Problema de la lliga espanyola. Examen juny-04

Josep M. Ribó

Juliol-04

(Extracte de solució del problema de l'examen de MTP-gestió de juny-2004)

1 Enunciat

L'objectiu d'aquest problema és dissenyar un programa per comptabilitzar la classificació dels equips que participen a la lliga espanyola de futbol i poder-hi afegir sempre que es requereixin els resultats de nous partits disputats.
En particular, el programa haurà de:

- Llegir de l'entrada estàndar el nombre d'equips participants i els seus noms.

  El format de lectura d'aquestes dades és el següent:

  ```
  4 Madrid Barcelona Bilbao Mallorca
  ```

- Mostrar un menú amb les opcions següents:

  1. Llegir més resultats de l'entrada estàndar.
     Aquests resultats tindran el format següent:

     ```
     Mallorca Madrid 2 1 Barcelona Bilbao 0 2 Barcelona Mallorca 0 1
     Madrid Bilbao 0 4 Bilbao Mallorca 2 0 Madrid Barcelona 0 0
     final
     ```

     Podem suposar que la separació entre cadenes sempre és un espai en blanc (agafem, doncs, els canvis de línia a l'exemple anterior com a espais en blanc). Podem suposar també que no hi haurá cap error a l'entrada. La cadena "final" indica que no hi ha més partits.
  2. Escriure per la sortida estàndar els punts, gols a favor i gols en contra que té acumulats fins el moment un determinat equip.
     Per exemple: Madrid: 1 punt, 1 gol a favor, 6 gols en contra
  3. Obtenir per la sortida estàndar la classificació ordenada per punts dels equips que participen a la lliga:
     En aquest exemple:

     ```
     Bilbao, 9
     Mallorca, 6
     Barcelona, 1
     Madrid, 1
     ```

Recordeu que les victòries valen 3 punts, els empats valen 1 punt i les derrotes, 0 punts.

4. Obtenir tots els resultats de tots els partits que ha disputat un equip.
   En aquest exemple:
   Barcelona: Barcelona-Bilbao: 0-2, Barcelona-Mallorca: 0-1, Madrid-Barcelona: 0-0
5. Sortir

Per tal de desenvolupar aquest programa ens ajudarem de dues classes: *Partit* i *ClassificacioEquip*

- La classe *Partit* modelitza un partit de futbol (evidentment, els aspectes que ens interessen d'un partit de futbol: Els equips que el juguen i el resultat del partit).
- La classe *ClassificacioEquip* modelitza els aspectes que ens interessen de la classificació d'un equip. Això és: els punts que ha aconseguit fins el moment, els gols a favor, els gols en contra i els partits que ha jugat.

Es demana:

1. (2 punts) Especifica i implementa la classe *Partit*. Tinguis cura en determinar quines operacions ha d'oferir aquesta classe. Indica a quin fitxer va cada part.
2. (3 punts) Especifica i implementa la classe *ClassificacioEquip*. Novament, cal que seleccionis molt acuradament les operacions que ha d'oferir aquesta classe.
3. Ara ha arribat el moment d'implementar el programa usuari que hem descrit anteriorment. Afortunadament, **no cal que implementis tot el programa.** De fet, el programa es presenta seguidament:

```
#include <iostream>
#include "partit.h"
#include "classifequip.h"
int main()
{
  int nequips, opcio;
  //Declaracio de vclassifequips????????
  obtenirEquips(nequips,vclassifequips);
  do{
  opcio=mostrarMenu();
  if (opcio==1) llegirResultats(vclassifequips,nequips);
  else if (opcio==2) obtenirPuntsEquip(vclassifequips,nequips);
  else if (opcio==3) obtenirClassificacio(vclassifequips,nequips);
  else if (opcio==4) obtenirResultatsPartitsEquip
                                        (vclassifequips,nequips);
  }while (opcio!=5);
  return 0;
}
```

D'aquest programa cal fer el següent:

(a) (1 punt) Declarar adequadament la variable `vclassifequips`

(b) (3 punts) Implementar l'acció `llegirResultats` que llegeix de l'entrada estàndar els resultats dels diferents partits disputats i els emmagatzema adequadament a `vclassifequips`.

4. (2 punts) Et proposo afegir una nova classe per tal d'obtenir una solució més elegant. Com es podria anomenar aquesta classe? Quines operacions oferiria?

Recorda que les operacions que associïs a cada classe hauran de permetre col.locar i obtenir les informacions que els programes usuaris requereixen d'aquella classe. Recorda també que, en general, les classes no han de tenir operacions d'entrada/sortida.

