

```

pointCol.cpp pointCol.h couleur.cpp couleur.h x point.cpp triangle.h
(Global Scope)
#pragma once
#include<iostream>
using namespace std;
#include<string>
class couleur
{
protected:
    int coul;
public:
    couleur(int =11);
    virtual ~couleur(void);
    virtual void afficher(string = "");
};
  
```

```

pointCol.cpp pointCol.h couleur.cpp couleur.h point.cpp triangle.h rectangle.h
(Global Scope)
#include "couleur.h"
couleur::couleur(int c): coul(c)
{
    cout<<"\n +++ constr couleur "<<this<<endl;
}
couleur::~couleur(void)
{
    cout<<"\n --- destr couleur "<<this<<endl;
}
void couleur::afficher(string msg)
{
    cout<<msg<<endl;
    cout<<"\n la couleur est "<<coul<<endl;
}
  
```

```

class point
{
protected:
    int x;
    int y;
public:
    point(int =99,int =88);
    virtual void afficher(string = "");
    virtual ~point();
    virtual void saisir();
};
  
```

```

point.cpp courbe.cpp segment.cpp main.cpp pointColMasse.h pointMasse.h pointColoreM
point
#include "point.h"
point::point(int abs,int ord): x(abs), y(ord)
{
    cout<<"\n +++ construc point "<<this<<endl;
}
void point::afficher(string msg)
{
    cout<<msg<<" ";
    cout<<this<<" coordonnees "<<x<<" "<<y<<endl;
}
void point::saisir()
{
    cout<<"\n saisir x et y "<<endl;
    cin>>x;
    cin>>y;
}
point::~point()
{
    cout<<"\n --- destruc point"<<this<<endl;
}
  
```

```

masse.h
(Global Scope)
#pragma once
#include<iostream>
using namespace std;
#include<string>
class masse
{
protected:
    int masses;
public:
    masse(int =33);
    virtual ~masse(void);
    virtual void afficher(string = "");
};

```

```

masse.cpp
(Global Scope)
#include "masse.h"
masse::masse(int m): masses(m)
{
    cout<<"\n +++ constr masse "<<this<<endl;
}
masse::~masse(void)
{
    cout<<"\n --- destr masse "<<this<<endl;
}
void masse::afficher(string msg)
{
    cout<<msg<<endl;
    cout<<"\n la masse est "<<masses<<endl;
}

```

```

pointCol.h
(Global Scope)
#pragma once
#include "couleur.h"
#include "point.h"
class pointCol: public couleur, virtual public point
{
public:
    pointCol(int =22,int=22,int=22);
    pointCol(couleur, point);
    ~pointCol(void);
    void afficher(string="");
};

```

```

pointCol.cpp
(Global Scope)
pointCol::pointCol(couleur c, point pt):
    couleur(c), point(pt)
{
}

```

```

pointCol.cpp
pointCol
#include "pointCol.h"
pointCol::pointCol(int abs, int ord, int c):
    point(abs,ord), couleur(c)
{
    // appel du constructeur de la classe point
    // appel du constructeur de la classe couleur
    cout<<"\n +++ constr pointCol "<<this<<endl;
}
pointCol::~pointCol(void)
{
    cout<<"\n --- destr pointCol "<<this<<endl;
}
void pointCol::afficher(string msg)
{
    cout<<msg<<endl;
    point::afficher();
    couleur::afficher();
}

```

```

pointMas.cpp
pointMas
#pragma once
#include "point.h"
#include "masse.h"
class pointMas : virtual public point, public masse
{
public:
    pointMas(int =44,int=44,int=44);
    pointMas(point, masse);
    ~pointMas(void);
    void afficher(string="");
};

```

```

pointMas::pointMas(point pt,masse m):
    point(pt), masse(m)
{
}

```

```

pointMasse.cpp
pointMasse
#include "pointMasse.h"
pointMasse::pointMasse(int abs, int ord, int m):
    point(abs,ord), masse(m)
{
    cout<<"\n +++ constr pointMasse "<<this<<endl;
}
pointMasse::~pointMasse(void)
{
    cout<<"\n --- destr pointMasse "<<this<<endl;
}
void pointMasse::afficher(string msg)
{
    cout<<msg<<endl;
    point::afficher();
    masse::afficher();
}

```

```

#pragma once
#include "pointCol.h"
#include "pointMasse.h"
class pointColMasse: public pointCol, public pointMasse
{
public:
    pointColMasse(int =66, int =66, int =66, int =66);
    ~pointColMasse(void);
    void afficher(string = "");
};

```

```

pointColMasse.cpp
pointColMasse
#include "pointColMasse.h"
pointColMasse::pointColMasse(int abs, int ord, int c, int m):
    pointCol(abs,ord,c), pointMasse(abs,ord,m), point(abs,ord)
{
    cout<<"\n +++ constr pointColMasse "<<this<<endl;
}
pointColMasse::~pointColMasse(void)
{
    cout<<"\n --- destr pointColMasse "<<this<<endl;
}
void pointColMasse::afficher(string msg)
{
    cout<<msg<<endl;
    couleur::afficher();
    point::afficher();
    masse::afficher();
}

