



P00 – Langage C++

Les classes et les objets

(Parties 4 et 5)

1^{ère} année ingénieur informatique

Mme Wiem Yaiche Elleuch

2017 - 2018

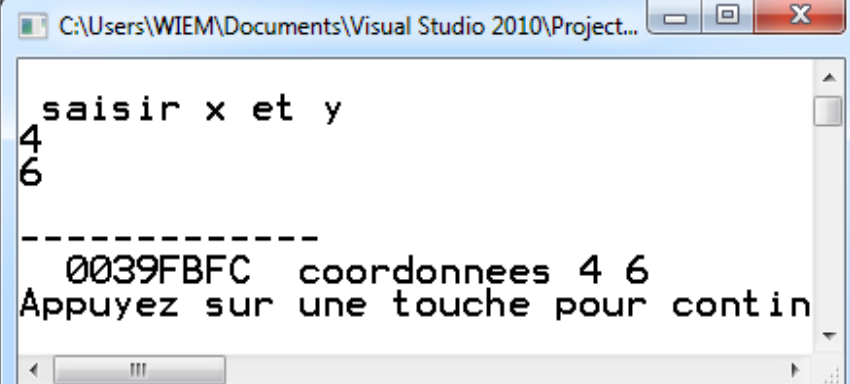
plan

1. Les structures en C++
2. Notion de classe
3. Affectation d'objets
4. Notion de constructeur et de destructeur
5. Les membres données et fonctions statiques
6. Protection contre les inclusions multiples
7. Surdéfinition des fonctions membres
8. Arguments par défaut des fonctions membres
9. Cas des objets transmis en argument d'une fonction (par valeur, par adresse, par référence)
10. Objet retourné par une fonction
11. Autoréférence: le mot clé this
12. Constructeur de copie
13. Objets membres
14. Tableau d'objets

Remarque

```
class point
{
private:
    int x;
    int y;
public:
    point(int =99,int =88);
    void deplacer(int,int);
    void afficher(string = "");
    bool coincide (point);
    point symetrique();
    void setX(int abs){x=abs;}
    void setY(int ord){y=ord;}
    int getX(){return x;}
    int getY(){return y;}
    ~point();
    void saisir_point();
};
```

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



```
saisir x et y
4
6
-----
0039FBFC coordonnees 4 6
Appuyez sur une touche pour contin
```

point.h point.cpp* courbe.h courbe.cpp main.cpp

(Global Scope)

```
void point::saisir_point()
{
    cout<<"\n saisir x et y "<<endl;
    cin>>x;
    cin>>y;
}
```

Tableau d'objets

La classe point doit avoir, soit:

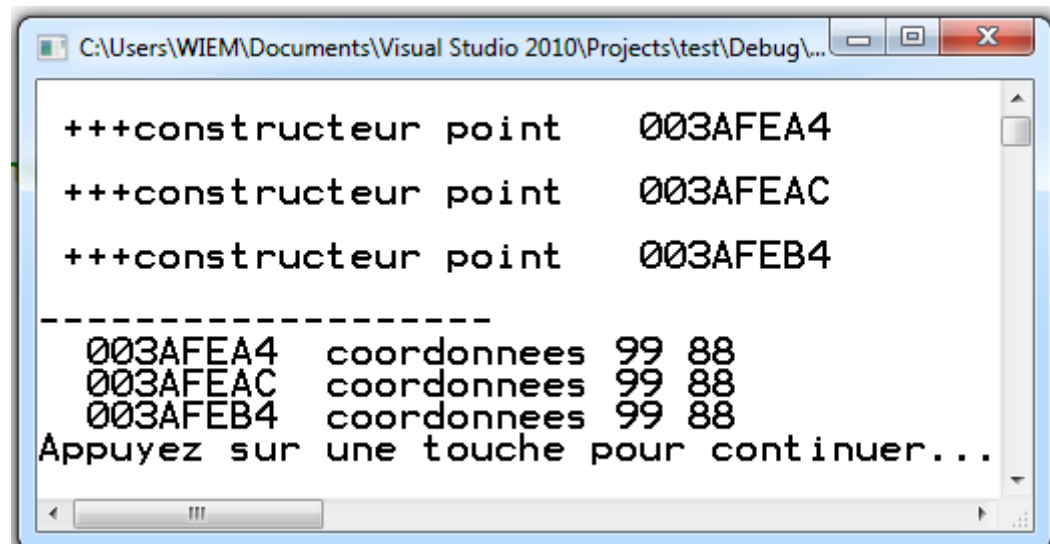
- Un constructeur avec 0 arguments
- un constructeur par défaut

tab est un tableau contenant 3 objets (3 points)

tab n'est pas objet

```
void main()
{
    point tab[3];
    // tab est un tableau d'objets
    cout<<"\n-----"<<endl;
    for(int i=0; i<3; i++)
        tab[i].afficher();

    system("PAUSE");
}
```



```
+++constructeur point 003AFE4
+++constructeur point 003AFEAC
+++constructeur point 003AFEB4
-----
003AFE4 coordonnees 99 88
003AFEAC coordonnees 99 88
003AFEB4 coordonnees 99 88
Appuyez sur une touche pour continuer...
```

Tableau d'objets

- En C++, un tableau peut posséder des éléments de n'importe quel type, y compris de type classe, ce qui conduit alors à un **tableau d'objets**.
- il reste toujours possible de définir une classe dont un des membres est un tableau d'objets.

3 schémas

// schéma 1

```
class courbe
{ point tab[3];
//méthodes
};
```

Objet
courbe

0	0	1	2	2	4
x	y	x	y	x	y

tab

// schéma 2

```
class courbe
{ int nb_points;
  point *tab;
};
```

Objet
courbe

3	F800
nb_points	tab

0	0	1	2	2	4
x	y	x	y	x	y

F800 F810 F820

Point d'indice i: tab[i] => tab[i].afficher();

*(tab+i).afficher(); // (tab+i)->afficher();

