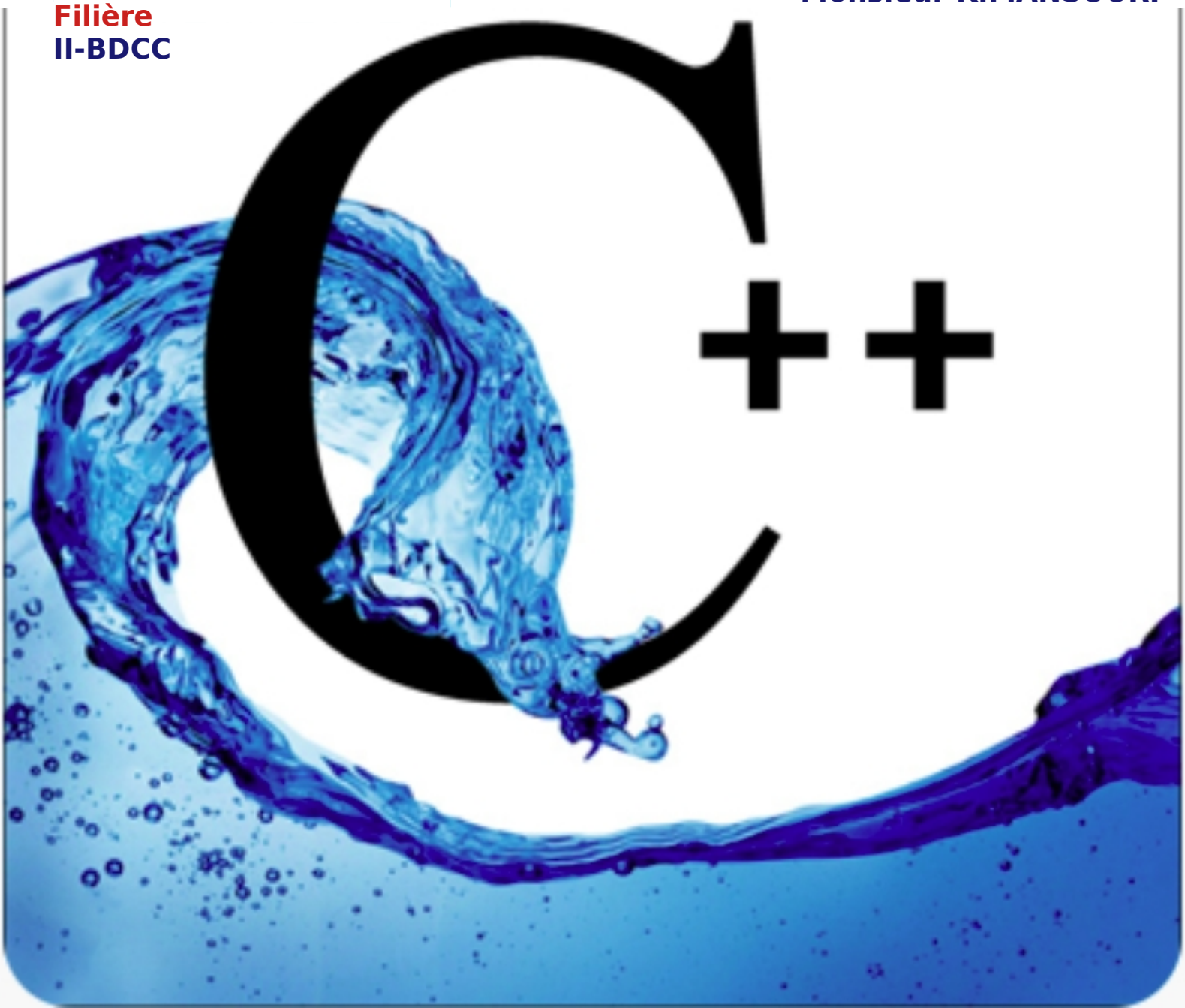
COMPTE RENDU DE CHAPITRE 5 + TPS

Réalisé par :
Abderrahmane ETTOUNANI
Filière
II-BDCC

Encadré par :
Monsieur K.MANSOURI



les exemples de cours & tps

----------project Class Complexe ----------

```cpp
#include <string.h>
#include <iostream>
#include <math.h>
using namespace std;
You, 10 seconds ago | 1 author (You)
class Complex
{
    double reel, imaginaire;

public:
    // Constructeur par défaut
    Complex()
    {
        reel = 0;
        imaginaire = 0;
    }
    // constructeur plein
    Complex(double re, double imag = 0)
    {
        reel = re;
        imaginaire = imag;
    }
    // partie real
    double real()
    {
        return reel;
    }
    // partie imaginaire
    double imag()
    {
        return imaginaire;      You, 46 minutes ago •
    }
    // the complex congugate
    Complex conj()
    {
        Complex res;
        res.reel = reel;
        res.imaginaire = -imaginaire;
        return res;
    }
```

```
40      // addition
41      Complex operator+(Complex c)
42      {
43          Complex res;
44          res.reel = c.real() + reel;
45          res.imaginaire = c.imag() + imaginaire;
46          return res;
47      }
48      //-----------------------------------------------
49      Complex operator-(Complex c)
50      {
51          Complex res;
52          res.reel = c.real() - reel;
53          res.imaginaire = c.imag() - imaginaire;
54          return res;
55      }
56      //-----------------------------------------------
57      Complex operator*(Complex c)
58      {
59          Complex res;
60          res.reel = (c.real() * reel) - imaginaire * c.imag();
61          res.imaginaire = reel * c.imag() + imaginaire * c.reel;
62          return res;
63      }
64      //-----------------------------------------------
65      int operator!=(Complex c)
66      {
67          if (c.imag() == imaginaire && c.real() == reel)
68              return 0;
69          else
70              return 1;
71      }
72
73      //-----------------------------------------------
74      int operator==(Complex c)
75      {
76          if (c.imag() == imaginaire && c.real() == reel)
77              return 1;
78          else
79              return 0;
80      }
```

```cpp
    // Magnitude of Square of a complex number
    double norm()
    {
        double res;
        res = imaginaire * imaginaire + reel * reel;
        return res;
    }
    // Complex number angle
    double arg()
    {
        double res = 0;
        res = atan(imaginaire / reel);
        return res;
    }

    // convert polar to Complex nimber
    Complex polar(double mag, double angle = 0)
    {
        Complex c(mag * cos(angle), mag * sin(angle));
        return c;
    }
    //-----------------------------------------
    Complex operator/(Complex c)
    {
        Complex res;
        res.reel = (c.real() * reel + c.imag() * imaginaire) / (c.imag() * c.imag() + c.real() * c.real());
        res.imaginaire = (c.real() * imaginaire + c.imag() * reel) / (c.imag() * c.imag() + c.real() * c.real());
        return res;
    }
    //-----------------------------------------