```

Quelques remarques:

Remarque 1:

Si la classe point est déclarée **non virtuelle** → création de 2 points

```
pointMasse.h pointColMasse.cpp pointMasse.cpp pointCol.cpp pointCol.h masse
(Global Scope)
#pragma once
#include "point.h"
#include "couleur.h"
class pointCol: public couleur, public point
{
public:
    pointCol(int =33, int =33, int =33);
    ~pointCol(void);
    virtual void afficher(string = "");
};
```

```
pointMasse.h pointColMasse.cpp pointMasse.cpp pointCol.cpp pointCol.h ma
(Global Scope)
#pragma once
#include "point.h"
#include "masse.h"
class pointMasse: public point, public masse
{
public:
    pointMasse(int =55, int =55, int =55);
    ~pointMasse(void);
    void afficher(string = "");
};
```

```
pointMasse.h pointColMasse.cpp pointMasse.cpp pointCol.cpp pointCol.h masse
(Global Scope)
pointColMasse::~pointColMasse(void)
{
    cout<<"\n --- destr pointColMasse "<<this<<endl;
}
void pointColMasse::afficher(string msg)
{
    cout<<msg<<endl;
    pointCol::afficher();
    pointMasse::afficher();
}
```

```
void main()
{
    pointColMasse a;
    cout<<"\n-----"<<endl;
    a.afficher();
    system("PAUSE");
}
```

```
#include "pointColMasse.h"
pointColMasse::pointColMasse(int abs, int ord, int c, int m):
    pointCol(abs,ord,c), pointMasse(abs,ord,m)
{
    cout<<"\n +++ constr pointColMasse "<<this<<endl;
}
```

Pas d'appel au constr de point

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingC2014\Rel...
+++ constr couleur 0037FE40
+++ construc point 0037FE48
+++ constr pointCol 0037FE40
+++ construc point 0037FE54
+++ constr masse 0037FE60
+++ constr pointMasse 0037FE54
+++ constr pointColMasse 0037FE40
-----
0037FE48 coordonnees 66 66

la couleur est 66
0037FE54 coordonnees 66 66

la masse est 66
Appuyez sur une touche pour continuer...
```

Pour l'objet pointColMasse, il y a eu création de 2 objets points se trouvant sur des adresses différentes → duplication des données (x et y)

Remarque 2:

1. La classe point est virtuelle → un seul point créé
2. s'il n'y a pas appel du constructeur de point dans le constructeur de pointColMasse → appel du constructeur de point sans arguments.

```
class pointCol: public couleur, virtual public point
{
public:
    pointCol(int =33, int =33, int =33);
    ~pointCol(void);
    virtual void afficher(string = "");
};
```

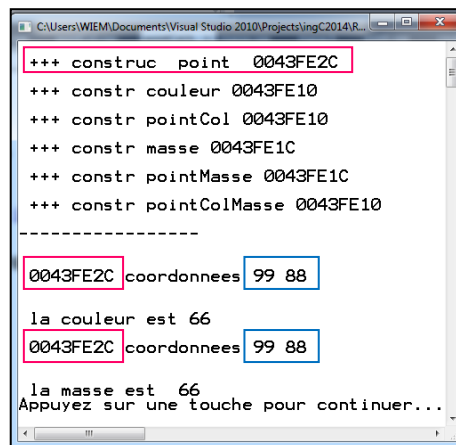
```
class pointMasse: virtual public point, public masse
{
public:
    pointMasse(int =55, int =55, int =55);
    ~pointMasse(void);
    void afficher(string = "");
};
```

```
#include "pointColMasse.h"
pointColMasse::pointColMasse(int abs, int ord, int c, int m):
    pointCol(abs,ord,c), pointMasse(abs,ord,m)
{
    cout<<"\n +++ constr pointColMasse "<<this<<endl;
}
```

```
pointMasse.h pointColMasse.cpp pointMasse.cpp pointCol.cpp pointCol.h masse
(Global Scope)
pointColMasse::~pointColMasse(void)
{
    cout<<"\n --- destr pointColMasse "<<this<<endl;
}
void pointColMasse::afficher(string msg)
{
    cout<<msg<<endl;
    pointCol::afficher();
    pointMasse::afficher();
}
```

```
#pragma once
#include "pointCol.h"
#include "pointMasse.h"
class pointColMasse: public pointCol, public pointMasse
{
public:
    pointColMasse(int =66, int =66, int =66, int =66);
    ~pointColMasse(void);
    void afficher(string = "");
};
```

```
void main()
{
    pointColMasse a;
    cout<<"\n-----"<<endl;
    a.afficher();
    system("PAUSE");
}
```



```
+++ construc point 0043FE2C
+++ constr couleur 0043FE10
+++ constr pointCol 0043FE10
+++ constr masse 0043FE1C
+++ constr pointMasse 0043FE1C
+++ constr pointColMasse 0043FE10
-----
0043FE2C coordonnees 99 88
la couleur est 66
0043FE2C coordonnees 99 88
la masse est 66
Appuyez sur une touche pour continuer...
```

Pour l'objet pointColMasse, il y a eu création d'un seul point → pas de duplication des données