



# POO – Langage C++

## Les fichiers

**1<sup>ère</sup> année ingénieur informatique**

Mme Wiem Yaiche Elleuch

2018 - 2019

# Définitions

- un fichier est une suite d'informations conservée sur un périphérique de stockage
- Chaque fichier a un début (la première information) et une fin (**le vide** après la dernière information).
- Entre ce début et cette fin se trouvent les informations que contient le fichier
- Ils existe 2 types de fichiers:
  - Fichiers textes
  - Fichiers binaires
- Un fichier texte est un fichier qui contient des informations sous la **forme de caractères**
- Un fichier binaire est un fichier qui contient directement la **représentation mémoire des informations**.

# Fichiers textes

- **Avantages d'un fichier texte**

- facile à manipuler
- facile à modifier (il suffit de prendre un éditeur de texte)
- facilement compréhensible par l'utilisateur (les données sont représentées sous leur forme textuelle)
- portable (transférable d'un ordinateur à un autre sans grande difficulté)

- **Inconvénients d'un fichier texte**

- perte de temps dans la transcription des informations en caractères
- une même information n'occupe pas toujours le même espace

# Fichiers binaires

- **Avantages des fichiers binaires**
  - plus rapide (pas de conversion)
  - accès direct aux données car longueur fixe des informations (pour accéder à la  $i$ -ème information, il suffit d'atteindre le  $i \times \text{taille}(\text{information})$  ème octet)
- **Inconvénients d'un fichier binaire**
  - non éditable par l'utilisateur
  - illisible pour l'utilisateur
  - données non directement transférables d'un ordinateur à l'autre

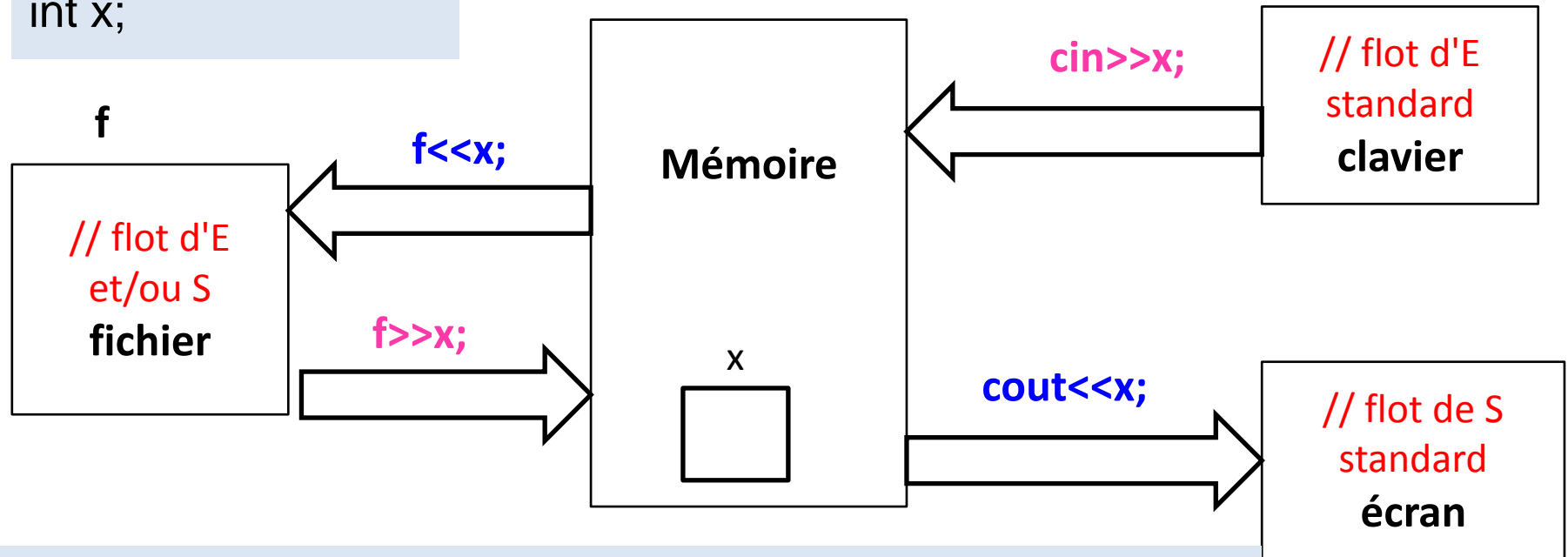
# Transfert d'informations

E: entrée  
S: sortie

```
#include<fstream>
```

```
fstream f;
```

```
int x;
```



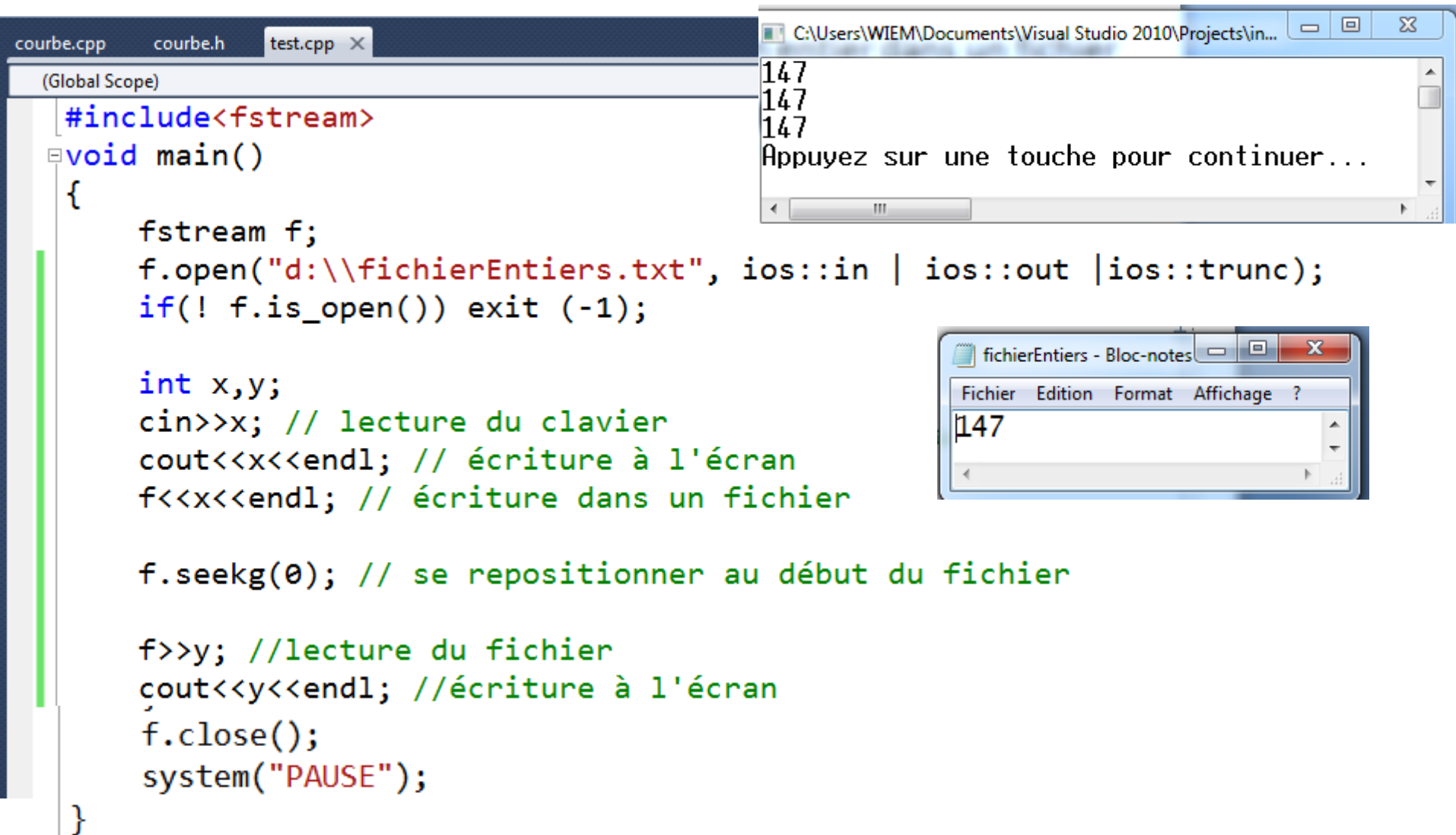
Écriture sur le flot de sortie: **operator<<** (envoyer dans le flux)

- **cout<<x;** // écriture à l'écran
- **f<<x;** // écriture dans un fichier

Lecture à partir du flot d'entrée: **operator>>** (extraction du flux)

- **cin>>x;** // lecture à partir du clavier
- **f>>x;** // lecture à partir d'un fichier

# Lecture et d'écriture d'un entier dans un fichier



```
courbe.cpp  courbe.h  test.cpp x
(Global Scope)
#include<fstream>
void main()
{
    fstream f;
    f.open("d:\\fichierEntiers.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);

    int x,y;
    cin>>x; // lecture du clavier
    cout<<x<<endl; // écriture à l'écran
    f<<x<<endl; // écriture dans un fichier

    f.seekg(0); // se repositionner au début du fichier

    f>>y; //lecture du fichier
    cout<<y<<endl; //écriture à l'écran
    f.close();
    system("PAUSE");
}
```

C:\Users\WIEM\Documents\Visual Studio 2010\Projects\in...  
147  
147  
147  
Appuyez sur une touche pour continuer...

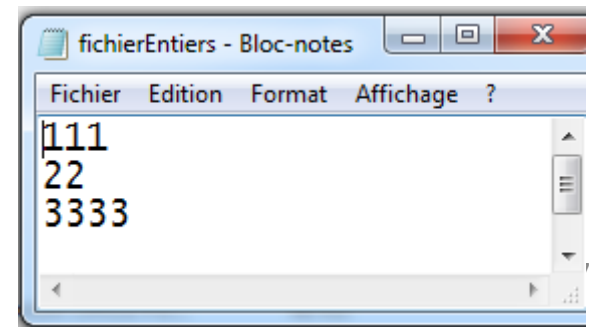
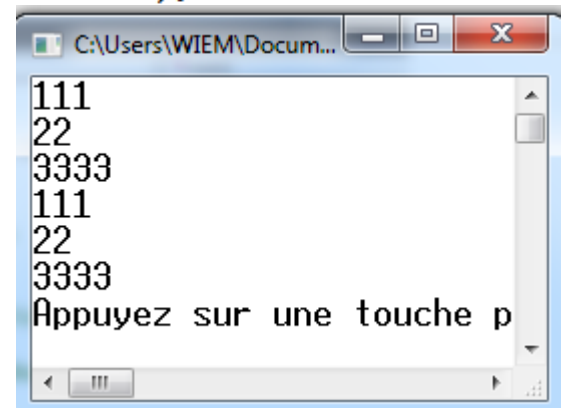
fichierEntiers - Bloc-notes  
Fichier Edition Format Affichage ?  
147

# Lire et Ecrire 3 entiers

```
courbe.cpp  courbe.h  test.cpp X
(Global Scope)  main()

void main()
{
    fstream f;    int x;
    f.open("d:\\fichierEntiers.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);

    for(int i=0; i<3;i++) // 3 itérations
    {
        cin>>x; // lecture du clavier
        f<<x<<endl; // écriture dans un fichier
    }
    f.seekg(0); // se repositionner au début du fichier
    while(1) // boucle infinie
    {
        f>>x; //lecture du fichier
        if(f.eof()) break; // tester si on a atteint la fin du fichier
        cout<<x<<endl; //écriture à l'écran
    }
    f.close();
    system("PAUSE");
}
```