    //-----------------------------------------
    // Friends
    //-----------------------------------------
    //-----------------------------------------
    friend ostream &operator<<(ostream &, Complex &);
    friend istream &operator>>(istream &, Complex &);
    friend Complex operator*(double, Complex);
    friend Complex operator+(Complex, double);
    friend Complex operator*(Complex, double);
    friend Complex operator/(double, Complex);
    friend Complex operator/(Complex, double);
    friend Complex operator-(double, Complex);
    friend Complex operator-(Complex, double);
    friend Complex operator+(double, Complex);
    friend Complex operator+(Complex, double);
};
```

```cpp
ostream &operator<<(ostream &os, Complex &c)
{
    os << "The Complex object is : " << c.reel << " + " << c.imaginaire << "i " << endl;
    return os;
}
istream &operator>>(istream &in, Complex &c)
{
    cout << "Enter Real Part ";
    in >> c.reel;
    cout << "Enter Imaginary Part ";
    in >> c.imaginaire;
    return in;
}
Complex operator+(double d, Complex c)
{
    Complex res;
    res.reel = c.real() + d;
    res.imaginaire = c.imag() + d;
    return res;
}
Complex operator+(Complex c, double d)
{
    Complex res;
    res.reel = c.real() + d;
    res.imaginaire = c.imag() + d;
    return res;
}
Complex operator-(double d, Complex c)
{
    Complex res;
    res.reel = c.real() - d;
    res.imaginaire = c.imag() - d;
    return res;
}
Complex operator-(Complex c, double d)
{
    Complex res;
    res.reel = c.real() - d;
    res.imaginaire = c.imag() - d;
    return res;
}
```

```cpp
Complex operator*(Complex c, double d){
    Complex res;
    res.reel = (c.real() * d) - d * c.imag();
    res.imaginaire = d * c.imag() + d * c.reel;
    return res;
}
Complex operator*(double d, Complex c){
    Complex res;
    res.reel = (c.real() * d) - d * c.imag();
    res.imaginaire = d * c.imag() + d * c.reel;
    return res;
}
Complex operator/(double d, Complex c){
    Complex res;
    res.reel = (c.real() * d + c.imag() * d) / (c.imag() * c.imag() + c.real() * c.real());
    res.imaginaire = (c.real() * d + c.imag() * d) / (c.imag() * c.imag() + c.real() * c.real());
    return res;
}
Complex operator/(Complex c, double d){
    Complex res;
    res.reel = (c.real() * d + c.imag() * d) / (c.imag() * c.imag() + c.real() * c.real());
    res.imaginaire = (c.real() * d + c.imag() * d) / (c.imag() * c.imag() + c.real() * c.real());
    return res;
}
```

```cpp
int main()
{
    Complex c1;
    cin >> c1;
    cout << c1 << endl;
    // angle of magnitude
    cout << "Angle φ  = " << c1.arg() << endl;
    // Polar to Complex
    Complex c2 = c1.polar(2, 0.93);
    cout << c2 << endl;
    // Multiplication
    Complex c3 = c2 * c1;
    cout << c3 << endl;
    // devision
    Complex c4 = c2 / c1;
    cout << c4 << endl;
    // addition
    Complex c5 = c2 + c1;
    cout << c5 << endl;
    // soustraction
    Complex c6 = c2 - c1;
    cout << c6 << endl;      You, 1 second ago • Un
    return 0;
}
```

```
Enter Real Part 2
Enter Imaginary Part 4
The Complex object is : 2 + 4i