// schéma 3

```
class courbe
{ int nb_points;
  point **tab;
};
```

Objet
courbe

2	F700
nb_points	tab

F800	F500
------	------

F700

F710

1	2
x	y

F800

2	4
x	y

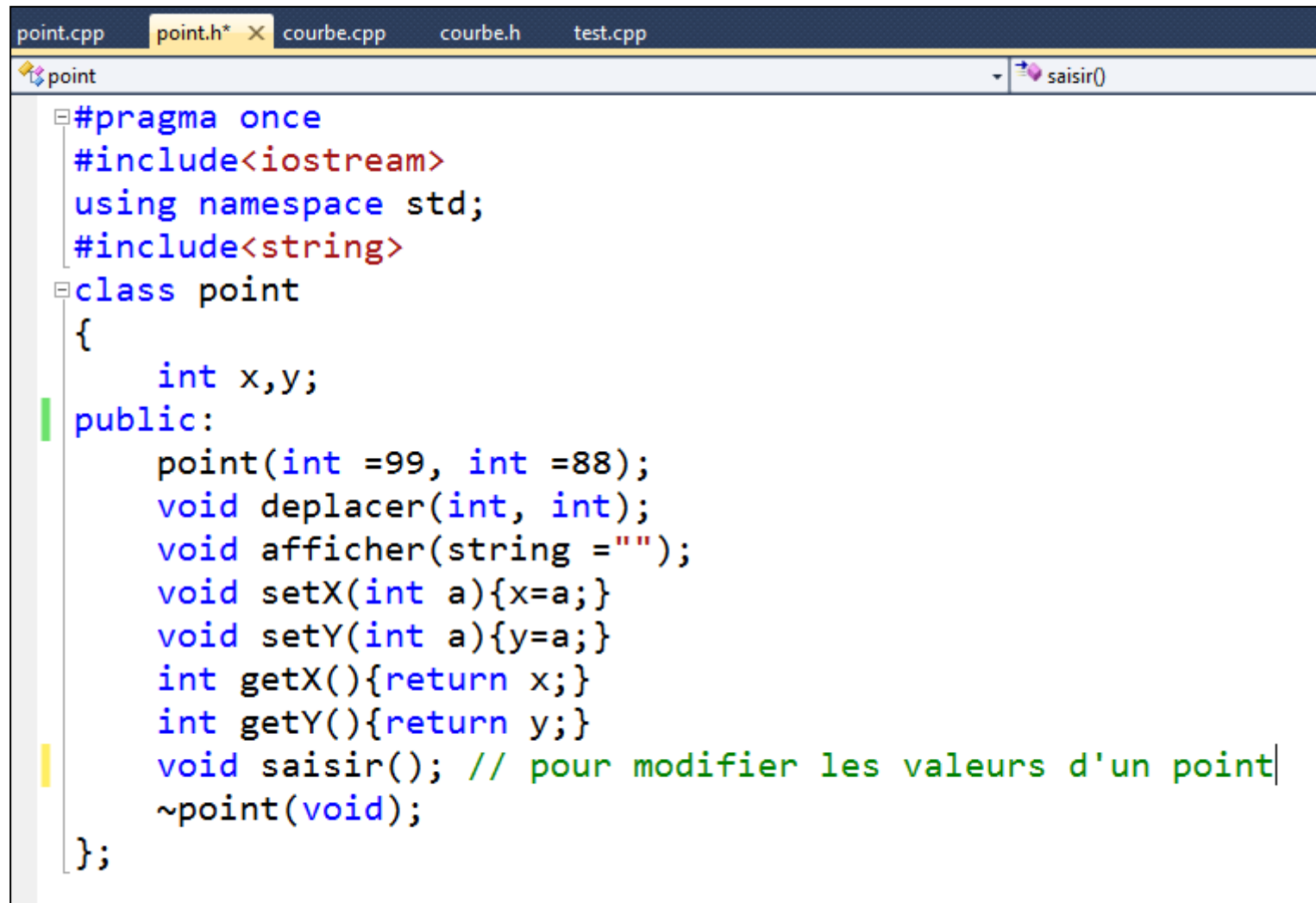
F500

tab[i]

*tab[i]

Point d'indice i: *tab[i] → (*tab[i]).afficher(); → tab[i]->afficher();

Rappel: classe point



```
point.cpp  point.h*  courbe.cpp  courbe.h  test.cpp
point
saisir()

#pragma once
#include<iostream>
using namespace std;
#include<string>
class point
{
    int x,y;
public:
    point(int =99, int =88);
    void deplacer(int, int);
    void afficher(string = "");
    void setX(int a){x=a;}
    void setY(int a){y=a;}
    int getX(){return x;}
    int getY(){return y;}
    void saisir(); // pour modifier les valeurs d'un point
    ~point(void);
};
```

Rappel: classe point

```
point.cpp* X point.h courbe.cpp courbe.h test.cpp
point saisir()

#include "point.h"
point::point(int x, int y)
{
    cout<<"\n +++ appel constr point +++"<<this<<endl;
    this->x=x;  this->y=y;
}
void point::deplacer(int dx, int dy)
{
    x=dx;  y=dy;
}
void point::afficher(string msg)
{
    cout<<msg<<"    "<<this<<" x:"<<x<<" y:"<<y<<"    "<<endl;
}
point::~~point(void)
{
    cout<<"\n--- appel destr point ---"<<this<<endl;
}
void point::saisir()
{
    cout<<"\n saisir x et y "<<endl;
    cin>>x;
    cin>>y;
}
```


Schéma 1

```
class courbe
{
    point tab[3];
public:
    courbe();
    void afficher(string = "");
    ~courbe(void);
};
```

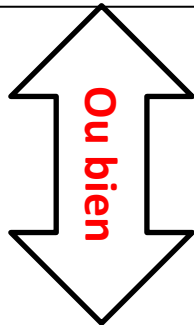
Objet
courbe

0	0	1	2	2	4
x	y	x	y	x	y
tab					

Point d'indice i: `tab[i] => tab[i].afficher();`

Schéma 1

```
▢ courbe::courbe()  
{  
    for(int i=0; i<3; i++)  
        tab[i].saisir();  
}
```



```
▢ void courbe::afficher(string msg)  
{  
    cout<<msg<<"  ";  
    for(int i=0; i<3; i++)  
        tab[i].afficher();  
}
```

```
▢ courbe::~~courbe(void)  
{  
    cout<<"\n --- destruct courbe "<<endl;  
}
```

```
▢ courbe::courbe()  
{  
    int abs, ord;  
    for(int i=0; i<3; i++)  
    {  
        cout<<"\n saisir x et y "<<endl;  
        cin>>abs;  
        cin>>ord;  
        tab[i]=point(abs,ord);  
    }  
}
```

Schéma 2

```
class courbe
{
    int nb_points;
    point *tab;
public:
    courbe(int =2);
    void afficher(string = "");
    ~courbe(void);
    courbe(const courbe&);
};
```

Objet
courbe

3	F800
nb_points	tab

0	0	1	2	2	4
x	y	x	y	x	y

F800 F810 F820

Point d'indice i: `tab[i] => tab[i].afficher();`

Schéma 2

```
▢ courbe::courbe(int nbPoint)
{
    this->nbPoint=nbPoint;
    tab=new point[nbPoint];
    for(int i=0; i<nbPoint; i++)
        tab[i].saisir();
}
```

```
▢ courbe::courbe(int n)
{
    nb_points=n;
    tab=new point[n];
    int abs, ord;
    for(int i=0; i<n; i++)
    {
        cout<<"\n saisir x et y "<<endl;
        cin>>abs;
        cin>>ord;
        tab[i]=point(abs,ord);
    }
}
```

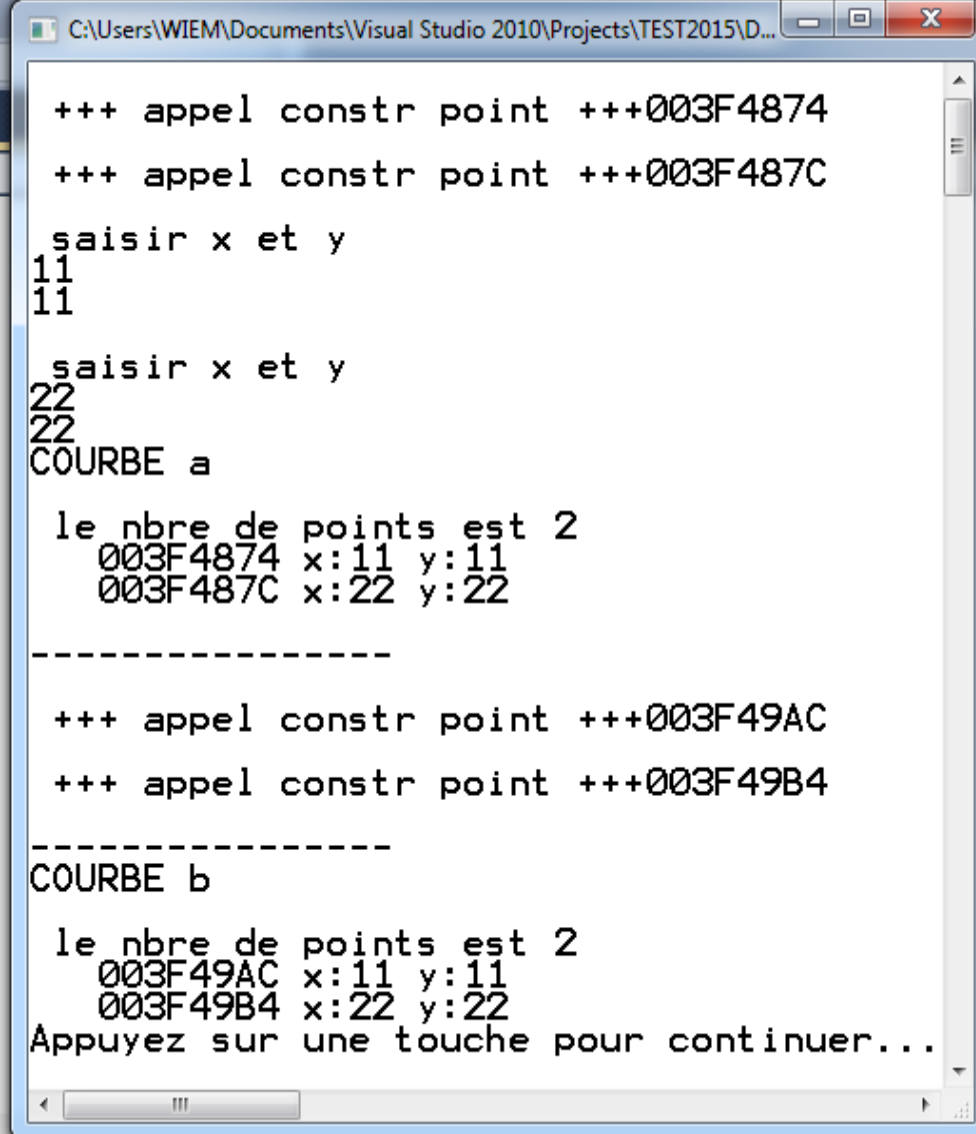
```
▢ courbe::courbe(const courbe &w)
{
    nb_points=w.nb_points;
    tab=new point[nb_points];
    for(int i=0; i<nb_points; i++)
        tab[i]=w.tab[i];
}
```

```
▢ courbe::~~courbe(void)
{
    cout<<"\n --- destruct courbe "<<endl;
    delete[]tab;
}
```

```
▢ void courbe::afficher(string msg)
{
    cout<<msg<<" ";
    cout<<"\n le nombre de points "<<nb_points<<endl;
    for(int i=0; i<nb_points; i++)
        tab[i].afficher();
}
```