# getline

```
courbe.cpp  courbe.h  test.cpp x
(Global Scope)  main()

void main()
{
    fstream f;  char ch[101];
    f.open("d:\\fichierEntiers.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);

    f<<"programmation orientee objet 2014 2015"<<endl;
    f<<"examen session principale juin 2015"<<endl;
    f<<"resultat juillet 2015"<<endl;
    f.seekg(0);

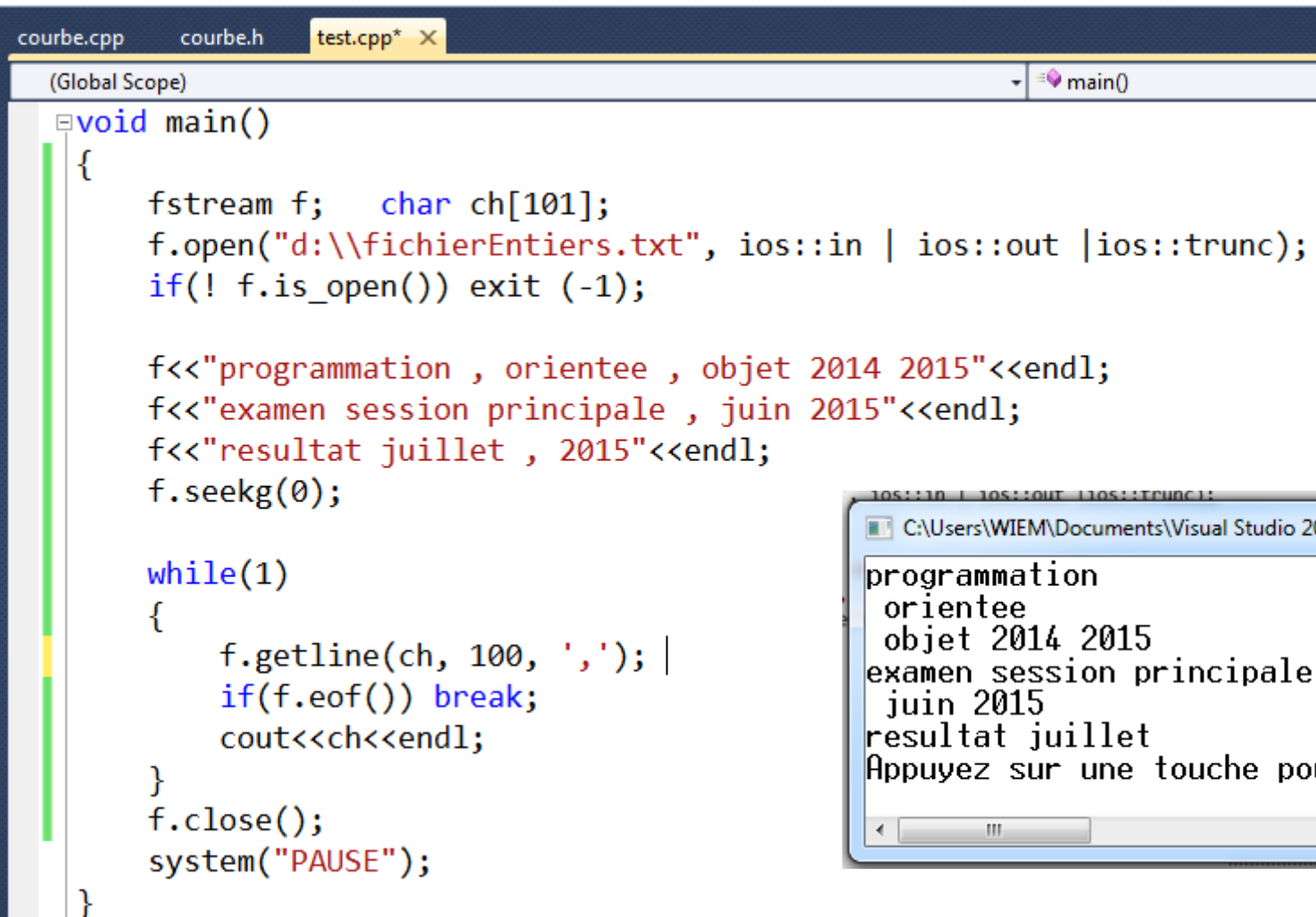
    while(1)
    {
        f.getline(ch, 100); // <=> f.getline(ch, 100, '\n');
        if(f.eof()) break;
        cout<<ch<<endl;
    }
    f.close();
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015...
programmation orientee objet 2014 2015
examen session principale juin 2015
resultat juillet 2015
Appuyez sur une touche pour continuer...
```

```
fichierEntiers - Bloc-notes
Fichier  Edition  Format  Affichage ?
programmation orientee objet 2014 2015
examen session principale juin 2015
resultat juillet 2015
```



# getline ','

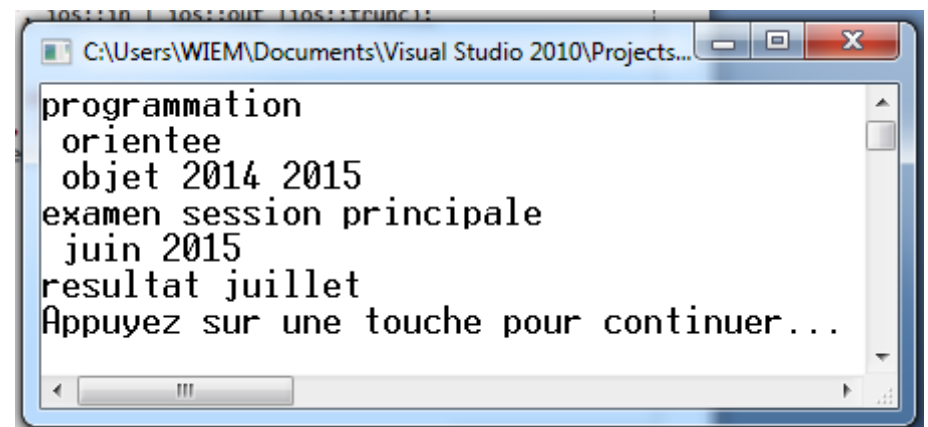


```
courbe.cpp  courbe.h  test.cpp* X
(Global Scope)  main()

void main()
{
    fstream f;  char ch[101];
    f.open("d:\\fichierEntiers.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);

    f<<"programmation , orientee , objet 2014 2015"<<endl;
    f<<"examen session principale , juin 2015"<<endl;
    f<<"resultat juillet , 2015"<<endl;
    f.seekg(0);

    while(1)
    {
        f.getline(ch, 100, ','); |
        if(f.eof()) break;
        cout<<ch<<endl;
    }
    f.close();
    system("PAUSE");
}
```



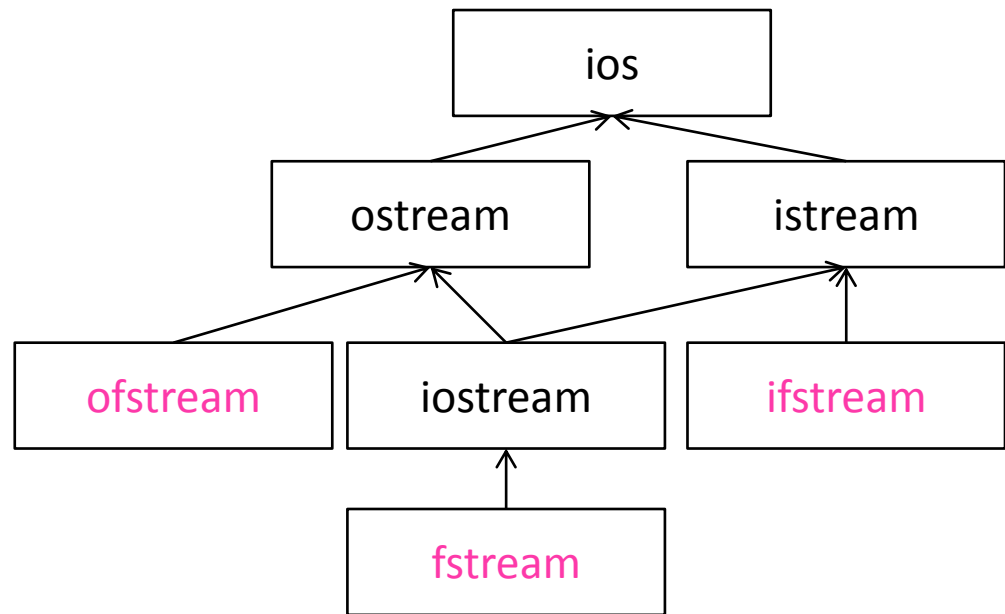
```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects...
programmation
,
orientee
,
objet 2014 2015
,
examen session principale
,
juin 2015
,
resultat juillet
,
Appuyez sur une touche pour continuer...
```

# Les entrées/sorties

- **ios** : classe de base des entrées/sorties.
- **istream** : classe dérivée de *ios* pour les flots en entrée.
- **ostream** : classe dérivée de *ios* pour les flots en entrée.

les 3 classes permettant de manipuler des fichiers.

- **ifstream**: l'accès en **lecture** seulement,
- **ofstream**: l'accès en **écriture** seulement,
- **fstream**: l'accès en **lecture** et **écriture**.



# Les méthodes associées aux fichiers

- **open(fich\_nom):** ouvre le fichier dont le nom est « fich\_nom », en utilisant les options par défaut
- **open(fich\_nom, flags):** ouvre le fichier « fic\_nom », en utilisant les options spécifiées par l'argument « flags ».
- **close():** ferme le fichier
- **is\_open():** retourne vrai (true) si le fichier a été ouvert, sinon faux (false) en cas d'échec.

# Exemples:

```
void ouvrir( fstream &es)
{
    es.open( "exemple.txt", ios::in | ios::out | ios::trunc);
    if ( ! es.is_open()) exit(-1);
}
```

==> Ce fichier sera créé dans le répertoire courant

```
void ouvrir( fstream &es)
{
    es.open( "D:\\divers\\exemple.txt", ios::in | ios::out | ios::trunc);
    if ( ! es.is_open()) exit(-1);
}
```

==> Ce fichier sera créé dans le répertoire indiqué par le chemin

```
void ouvrir( fstream &es)
{
    char nom_fich[30];
    cout<<"\n saisir le nom du fichier "<<endl;
    cin>>nom_fich;
    es.open( nom_fich, ios::in | ios::out | ios::trunc);
    if ( ! es.is_open()) exit(-1);
}
```

==> Ce fichier sera créé dans le répertoire courant

# Les bits associés à l'opération d'ouverture de fichiers

bit	signification
in	lecture, le fichier doit exister
out	si le fichier existe, écrire par dessus, le créer si nécessaire
out   trunc	si le fichier existe son contenu est perdu, le créer si nécessaire
out   app	ajouter à la fin du contenu actuel du fichier, le créer si nécessaire
in   out	lecture et écriture : position de départ est le début de fichier, le fichier doit exister
in   out   trunc	si le fichier existe son contenu est perdu, lecture et écriture dans un fichier, le fichier est créé si nécessaire

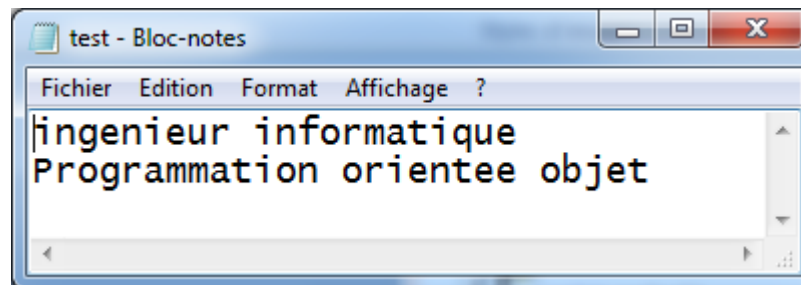
- On peut ajouter dans cette description le bit « binary » qui signifie mode d'entrée (et/ou) sortie binaire.
- Par exemple un « in | binary » signifie lecture (entrée) binaire, un « out | binary » écriture (sortie) binaire dans le fichier, etc.