Angle φ  = 1.10715
The Complex object is : 1.19567 + 1.60324i

The Complex object is : -4.02162 + 7.98915i

The Complex object is : 0.440215 + 0.399458i

The Complex object is : 3.19567 + 5.60324i

The Complex object is : 0.804332 + 2.39676i
```

----------exemple de cours----------

```
You, 1 hour ago | 1 author (You)
#include <iostream>
using namespace std;
You, 1 hour ago | 1 author (You)
class vecteur
{
    float x, y;

public:
    vecteur(float, float);
    void afficher();
    vecteur operator+(vecteur);
};
vecteur::vecteur(float abs = 0, float ord = 0)
{
    x = abs;
    y = ord;
}
void vecteur::afficher()
{
    cout << "x =" << x << " y=" << y << "\n";
}
vecteur vecteur::operator+(vecteur v)
{
    vecteur res;
    res.x = v.x + x;        You, 1 hour ago • tp5 …
    res.y = v.y + y;
    return res;
}
int main()
{
    vecteur a(2, 6), b(4, 8), c, d, e, f;
    c = a + b;
    c.afficher();
    d = a.operator+(b);
    d.afficher();
    e = b.operator+(a);
    e.afficher();
    f = a + b + c;
    f.afficher();

    return 1;
}
```

```
x =6 y=14
x =6 y=14
x =6 y=14
x =12 y=28
```

```cpp
#include <iostream>
using namespace std;
class vecteur
{
    float x, y;
public:
    vecteur(float, float);
    void afficher();
    vecteur operator+(vecteur);
    float operator*(vecteur);
};
vecteur::vecteur(float abs = 0, float ord = 0)
{
    x = abs;
    y = ord;
}
void vecteur::afficher()
{
    cout << x << " , " << y;
}
vecteur vecteur::operator+(vecteur v)
{
    vecteur res;
    res.x = v.x + x;
    res.y = v.y + y;
    return res;
}
float vecteur::operator*(vecteur v)
{
    float res;
    res = v.x * x + v.y * y;
    return res;
}
int main()
{
    vecteur a(2, 6), b(4, 8);
    float f;
    f = a * b;
    cout << "produit scalaire de : vecteur A:( ";
    a.afficher();
    cout << " ) et vecteur B: (";
    b.afficher();
    cout << ") est  " << f;
    return 1;
}
```

```
produit scalaire de : vecteur A:( 2 , 6 ) et vecteur B: (4 , 8) est  56
```

```
You, 1 hour ago | 1 author (You)
#include <iostream>
using namespace std;
You, 1 hour ago | 1 author (You)
class liste
{
    int taille;
    float *adr;
public:
    liste(int);
    liste(liste &);
    void saisie();
    void affiche();
    void operator=(liste &);
    //  ~liste();
};
liste::liste(int t)
{
    taille = t;
    adr = new float[taille];
    cout << "Construction \n";
    cout << " Adresse de l'objet: " << this;
    cout << " Adresse de liste: " << adr << "\n";
}
liste::liste(liste &v)
{       You, 1 hour ago • tp5 …
    taille = v.taille;
    adr = new float[taille];
    for (int i = 0; i < taille; i++)
        adr[i] = v.adr[i];
    cout << "\nConstructeur par recopie";
    cout << " Adresse de l'objet:" << this;
    cout << " Adresse de liste:" << adr << "\n";
}
void liste::saisie()
{
    int i;
    for (i = 0; i < taille; i++)
    {
        cout << "Entrer un nombre:";
        cin >> *(adr + i);
    }
}
  void liste::affiche()
  {
      int i;
      cout << "Adresse:" << this << " ";
      for (i = 0; i < taille; i++)
          cout << *(adr + i) << " ";
      cout << "\n\n";
  }
  void liste::operator=(liste &lis)
  {