Test du constr de recopie (schéma 2)

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



```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\TEST2015\D...
+++ appel constr point +++003F4874
+++ appel constr point +++003F487C
saisir x et y
11
11
saisir x et y
22
22
COURBE a

le nbre de points est 2
003F4874 x:11 y:11
003F487C x:22 y:22

-----

+++ appel constr point +++003F49AC
+++ appel constr point +++003F49B4

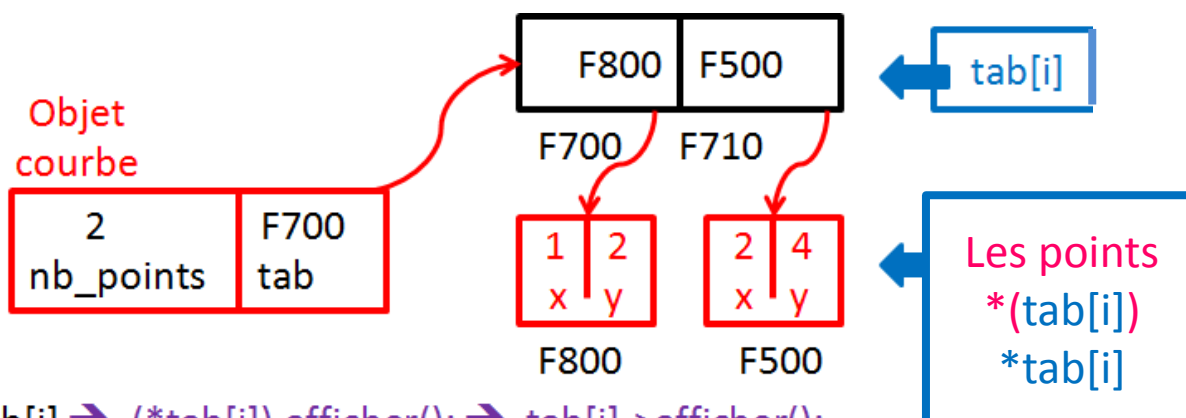
-----
COURBE b

le nbre de points est 2
003F49AC x:11 y:11
003F49B4 x:22 y:22
Appuyez sur une touche pour continuer...
```

Schéma 3

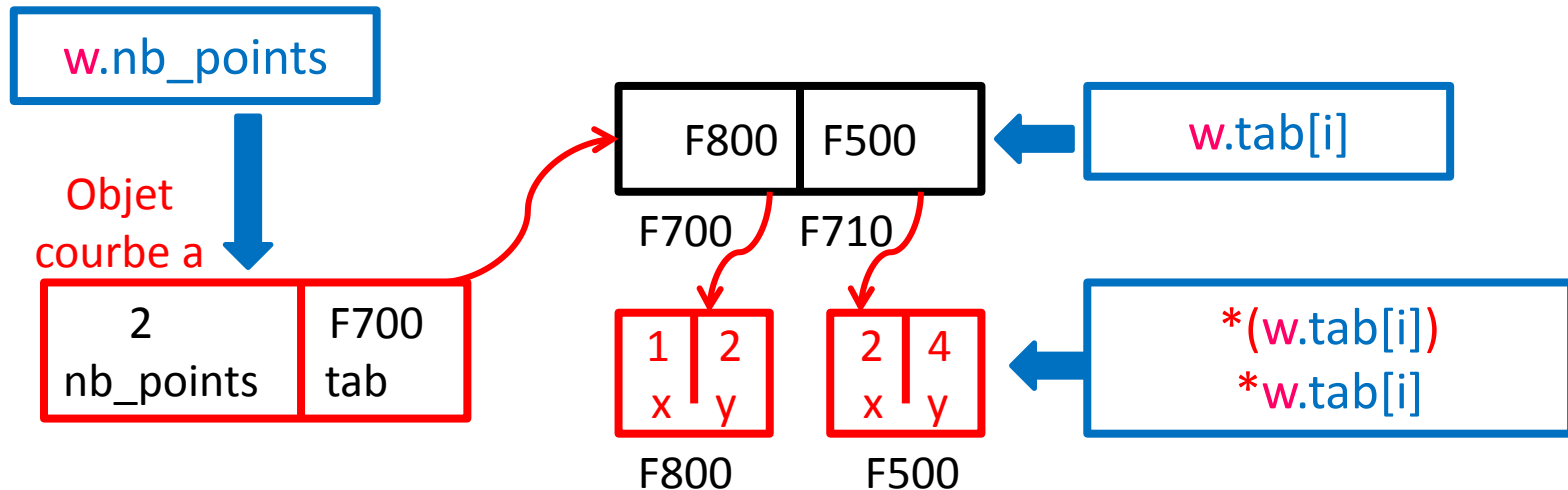
```

courbe.h  courbe.cpp  main.cpp*
(Global Scope)
class courbe
{
    int nb_points;
    point **tab;
public:
    courbe(int =2);
    void afficher(string = "");
    ~courbe(void);
    courbe(const courbe&);
};
    
```



Point d'indice i: $*tab[i] \rightarrow (*tab[i]).afficher(); \rightarrow tab[i] \rightarrow afficher();$

courbe::courbe(const courbe &w)



```
 courbe::courbe(const courbe &w)
{
    nb_points=w.nb_points;
    tab=new point*[nb_points];
    for(int i=0; i<nb_points; i++)
        tab[i]=new point(*w.tab[i]);
}
```

Schéma 3

```
▢ courbe::courbe(int nbPoint)
{
    this->nbPoint=nbPoint;
    tab=new point*[nbPoint];
    for(int i=0; i<nbPoint; i++)
    {
        tab[i]=new point();
        tab[i]->saisir();
    }
}
```

```
▢ courbe::~~courbe(void)
{
    cout<<"\n --- destruct courbe "<<endl;
    for(int i=0; i<nb_points; i++)
        delete tab[i];
    delete[]tab;
}
```

```
▢ void courbe::afficher(string msg)
{
    cout<<msg<<" ";
    cout<<"\n le nombre de points "<<nb_points<<endl;
    for(int i=0; i<nb_points; i++)
        tab[i]->afficher();
}
```

```
▢ courbe::courbe(int n)
{
    nb_points=n;
    tab=new point*[n];
    int abs, ord;
    for(int i=0; i<n; i++)
    {
        cout<<"\n saisir x et y "<<endl;
        cin>>abs;
        cin>>ord;
        tab[i]=new point(abs,ord);
    }
}
```

```
▢ courbe::courbe(const courbe &w)
{
    nb_points=w.nb_points;
    tab=new point*[nb_points];
    for(int i=0; i<nb_points; i++)
        tab[i]=new point(*w.tab[i]);
}
```


Test du constr de recopie (schéma 3)

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\TEST2015\Debug\TE...