# Ouverture en mode **écriture seulement**

```
// ou bien  
fstream f;  
f.open("D:test.txt", ios::out);
```

```
void main()  
{  
    ofstream f; // o: out  
    f.open("D:test.txt"); // ouverture en mode écriture  
    f<<"ingenieur informatique"<<endl;  
    f<<"Programmation orientee objet"<<endl;  
    f.close();  
    system("PAUSE");  
}
```

Contenu du  
fichier  
test.txt



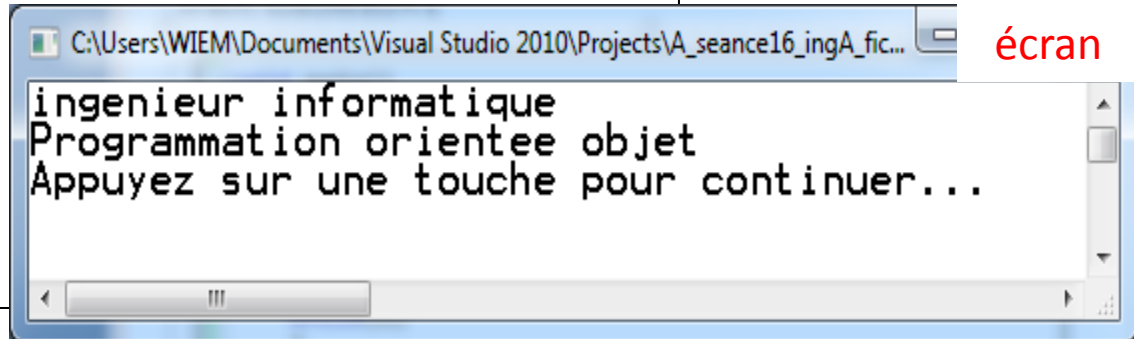
On peut ne  
pas utiliser  
open

```
void main()  
{  
    ofstream f("D:\\test.txt");  
    if (!f) cout<<"ERREUR ouverture "<<endl;
```

## Ouverture en mode lecture seulement

```
// ou bien  
fstream f;  
f.open("D:test.txt", ios::in);
```

```
void main()  
{  
    ifstream f; // i: in  
    f.open("D:test.txt"); // ouverture en mode lecture  
    // le fichier existe déjà et contient des données  
    char ch[100];  
    while(1)  
    {  
        f.getline(ch, 100);  
        //extraction de 100 caracteres à partir du fichier  
        //cette extraction s'arrête si elle rencontre \n  
        // mettre les informations extraites dans la chaîne ch  
        if (f.eof()) break;  
        cout<<ch<<endl;  
    }  
    f.close();  
    system("PAUSE");  
}
```



écran

On peut ne  
pas utiliser  
open

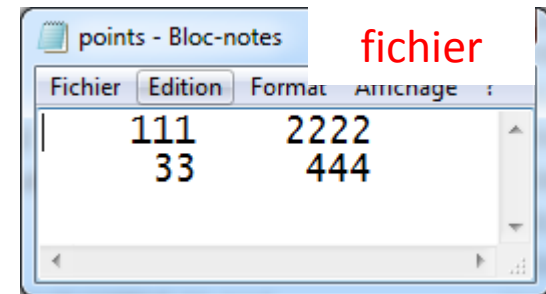
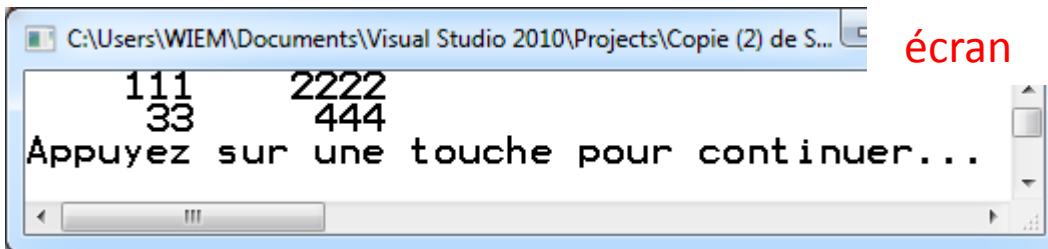
```
void main()  
{  
    ifstream f("D:\\test.txt");  
    if (!f) cout<<"ERREUR ouverture "<<endl;
```

# SURCHARGE DE L'OPÉRATEUR <<

```
ostream& operator<< (ostream& out, point& pt)
{
    out<<setw(10)<<pt.x<<" "<<setw(10)<<pt.y;
    return out;
}
```

**Gabarit** d'écriture d'un point sur un flot de sortie (**écran, fichier, etc**)

- l'abscisse sur 10 positions
- Espace entre abscisse et ordonnée (**voir remarque en bas**)
- l'ordonnée sur 10 positions
- Sans endl à la fin pour pouvoir écrire la couleur sur la même ligne lors de la surcharge de operator<< dans la classe pointCouleur



**Remarque:** cet espace garantit que l'abscisse et l'ordonnée soient toujours séparées (cas où y est une valeur sur 10 positions)

Ajouter: `#include<iomanip>` pour utiliser setw



# SURCHARGE DE L'OPÉRATEUR >>

```
istream& operator>> (istream& entree, point &pt)
{
    entree>>pt.x>>pt.y;
    return entree;
}
```

Lecture d'informations à partir d'un flot d'entrée (**clavier, fichier**, etc)

- la 1<sup>ère</sup> valeur lue sera affectée à pt.x
- La 2<sup>ème</sup> valeur lue sera affectée à pt.y

# Surcharge de operator<< et operator>>

```
void main()
{
    point a;
    cin>>a; // cin>>Objet
    cout<<a; // cout<<objet
    // friend ostream& operator<< (ostream&, point&);
    // friend istream& operator>> (istream&, point&);
    cout<<"\n-----"<<endl;
    point*q = new point;

    cin>>q; // cin>>adresseObjet
    cout<<q; // cout<<adresseObjet
    // friend ostream& operator<< (ostream&, point*);
    // friend istream& operator>> (istream&, point*);
    system("PAUSE");
}
```

```

class point
{
protected:
    int x;
    int y;

public:
    friend ostream& operator<< (ostream&, point&);
    friend istream& operator>> (istream&, point&);
    friend ostream& operator<< (ostream&, point*);
    friend istream& operator>> (istream&, point*);

```

point coincider(c

```

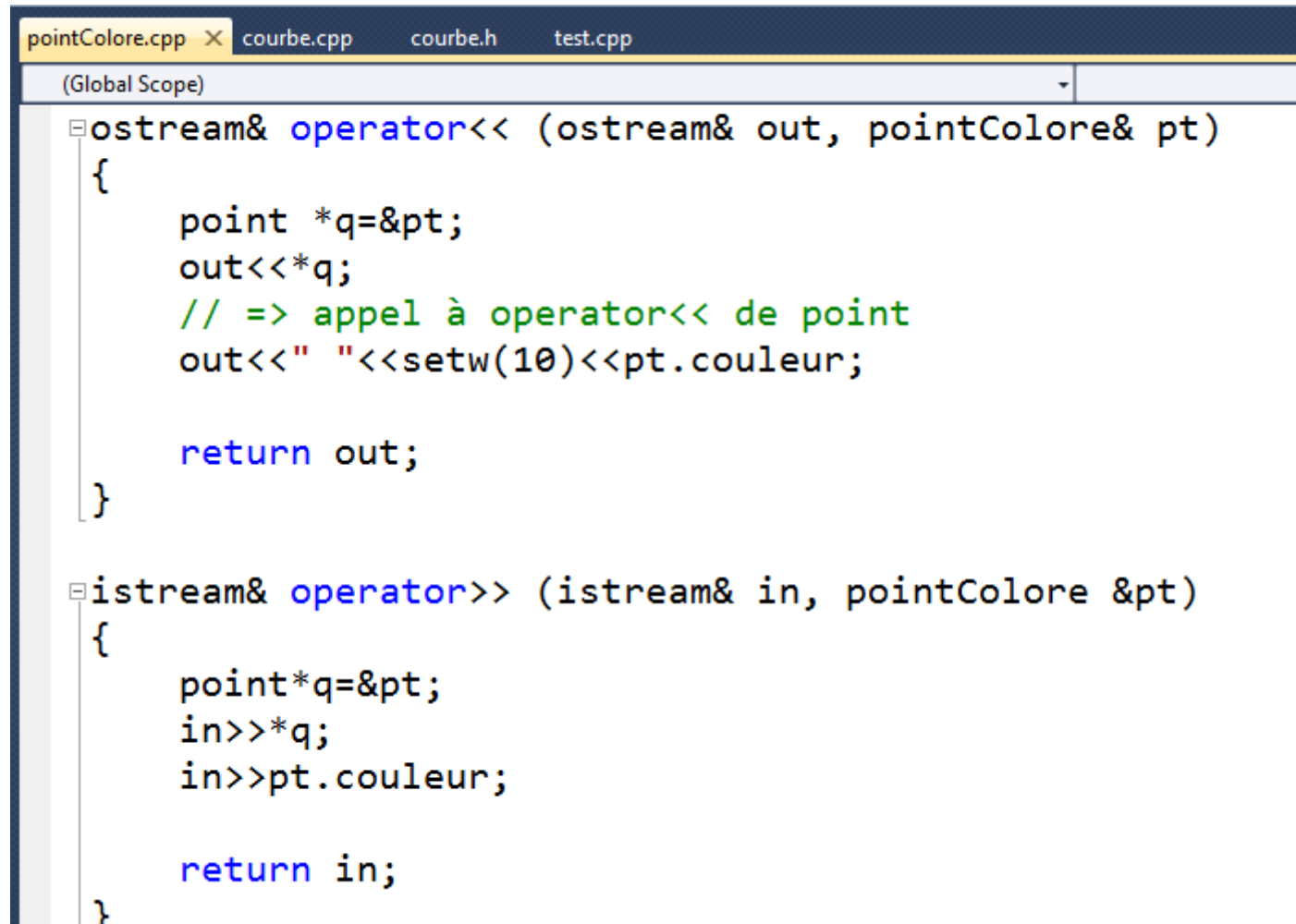
ostream& operator<< (ostream& out, point *pt)
{
    out<<setw(10)<<pt->x<<" "<<setw(10)<<pt->y;
    return out;
}

istream& operator>> (istream& in, point *pt)
{
    in>>pt->x;
    in>>pt->y;

    return in;
}