      cout << " \n hello from operator \n";
      int i;
      taille = lis.taille;
      delete adr;
      adr = new float[taille];
      for (i = 0; i < taille; i++)
          adr[i] = lis.adr[i];
  }
  int main()
  {
      cout << "Debutde main()\n";
      liste a(5);
      liste b(2);
      a.saisie();
      a.affiche();
      b.saisie();
      b.affiche();
      b = a;
      b.affiche();
      a.affiche();
      cout << "Fin de main() \n";
  }
```

```
Debutde main()
Construction
 Adresse de l'objet: 0x7ffc596cc0d0 Adresse de liste: 0x55a9490e22c0
Construction
 Adresse de l'objet: 0x7ffc596cc0e0 Adresse de liste: 0x55a9490e22e0
Entrer un nombre:2
Entrer un nombre:4
Entrer un nombre:1
Entrer un nombre:5
Entrer un nombre:7
Adresse:0x7ffc596cc0d0 2 4 1 5 7

Entrer un nombre:2
Entrer un nombre:5
Adresse:0x7ffc596cc0e0 2 5


 hello from operator
Adresse:0x7ffc596cc0e0 2 4 1 5 7

Adresse:0x7ffc596cc0d0 2 4 1 5 7

Fin de main()
```

```cpp
#include <iostream>
using namespace std;
You, 1 hour ago | 1 author (You)
class liste
{
    int taille;
    float *adr;

public:
    liste(int);
    liste(liste &);
    void saisie();
    void affiche();
    void operator=(liste &);
    float &operator[](int);
    //  ~liste();
};
float &liste::operator[](int i)
{
    return this->adr[i];
}
liste::liste(int t)
{
    taille = t;
    adr = new float[taille];
}
liste::liste(liste &v)
{

    taille = v.taille;
    adr = new float[taille];
    for (int i = 0; i < taille; i++)
        adr[i] = v[i];
}
void liste::saisie()
{
    int i;
    for (i = 0; i < taille; i++)
    {
        cout << "Entrer un nombre:";
        cin >> (*this)[i];
    }
}
```

```
43  void liste::affiche()
44  {
45      int i;
46      cout << "Adresse:" << this << " ";
47      for (i = 0; i < taille; i++)
48          cout << (*this)[i] << " ";
49  }
50
51  void liste::operator=(liste &lis)
52  {
53      int i;
54      taille = lis.taille;
55      delete adr;
56      adr = new float[taille];
57      for (i = 0; i < taille; i++)
58          adr[i] = lis[i];
59  }
60
61  int main()
62  {
63      cout << "Debutde main()\n";
64      liste a(5);
65      liste b(2);
66      a.saisie();
67      cout << "\n-------------------------------------\n";
68      a.affiche();
69      cout << "\n-------------------------------------\n";
70      b.saisie();
71      b.affiche();
72      cout << "\n-------------------------------------\n";
73      b = a;      You, 1 hour ago • tp5 …
74      b.affiche();
75      cout << "\n-------------------------------------\n";
76      a.affiche();
77      cout << "\n-------------------------------------\n";
78      cout << "Fin de main() \n";
79  }
80
```

```
Debutde main()
Entrer un nombre:2
Entrer un nombre:4
Entrer un nombre:5
Entrer un nombre:6
Entrer un nombre:8

-------------------------------------
Adresse:0x7ffdd82d15f0 2 4 5 6 8
-------------------------------------
Entrer un nombre:7
Entrer un nombre:9
Adresse:0x7ffdd82d1600 7 9
-------------------------------------
Adresse:0x7ffdd82d1600 2 4 5 6 8
-------------------------------------
Adresse:0x7ffdd82d15f0 2 4 5 6 8
-------------------------------------
