

24/04/2022

# Compte rendu

Des travaux pratique de la  
chapitre 5

**Surdefinition des operateurs**

PRÉPARÉE POUR  
TAFFAH ACHRAF

ENCADRÉ PAR  
PR. KHALIFA MANSOURI

# TABLE DES MATIERES

introduction .....	2
Partie pratique .....	3
1. Exemples de cour : .....	3
REMARQUE : .....	3
Exemple 1 : .....	3
Résultats : .....	4
la declaration de la classe liste : .....	4
la definition des methodes de la classe : .....	5
les fichier main pour tester les : .....	5
1. TP5 (Problème : Formes géométriques) : .....	6
la declaration de la classe <b>COORDONNE, FORME, CERCLE, TRIANGLE, RECTANGLE</b> et <b>carre</b> : .....	6
la definition des methodes des METHODES DES CLASSES DEFINIES : .....	7
les fichier main pour tester les METHODES DES CLASSES : .....	10
Résultats : .....	10
Conclusion .....	11

# INTRODUCTION

VOUS POUVEZ REDEFINIR OU SURCHARGER LA PLUPART DES OPERATEURS INTEGRES DISPONIBLES EN C++. AINSI, UN PROGRAMMEUR PEUT EGALEMENT UTILISER DES OPERATEURS AVEC DES TYPES DEFINIS PAR L'UTILISATEUR.

LES OPERATEURS SURCHARGES SONT DES FONCTIONS AVEC DES NOMS SPECIAUX : LE MOT CLE "**OPERATOR**" SUIVI DU SYMBOLE DE L'OPERATEUR EN COURS DE DEFINITION. COMME TOUTE AUTRE FONCTION, UN OPERATEUR SURCHARGE A UN TYPE DE RETOUR ET

# PARTIE PRATIQUE

## 1. Exemples de cour :

REMARQUE :

La surdéfinition des opérateurs est une technique qui nous permet de créer par le biais des classes, des types à part entière, c'est-à-dire des types munis, comme les types de base, d'opérateurs parfaitement intégrés.

## Le mécanisme de surdéfinition d'opérateurs

EXEMPLE 1 :

```
exemple1.cpp
1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 class vecteur{
5     float x,y;
6     public:
7         vecteur(float,float);
8         void affiche();
9         vecteur operator + (vecteur); // surdefinition de l'opérateur somme
10                                     // On passe un paramètre vecteur
11                                     // La fonction retourne un vecteur
12 };
13 vecteur::vecteur(float abs=0,float ord=0){
14     x=abs;
15     y=ord;
16 }
17 void vecteur::affiche(){
18     cout<<"x = "<<x<<" y = "<<y<<"\n";
19 }
20 vecteur vecteur::operator+(vecteur v){
21     vecteur res;
22     res.x=v.x+x;
23     res.y=v.y+y;
24     return res;
25 }
26 int main(){
27     vecteur a(2,6),b(4,8),c,d,e,f;
28     c=a+b;
29     c.affiche();
30     e=b.operator +(c);
31     e.affiche();
32     getch();
33     return 0;
34 }
```

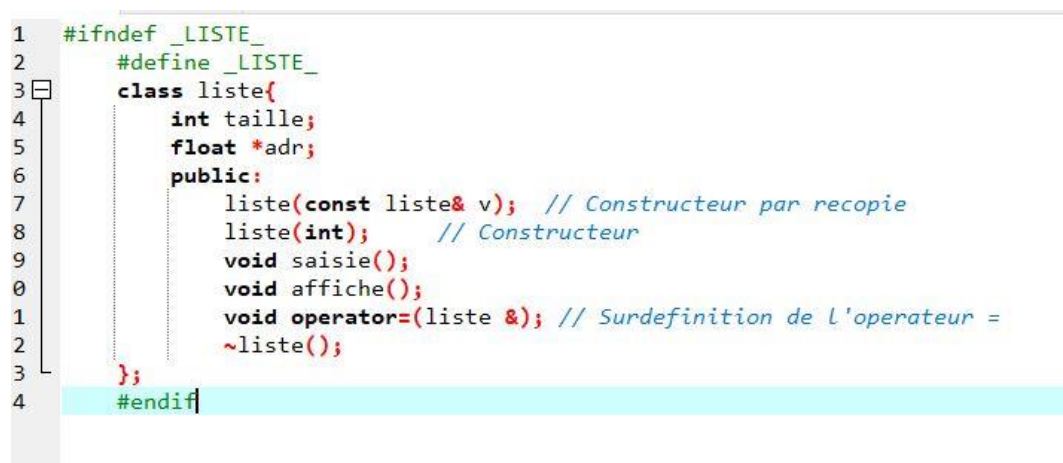
RESULTATS :



```
C:\Users\taffa\OneDrive\Bureau\C++\Achraf-TAFFAH-GLSID1-2022-CHAPITRE5\Cour && Exercices\Cour\exemple1.exe
x = 6 y = 14
x = 10 y = 22
```