```

# Surcharge de operator<< et operator>> dans la classe pointCouleur



The image shows a code editor window with four tabs: 'pointCouleur.cpp' (active), 'courbe.cpp', 'courbe.h', and 'test.cpp'. The editor displays the implementation of two overloaded operators for the 'pointCouleur' class. The first function, 'operator<<', takes an 'ostream& out' and a 'pointCouleur& pt' as arguments. It creates a 'point' pointer 'q' pointing to 'pt', writes '\*q' to the stream, then writes a space followed by 'pt.couleur' with a width of 10. The second function, 'operator>>', takes an 'istream& in' and a 'pointCouleur &pt' as arguments. It creates a 'point' pointer 'q', reads from the stream into '\*q', then reads 'pt.couleur' from the stream. Both functions return the stream object.

```
pointCouleur.cpp x courbe.cpp courbe.h test.cpp
(Global Scope)
ostream& operator<< (ostream& out, pointCouleur& pt)
{
    point *q=&pt;
    out<<*q;
    // => appel à operator<< de point
    out<<" "<<setw(10)<<pt.couleur;

    return out;
}

istream& operator>> (istream& in, pointCouleur &pt)
{
    point*q=&pt;
    in>>*q;
    in>>pt.couleur;

    return in;
}
```

# Surcharge de operator<< et operator>> dans la classe pointCouleurMasse

```
pointCouleurMasse.cpp × pointCouleur.cpp  courbe.cpp  courbe.h  test.cpp
(Global Scope)
ostream& operator<< (ostream& out, pointCouleurMasse& pt)
{
    pointCouleur *q=&pt;
    out<<*q;
    out<<" "<<setw(10)<<pt.masse;

    return out;
}

istream& operator>> (istream& in, pointCouleurMasse &pt)
{
    pointCouleur*q=&pt;
    in>>*q;
    in>>pt.masse;

    return in;
}
```

# REMARQUE

```
void main()
{ fstream f;
....
point p(111,2222);
cout<<p;
f<<p;
...
}
```

```
ostream& operator<< (ostream& sortie, point &pt)
{
    sortie<<"\n DEBUT SURDEFINITION << point"<<endl;
    sortie<<setw(7)<<pt.x<<" "<<setw(7)<<pt.y<<"\n";
    sortie<<"\n FIN SURDEFINITION << point"<<endl;
    return sortie;
}
```



```
DEBUT SURDEFINITION << point
111      2222
FIN SURDEFINITION << point
Appuyez sur une touche pour continuer...
```

écran



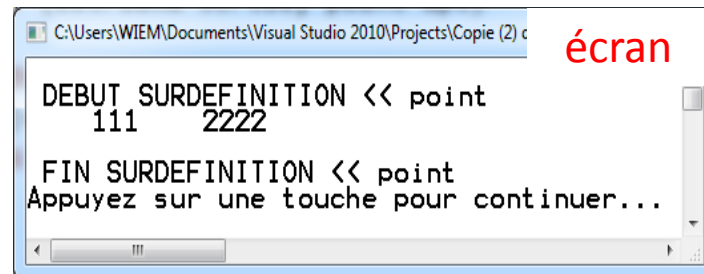
```
DEBUT SURDEFINITION << point
111      2222
FIN SURDEFINITION << point
```

fichier

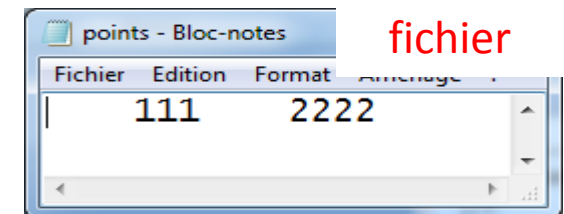
# REMARQUE

```
void main()
{ fstream f;
....
point p(111,2222);
cout<<p;
f<<p;
...
}
```

```
ostream& operator<< (ostream& sortie, point &pt)
{
    cout<<"\n DEBUT SURDEFINITION << point"<<endl;
    sortie<<setw(7)<<pt.x<<" "<<setw(7)<<pt.y<<"\n";
    cout<<"\n FIN SURDEFINITION << point"<<endl;
    return sortie;
}
```



écran



fichier

# Lecture et ecriture d'un seul point

```
point.cpp  courbe.h  test.cpp* X
(Global Scope)
void main()
{
    fstream f;
    f.open("d:\\fichierPoints.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);

    point a,b;

    cin>>a; // lecture du clavier
    f<<a<<endl; // ecriture dans le fichier

    f.seekg(0);

    f>>b; //lecture du fichier
    cout<<b<<endl; // ecriture à l'ecran

    f.close();
    system("PAUSE");
}
```

C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest20...

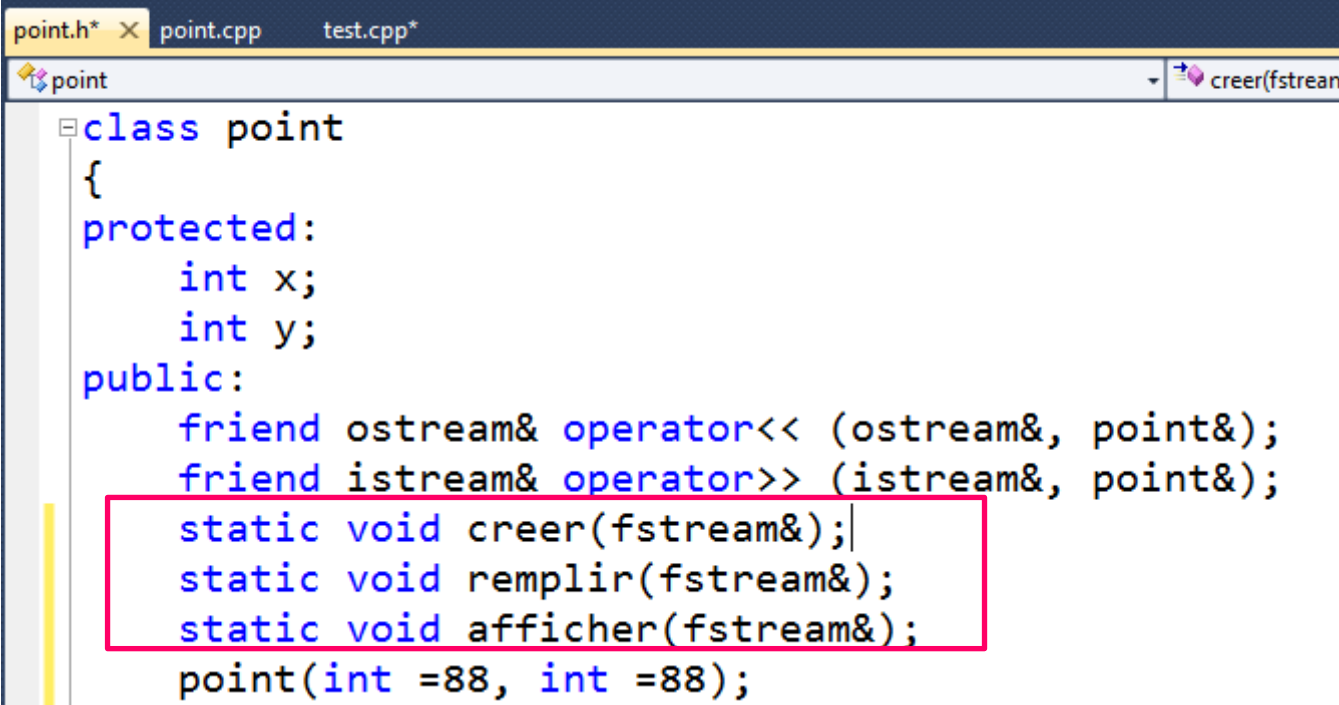
```
11
111
Appuyez sur une touche pour continuer...
```

fichierPoints - Bloc-notes

Fichier	Edition	Format	Affichage ?
	11	111	



# Lecture et ecriture de plusieurs points dans un fichier



The screenshot shows a C++ IDE with three tabs: 'point.h\*', 'point.cpp', and 'test.cpp\*'. The 'point' tab is active, displaying the following code:

```
class point
{
protected:
    int x;
    int y;
public:
    friend ostream& operator<< (ostream&, point&);
    friend istream& operator>> (istream&, point&);
    static void creer(fstream&);
    static void remplir(fstream&);
    static void afficher(fstream&);
    point(int =88, int =88);
```

The three static member function declarations are highlighted with a red rectangle:

- `static void creer(fstream&);`
- `static void remplir(fstream&);`
- `static void afficher(fstream&);`

```
point.h  point.cpp  test.cpp x
(Global Scope)
void main()
{
    fstream f;

    point::creer(f);
    point::remplir(f);
    point::afficher(f);

    f.close();
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\De...
saisir un point
11
111

saisir un point
222
2

saisir un point
3333
33

          11          111
          222          2
          3333         33
Appuyez sur une touche pour continuer...
```

```
fichierPoints - Bloc-notes
Fichier  Edition  Format  Affichage  ?
          11          111
          222          2
          3333         33
```

```
void point::creer(fstream& f)
{
    f.open("d:\\fichierPoints.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit (-1);
}
```

```
void point::remplir(fstream& f)
{
    point a;
    for(int i=0; i<3 ;i++) // 3 iérations
    {
        cout<<"\n saisir un point "<<endl;
        cin>>a;
        f<<a<<endl;
    }
}
```

```
void point::afficher(fstream &f)
{
    point a;
    f.seekg(0);
    while(1)
    {
        f>>a;
        if(f.eof()) break;
        cout<<a<<endl;
    }
}
```

```

void point::afficher(fstream &f)
{
    point a;
    f.seekg(0);
    while(1)
    {
        f>>a;
        if(f.eof()) break;
        cout<<a<<endl;
    }
}

```

La ligne est copiée dans l'objet a (de type point).

```

void point::afficher(fstream& f)
{
    char ch[100];
    f.seekg(0);
    while(1)
    {
        //à chaque itération: lire une ligne et l'afficher
        f.getline(ch, 100);
        if(f.eof()) break;
        cout<<ch<<endl;
    }
}

```

La ligne est copiée dans une chaîne de caractères ch.

