```
/*                    classe « vecteur » (exemple du cours C++)    */
/*                    Patrick TRAU ULP-IPST Strasbourg novembre 04 */
#include <iostream.h>

class vecteur
 {
 private ://la structure interne d'un vrai objet n'a pas à être publique
  float x,y,z;
 protected : //à la rigueur les héritiers peuvent accéder
             //directement aux attributs
  void setx(float a=0) {x=a;}
  void sety(float a=0) {y=a;}
  void setz(float a=0) {z=a;}
 public :
  vecteur(float a=0,float b=0,float c=0) {x=a;y=b;z=c;} //constructeur

//les accesseurs en lecture sont publics
  float getx(void) {return(x);}
  float gety(void) {return(y);}
  float getz(void) {return(z);}

  void affiche(ostream &flux) //flux est en argument car je veux
                //pouvoir utiliser cout mais aussi tout fichier texte
      {flux<<"["<<x<<","<<y<<","<<z<<"]";}
  void saisie(istream &flux); //si le flux est cin on pose des
                     //questions sur cout, sinon on saisit sans question

  void additionner(float a)
      {x=a+x;y=a+y;z=a+z;}
  void additionner(float a,float b,float c) //ceci est une surcharge.
      {x=a+x;y=b+y;z=c+z;}
  void additionner(vecteur a)               //encore une surcharge.
      {x=x+a.x;y=y+a.y;z=z+a.z;}

  float norme(void) {return(sqrt(x*x+y*y+z*z));}  //calcule la norme
  void normer(void)  //le modifie (le rend unitaire)
    {float n=norme();x/=n;y/=n;z/=n;}

  void multiplier_par(float a) {x=a*x;y=a*y;z=a*z;}
  float prodscal(vecteur v) {return(x*v.x+y*v.y+z*v.z);}

  vecteur operator = (int arg)  //opérateur de copie, prévu ici
                                // uniquement pour écrire V=0
    { if(arg==0) x=y=z=0; else cout<<"affectation suspecte\n"; }
 };  //n'oubliez pas ce ; c'est la fin d'une déclaration

void vecteur::saisie(istream &f)
 {
  if(f==cin)cout<<"entrez x : ";
  f>>x;
  if(f==cin)cout<<"entrez y : ";
  f>>y;
  if(f==cin)cout<<"entrez z : ";
  f>>z;
 }

//redéfinition des opérateurs (sous forme de fonctions)
ostream& operator << (ostream &f,vecteur v)
   {v.affiche(f);return(f);}
istream& operator >> (istream &f,vecteur &v) //v est modifié!
 { v.saisie(f); return(f); }

vecteur operator ^ (vecteur v,vecteur w)     //produit vectoriel
 {vecteur z(
           v.gety()*w.getz()-w.gety()*v.getz() ,
           v.getz()*w.getx()-w.getz()*v.getx() ,
           v.getx()*w.gety()-w.getx()*v.gety()
          );
  return(z);
 }

vecteur operator * (float f,vecteur v)     //produit par un réel
 {vecteur z=v;
  z.multiplier_par(f);
  return(z);
 }
vecteur operator * (vecteur v,float f)  //le prod par un réel est
commutatif !!!
 {return(f*v);}   //déjà défini dans l'autre sens, autant s'en servir !

vecteur operator / (vecteur v,float f)
 {return(v*(1/f));}

float operator * (vecteur v,vecteur w)     //produit scalaire
 {return v.prodscal(w);}

vecteur operator + (vecteur v,vecteur w)     //somme vectorielle
 {vecteur z=v;
  v.additionner(w);
  return(z);
 }

vecteur operator - (vecteur v,vecteur w)     //différence vectorielle
 {return(v+((-1)*w));}

/* petit programme main si l'on veut tester l'objet ***********
int main(void)
 {
  vecteur v(1,1,0),w,z;
  cout<<"la norme de "<<v<<" vaut "<<v.norme()<<"\n";
  cout<<"entrez vos nouvelles coordonnées :\n";
  cin>>w;
  cout<<"la norme de "<<w<<" vaut "<<w.norme()<<"\n";
  cout<<"le prod scal de "<<v<<" et "<<w<<" est "<<v.prodscal(w)
                                 <<" (ou "<<v*w<<")\n";
  z=2*(v^w);
  cout<<"le double de leur prod vect vaut "<<z<<"\n";
 }
/* ouf, c'est fini ************************************/
```