## Exercice 1 :

LA DECLARATION DE LA CLASSE LISTE :



```
1  #ifndef _LISTE_
2  #define _LISTE_
3  class liste{
4      int taille;
5      float *adr;
6  public:
7      liste(const liste& v); // Constructeur par recopie
8      liste(int);           // Constructeur
9      void saisie();
10     void affiche();
11     void operator=(liste &); // Surdefinition de l'opérateur =
12     ~liste();
13 };
14 #endif
```

## LA DEFINITION DES METHODES DE LA CLASSE :

```
1  #include "liste.hpp"
2  #include <iostream>
3  #include <conio.h>
4  using namespace std;
5  liste::liste(int t){
6      taille=t;
7      adr=new float(taille);
8      cout<<"Construction";
9      cout<<" Adresse de l'objet:"<<this;
10     cout<<" Adresse de liste:"<<adr<<"\n";
11 }
12 liste::~liste(){
13     cout<<"Destruction Adresse de l'objet:"<<this;
14     cout<<" Adresse de liste:"<<adr<<"\n";
15     delete adr;
16 }
17 liste::liste(const liste& v){
18     taille=v.taille;
19     adr=new float[taille];
20     for(int i=0;i<taille;i++){
21         adr[i]=v.adr[i];
22     }
23     cout<<"\nConstructeur par recopie";
24     cout<<" Adresse de l'objet:"<<this;
25     cout<<" Adresse de liste:"<<adr<<"\n";
26 }
27 void liste::saisie(){
28     int i;
29     for(i=0;i<taille;i++){
30         cout<<"Entrer un nombre:";
31         cin>>*(adr+i);
32     }
33 }
34 void liste::affiche(){
35     int i;
36     cout<<"Adresse:"<<this<<" ";
37     for(i=0;i<taille;i++){
38         cout<<*(adr+i)<<" ";
39     }
40     cout<<"\n\n";
41 }
42 void liste::operator=(liste &lis){ // passage par reference pour
43     int i; // Eviter l'appel au constructeur par recopie
44     taille=lis.taille; // Et la double libération d'un meme emplacement memoire
45     delete adr;
46     adr=new float[taille];
47     for(i=0;i<taille;i++){
48         adr[i]=lis.adr[i];
49     }
50 }
```

## LES FICHER MAIN POUR TESTER LES :

```
48 int main(){
49     cout<<"Debut de main ()\n";
50     liste a(5);
51     liste b(2);
52     a.saisie();
53     a.affiche();
54     b=a;
55     b.affiche();
56     a.affiche();
57     cout<<"Fin de main() \n";
58     return 0;
59 }
```