# Exemples

- Lecture et écriture d'un fichier à partir d'un vector d'éléments (exemple courbe)
- Lecture et écriture d'un fichier à partir d'une liste chaînée d'éléments (exemple voiture-parcVoiture)

```
class courbe
{
    vector<point*> tab;
public:
    courbe();

    static void creer(fstream&);
    // surcharge de operator<<
    friend ostream& operator<<(ostream&, courbe &); // ecrire à l'ecran
    friend ostream& operator<<(ostream&, courbe *); // ecrire dans un fichier

    // surcharge de operator>>
    friend istream& operator>>(istream&, courbe&); //lire du clavier
    friend istream& operator>>(istream&, courbe*); //lire du fichier

    courbe& operator=(courbe&);
    courbe(const courbe&);

    int taille(){ return tab.size();}
```

```
void courbe::creer(fstream &f)
{
    f.open("d:\\fichierCourbe.txt", ios::in | ios::out | ios::trunc);
    if(!f.is_open()) exit;
}
```

```

istream& operator>> (istream& in, courbe& w)
{
    int choix;  char rep;
    do
    {
        cout<<"\n taper 1: point, 2: pointColore, 3: pointColoreMasse"<<endl;
        in>>choix;
        if(choix==1)
        {
            point* q=new point();
            in>>*q; // appel de operator>> de la classe point
            w.tab.push_back(q);
        }
        else if (choix==2)
        {
            pointColore*q=new pointColore();
            in>>*q; // appel de operator>> de la classe pointColore
            w.tab.push_back(q);
        }
        else if(choix==3)
        {
            pointColoreMasse*q=new pointColoreMasse();
            in>>*q; // appel de operator>> de la classe pointColoreMasse
            w.tab.push_back(q);
        }
        cout<<"\n ajouter ? "<<endl;
        in>>rep;
    }
    while(rep=='o' || rep=='O');

    return in;
}

```

Utilisé pour lire du clavier  
 courbe c;  
 cin>>c;

Utilisé pour écrire à l'écran  
courbe c;  
cout<<c;

```
courbe.cpp* x courbe.h test.cpp*
(Global Scope) operator<<(ostream & out, courbe & w)
ostream& operator<< (ostream& out, courbe& w)
{
    for(int i=0; i<w.tab.size(); i++)
        if (typeid(*w.tab[i])==typeid(point))
            out<<*w.tab[i]<<endl;
        // ==> appel de operator<< de la classe point

        else if (typeid(*w.tab[i])==typeid(pointCouleur))
            out<<static_cast< pointCouleur&>(*w.tab[i])<<endl;
        // ==> appel de operator<< de la classe pointCouleur

        else if (typeid(*w.tab[i])==typeid(pointCouleurMasse))
            out<<static_cast< pointCouleurMasse&>(*w.tab[i])<<endl;
        // ==> appel de operator<< de la classe pointCouleurMasse
    return out;
}
```



```
#include<fstream>
void main()
{
    courbe c;
    cin>>c;
    cout<<c;
    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debug\ingTest2015V2.exe

taper 1: point, 2: pointColore, 3: pointColoreMasse
1
11
11
ajouter ?
o
taper 1: point, 2: pointColore, 3: pointColoreMasse
2
22
22
22
ajouter ?
o
taper 1: point, 2: pointColore, 3: pointColoreMasse
3
33
33
33
33
ajouter ?
n
11 11 22
22 22 33
33 33 33
Appuyez sur une touche pour continuer... 33
```

Utilisé pour lire du fichier  
courbe c;  
f>>&c;

```
courbe.cpp* X courbe.h test.cpp
(Global Scope) operator>>(istream
istream& operator>> (istream& in, courbe* w)
{
    int choix;
    in.seekg(0);
    while(1)
    {
        in>>choix;
        if(in.eof()) break;
        if(choix==1)
        {
            point* q=new point();
            in>>*q; // appel de operator>> de la classe point
            w->tab.push_back(q);
        }
        else if (choix==2)
        {
            pointColore*q=new pointColore();
            in>>*q; // appel de operator>> de la classe pointColore
            w->tab.push_back(q);
        }
        else if(choix==3)
        {
            pointColoreMasse*q=new pointColoreMasse();
            in>>*q; // appel de operator>> de la classe pointColoreMasse
            w->tab.push_back(q);
        }
    }
    return in;
}
```

Utilisé pour écrire dans le fichier  
courbe c;  
f<<&c;

```
courbe.cpp x courbe.h test.cpp
(Global Scope) operator>>(istream & in, courbe * w)

ostream& operator<< (ostream& out, courbe* w)
{
    for(int i=0; i<w->tab.size(); i++)
        if (typeid(*w->tab[i])==typeid(point))
            out<<"1 "<<*w->tab[i]<<endl;
        // ==> appel de operator<< de la classe point

        else if (typeid(*w->tab[i])==typeid(pointColore))
            out<<"2 "<< static_cast< pointColore&>(*w->tab[i])<<endl;
        // ==> appel de operator<< de la classe pointColore

        else if (typeid(*w->tab[i])==typeid(pointColoreMasse))
            out<<"3 "<< static_cast< pointColoreMasse&>(*w->tab[i])<<endl;
        // ==> appel de operator<< de la classe pointColoreMasse
    return out;
}
```

```
courbe.cpp  courbe.h  test.cpp X
(Global Scope)
#include<fstream>
void main()
{
    fstream f;
    courbe::creer(f);

    courbe c;
    cin>>c; //lecture du clavier
    f<<&c; // ecriture dans le fichier

    f.seekg(0);

    courbe d;
    f>>&d; // lecture du fichier
    cout<<d; // ecriture à l'écran

    system("PAUSE");
}
```

```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debug\ingTest2015V2.exe
taper 1: point, 2: pointColore, 3: pointColoreMasse
1
11
11
ajouter ?
o
taper 1: point, 2: pointColore, 3: pointColoreMasse
2
22
22
22
ajouter ?
o
taper 1: point, 2: pointColore, 3: pointColoreMasse
3
33
33
33
33
ajouter ?
n
11 11 22 33
22 22 33 33
33 33 33 33
Appuyez sur une touche pour continuer...
```

fichierCourbe - Bloc-notes

Fichier	Edition	Format	Affichage	?
1	11	11		
2	22	22	22	
3	33	33	33	33

```
#include "courbe.h"
```

```
void main()
```

```
{
```

```
    courbe c;
```

```
    cin>>c;
```

```
    cout<<"\n-----"<<endl;
```

```
    c.enregistrer();
```

```
    system("PAUSE");
```

```
}
```

# enregistrer

point.cpp

main.cpp

courbe

enregistrer()

```
void courbe::enregistrer()
```

```
{
```

```
    fstream f;
```

```
    f.open("d:\\courbe.txt", ios::in | ios::out | ios::trunc);
```

```
    if(! f.is_open()) exit(-1);
```

```
    for(int i=0; i<tab.size(); i++)
```

```
    {
```

```
        if( typeid( *tab[i] ) ==typeid(point))
```

```
            f<<"1 "<<*tab[i]<<endl;
```

```
        else if( typeid( *tab[i] ) ==typeid(pointColore))
```

```
            f<<"2 "<<static_cast< pointColore&> (*tab[i])<<endl;
```

```
        else if( typeid( *tab[i] ) ==typeid(pointColoreMasse))
```

```
            f<<"3 "<<static_cast< pointColoreMasse&> (*tab[i])<<endl;
```

```
    }
```

```
    f.close();
```

```
}
```

```
void courbe::enregistrer()
```

```
{
```

```
    fstream f;
```

```
    f.open("d:\\courbe.txt", ios::out | ios::trunc);
```

```
    if(! f.is_open()) exit(-1);
```

```
    f<<this;
```

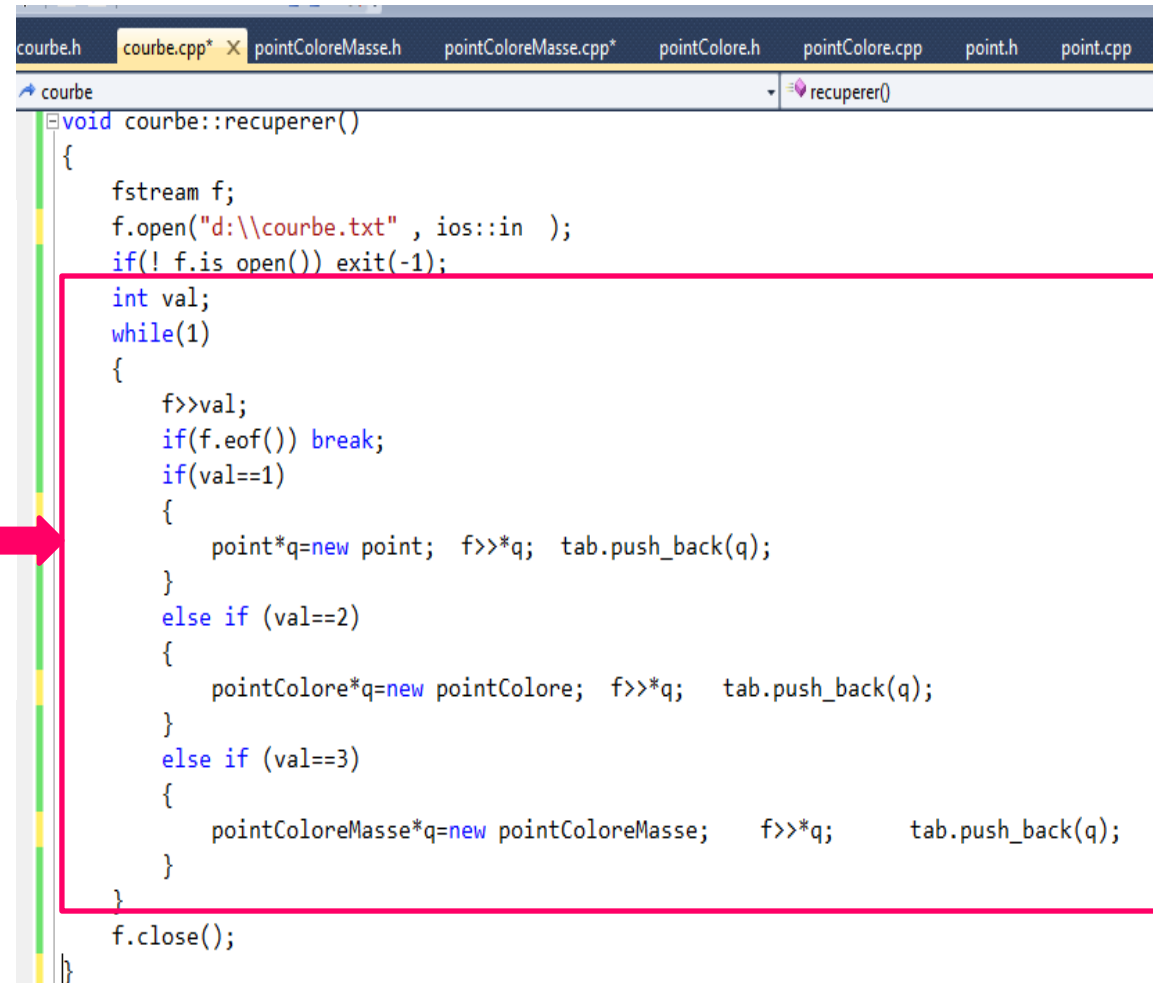
```
    f.close();
```

```
}
```

# Charger/recuperer

```
#include "courbe.h"
void main()
{
    courbe c;
    cin>>c;
    cout<<"\n-----"<<endl;
    c.enregistrer();
    cout<<"\n-----"<<endl;
    courbe d;
    d.recuperer();
    cout<<d;
    system("PAUSE");
}
```

```
void courbe::recuperer()
{
    fstream f;
    f.open("d:\\courbe.txt", ios::in );
    if(! f.is_open()) exit(-1);
    f>>this;
}
```



```
void courbe::recuperer()
{
    fstream f;
    f.open("d:\\courbe.txt", ios::in );
    if(! f.is_open()) exit(-1);
    int val;
    while(1)
    {
        f>>val;
        if(f.eof()) break;
        if(val==1)
        {
            point*q=new point; f>>*q; tab.push_back(q);
        }
        else if (val==2)
        {
            pointColore*q=new pointColore; f>>*q; tab.push_back(q);
        }
        else if (val==3)
        {
            pointColoreMasse*q=new pointColoreMasse; f>>*q; tab.push_back(q);
        }
    }
    f.close();
}
```

# Classe point

```
class point
{
    static int taille; // taille d'une ligne dans le fichier
    static int nbLignes; // nbre de lignes dans le fichiers
protected:
    int x;
    int y;
public:
    friend ostream& operator<< (ostream&, point&);
    friend istream& operator>> (istream&, point&);
    static void creer(fstream&);
    static void remplir(fstream&);
    static void afficher(fstream&);

    static void supprimer(fstream&,int); //supprimer une ligne du fichier
    void modifier(fstream&,int); // modifier une ligne par une autre
    static int chercher(fstream&, int); // retourner un num ligne selon un identificateur
    void inserer (fstream&, int); // insérer une ligne à une position donnée

    point(int =88, int =88):
```