+++ appel constr point +++008D48B8
saisir x et y
11
11

+++ appel constr point +++008D49E8
saisir x et y
22
22
COURBE a

le nbre de points est 2
008D48B8 x:11 y:11
008D49E8 x:22 y:22

-----

-----
COURBE b

le nbre de points est 2
008D48B8 x:11 y:11
008D4BC0 x:22 y:22
Appuyez sur une touche pour continuer... ■
```

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

Introduction (1/2)

- Le langage C++ offre une bibliothèque nommée **Standard Template Library** (en abrégé **STL**) développée chez Hewlett Packard.
- Cette bibliothèque fournit un ensemble de **classes** dites **conteneurs**, permettant de représenter les structures de données les plus répandues telles que les tableaux (classe: vector), les listes (classe: list), les piles (classe: stack), les files (classe: queue) etc.
- **Les vectors sont des tableaux** avec une gestion de la mémoire et des méthodes de gestion intégrés.

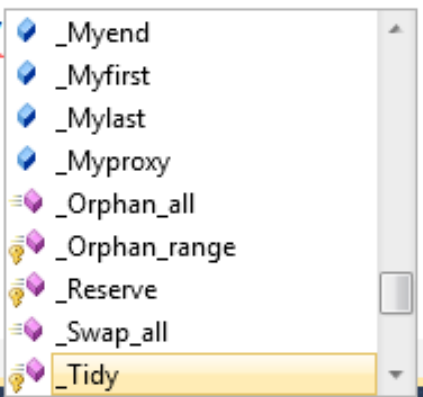
Introduction (2/2)

- Il est possible de construire un vecteur d'entiers, un vecteur de points, un vecteur de pointeurs sur point, etc par les déclarations suivantes:
- **vector<int> v;** // vecteur vide d'éléments de type **int**
- **vector<point> v;** // vecteur vide d'éléments de type **point**
- **vector<point*> v;** // vecteur vide d'éléments de type **point***

La classe vector

```
#include<vector>
// classe vector : conteneur
void main()
{
    vector<int> v;

    v.|
    sy
```

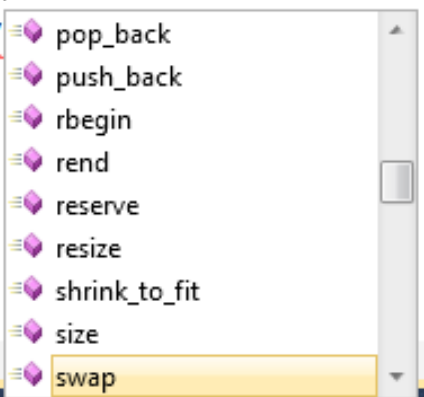


- _Myend
- _Myfirst
- _Mylast
- _Myproxy
- _Orphan_all
- _Orphan_range
- _Reserve
- _Swap_all
- _Tidy

Quelques attributs (en bleu) de l'objet v: `_Myfirst`, `_Mylast`, etc

```
#include<vector>
// classe vector : conteneur
void main()
{
    vector<int> v;

    v.|
    sy
```



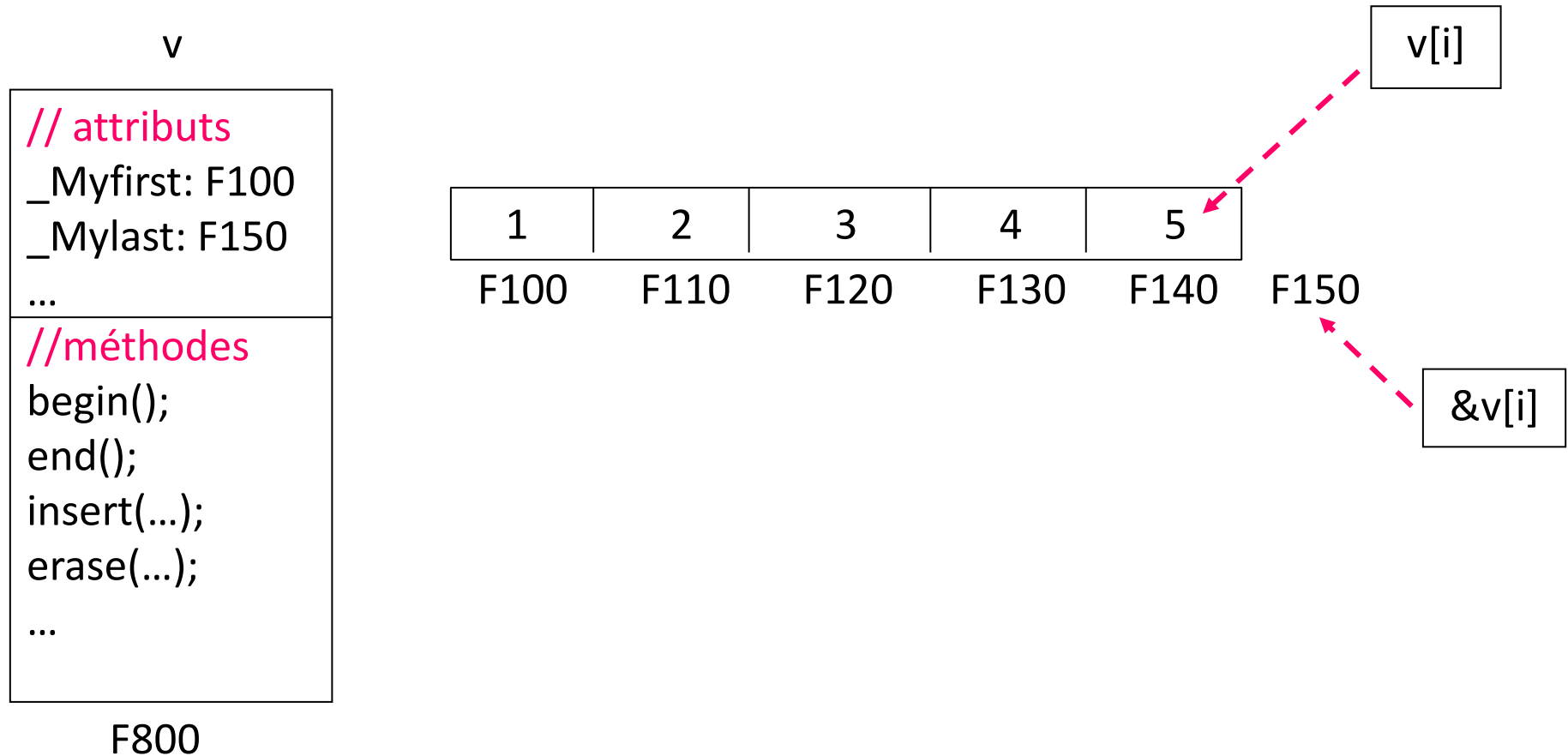
- pop_back
- push_back
- rbegin
- rend
- reserve
- resize
- shrink_to_fit
- size
- swap

Quelques méthodes (en rose) de l'objet v: `pop_back`, `push_back`, etc

Principales méthodes

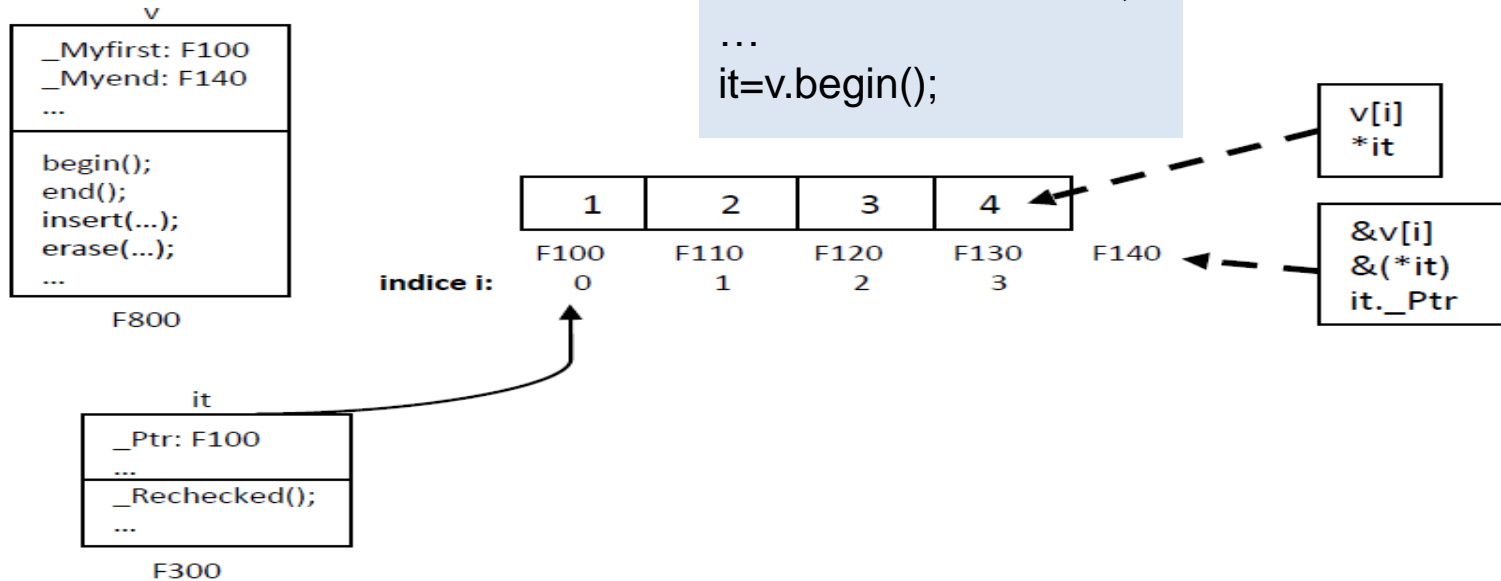
fonctions	description
<code>v.push_back(val);</code>	ajoute la valeur val à la fin du vecteur v.
<code>v.size();</code>	retourne le nombre d'éléments dans le vecteur v.
<code>v[i]</code>	l'élément d'indice i dans le vecteur v (1 ^{er} élément d'indice 0).
<code>v.erase(v.begin()+i);</code>	supprime l'élément d'indice i
<code>v.insert(v.begin()+i, val);</code>	insère la valeur val à la position d'indice i
<code>v.pop_back();</code>	supprime le dernier élément
<code>v.clear();</code>	supprime tous les éléments du vecteur (vider le vecteur)

Schéma de l'objet v, et du tableau d'entiers



La classe vector