2 Fitxer *partit.txt*

Classe: `Partit`
Objectiu: Modelitza un partit de futbol entre dos equips, del qual ens interessa el nom dels equips que juguen i el resultat del partit.
Trio usar la classe `Cadena` en lloc d'arrays de caràcters per emmagatzemar cadenes de caràcrters. S'hauria pogut treballar també amb arrays de caràcters.
Operacions:

- `Partit(Cadena& equip1, Cadena& equip2, unsigned int gols1, int gols2);`

  `Partit p(equip1,equip2,gols1,gols2);`

  Post: `p` és un partit jugat entre els equips `equip1` i `equip2`. Al final, `equip1` va aconseguir `gols1` gols i `equip2` en va aconseguir `gols2`.

- `void getEquip1(Cadena& equip1);`

  `p.getEquip1(equip1);`

- `void getEquip2(Cadena& equip2);`

  `p.getEquip2(equip2);`

- `int getGols1();`

  `ngols1=p.getGols1();`

- `int getGols2();`

  `ngols2=p.getGols2();`

- `void setEquip1(Cadena& equip1);`

  `p.setEquip1(equip1);`

- `void setEquip2(Cadena& equip2);`

  `p.setEquip2(equip2);`

- `void setGols1(unsigned int ng1);`

  `p.setGols1(ng1);`

- `void setGols2(unsigned int ng2);`

  `p.setGols2(ng2);`

3 Fitxer *partit.h*

```
#include "cadena.h"

class Partit{

  Cadena equip1;
  Cadena equip2;
  unsigned int gols1;
  unisgned int gols2;

public:

  Partit (Cadena& pequip1,Cadena& pequip2,
          unsigned int pgols1,unsigned int pgols2);
  void getEquip1(Cadena& pequip1);
  void getEquip2(Cadena& pequip2);
  unsigned int getGols1();
  unsigned int getGols2();
  void setEquip1(Cadena& pequip1);
  void setEquip2(Cadena& pequip2);
  void setGols1(unsigned int ng1);
  void setGols2(unsigned int ng2);
};
```

4 Fitxer *partit.cpp*

```
#include "partit.h"

  Partit::Partit(Cadena& pequip1,Cadena& pequip2,
           unsigned int pgols1,unsigned int pgols2)
  {
    equip1=pequip1; //Si equip1 i equip2 fossin array
                    //de caracters:
                    //strcpy(equip1,pequip1);

    equip2=pequip2;
    gols1=pgols1;
    gols2=pgols2;
  }


  void Partit::getEquip1(Cadena& pequip1)
  {
    pequip1=equip1;
  }
```

```
  void Partit::getEquip2(Cadena& pequip2)
  {
    pequip2=equip2;
  }

  unsigned int Partit::getGols1()
  {
    return gols1;
  }

  unsigned int getGols2()
  {
    return gols2;
  }

.......
```

5 Fitxer *classificacioEq.txt*

Classe: `ClassificacioEquip`
Objectiu: Modelitza la classificació d'un equip de futbol, considerant els punts que té, els gols a favor, els gols en contra i els partits jugats.
La classe `ClassificacioEquip` considera que un equip pot jugar en una lliga un màxim de `N=100` partits.
Operacions:

- `ClassificacioEquip(Cadena& equip);`

  ClassificacioEquip ceq(equip);

  Post: `ceq` és un objecte que contindrà la classificació de l'equip `ceq`. En aquests moments aquest equip té 0 punts,, 0 gols a favor, 0 gols en contra i cap partit realitzat.

- `void afegirPartit(Partit& p, bool& err);`

  ceq.afegirPartit(p);

  Post: S'ha afegit el partit `p` als partits que ha jugat l'equip que té per classificació `ceq` i s'han actualitzat convenientment els punts, gols a favor i gols en contra d'aquest equip. `err=fals`

  Si el partit ja s'havia introduït prèviament a la classificació d'aquest equip, aleshores, `err=cert` i `ceq=ceq'`

  Si l'equip que té per classificació `ceq` no és cap dels dos que juguen el partit `p`, `err=cert` i `ceq=ceq'`

  Si ja s'han introduït `N` partits a la classificació d'aquest equip, `err=cert` i `ceq=ceq'`

- `void getNomEquip(Cadena& equip);`

  `ceq.getNomEquip(equip);`

  Nota: Aquesta operació també hagués pogut tenir la capcelera:

```
Cadena getNomEquip();
```

