Corrigé des exercices

Remarque : les classes sont écrites sous forme “condensée”, avec le code directement dans la déclaration, pour raccourcir le texte.

Correction exercice 1

```
#include<iostream.h>
#include<string>
#include<vector.h>

class ObjetPostal
{
  private:
  string destination;
  float volume;

  public:
  ObjetPostal(string d="inconnue", float v=0){destination=d; volume=v;}
  virtual ~ObjetPostal(){}
  virtual string getDestination()const   {return destination;}
  virtual void setDestination(string d) {destination=d;}
  virtual float getVolume()const   {return volume;}
  virtual void setVolume(float v) {volume=v;}
  virtual void affiche(ostream& os)const
  {os << nomObjetPostal()+" "+getDestination()+"/"<<getVolume();}
  virtual string nomObjetPostal()const=0;
};

ostream& operator<<(ostream& os, const ObjetPostal* o)
{o->affiche(os); return os;}

class Lettre : public virtual ObjetPostal
{
  private:
  bool urgente;

  public:
  Lettre(string d="inconnue", float v=0, bool u=false):ObjetPostal(d,v){urgente=u;}
  virtual ~Lettre(){}
  virtual bool getUrgente()const   {return urgente;}
  virtual void setUrgente(bool u) {urgente=u;}
  virtual void affiche(ostream& os)const
  {ObjetPostal::affiche(os); os << " " <<getUrgente()<<endl;}
  virtual string nomObjetPostal()const{return "Lettre";}
};

class Colis : public virtual ObjetPostal
{
  private:
  string declareContenu;
```

```
  public:
  Colis(string d="inconnue", float v=0, string dc="mystere"):ObjetPostal(d,v){declareContenu=dc;}
  virtual ~Colis(){}
  virtual string getDeclareContenu()const{return declareContenu;}
  virtual void setDeclareContenu(string d) {declareContenu=d;}
  virtual void affiche(ostream& os)const
  {ObjetPostal::affiche(os); os << " " <<getDeclareContenu()<<endl;}
  virtual string nomObjetPostal()const{return "Colis";}
};


class SacPostal
{
  private:
  vector<ObjetPostal*> contenu;

  public:
  SacPostal(){}
  virtual ~SacPostal(){}
  virtual void ajoute(ObjetPostal* po)
  {contenu.push_back(po);}
  virtual void affiche(ostream& os)const
   {for (int i=0; i<contenu.size(); i++)
    os << contenu[i] << " ";
   }
  virtual float volume()const
   {float v=0;
    for (int i=0; i<contenu.size(); i++)
      v+=contenu[i]->getVolume();
   }
};

ostream& operator<<(ostream& os, const SacPostal& s)
{s.affiche(os); return os;}

void main()
{
  SacPostal *pSac = new SacPostal();

  Lettre *pl1 = new Lettre("Le pere Noel",0.05,1);
  Lettre *pl2 = new Lettre("Montpellier",0.07,0);
  Colis *pc1 = new Colis("Yolande",0.6,"sun-ray");
  Colis *pc2 = new Colis("Yolande",0.4,"bureau");

  pSac->ajoute(pc1); pSac->ajoute(pc2);
  pSac->ajoute(pl1); pSac->ajoute(pl2);
  cout << *pSac <<endl;
}
```

Correction exercice 2

```
#include<iostream.h>
#include<string>
#include<map.h>
#include <pair.h>

ostream& operator<<(ostream& os, const pair<const string,int>& p)
{os<<p.first<<" "<<p.second<<endl; return os;}

void main()
{
  map<string, int, less<string> > dico;
  string Mot;

  ostream_iterator<pair<const string,int> > outDate(cout," ");
  copy(dico.begin(), dico.end(), outDate);

  while(cin>>Mot)
   {if (dico.find(Mot)!=dico.end()) ++(*(dico.find(Mot))).second;
    else  dico[Mot]=1;}

  copy(dico.begin(), dico.end(), outDate);
}
```

Correction exercice 3

On définit une fonction de comparaison entre objets postaux.

```
bool compareObjets(ObjetPostal *po1, ObjetPostal *po2)
 {return (po1->getDestination() < po2->getDestination());}
```

On ajoute à la classe `ObjetPostal` la méthode suivante `trieEtAffiche`.

```
  virtual void trieEtAffiche(ostream& os)
  {
    sort(contenu.begin(),contenu.end(),compareObjets);
    ostream_iterator<ObjetPostal*> outInt(cout, " ");
    copy(contenu.begin(),contenu.end(), outInt);
  }
```

On complète le `main` par :

```
  pSac->trieEtAffiche(cout);
```

Correction exercice 4.

```
  // à partir de (x,y)-> x==y, crée x -> x==7
  replace_if(v.begin(),v.end(),bind2nd(equal_to<int>(), 7),8);
```