## 1. TP5 (Problème : Formes géométriques) :

LA DECLARATION DE LA CLASSE COORDONNE, FORME, CERCLE, TRIANGLE, RECTANGLE ET CARRE :

```
main.cpp classes.hpp classes.cpp
1  #ifndef _CLASSES_
2  #define _CLASSES_
3  #include <math.h>
4  #define PI 3.14159265359
5  class coordonne{
6      int x,y;
7      public:
8          coordonne(int a=0,int b = 0):x(a),y(b){}
9          void deplacer(int,int);
10         void afficher();
11         int get_x() const;
12         int get_y() const;
13         friend distance(coordonne const & a,coordonne const & b){
14             return sqrt(((a.x-b.x)*(a.x-b.x))+((a.y-b.y)*(a.y-b.y)));
15         }
16     };
17     class forme{
18     protected:
19         short couleur;
20     public:
21         forme(short s = 1):couleur(s){}
22         forme(forme & f){
23             this->couleur = f.couleur;
24         }
25         void affiche();
26         forme operator=(forme &f);
27     };
28     class cercle : public forme{
41
42     class triangle: public forme{
54
55     class rectangle: public forme{
```

```
28     class cercle : public forme{
29     protected:
30         coordonne centre;
31         short rayon;
32     public:
33         cercle(int x,int y,short rayon,short couleur):forme(couleur),centre(x,y),rayon(rayon){}
34         cercle(cercle & c):forme(c.couleur),centre(c.centre),rayon(c.rayon){}
35         void affiche();
36         void deplacer(int x,int y);
37         float surface() const;
38         float perimetre() const;
39         cercle operator=(cercle &f);
40     };
41
42     class triangle: public forme{
43     protected:
44         coordonne a,b,c;
45     public:
46         triangle(int a_x,int a_y,int b_x,int b_y,int c_x,int c_y,short couleur):
47             forme(couleur),a(a_x,a_y),b(b_x,b_y),c(c_x,c_y){}
48         triangle(triangle & t):forme(t.couleur),a(t.a),b(t.b),c(t.c){}
49         triangle operator =(triangle & t);
50         void affiche();
51         void deplacer(int x,int y);
52         float surface() const;
53         float perimetre() const;
54     };
55
```

Activer Windows

```

5 class rectangle: public forme{
7     protected:
8         coordonne a,b;
9     public:
10        rectangle(int a_x,int a_y,int b_x,int b_y,short couleur):forme(couleur),a(a_x,a_y),b(b_x,b_y){}
11        rectangle(rectangle & r):forme(r.couleur),a(r.a),b(r.b){}
12        rectangle operator =(rectangle & t);
13        void affiche();
14        void deplacer(int x,int y);
15        float surface() const;
16        float perimetre() const;
17    };
18
19 class carre: public forme{
20     protected:
21         coordonne a;
22         short cote;
23     public:
24        carre(int a_x,int a_y,short cote,short couleur):forme(couleur),cote(cote),a(a_x,a_y){}
25        carre(carre & c):forme(c.couleur),a(c.a){}
26        void affiche();
27        void deplacer(int x,int y);
28        float surface() const;
29        float perimetre() const;
30    };
31 #endif

```

Activer Windows  
Accédez aux paramètres pour ac

## LA DEFINITION DES METHODES DES METHODES DES CLASSES DEFINIES :

```

1  #include "classes.hpp"
2  //Classe coordonne
3  void coordonne::deplacer(int a,int b){
4      x+=a;
5      y+=b;
6  }
7  void coordonne::afficher(){
8      cout <<"x = "<<x<<" y = "<<y<<endl;
9  }
10 int coordonne::get_x() const{
11     return this->x;
12 }
13 int coordonne::get_y() const{
14     return this->y;
15 }
16 //Classe forme
17 void forme::affiche(){
18     cout <<"couleur : "<< couleur << endl;
19 }
20 forme forme::operator=(forme &f){
21     forme nf(f.couleur);
22     return nf;
23 }
24 //Classe cercle
25 void cercle::affiche(){
26     cout << "cercle"<<endl;
27     centre.afficher();
28     forme::affiche();
29     cout << "rayon : "<< rayon << endl;
30 }

```

Activer Wi

```

31 void cercle::deplacer(int x,int y){
32     centre.deplacer(x,y);
33 }
34 float cercle::surface() const{ return (PI*rayon*rayon); }
35 float cercle::perimetre() const{ return (2*PI*rayon) ; }
36 cercle cercle::operator=(cercle &f){
37     cercle c(f);
38     return c;
39 }
40 //Classe triangle
41 triangle triangle::operator =(triangle & t){
42     triangle ts(t);
43     return ts;
44 }
45 void triangle::affiche(){
46     cout << "triangle"<<endl;
47     a.afficher();
48     b.afficher();
49     c.afficher();
50     forme::affiche();
51 }
52 void triangle::deplacer(int x,int y){
53     a.deplacer(x,y);
54     b.deplacer(x,y);
55     c.deplacer(x,y);
56 }