# Clear()

```
void main()
{
    fstream f;
    point::creer(f);
    point::remplir(f);
    point::afficher(f);

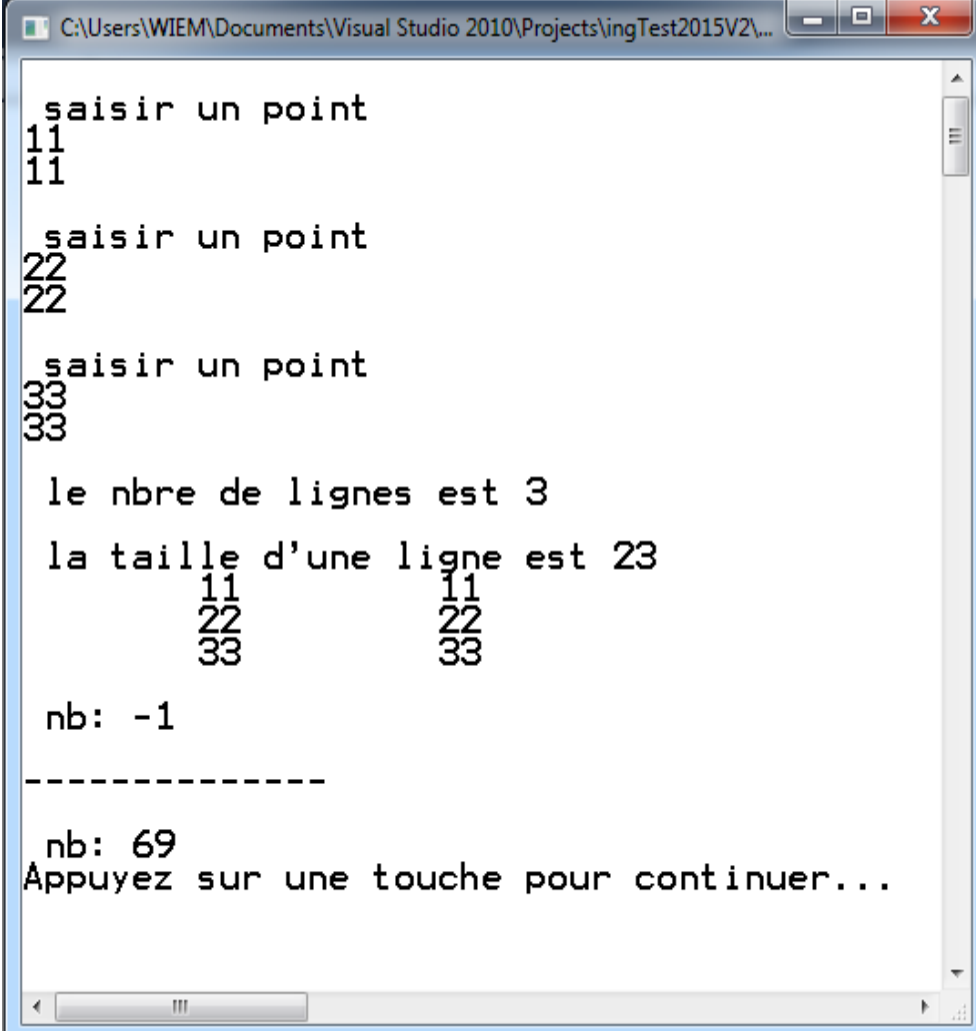
    int nb=f.tellg();
    cout<<"\n nb: "<<nb<<endl;

    cout<<"\n-----"<<endl;

    f.clear();

    nb=f.tellg();
    cout<<"\n nb: "<<nb<<endl;

    f.close();
    system("PAUSE");
}
```



```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\...

saisir un point
11
11

saisir un point
22
22

saisir un point
33
33

le nbre de lignes est 3
la taille d'une ligne est 23
11 11
22 22
33 33

nb: -1
-----

nb: 69
Appuyez sur une touche pour continuer...
```



# Chercher un élément dans le fichier courbe

```
class courbe
{
    vector<point*> tab;

public:
    courbe();
    static void creer(fstream&);

    static int chercher(fstream&, point);
    int chercher(point);
}
```

```
int courbe::chercher(fstream& f, point a)
{
    courbe d;
    f>>&d;

    int res=d.chercher(a);
    return res;
}
```

```
int courbe::chercher(point p)
{
    for(int i=0; i<tab.size(); i++)
        if( (p.getX()== tab[i]->getX() )&& (p.getY()== tab[i]->getY() ) )
            return i;
    return -1;
}
```

```

void main()
{
    fstream f;
    courbe::creer(f);
    courbe c;
    cin>>c;
    f<<&c;
    cout<<"\n-----"<<endl;
    // chercher le numero de la ligne où
    // se trouve l'objet a
    point a(33,33);
    int pos=courbe::chercher(f,a);
    if(pos==-1) cout<<"\n objet n'existe pas "<<endl;
    else cout<<"\n objet existe ligne "<<pos<<endl;

    f.close();
    system("PAUSE");
}

```

```

C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debug\ingTest201...
taper 1: point, 2: pointColore, 3: pointColoreMas
1
22
22
ajouter ?
o
taper 1: point, 2: pointColore, 3: pointColoreMas
1
33
33
ajouter ?
n
-----
objet existe ligne 1
Appuyez sur une touche pour continuer... █

```

# Insérer un pointCouleur dans le fichier courbe à une ligne donnée

The image shows a Visual Studio 2010 environment with a C++ project. The main window displays the code for `main()` in `test.cpp`. The code involves file operations using `fstream` to create, read, and write to a file named `courbe`. Comments in green indicate the purpose of each step: filling a vector `c` from the keyboard, copying it to a file, reading it back, adding a new colored point, and then copying the updated data back to the file. A `PAUSE` system call is used to halt execution.

```
void main()
{
    fstream f;
    courbe::creer(f);
    courbe c;
    cin>>c; // remplir c du clavier
    f<<&c; // copier c dans le fichier
    cout<<"\n-----"<<endl;
    f.clear();
    f.seekg(0);
    courbe d;
    f>>&d; // copier le fichier dans d
    pointCouleur pc(99,99,99);
    d.ajouter(pc,1); // faire le traitement dans d
    f.clear();
    f.seekg(0);
    f<<&d; // recopier d dans le fichier
    system("PAUSE");
}
```

The console window shows the program's execution flow. It prompts the user to enter data for three points (1, 2, 3) with their respective x, y, and mass values. The user enters 11, 11, 11 for the first point. Then, it asks 'ajouter ?' (add?). The user enters 0. The process repeats for the second point (22, 22, 22) and the third point (33, 33, 33). After the third point, the user enters 'n' for 'ajouter ?'. The program then displays a separator line and asks the user to press a key to continue.

The 'fichierCourbe - Bloc-notes' window shows the contents of the file `courbe` after the first two points have been added. The file contains the following data:

Fichier	Edition	Format	Affichage	?
1	11	11		
2	99	99	99	
2	22	22	22	
3	33	33	33	33

# Exemples

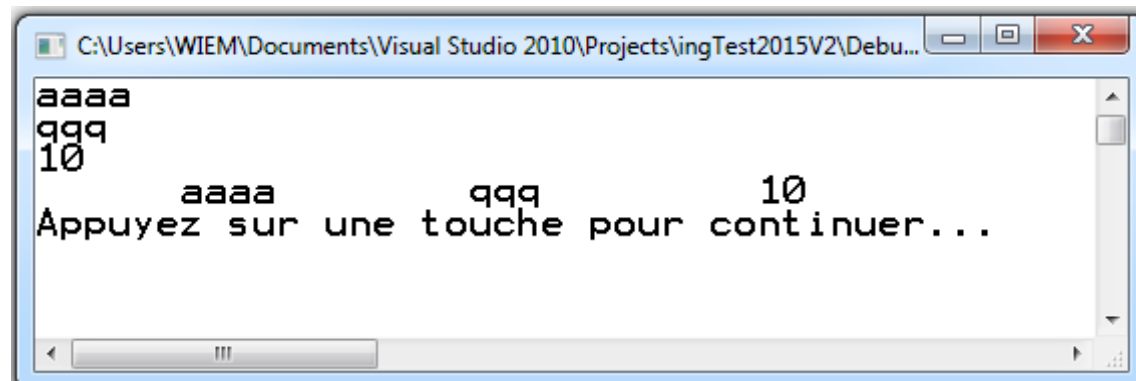
- Lecture et écriture d'un fichier à partir d'un vector d'éléments (exemple courbe)
- Lecture et écriture d'un fichier à partir d'une liste chaînée d'éléments (exemple voiture-parcVoiture)

```
voiture.cpp  voiture.h  parcVoiture.h  parcVoiture.cpp  test.cpp
(Global Scope)
class voiture
{
    string marque;
    string modele;
    float prix;
public:
    voiture*suivant;
    voiture(string = "", string="", float=0.0);
    ~voiture(void);
    friend ostream& operator<< (ostream&, voiture&);
    friend istream& operator>> (istream&, voiture&);
};
```

```
voiture.cpp*  voiture.h  parcVoiture.h  parcVoiture.cpp  test.cpp
voiture
voiture(string ma, string mo, float p)
{
    marque=ma;
    modele=mo;
    prix=p;
    suivant=NULL;
}
voiture::~voiture(void) { ... }
ostream& operator<< (ostream& out, voiture& v)
{
    out<<setw(10)<<v.marque<<" "<<setw(10)<<v.modele<<" "<<setw(10)<<v.prix;
    return out;
}
istream& operator>> (istream& in, voiture& v)
{
    in>>v.marque;
    in>>v.modele;
    in>>v.prix;
    return in;
}
```

# test

```
void main()
{
    voiture v;
    cin>>v;
    cout<<v<<endl;
    system("PAUSE");
}
```

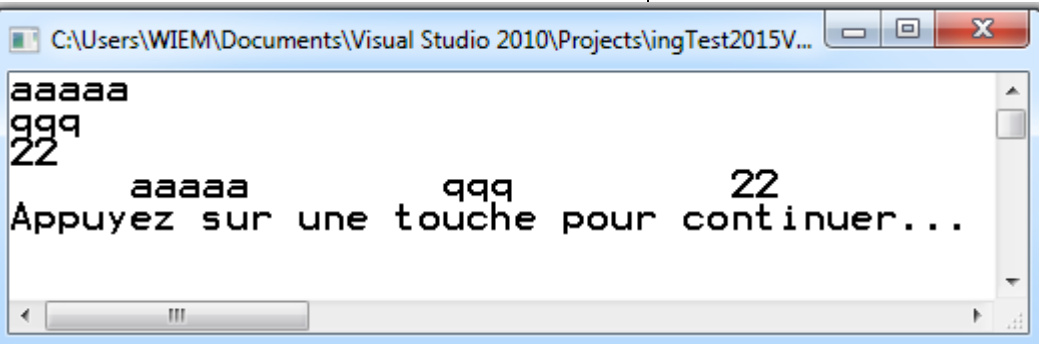


```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debu...
aaaa
qqq
10
Appuyez sur une touche pour continuer...
```