```
vector<int> v;
vector<int>::iterator it;
...
it=v.begin();
```

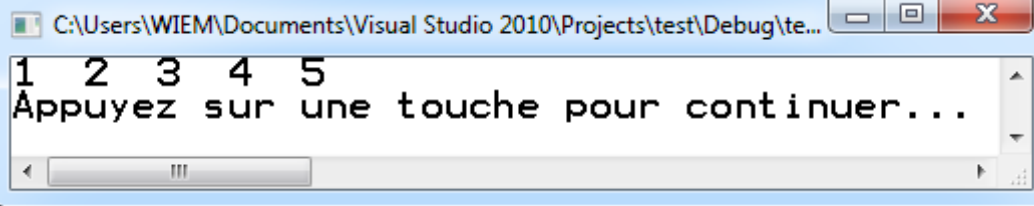


fonctions	description
<code>v.push_back(val);</code>	ajoute la valeur <code>val</code> à la fin du vecteur <code>v</code> .
<code>v.size();</code>	retourne le nombre d'éléments dans le vecteur <code>v</code> .
<code>v[i]</code>	l'élément d'indice <code>i</code> dans le vecteur <code>v</code> (1 ^{er} élément d'indice 0).
<code>v.begin()</code>	Retourne <u>un itérateur</u> qui pointe sur le premier élément du vecteur
<code>v.end()-1</code>	Retourne <u>un itérateur</u> qui pointe sur le dernier élément du vecteur
<code>v.erase(v.begin()+i);</code>	supprime l'élément d'indice <code>i</code>
<code>v.insert(v.begin()+i, val);</code>	insère la valeur <code>val</code> à la position d'indice <code>i</code>
<code>v.pop_back();</code>	supprime le dernier élément
<code>v.clear();</code>	supprime tous les éléments du vecteur (vider le vecteur)

Exemple 1: push_back, size, []

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

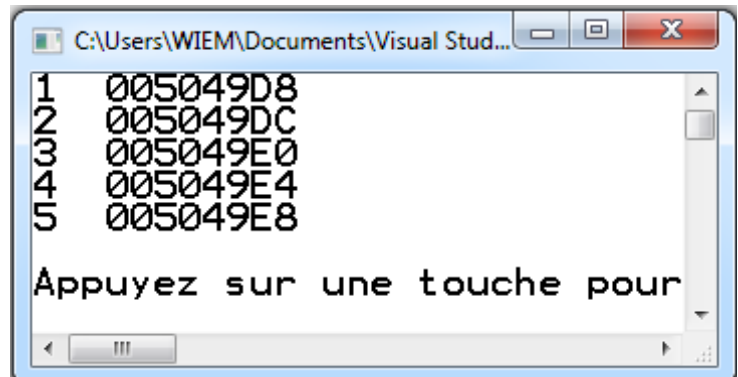
    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;
    system("PAUSE");
}
```



C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\Debug\te...
1 2 3 4 5
Appuyez sur une touche pour continuer...

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" "<<&v[i]<<endl;
    cout<<endl;
    system("PAUSE");
}
```



C:\Users\WIEM\Documents\Visual Stud...
1 005049D8
2 005049DC
3 005049E0
4 005049E4
5 005049E8
Appuyez sur une touche pour

Exemple 2: _Myfirst, _Mylast

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" "<<&v[i]<<endl;
    cout<<endl;

    cout<<"\n l'adresse de v: "<<&v<<endl;
    cout<<"\n la valeur de Myfirst "<<v._Myfirst<<endl;
    cout<<"\n la valeur de Mylast "<<v._Mylast<<endl;
    cout<<"\n la valeur de Mylast-1 "<<v._Mylast-1<<endl;
    system("PAUSE");
}
```

C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\Debug\te...