Fin de main()
```

```cpp
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
using namespace std;
You, 1 hour ago | 1 author (You)
class chaine
{
    int l;
    char *adr;
public:
    chaine()
    {
        l = 0;
        adr = (char *)malloc(10 * sizeof(char));
        adr = NULL;
    }
    chaine(char *);
    char *getChaine();
    chaine(chaine &);
    chaine operator=(char *);
    bool operator==(char *);
    chaine operator+(char *);
    char operator[](int i);
    void afficher();
};
chaine::chaine(char *t)
{
    l = strlen(t);
    strcpy(adr, t);
}       You, 1 hour ago • tp5 …
chaine::chaine(chaine &a)
{
    l = a.l;
    for (int i = 0; i < l; i++)
        adr[i] = a[i];
}
chaine chaine::operator=(char *cp)
{
    chaine c;
    c.l = strlen(cp);
    strcpy(c.adr, cp);
    return c;
}
```

```
44  bool chaine::operator==(char *t)
45  {
46      if (strcmp(adr, t))
47          return true;
48      return false;
49  }
50  chaine chaine::operator+(char *b)
51  {
52      chaine c;
53      c.adr = strcat(this->adr, b);
54      c.l = strlen(b) + l;
55      return c;
56  }
57  char chaine::operator[](int i)
58  {
59      return adr[i];
60  }
61  void chaine::afficher()
62  {
63      cout << adr << endl;
64  }
65  char *chaine::getChaine()
66  {
67      return adr;
68  }
69  int main()
70  {
71      char *test = (char *)"hi";
72      char *test2 = (char *)"hello";
73      chaine a(test);
74      chaine b(test2);
75      if (a.operator==(b.getChaine()))
76          printf("oui \n");
77      else
78          printf("non \n");
79      chaine c;
80      c = a.operator+(b.getChaine());
81      c.afficher();
82      return 0;
83  }
84
```

---------- ----------tp----------

```cpp
// partie1
class Coordonee
{
private:
    int x, y;

public:
    Coordonee(int, int);
    void deplace(int, int);
    void affichage();
};
Coordonee::Coordonee(int a = 0, int b = 0)
{
    x = a;
    y = b;
}
void Coordonee::deplace(int a, int b)
{
    x += a;
    y += b;
}
void Coordonee::affichage()
{
    cout << "X = " << x << " Y = " << y << endl;
}
```

```cpp
// la class Forme
class Forme
{
protected:
    short couleur;

public:
    Forme(short);
    Forme(const Forme &);
    void affichage();
    Forme operator=(Forme &);
};
Forme::Forme(short c) : couleur(c) {}
Forme::Forme(const Forme &forme)
{
    cout << "Copy Constructor" << endl;
}
void Forme::affichage()
{
    cout << "Couleur : " << couleur << endl;
}
Forme Forme::operator=(Forme &forme)
{
    cout << "Operator =" << endl;
    forme.couleur = this->couleur;
    return forme;
}
```

```cpp
// partie3
class Triangle : public Forme
{
protected:
    Coordonee a, b, c;
public:
    Triangle(int, int, int, int, int, int, short);
    Triangle(Triangle &);
    Triangle operator=(Triangle &);
    void affichage();
    void deplace(int, int);
    float surface();
    float perimetre();
};
Triangle::Triangle(int d, int e, int f, int g, int h, int i, short couleur) : Forme(couleur)
{
    a = Coordonee(d, e);
    b = Coordonee(f, g);
    c = Coordonee(h, i);
}
Triangle::Triangle(Triangle &triangle) : Forme(triangle)
{
}
void Triangle::affichage()
{