# remarque

```
class voiture
{
    string marque;
    string modele;
    float prix;
public:
    voiture*suivant;
    voiture(string = "", string = "", float=10);
    ~voiture(void);
    friend ostream& operator<< (ostream&, voiture&);
    friend istream& operator>> (istream&, voiture&);
    // on peut surcharger operator<< et operator>>
    friend ostream& operator<< (ostream&, voiture*);
    friend istream& operator>> (istream&, voiture*);
};
```

```
void main()
{
    voiture *v=new voiture;
    cin>>v;
    cout<<v<<endl;
    system("PAUSE");
}
```



```
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V...
aaaaa
qqq
22
aaaaa      qqq      22
Appuyez sur une touche pour continuer...
```

```

#include "voiture.h"
voiture::voiture(string ma, string mo, float p)
{
    marque=ma;
    modele=mo;
    prix=p;
    suivant=NULL;
}
voiture::~voiture(void)
{
}

```

```

ostream& operator<< (ostream& out, voiture &v)
{
    out<<setw(10)<<v.marque<<" "<<setw(10)<<v.modele<<" "<<setw(7)<<v.prix<<endl;
    return out;
}
istream& operator>> (istream& in, voiture& v)
{
    in>>v.marque;
    in>>v.modele;
    in>>v.prix;
    return in;
}

```



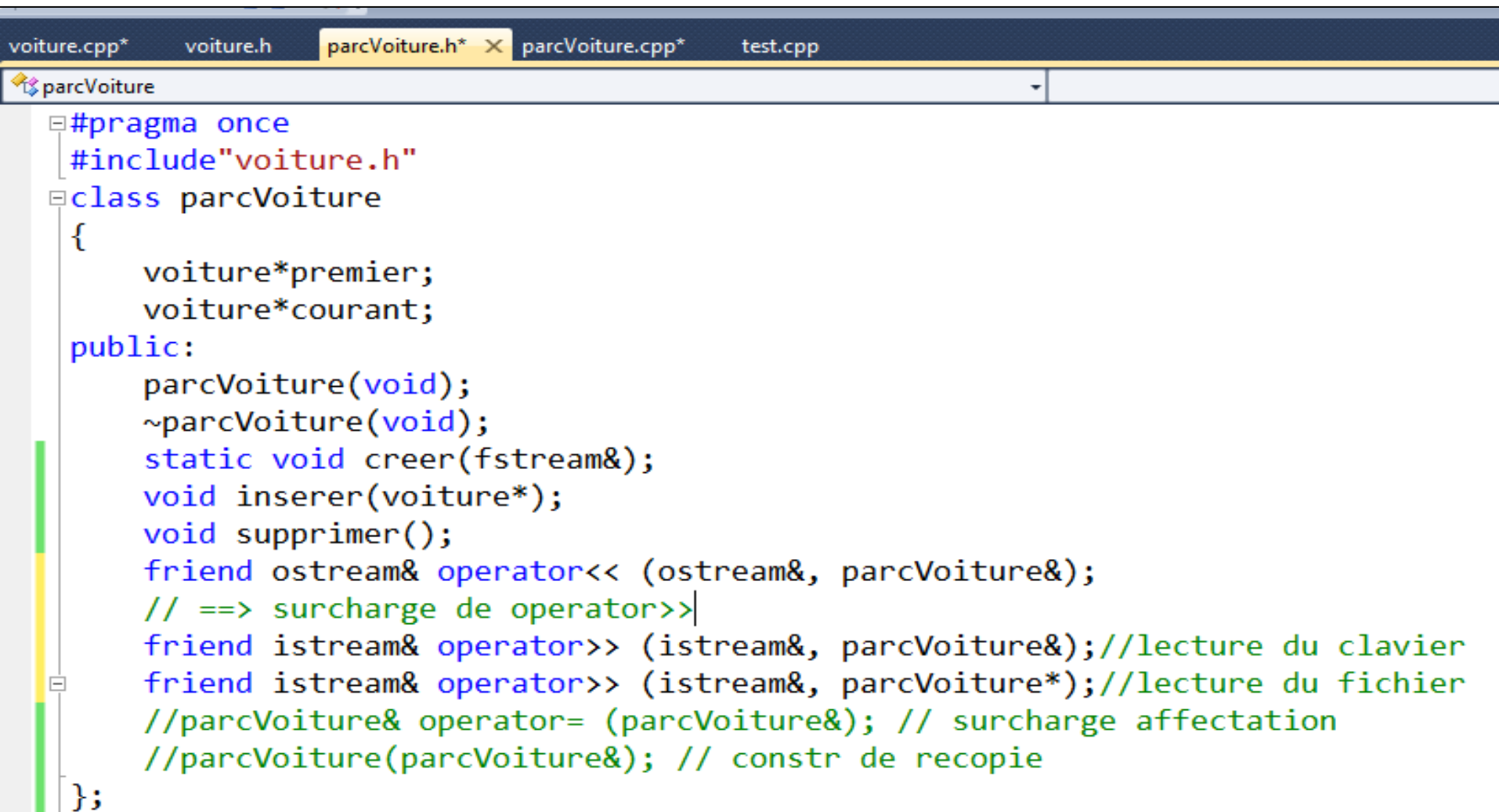
```

ostream& operator<< (ostream& out, voiture *v)
{
    out<<setw(10)<<v->marque<<" "<<setw(10)<<v->modele<<" "<<setw(7)<<v->prix<<endl;
    return out;
}

istream& operator>> (istream& in, voiture* v)
{
    in>>v->marque;
    in>>v->modele;
    in>>v->prix;
    return in;
}

```

# Classe parcVoiture



The screenshot shows a code editor with several tabs at the top: `voiture.cpp*`, `voiture.h`, `parcVoiture.h*` (selected), `parcVoiture.cpp*`, and `test.cpp`. The editor displays the content of `parcVoiture.h`. The code defines a `parcVoiture` class with two private pointers, `voiture*premier` and `voiture*courant`. It includes a `public` section with several methods: a default constructor `parcVoiture(void)`, a destructor `~parcVoiture(void)`, a static method `creer(fstream&)`, and three non-static methods `inserer(voiture*)`, `supprimer()`, and a friend function `operator<<`. Additionally, there are three green comments: `// ==> surcharge de operator>>`, `//lecture du clavier`, and `//lecture du fichier`. The class ends with a closing brace and a semicolon.

```
#pragma once
#include"voiture.h"
class parcVoiture
{
    voiture*premier;
    voiture*courant;
public:
    parcVoiture(void);
    ~parcVoiture(void);
    static void creer(fstream&);
    void inserer(voiture*);
    void supprimer();
    friend ostream& operator<< (ostream&, parcVoiture&);
    // ==> surcharge de operator>>
    friend istream& operator>> (istream&, parcVoiture&); //lecture du clavier
    friend istream& operator>> (istream&, parcVoiture*); //lecture du fichier
    //parcVoiture& operator= (parcVoiture&); // surcharge affectation
    //parcVoiture(parcVoiture&); // constr de copie
};
```

```
voiture.cpp*  voiture.h  parcVoiture.h  parcVoiture.cpp*  test.cpp
(Global Scope)
#include "parcVoiture.h"
parcVoiture::parcVoiture(void)
{
    premier=NULL;
    courant=NULL;
}
parcVoiture::~parcVoiture(void)
{
    while(premier!=NULL)
        supprimer();
}
void parcVoiture::inserer(voiture *v)
{
    v->suivant=premier;
    premier=v;
}
```

```
void parcVoiture::supprimer()
{
    if(premier!=NULL)
    {
        courant=premier;
        premier=premier->suivant;
        delete courant;
    }
}
void parcVoiture::creer(fstream& f)
{
    f.open("d:\\parcVoitures.txt", ios::in | ios::out | ios::trunc);
    if(! f.is_open()) exit(-1);
}
```

```
voiture.cpp*  voiture.h  parcVoiture.h  parcVoiture.cpp*  test.cpp
(Global Scope)
ostream& operator<< (ostream& out, parcVoiture& p)
{
    p.courant=p.premier;
    while(p.courant!=NULL)
    {
        out<<*p.courant<<endl;
        p.courant=p.courant->suivant;
    }
    return out;
}
```

# Surcharge operator>>

```
// operateur de lecture à partir du clavier
istream& operator>> (istream& in, parcVoiture& p)
{
    voiture*v;
    char rep;
    do
    {
        v=new voiture;
        in>>*v;
        p.inserer(v);
        cout<<"\n ajouter voiture? "<<endl;
        cin>>rep;
    }
    while(rep=='o' || rep=='O');
    return in;
}
```

```
// operateur de lecture à partir du fichier
istream& operator>> (istream& in, parcVoiture* p)
{
    voiture*v;
    char rep;
    in.seekg(0);
    while(1)
    {
        v=new voiture;
        in>>*v;
        if(in.eof()) break;
        p->inserer(v);
    }
    return in;
}
```

```
void main()
{
    parcVoiture p;
    cin>>p;
    // istream& operator>> (istream&, parcVoiture&);
    cout<<p;
    system("PAUSE");
}
```

```
void main()
{
    fstream f;
    parcVoiture::creer(f);
    // le fichier existe et contient des données
    parcVoiture p;
    f>>p;
    // istream& operator>> (istream&, parcVoiture*);
    cout<<p;
    system("PAUSE");
}
```

```

test.cpp x
(Global Scope)
void main()
{
    fstream f;
    parcVoiture::creer(f);
    parcVoiture p,q;
    cin>>p;
    cout<<p;
    f<<p;
    cout<<"\n-----"<<endl;
    f>>q;
    cout<<q;
    system("PAUSE");
}

```

```

C:\Users\WIEM\Documents\Visual Studio 2010\Projects\TEST2015\Debug\TE...
aaa
aaaa
11

ajouter voiture?
o
bbb
bbbbbb
22

ajouter voiture?
n
      bbb      bbbbbb      22
      aaa      aaaa      11
-----
      aaa      aaaa      11
      bbb      bbbbbb      22
Appuyez sur une touche pour continuer...