1	000349D8
2	000349DC
3	000349E0
4	000349E4
5	000349E8

l'adresse de v: 002AF8B4
la valeur de Myfirst 000349D8
la valeur de Mylast 000349EC
la valeur de Mylast-1 000349E8
Appuyez sur une touche pour continuer

Exemple 3: pop_back

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;

    v.pop_back();
    // supprimer le dernier élément
    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;
    system("PAUSE");
}
```



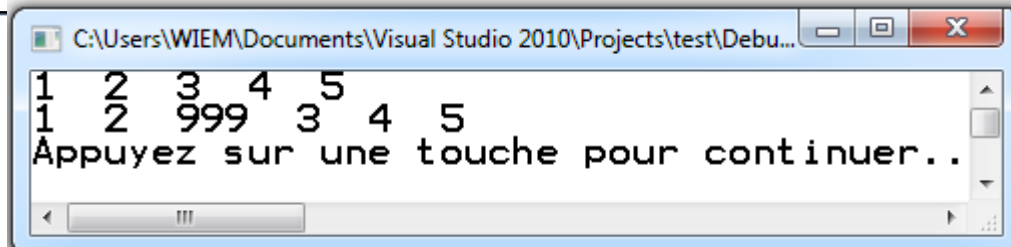
Exemple 4: insert

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;

    v.insert(v.begin()+2, 999);
    // insérer la valeur 999 à l'indice 2

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;
    system("PAUSE");
}
```



C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\Debu...

1 2 3 4 5
1 2 999 3 4 5
Appuyez sur une touche pour continuer..

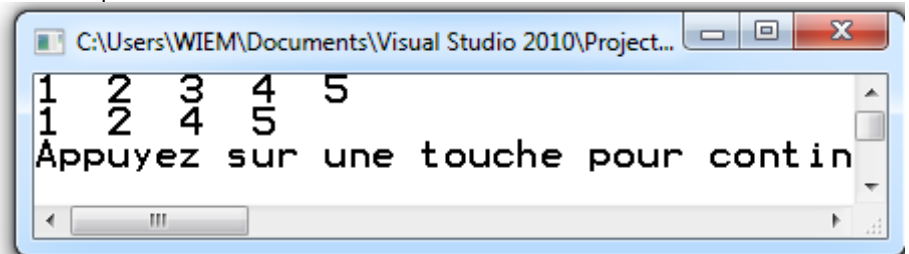
Exemple 5: erase

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;

    v.erase( v.begin()+2);
    // supprimer l'élément qui a l'indice 2

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;
    system("PAUSE");
}
```



Exemple 6: clear

```
#include<vector>
void main()
{
    vector<int> v;
    for(int i=1; i<6; i++)
        v.push_back(i);

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;

    v.clear();
    // vider le tableau

    for(int i=0; i<v.size(); i++)
        cout<<v[i]<<" ";
    cout<<endl;
    system("PAUSE");
}
```



Tableau dynamique d'objets (de points)

```
#include<vector> // bibliothèque STL
void main()
{
    vector<point> v;

    for(int i=1; i<3 ; i++)
    { // 2 itérations
        point a;
        a.saisirPoint();
        v.push_back(a);
    }

    for(int i=0; i<v.size() ; i++)
        v[i].afficher();

    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\POOD...
+++ appel constr 2 arg point +++ 0052FE8C
saisir x et y
99
99

--- appel destr point --- 0052FE8C
+++ appel constr 2 arg point +++ 0052FE8C
saisir x et y
88
88

--- appel destr point --- 007C4A00
--- appel destr point --- 0052FE8C

abscisse 99 ordonnee 99 007C4A48

abscisse 88 ordonnee 88 007C4A50
Appuyez sur une touche pour continuer...
```

Tableau dynamique d'adresses d'objets (adresses de points)

```
(Global Scope)
#include<vector> // bibliothèque STL
void main()
{
    vector<point*> v;

    for(int i=1; i<3 ; i++)
    { // 2 itérations
        point *q=new point;
        q->saisirPoint();
        v.push_back(q);
    }

    for(int i=0; i<v.size() ; i++)
        v[i]->afficher();

    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\POOD2018\De...
+++ appel constr 2 arg point +++ 005D4918
saisir x et y
11
11
+++ appel constr 2 arg point +++ 005D4A88
saisir x et y
22
22

abscisse 11 ordonnee 11 005D4918

abscisse 22 ordonnee 22 005D4A88
Appuyez sur une touche pour continuer...
```

Exemple: classe courbe

Objet courbe

F700	F720	...
_Myfirst	_Mylast	

tab

// les adresses
// des points
tab[i]
de type
(point*)

vecteur
de point*

F800	F500
------	------

F700 F710

11	11
x	y

22	22
x	y

F800

F500

// les points
***tab[i]**
de type
(point)

```
class courbe
{
    vector<point*> tab;
    // un seul attribut privé: tab
    // tab est un objet: instance de vector
    // les éléments du tableau sont
    // de type point*
public:
    courbe(void);
    void remplir();
    courbe(const courbe&);
    void ajouter(point,int =0);
    // par défaut ajouter au début
    void supprimer(int =0);
    // par défaut supprimer le 1er point
    int chercher(point);
    int taille(){return tab.size();}
    void afficher(string = "");
    ~courbe(void);
};
```

Les **éléments** du vecteur sont de **type (point*)**. les **valeurs** du tableau sont: **F800, F500**

Nombre d'éléments: `tab.size()`

Pour afficher les points: `tab[i]->afficher();`
`(*tab[i]).afficher();`

Pour ajouter un élément au vecteur:
`tab.push_back(adressePoint);`


```
// l'interface
class courbe
{
    vector<point*> tab;
public:
    courbe();
    void remplir();
    courbe(const courbe&);
    ~courbe(void);
    void afficher(string = "");
    void ajouter (point, int=0);
    void ajouter (point*, int=0);
    void supprimer(int =0);
    int taille(){ return tab.size();}
    bool chercher(point);
};
```

```
void courbe::ajouter(point pt, int ind)
{
    point*q=new point(pt);
    tab.insert(tab.begin()+ind, q);
}
void courbe::ajouter(point*q, int ind)
{
    tab.insert(tab.begin()+ind, q);
}
```

```

courbe::courbe()
{
    int abs,ord; char rep;
    do
    {
        cout<<"\n saisir x et y "<<endl;
        cin>>abs; cin>>ord;
        point *q=new point(abs,ord);
        tab.push_back(q);
        cout<<"\n rajouter?"<<endl;
        cin>>rep;
    }
    while (rep=='o' || rep=='O');
}

```

```

courbe::courbe(const courbe &w)
{
    for(int i=0; i<w.tab.size(); i++)
    {
        point *q=new point(*w.tab[i]);
        tab.push_back(q);
    }
}

```

```

void courbe::ajouter(point pt, int ind)
{
    point *q=new point(pt);
    tab.insert(tab.begin()+ind, q);
}

```

```

void courbe::afficher(string msg)
{
    cout<<msg<<endl;
    for(int i=0; i<tab.size(); i++)
        tab[i]->afficher();
}

```

```

courbe::~~courbe()
{
    for(int i=0; i<tab.size(); i++)
        delete tab[i];
    tab.clear();
}

```