```

```

45 void triangle::affiche(){
46     cout << "triangle"<<endl;
47     a.afficher();
48     b.afficher();
49     c.afficher();
50     forme::affiche();
51 }
52 void triangle::deplacer(int x,int y){
53     a.deplacer(x,y);
54     b.deplacer(x,y);
55     c.deplacer(x,y);
56 }
57 float triangle::surface() const{
58     float ab = distance( this->a , this->b );
59     float ac = distance( this->a , this->c );
60     float bc = distance( this->b , this->c );
61     float h = 0.5 * sqrt(((ab + bc + ac) * (-ab + bc + ac) * (ab - bc + ac) * (ab + bc - ac)) / bc);
62     return (bc * h) / 2;
63 }
64 float triangle::perimetre() const{
65     float ab = distance( this->a , this->b );
66     float ac = distance( this->a , this->c );
67     float bc = distance( this->b , this->c );
68     return ( ab + ac + bc );
69 }
70 // Classe rectangle
71 rectangle rectangle::operator =(rectangle & t){
72     rectangle ts(t);
73     return ts;
74 }

```

Activer Windows



```

75 void rectangle::affiche(){
76     cout << "rectangle"<<endl;
77     a.afficher();
78     b.afficher();
79     forme::affiche();
80 }
81 void rectangle::deplacer(int x,int y){
82     a.deplacer(x,y);
83     b.deplacer(x,y);
84 }
85 float rectangle::surface() const{
86     float w = abs(a.get_x()-b.get_x());
87     float l = abs(a.get_y()-b.get_y());
88     return (w*l);
89 }
90 float rectangle::perimetre() const{
91     float w = abs(a.get_x()-b.get_x());
92     float l = abs(a.get_y()-b.get_y());
93     return (w+l)*2;
94 }
95 //Classe carre
96 void carre::affiche(){
97     cout << "carre"<<endl;
98     a.afficher();
99     cout <<"cote : "<<cote<<endl;
100     forme::affiche();
101 }
102 void carre::deplacer(int x,int y){
103     a.deplacer(x,y);
104 }
105 float carre::surface() const{
106     return (cote*cote);
107 }
108 float carre::perimetre() const{
109     return cote*4;
110 }

```

LES FICHER MAIN POUR TESTER LES METHODES DES CLASSES :

```
[*] main.cpp  classes.hpp  classes.cpp
1  #include <iostream>
2  #include <conio.h>
3  #include <math.h>
4  #include<string>
5  using namespace std;
6  #include"classes.cpp"
7
8  int main(){
9      cercle c1(14,0,19,1001);
10     c1.affiche();
11     c1.deplacer(5,4);
12     c1.affiche();
13     getch();
14
15     triangle t(17,97,4,897,50,60,4);
16     t.affiche();
17     t.deplacer(9,-4);
18     t.affiche();
19     getch();
20
21     return 0;
22 }
```

RESULTATS :

```
C:\Users\taffa\OneDrive\Bureau\C++\Achraf-TAFFAH-GLSID1-2022-CHAPITRE5\Cour && Exercices\Tp\main.exe
cercle
x = 14 y = 0
couleur : 1001
rayon : 19
cercle
x = 19 y = 4
couleur : 1001
rayon : 19
```

# CONCLUSION

ON ENTEND PAR SURCHARGE DES OPERATEURS LA REDEFINITION DE CES OPERATEURS, POUR LES ADAPTES A UNE UTILISATION AVEC DES TYPES UTILISATEURS, DIFFERENTS DES TYPES SUR LESQUELS ILS TRAVAILLAIENT DANS LEURS VERSIONS D'ORIGINES. LES TYPES UTILISATEURS CONCERNENT NOTAMMENT LES CLASS ET LES ENUMERATION.