```

```

parcVoitures - Bloc-notes
Fichier  Edition  Format  Affichage  ?
|
      bbb      bbbbbb      22
      aaa      aaaa      11

```

# La méthode enregistrer

```
void main()
{
    parcVoiture p;
    cin>>p;
    p.enregistrer("fichierParc");
    cout<<"\n-----"<<endl;

    system("PAUSE");
}
```

```
void parcVoiture::enregistrer()
{
    fstream f;
    string nom;
    cout<<"\n saisir nom fichier "<<endl;
    cin>>nom;
    f.open(nom, ios::in | ios::out | ios::trunc);
    if( !f.is_open()) exit(-1);

    f<<*this;

    f.close();
}
```

```
void parcVoiture::enregistrer()
{
    fstream f;
    string nom;
    cout<<"\n saisir nom fichier "<<endl;
    cin>>nom;
    f.open(nom, ios::in | ios::out | ios::trunc);
    if( !f.is_open()) exit(-1);
    courant=premier;
    while(courant!=NULL)
    {
        f<<courant;
        courant=courant->suivant;
    }
    f.close();
}
```

# La méthode recuperer/ charger

```
parcVoiture.h  parcVoiture.cpp  voiture.h  voiture.cpp  main.cpp
(Global Scope)

void main()
{
    parcVoiture p;

    p.recuperer();
    cout<<"\n-----"<<endl;
    cout<<p;

    system("PAUSE");
}
```

```
void parcVoiture::recuperer()
{
    fstream f;
    string nom;
    cout<<"\n saisir nom fichier "<<endl;
    cin>>nom;
    f.open(nom, ios::in);
    if(! f.is_open()) exit(-1);
    f>>this;
    f.close();
}
```

```
parcVoiture.h  parcVoiture.cpp*  voiture.h  voiture.cpp  main.cpp
parcVoiture

void parcVoiture::recuperer()
{
    fstream f;
    string nom;
    cout<<"\n saisir nom fichier "<<endl;
    cin>>nom;
    f.open(nom, ios::in);
    if(! f.is_open()) exit(-1);
    while(1)
    {
        voiture*q=new voiture;
        f>>q;
        if(f.eof()) break;
        inserer(q);
    }
    f.close();
}
```

# Les accès directs:

- Si le fichier est ouvert en mode **lecture seulement**, alors utiliser **seekg** et **tellg (g:get)**
- Si le fichier est ouvert en mode **écriture seulement**, alors utiliser **seekp** et **tellp (p:put)**
- Si le fichier est ouvert en mode **lecture et écriture** alors:  
(**seekg** ⇔ **seekp**) et (**tellg** ⇔ **tellp**)



# Seekp & seekg

- seekp(n) (seekg idem): se positionner directement après le n-ème caractère du fichier

```
f.seekp(0); // se positionner au début du fichier  
f.seekp(5); // se positionner après le 5ème caractère
```

- seekp( n, dir); (avec dir prend 3 valeurs possibles)
  - 0 (ios::beg): déplacement de n caractères par rapport au début du fichier
  - 1 (ios::cur): déplacement de n caractères par rapport à la position courante. Si n est négatif, alors c'est un déplacement vers l'arrière
  - 2 (ios::end): déplacement de n caractères par rapport à la fin du fichier

```
f.seekp(20, 0); // avancer de 20 caractères par rapport au début du fichier  
f.seekp(6, 1); // avancer de 6 caractères par rapport à la position courante  
f.seekp(-8, 1); // reculer de 8 caractères par rapport à la position courante  
f.seekp(0,2); // se positionner à la fin  
f.seekp(-12, 2); // reculer de 12 caractères par rapport à la fin
```

# Les accès directs

- La table suivante présente les différentes fonctions permettant de donner ou de modifier les positionnements dans un flux.

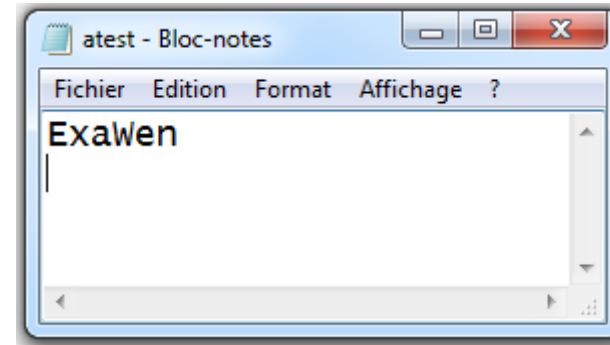
Classe	fonction membre	tâche
<code>basic_istream&lt;&gt;</code>	<code>tellg()</code>	retourne la position courante dans le flot en lecture
	<code>seekg(pos)</code>	se positionne en lecture à l'index <code>pos</code>
	<code>seekg(offset, dir)</code>	se positionne en lecture à l'index <code>offset</code> , par rapport à la direction définie par <code>dir</code> .
<code>basic_ostream&lt;&gt;</code>	<code>tellp()</code>	retourne la position courante dans le flot en écriture
	<code>seekp(pos)</code>	se positionne en écriture à l'index <code>pos</code>
	<code>seekp(offset, dir)</code>	se positionne en écriture à l'index <code>offset</code> , par rapport à la direction définie par <code>dir</code> .

```

void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl; // 6 caracteres
    f.seekg(3);
    f<<"W";
    f.close();
    system("PAUSE");
}

```

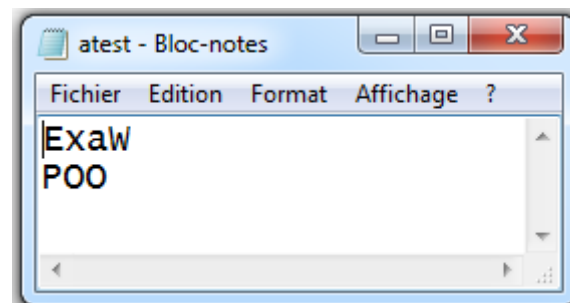


```

void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"ExamenPOO"<<endl;
    f.seekg(3);
    f<<"W"<<endl;
    f.close();
    system("PAUSE");
}

```



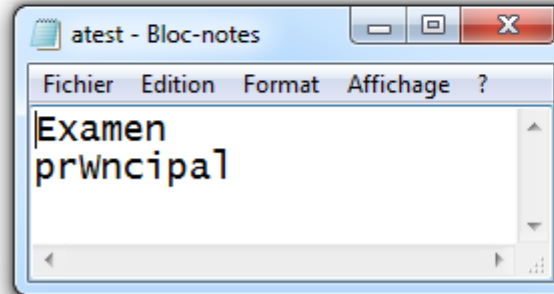
# exemple2

```
void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl; // 6 caractères
    f<<"principal"<<endl; // 9 caractères

    f.seekg(10);

    f<<"W";
    f.close();
    system("PAUSE");
}
```

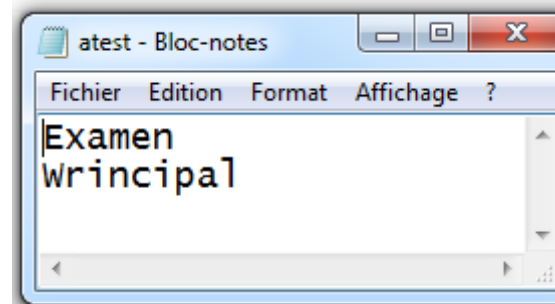


```

void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl;
    f<<"principal"<<endl;
    f.seekg(8);
    f<<"W";
    f.close();
    system("PAUSE");
}

```

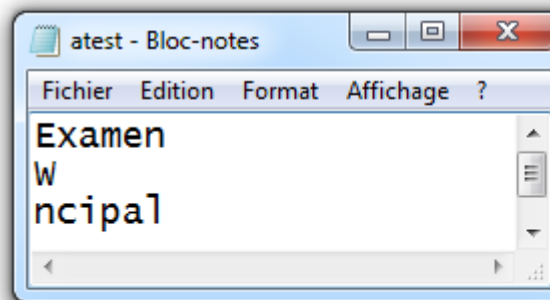


```

void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl;
    f<<"principal"<<endl;
    f.seekg(8);
    f<<"W"<<endl;
    f.close();
    system("PAUSE");
}

```



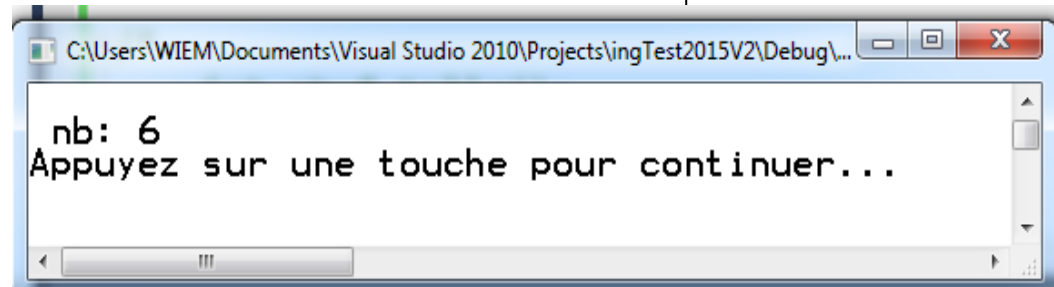
# tellg tellp

```
void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"; // 6 caractères

    int nb=f.tellg();
    cout<<"\n nb: "<<nb<<endl;

    f.close();
    system("PAUSE");
}
```



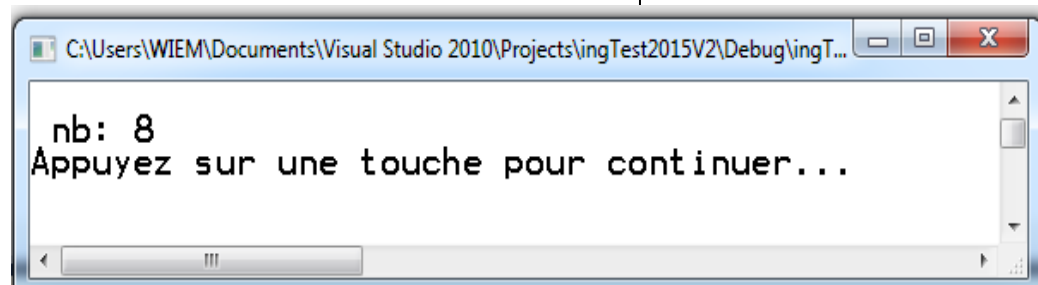
C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debug\...  
nb: 6  
Appuyez sur une touche pour continuer...

```
void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl; // 6 caractères

    int nb=f.tellg();
    cout<<"\n nb: "<<nb<<endl;

    f.close();
    system("PAUSE");
}
```



C:\Users\WIEM\Documents\Visual Studio 2010\Projects\ingTest2015V2\Debug\ingT...  
nb: 8  
Appuyez sur une touche pour continuer...

# exemple

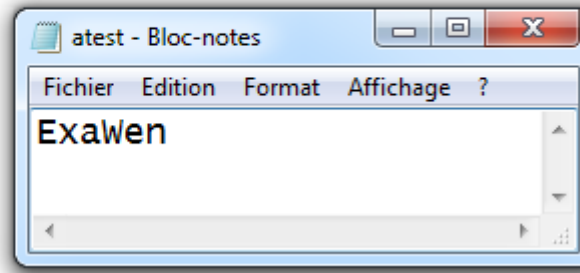
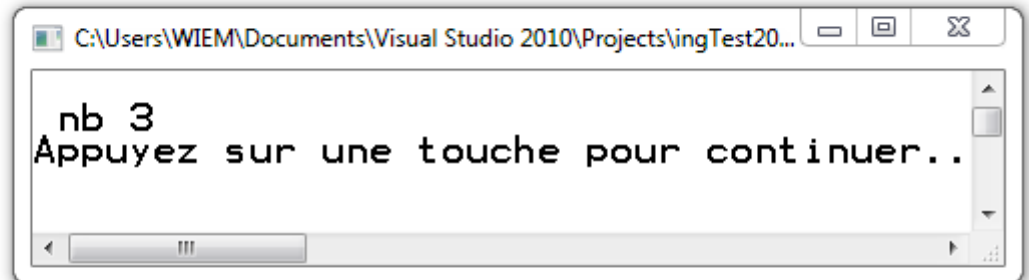
```
void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl;

    f.seekg(3);

    int nb=f.tellg();
    cout<<"\n nb " <<nb<<endl;

    f<<"W";
    f.close();
    system("PAUSE");
}
```



# Nombre total de caracteres: seek et tell

```
void main()
{
    fstream f;
    f.open("d:\\atest.txt", ios::in | ios::out | ios::trunc);

    f<<"Examen"<<endl; // 6 caractères
    f<<"POO"<<endl;    // 3 caractères

    f.seekg(0,2);

    int nb=f.tellg();
    cout<<"\n nbre total de caracteres: "<<nb<<endl;

    f.close();
    system("PAUSE");
}
```

