants espais i línies en blanc entre les diferents marques com es desitgi.

Per exemple:

```
...
  <person>


    <name>        pep </name>
    <nif>11111111C</nif>

    <age> 29 </age>


</person>
...
```

seria un fragment de fitxer acceptable.

3.2 El que cal fer

Farem una acció `readPersons` que:

- Llegirà un fitxer xml de persones amb el format descrit anteriorment.
- Generarà un fitxer de la classe `SeqFileI<Person>` amb les dades de les persones llegides del fitxer xml.

Posteriorment implementarem una altra acció `applyFSeq` que obrirà el fitxer acabat de crear com a fitxer `SeqFileC<Person>` i n'escriurà el seu contingut per la sortida estàndar.

4 readPersons

L'acció `readPersons` tindrà la capcelera següent:

```
void readPersons(const MyString& fileName, const MyString& fileName2)
```

llegirà un fitxer xml de persones anomenat `fileName` amb el format descrit i posarà el seu contingut en un fitxer `SeqFileI<Person>` anomenat `fileName2`.

La classe `Person` està definida de la manera següent:

```
class Person :public Object{
public:
  MyString name;
  MyString nif;
  int age;
```

```
  Person(){}
  void copy(...){...}
  Object* clone(){...}
  bool operator==(...){...}
  MyString toString(){...}
};
```

I l'operació que proposem pot tenir un codi semblant al següent:

```
void readPersons(const MyString& fileName,
                 const MyString& fileName2){

  ifstream fpers;
  MyString label;
  Person pers;
  SeqFileI<Person> seqfp(fileName2);

  fpers.open(fileName.toArrayChar(),ios::in);

  if (fpers.fail()){throw FileException();}

  getNextLabel(fpers,label);
   /***
     getNextLabel obte sobre label la seguent etiqueta del fitxer.
     Aquesta etiqueta SEGUR que sera <persons> (doncs suposem que
     llegirem un document ben format.
     getNextLabel obte l'etiqueta i la salta.
   ***/

  readPerson(fpers, pers);
   /***
     Llegeix la propera persona que troba al fitxer fpers i la
     retorna a l'objecte de sortida pers.

     readPerson(..) suposa que la propera marca que es trobara
     al fitxer xml es <person>.

     Si al fitxer fpers ja no hi ha cap altra persona (i, per tant,
     la primera marca que es troba es </persons>, aleshores l'objecte
     pers de sortida conte MyString("") a pers.nif
   ***/

  while (!(pers.nif == MyString("")) ){

    //(1): escriure pers a seqfp

    readPerson(fpers,pers);
  }

  //(2):Tancar seqfp
}
```

Es demana:

- Escriure el codi de la instrucció (1) que manca a l'acció anterior.
- Assegura't que el fitxer `seqfp` queda ben tancat (instrucció (2)).

- Com ha d'estar representada la classe `MyString` per tal que l'escriptura a `seqfp` d'un objecte `Person` escrigui físicament al fitxer el nom i el nif?

5 readPerson

Ha arribat el moment d'implementar l'acció `readPerson`. O sigui, l'acció que llegirà **la propera persona** del fitxer de persones.

Aquesta acció té la capcelera següent:

```
void readPerson(ifstream& fpers,Person& pers)
```

I obtindrà a l'objecte `pers` la propera persona que trobi al fitxer `fpers`.

Recordem que aquesta acció es pot trobar en dues situacions diferents:

- La propera marca que llegeixi del fitxer `fpers` és `<person>`. En aquest cas, llegirà la persona i la retornarà a `pers`.
- La propera marca que llegeixi del fitxer `fpers` és `</persons>`. Això vol dir que ja no hi ha més persones per llegir i, en aquest cas, retornarà un objecte `pers` amb `nif==''''`.

Veiem el codi d'aquesta acció:

```
void readPerson(ifstream& fpers, Person& pers){

  Pair pair;
   /***
    Pair indicara un parell (property,value).

    Exemple: En el cas <name>pep</name>

    la property seria 'name' i el value 'pep'


   ***/

  MyString label;


  getNextLabel(fpers,label);
   /***
     LLegeix la seguent marca de fpers (que pot ser
     "person" o "/persons"
   ***/

  if (label==MyString("/persons")) pers.setNif("");
  else{
    getNextPair(fpers,pair);
       /***
        Llegeix el parell (name,nomPersona), que sera la primera
        propietat de la persona
       ***/

    //(1): Establir l'atribut name de pers amb el nom llegit a fpers
```

```
  //(2): Llegir el parell (nif,nifPersona) i establir l'atribut nif
  //     de pers amb  el valor nifPersona

  //(3): Llegir el parell (age,edatPersona) i establir l'atribut age
  //     de pers amb  el valor edatPersona

 }
}
```

Notes:

- L'acció `getNextPair(fpers,pair)` obté el proper parell del tipus (propietat,valor) del fitxer `fpers` i el col.loca a `pair`.
- Un parell del tipus (propietat, valor) pot ser:
  - `<name>pep</name>`
    propietat= name, valor= pep
  - `<age>23</age>`
    propietat=age, valor=23
- `pair` és un objecte de la classe `Pair`, definida seguidament:

```
class Pair{
 public:

  MyString property;
  MyString value;

};
```

Es demana:

- Escriure les instruccions (1), (2) i (3) del codi de `readPerson`. Per fer això, us caldrà cridar l'acció `getNextPair(fpers,pair)` per tal d'obtenir el nif i l'edat. Un cop hagueu obtingut els valors d'aquestes propietats, us caldrà escriure'ls a l'objecte `pers`

6 Les accions `getNextLabel` i `getNextPair`

Fins ara he fet jo la major part de la feina. Ara us toca a vosaltres. Implementeu les accions:

- `void getNextLabel(ifstream& fpers, MyString& label)`
  Llegeix la propera etiqueta del fitxer `fpers` i la retorna a `label`.
- `void getNextPair(ifstream& fpers, Pair& pair)`
  Llegeix la propera parella (propietat, valor) de `fpers` i la retorna a `pair`.

Per implementar aquestes accions us serà útil l'operació dels `istreams`: `fpers.getLine(...)`

7 L'escriptura del fitxer seqüencial de persones

Implementa una acció:

```
template<class Action>
void applyFSeq(const MyString& fileName Action& action)
```

que:

- Llegeixi el fitxer seqüencial de persones creat per `readPersons(..)`
- Apliqui a cada persona llegida del fitxer l'acció `action`.

  Per fer això requerirem que la classe que instancïi `action` sobrecarregui `operator()`. Vegeu apunts 2.6.3

  Una acció exemple que us proposo simplement escriurà a la sortida estàndar cadascuna de les persones del fitxer.