

# Openframeworks

**CrasLab - Mains d'Œuvres**

*28/02/2009 - 01/03/2009*

# OF : Introduction

- <http://www.openframeworks.cc>
- <http://www.openframeworks.cc/download>
- <http://www.openframeworks.cc/forum>
- <http://wiki.openframeworks.cc>

# OF : Introduction

- Zachary Lieberman ( <http://www.thesystemis.com> )
- Theo Watson ( <http://muonics.net> )
- Communauté de développeurs

# OF : Introduction

- Langage : C++
- Open-source
- Ensemble de bibliothèques
  - création d'images dynamiques (OpenGL comme API graphique)
  - manipulation de sons, de fontes, de données diverses
  - Input / Output

# OF vs Processing

- Java ( Processing )
  - compilation en code intermédiaire, identique pour chaque type de processeur / machines ( “*write once, run everywhere*” )
  - exécution par une “machine virtuelle”
  - fichier .java seulement pour décrire une classe
  - éditeur simple pour créer des programmes

# OF vs Processing

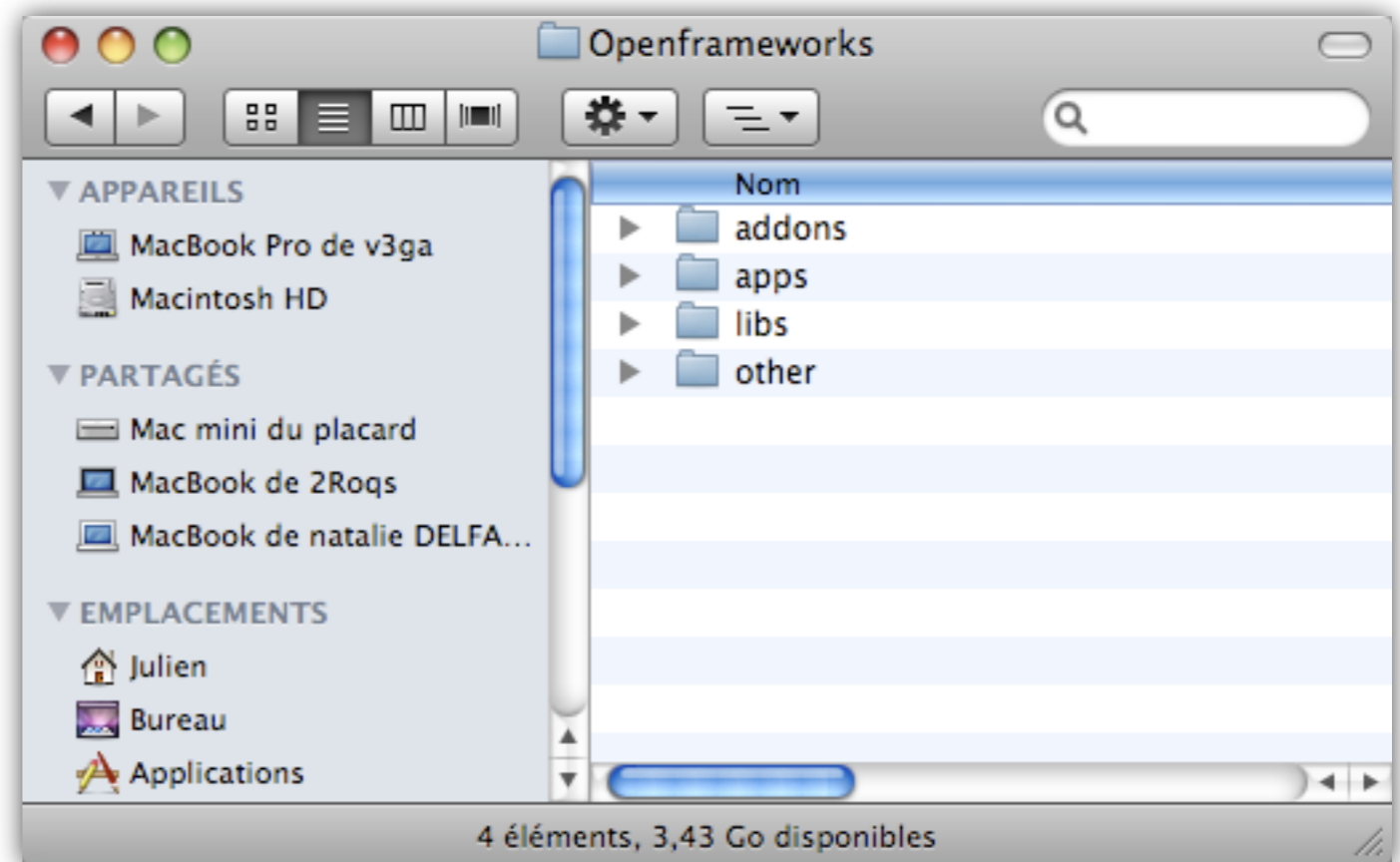
- C++
  - langage “bas-niveau” (vs *langage de script*)
  - compilation en langage machine, dépend de l’architecture du processeur (intel, powerpc, ...)
  - 2 fichiers pour une classe :
    - entête ou header (.h ou .hpp)
    - source (.cpp)

# OF vs Processing

- OF
  - pas de possibilités d'export pour le web
  - pas multi-platerforme (ou moins facilement qu'avec Processing)
  - langage c++ plus difficile à apprendre mais plus puissant

# OF : Dossiers

- **addons**  
Dossier contenant les librairies additionnelles (sources, exemples, doc)
- **apps**  
Dossier contenant les exemples et où l'on stockera nos propres applications (code source, data, fichiers de projet)
- **libs**  
Dossier contenant les entêtes (définitions de classe) + librairies/bibliothèques compilées.