    cout << "Affichage d'un Triangle" << endl;
    Forme::affichage();
    a.affichage();
    b.affichage();
    c.affichage();
}
void Triangle::deplace(int x, int y)
{
    a.deplace(x, y);
    b.deplace(x, y);
    c.deplace(x, y);
}
float Triangle::surface()
{
    return 1; // formule de surface d'un triangle
}
float Triangle::perimetre()
{
    return 2; // formule de perimetre d'un triangle
}
float Cercle::perimetre()
{
    return 2 * 3.14 * rayon;
}
```

```cpp
// partie4
class Rectangle : public Forme
{
protected:
    Coordonee a, b;

public:
    Rectangle(int, int, int, int, short);
    Rectangle(Rectangle &);
    Rectangle operator=(Rectangle &);
    void affichage();
    void deplace(int, int);
    float surface();
    float perimetre();
};
Rectangle::Rectangle(int d, int e, int f, int g, short couleur) : Forme(couleur)
{
    a = Coordonee(d, e);
    b = Coordonee(f, g);
}
Rectangle::Rectangle(Rectangle &rectangle) : Forme(rectangle)
{
}
void Rectangle::affichage()
{
    cout << "Affichage d'un Rectangle" << endl;
    Forme::affichage();
    a.affichage();
    b.affichage();
}
void Rectangle::deplace(int x, int y)
{
    a.deplace(x, y);
    b.deplace(x, y);
}
float Rectangle::surface()
{
    return 10; // formule de surface d'un rectangle
}
float Rectangle::perimetre()
{
    return 20; // formule de perimetre d'un rectangle
}
```

```cpp
// Partie5
class Carre : public Forme
{
protected:
    Coordonee a;
    short cote;

public:
    Carre(int, int, short, short);
    Carre(Carre &);
    Carre operator=(Carre &);
    void affichage();
    void deplace(int, int);
    float surface();
    float perimetre();
};
Carre::Carre(int x, int y, short c, short couleur) : Forme(couleur)
{
    a = Coordonee(x, y);
    cote = c;
}
Carre::Carre(Carre &carre) : Forme(carre)
{
}
void Carre::affichage()
{
    cout << "Affichage d'un Carre" << endl;
    Forme::affichage();
    a.affichage();
    cout << "Cote : " << cote << endl;
}
void Carre::deplace(int x, int y)
{
    a.deplace(x, y);
}
float Carre::surface()
{
    return cote * cote;
}
float Carre::perimetre()
{
    return 4 * cote;
}
```

main.cpp > main(int, char **)

```cpp
#include "classes.hpp"

int main(int argc, char **argv)
{
    Cercle cl(10, 20, 5, 12);
    cl.affichage();
    cl.deplace(5, 4);
    cl.affichage();
    Triangle t(10, 20, 30, 40, 50, 50, 11);
    t.affichage();
    t.deplace(4, 5);
    t.affichage();
    Rectangle r(10, 20, 30, 40, 50);
    r.affichage();
    r.deplace(4, 5);
    r.affichage();
    Carre cr(10, 20, 5, 10);
    cr.affichage();
    cr.deplace(4, 5);
    cr.affichage();
    return 0;
}
```

```
Affichage d'un Cercle
Couleur : 12
X = 10 Y = 20
Rayon : 5
Affichage d'un Cercle
Couleur : 12
X = 15 Y = 24
Rayon : 5
Affichage d'un Triangle
Couleur : 11
X = 10 Y = 20
X = 30 Y = 40
X = 50 Y = 50
Affichage d'un Triangle
Couleur : 11
X = 14 Y = 25
X = 34 Y = 45
X = 54 Y = 55
Affichage d'un Rectangle
Couleur : 50
X = 10 Y = 20
X = 30 Y = 40
Affichage d'un Rectangle
Couleur : 50
X = 14 Y = 25
X = 34 Y = 45
Affichage d'un Carre
Couleur : 10
X = 10 Y = 20
Cote : 5
Affichage d'un Carre
Couleur : 10
X = 14 Y = 25
Cote : 5
```