- `unsigned int getPunts();`

  `np=ceq.getPunts();`

- `unsigned int getGolsFavor();`

  `ngf=ceq.getGolsFavor();`

- `unsigned int getGolsContra();`

  `ngc=ceq.getGolsContra();`

- `void getPartits(Partit vpart[], int npart);`

  `ceq.getPartits(vpart,npart);`

6 Fitxer *classificacioEq.h*

```
#include "partit.h"
#include "cadena.h"

const int N=100;

class ClassificacioEquip{

  Cadena nomequip;
  Partit vp[N];
  int npartits;

  int punts;
  int golsf;
  int golsc;

  void actualitzarClassif (unsigned int golsfav,
                           unsigned int golscon);
  bool partitRepetit(Partit& p);

public:

 ClassificacioEquip(Cadena& equip);

 void afegirPartit(Partit& p, bool& err);
 void  getNomEquip(Cadena& equip);
 unsigned int getPunts();
 unsigned int getGolsFavor();
 unsigned int getGolsContra();
 void getPartits(Partit vpart[], int npart);
};
```

7 Fitxer *classificacioEq.cpp*

```
#include "classificacioEq.h"


 ClassificacioEquip::ClassificacioEquip(Cadena& equip)
 {
   nomequip=equip;
   golsf=0;
   golsc=0;
   punts=0;
   npartits=0;
 }

 void  ClassificacioEquip::afegirPartit(Partit& p, bool& err)
 {
   Cadena eq1, eq2;

   p.getEquip1(eq1);
   p.getEquip2(eq2);

   if (partitRepetit(p) ||
       (!(eq1==nomequip) && !(eq2==nomequip) ) ||
        npartits==N)
     err=true;
   else{
     err=false;
     vp[npartits]=p;
     npartits++;
     if (eq1==nomequip)
       actualitzarClassif(p.getGols1(),p.getGols2());
     else
       actualitzarClassif(p.getGols2(),p.getGols1());

     }

 }


 void   ClassificacioEquip::getNomEquip(Cadena& equip)
 {...}
 unsigned int  ClassificacioEquip::getPunts()
 {...}
 unsigned int  ClassificacioEquip::getGolsFavor()
 {...}
 unsigned int  ClassificacioEquip::getGolsContra()
 {...}

 void  ClassificacioEquip::getPartits(Partit vpart[], int npart);
 {
   int i;
   for (npart=0; npart<npartits;npart++){
```

```
    vpart[npart]=vp[npart];
   }
 }

/*****************************************************

OPERACIONS PRIVADES


*****************************************************/


/*****************************************************

void ClassificacioEquip::actualitzarClassif
                            (unsigned int golsfav,
                             unsigned int golscon);

Afegeix a la classificaci de l'equip golsfav gols a favor
i golscon gols en contra. Actualitza tamb els punts de
l'equip segons sigui golsfav>golscon o viceversa.


*****************************************************/


void ClassificacioEquip::actualitzarClassif
                            (unsigned int golsfav,
                             unsigned int golscon)
 {
   golsf=golsf+golsfav;
   golsc=golsc+golscon;
   if (golsfav>golscon) punts=punts+3;
   else if (golsfav==golscon) punts=punts+1;
 }


/*************************************************

  bool partitRepetit(Partit& p);

  Retorna cert si el partit p ja est considerat a la
  classficaci d'aquest equip i fals altrament.

*************************************************/

  bool ClassificacioEquip::partitRepetit(Partit& p)
  {
   ...
  }
```

8 Acció `llegirResultats(...)`

Suposem, tal com diu l'enunciat, que no hi ha cap error a l'entrada de dades (en particular, suposarem que tots els equips s'han introduït correctament anteriorment).

```
void llegirResultats(ClassificacioEquip vclassifequips[], int nequips)
{
  Cadena equip1, equip2;
  unsigned int gols1, gols2, i;
  Cadena cadfinal("final");
  bool err;

  equip1.llegirCadena(' ');  //operacio de la classe Cadena que llegeix fins
                             //trobar el caracter que apareix al
                             //parametre. En aquest cas: ' '
  while (!(equip1==cadfinal)){
    equip2.llegirCadena(' ');

    cin>>gols1;
    cin>>gols2;

    p.setEquip1(equip1);
    p.setEquip2(equip2);
    p.setGols1(gols1);
    p.setGols2(gols2);

    i=obtenirIndexEquip(vclassifequips,nequips,equip1);
    vclassifequips[i].afegirPartit(p,err);

    i=obtenirIndexEquip(vclassifequips,nequips,equip2);
    vclassifequips[i].afegirPartit(p,err);

    equip1.llegirCadena(' ');
  }
}

unsigned int obtenirIndexEquip(ClassificacioEquip vclassifequips[],
                               int nequips, Cadena equip){
  Cadena eq;
  bool trobat;
  int i;

  i=0; trobat=false;

  while (! trobat){
    vclassifequips[i].getEquip(eq);
    if (eq==equip) trobat=true;
    else i++;
  }
  return i;
}
```