Solution Exercise 3.2

```
#include <iostream>
using namespace std;

int main()
{
  int entree,max;

  cout<<"Entrer un entier: ";
  cin>>entree;

  max=entree;

  cout<<"Entrer un entier: ";
  cin>>entree;

  if(entree>max) max=entree;

  cout<<"Entrer un entier: ";
  cin>>entree;

  if(entree>max) max=entree;

  cout<<"Entrer un entier: ";
  cin>>entree;

  if(entree>max) max=entree;

  cout<<"Entrer un entier: ";
  cin>>entree;

  if(entree>max) max=entree;

  cout<<"Le plus grand entier entrée est "<<max<<endl;

  return 0;
}
```

On remarque que cette façon de faire est répétitive onéreuse.
Les boucles, traitées dans le chapitre suivant, vont nous permettre de faire la même chose de manière beaucoup plus compacte et expéditive.

Solution Exercice Chap3.3

```
#include <iostream>
#include <cmath>
using namespace std;

int main()
{
  double a,b,c;
  double delta;
  double sol,r1,r2;
  double g,h;

  cout<<"Nous cherchons les racines de l'equation ax^2+bx+c" << endl;

  cout<<"Entrer la valeur de a: ";
  cin>>a;

  cout<<"Entrer la valeur de b: ";
  cin>>b;

  cout<<"Entrer la valeur de c: ";
  cin>>c;

  if(a==0)
  {
   if(b==0)
          {
           if(c==0)
           {
              cout<<"Tout x est solution" << endl;
           }
           else
           {
              cout<<"Il n'y a aucune solution" << endl;
           }
          }
     else
          {
          sol=-c/b;
          cout << "Il existe une racine unique = : " << sol << endl;
          }
  }

  else
  {
     delta=b*b-4.*a*c;

     if(delta==0)
          {
```

```
            sol=-b/(2.*a);
            cout<<"Il existe une racine double = : "<<sol<<endl;
           }

    else
          {
           if(delta>0)
           {
              r1=(-b-sqrt(delta))/(2.*a);
              r2=(-b+sqrt(delta))/(2.*a);
              cout<<"Il existe deux racines reelles = : " << r1 <<" et = : "  << r2 << endl;
           }
           else
           {
             g=-b/(2.*a);
             h=sqrt(-delta)/(2.*a);
             cout << "Il existe deux racines = : " << g << " + i " << h ;
             cout << " et = : "<<g<<" - i "<<h <<endl;
           }
          }
 }


 return 0;
}
```

(N.B. Erreur fréquent!)
Remarquez bien l'utilisation de deux signes d'égalité cote à cote, "==" comme conditionnelle pour vérifier si deux nombres sont égaux. Un seule signe d'égalité "=" aurait fait une affectation, ce qu'on ne veut pas dans cet exercice.

Solution Exercice Chap3.4

```
#include <iostream>
using namespace std;

int main()
{
  double xa,ya;
  double xb,yb;
  double xc,yc;
  double xd,yd;

  int test=0;
  int vertab=0,vertcd=0;
  double dirab,ordab;
  double dircd,ordcd;

  double xint,yint;

  cout<<"Entrer l'abscisse de A, puis l'ordonnee de A :"<<endl;
  cin>> xa >> ya;

  cout<<"Entrer l'abscisse de B, puis l'ordonnee de B :"<<endl;
  cin>>xb >> yb;

  cout<<"Entrer l'abscisse de C, puis l'ordonnee de C :"<<endl;
  cin>> xc >>yc;
  cout<<"Entrer l'abscisse de D, puis l'ordonnee de D :"<<endl;
  cin>>xd >>yd;

  if((xa==xb)&&(ya==yb))
  {
     test++;
     cout<<"A et B sont confondus"<<endl;
  }

  if((xc==xd)&&(yc==yd))
  {
     test++;
     cout<<"C et D sont confondus"<<endl;
  }

  if(test==0)
  {
     if(xa!=xb)
     {
              dirab=(yb-ya)/(xb-xa);
              ordab=ya-dirab*xa;
            }
     else
```

```
        {
         vertab++;
        }

    if(xc!=xd)
        {
         dircd=(yd-yc)/(xd-xc);
         ordcd=yc-dircd*xc;
        }
    else
        {
         vertcd++;
        }

    if((vertab==0)&&(vertcd==0))
        {
          if(dirab==dircd)
          {
           if(yc==ya+dirab*(xc-xa))
                {
                 cout<<"Les droites (AB) et (CD) sont confondues"<<endl;
                }
           else
                {
                 cout<<"Les droites (AB) et (CD) sont parallales"<<endl;
                }
          }
          else
          {
            xint=-(ordab-ordcd)/(dirab-dircd);
            yint=dirab*xint+ordab;

             cout<<"Les droites (AB) et (CD) s'intersectent au point de coordonnees
("<<xint<<", "<<yint<<")"<<endl;
          }
        }

    else
        {
         if((vertab==0)&&(vertcd!=0))
         {
           xint=xa;
           yint=dircd*xint+ordcd;

            cout<<"Les droites (AB) et (CD) s'intersectent au point de coordonnees
("<<xint<<", "<<yint<<")"<<endl;
         }

         else
         {
```

```
            if((vertab!=0)&&(vertcd==0))
                  {
                   xint=xc;
                   yint=dirab*xint+ordcd;

                   cout<<"Les droites (AB) et (CD) s'intersectent au point de coordonnees
("<<xint<<", "<<yint<<")"<<endl;
                  }

            else
                  {
                   if(xa==xc)
                   {
                     cout<<"Les droites (AB) et (CD) sont confondues"<<endl;
                   }
                   else
                   {
                     cout<<"Les droites (AB) et (CD) sont parallales"<<endl;
                   }
                  }
              }
          }
    }

  return 0;
}
```

Cet exercice montre comment un problème en apparence assez simple peut vite devenir compliqué avec de multiple accolades imbriqués. Une programmation structurée faisant appel à des fonctions peut simplifier cette démarche.