```

void courbe::supprimer(int i)
{
    delete tab[i];
    // supprimer le point d'indice i
    tab.erase(tab.begin()+i);
    // supprimer l'adresse de ce point
    // du tableau d'adresses
}

```

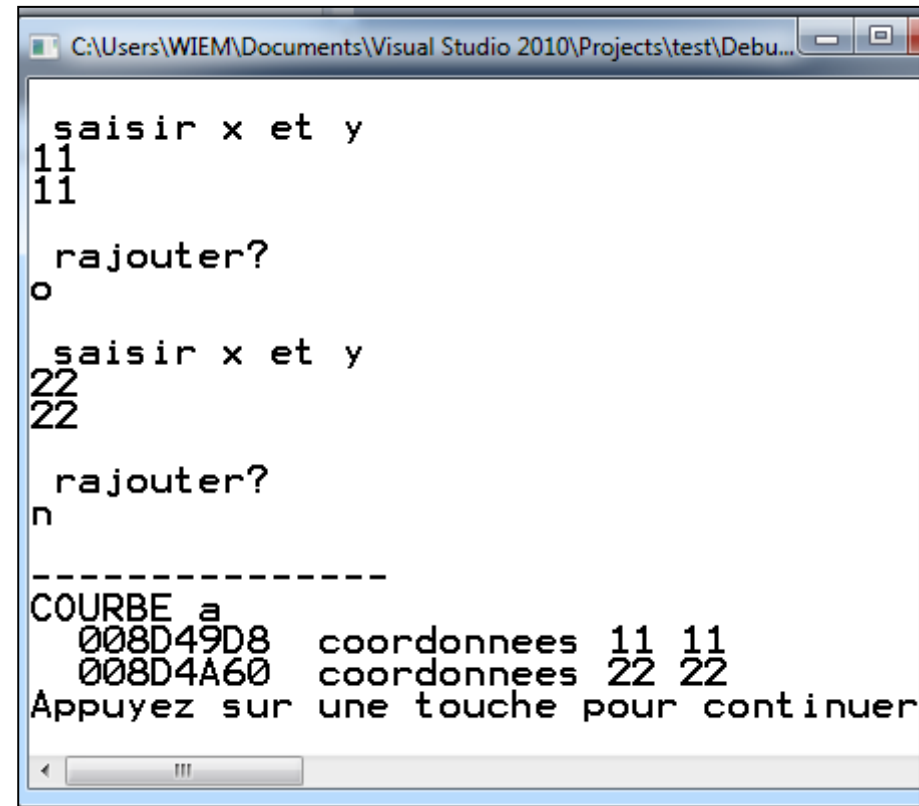
```

bool courbe::chercher(point pt)
{
    // x et y sont des attributs privés
    // de la classe point
    for(int i=0; i<tab.size(); i++)
        if(tab[i]->getX() ==pt.getX() &&
            tab[i]->getY() == pt.getY())
            return true;
    return false;
}

```

tests

```
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```



```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\Debu...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
008D49D8 coordonnees 11 11
008D4A60 coordonnees 22 22
Appuyez sur une touche pour continuer
```

```

courbe.h courbe.cpp main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    // test du constructeur de recopie
    courbe b=a;
    b.afficher("COURBE b");
    system("PAUSE");
}

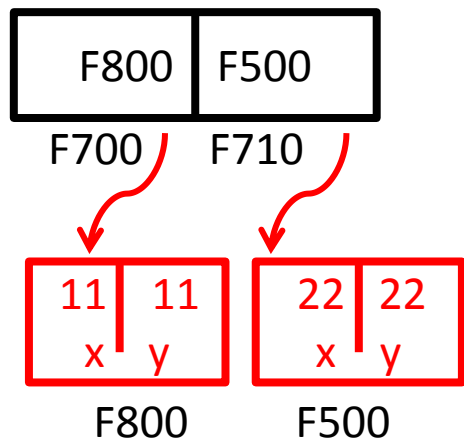
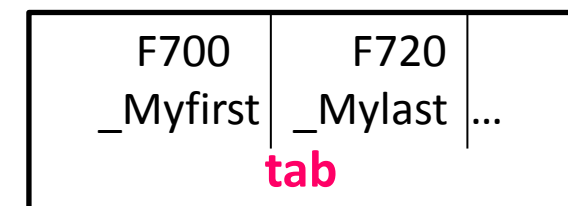
```

```

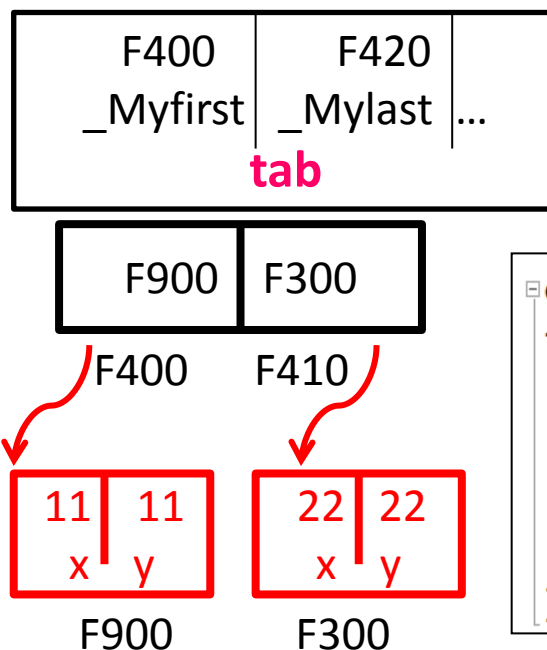
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\Debug\te...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
007749D8 coordonnees 11 11
00774A60 coordonnees 22 22
-----
COURBE b
00774C90 coordonnees 11 11
00774CD8 coordonnees 22 22
Appuyez sur une touche pour continuer...

```

Objet courbe a



Objet courbe b



```

courbe::courbe(const courbe &w)
{
    for(int i=0; i<w.tab.size(); i++)
    {
        point *q=new point(*w.tab[i]);
        tab.push_back(q);
    }
}

```

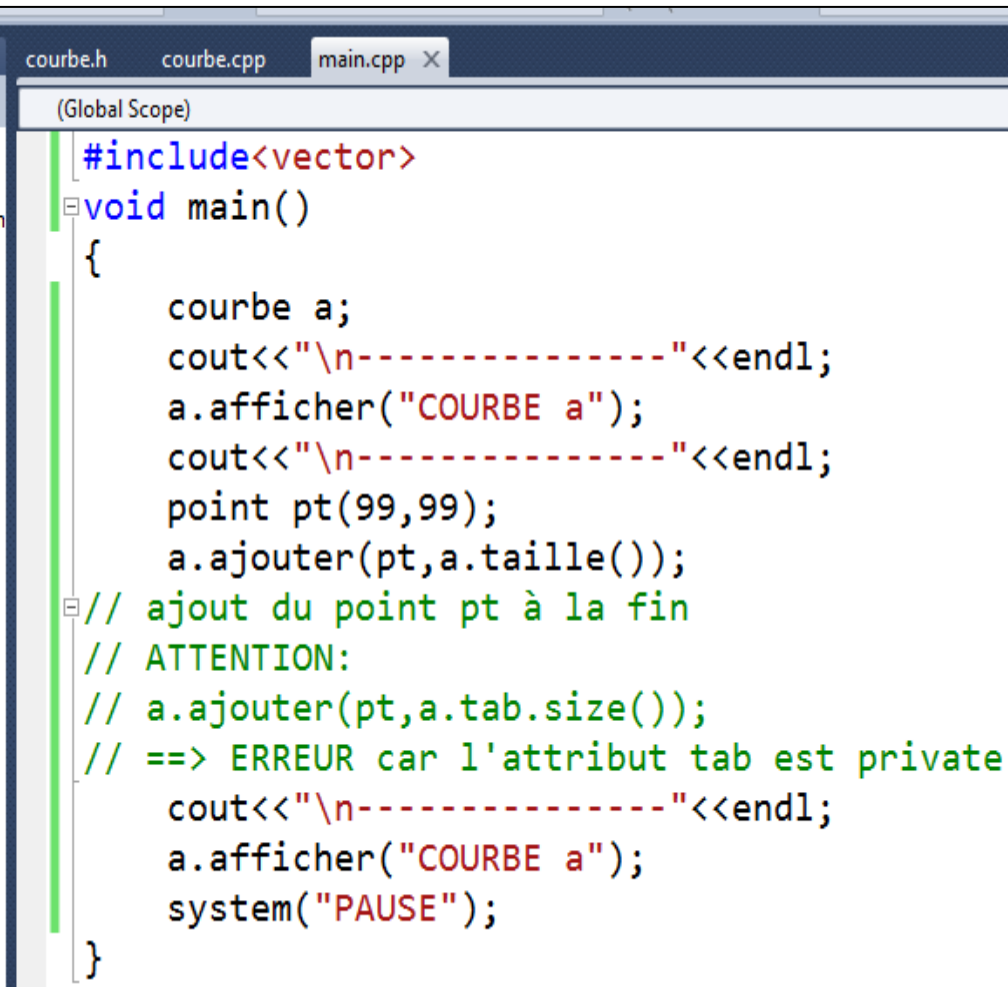
Test de la méthode ajouter: ajout à un indice donné

```
courbe.h  courbe.cpp  main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    point pt(99,99);
    a.ajouter(pt,1);
    // ajout du point pt à l'indice 1
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```

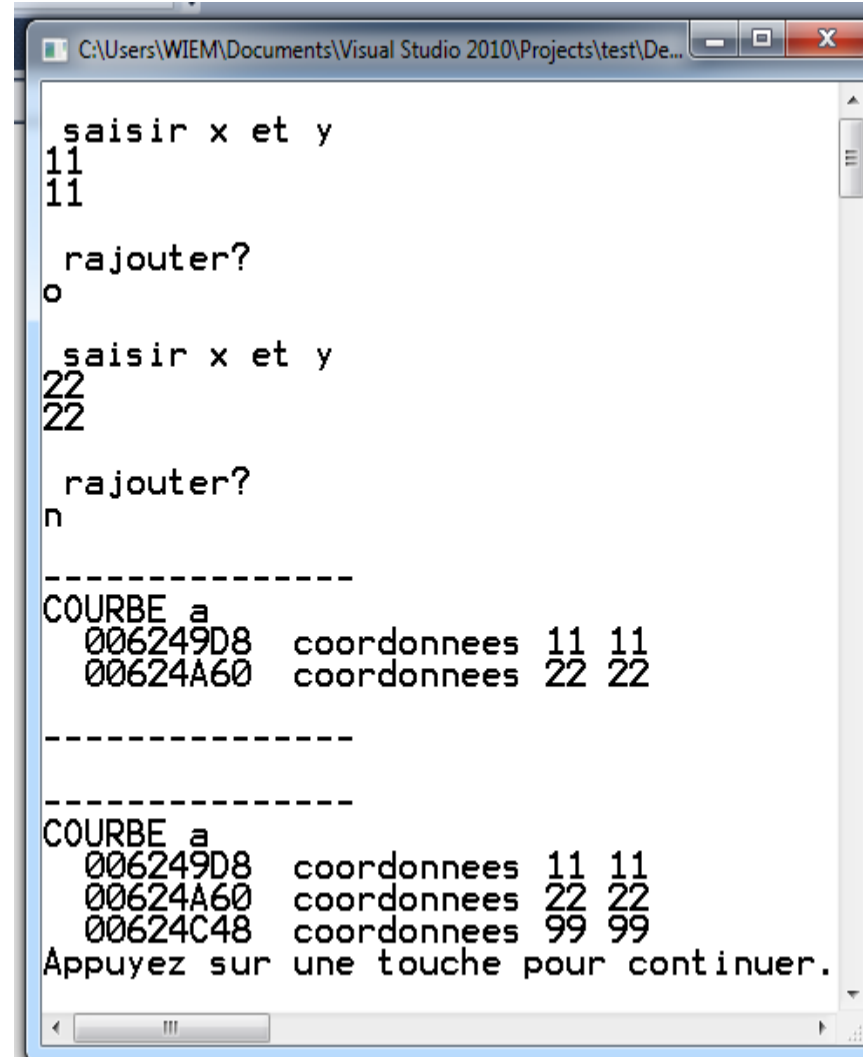
```
void courbe::ajouter(point pt, int ind)
{
    point *q=new point(pt);
    tab.insert(tab.begin()+ind, q);
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\tes...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
006349D8 coordonnees 11 11
00634A60 coordonnees 22 22
-----
COURBE a
006349D8 coordonnees 11 11
00634C48 coordonnees 99 99
00634A60 coordonnees 22 22
Appuyez sur une touche pour continue
```

Test de la méthode ajouter: ajout à la fin



```
courbe.h  courbe.cpp  main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    point pt(99,99);
    a.ajouter(pt,a.taille());
    // ajout du point pt à la fin
    // ATTENTION:
    // a.ajouter(pt,a.tab.size());
    // ==> ERREUR car l'attribut tab est private
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```



```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\De...
saisir x et y
11
11

rajouter?
o

saisir x et y
22
22

rajouter?
n

-----
COURBE a
006249D8 coordonnees 11 11
00624A60 coordonnees 22 22

-----

COURBE a
006249D8 coordonnees 11 11
00624A60 coordonnees 22 22
00624C48 coordonnees 99 99
Appuyez sur une touche pour continuer.
```

Test de la méthode ajouter: ajout au début

```
courbe.h  courbe.cpp  main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    point pt(99,99);
    a.ajouter(pt);

    // l'argument par défaut =0
    // L'ajout par défaut se fait à la
    // position 0 (au début)
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\De...
saisir x et y
11
11

rajouter?
o

saisir x et y
22
22

rajouter?
n

-----
COURBE a
004B49D8 coordonnees 11 11
004B4A60 coordonnees 22 22

-----

-----
COURBE a
004B4C48 coordonnees 99 99
004B49D8 coordonnees 11 11
004B4A60 coordonnees 22 22
Appuyez sur une touche pour continuer.
```

Test de la méthode supprimer: suppression du 1^{er} élément

```
courbe.h  courbe.cpp  main.cpp* X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    a.supprimer();
    // par défaut suppression du 1er élément
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}

void courbe::supprimer(int i)
{
    delete tab[i];
    // supprimer le point d'indice i
    tab.erase(tab.begin()+i);
    // supprimer l'adresse de ce point
    // du tableau d'adresses
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\De...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
007749D8 coordonnees 11 11
00774A60 coordonnees 22 22
-----
-----
COURBE a
00774A60 coordonnees 22 22
Appuyez sur une touche pour continuer.
```


Test de la méthode supprimer: suppression de l'élément se trouvant à un indice donné

```
courbe.h  courbe.cpp  main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    a.supprimer(1);
    // suppression de l'élément d'indice 1
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\t...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
o
saisir x et y
33
33
rajouter?
n
-----
COURBE a
001949D8 coordonnees 11 11
00194A60 coordonnees 22 22
00194D18 coordonnees 33 33
-----
COURBE a
001949D8 coordonnees 11 11
00194D18 coordonnees 33 33
Appuyez sur une touche pour continu
```

Test de la méthode supprimer: suppression du dernier élément

```
courbe.h  courbe.cpp  main.cpp x
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    a.supprimer(a.taille()-1);
    // suppression du dernier élément
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
001D49D8 coordonnees 11 11
001D4A60 coordonnees 22 22
-----
-----
COURBE a
001D49D8 coordonnees 11 11
Appuyez sur une touche pour continuer
```

Test de la méthode chercher

```
courbe.h  courbe.cpp  main.cpp X
(Global Scope)
#include<vector>
void main()
{
    courbe a;
    cout<<"\n-----"<<endl;
    a.afficher("COURBE a");
    cout<<"\n-----"<<endl;
    point pt(44,44);
    bool res=a.chercher(pt);
    cout<<"\n-----"<<endl;
    if(res==1) cout<<"\n le point existe "<<endl;
    else cout<<"\n le point n'existe pas "<<endl;
    system("PAUSE");
}
```

```
bool courbe::chercher(point pt)
{
    // x et y sont des attributs privés
    // de la classe point
    for(int i=0; i<tab.size(); i++)
        if(tab[i]->getX() ==pt.getX() &&
            tab[i]->getY() == pt.getY())
            return true;
    return false;
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\test\De...
saisir x et y
11
11
rajouter?
o
saisir x et y
22
22
rajouter?
n
-----
COURBE a
006D4900 coordonnees 11 11
006D4948 coordonnees 22 22
-----
le point n'existe pas
Appuyez sur une touche pour continuer.
```

x et y sont privés, donc ils sont accessibles uniquement par les méthodes de la classe point → Ils ne sont pas accessibles à partir des méthodes de la classe courbe.