# OF : add-ons

- Bibliothèques de base
- Possibilités d'extensions des fonctionnalités (add-ons)
- <http://addons.openframeworks.cc>

# OF : add-ons

- OSC
  - protocole de communication.
- Thread
  - gestion de taches en parallèle.
- OpenCV
  - détection de contour, de visages, de mouvements.

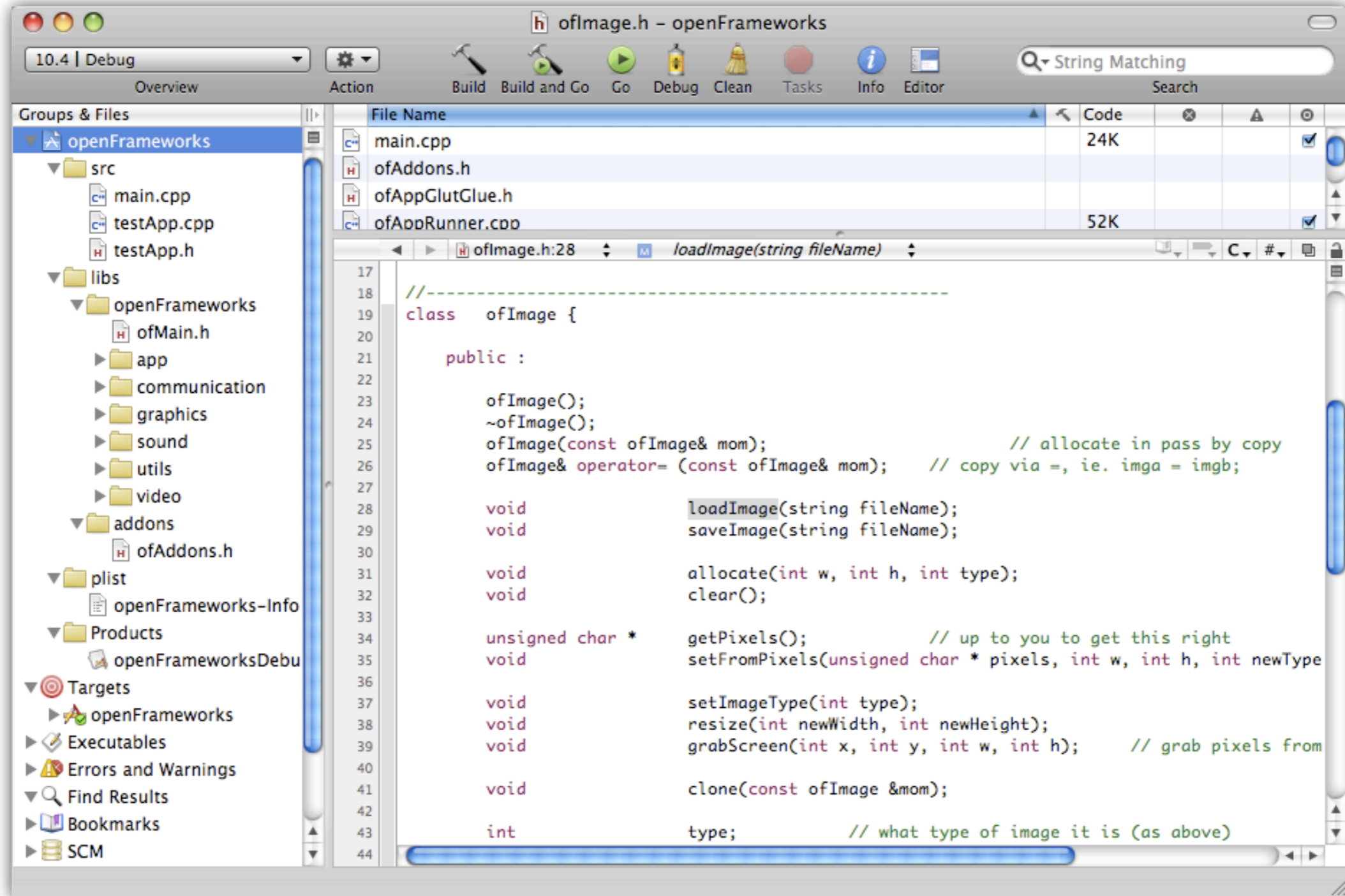
# OF : add-ons et plus!

- Compilation et utilisation d'autres bibliothèques écrites en C++ (ou C)
- Exemples :
  - Opensteer ( <http://opensteer.sourceforge.net> )
  - Box2D ( <http://www.box2d.org> )
  - ODE ( <http://www.ode.org> )
  - ... et d'autres !

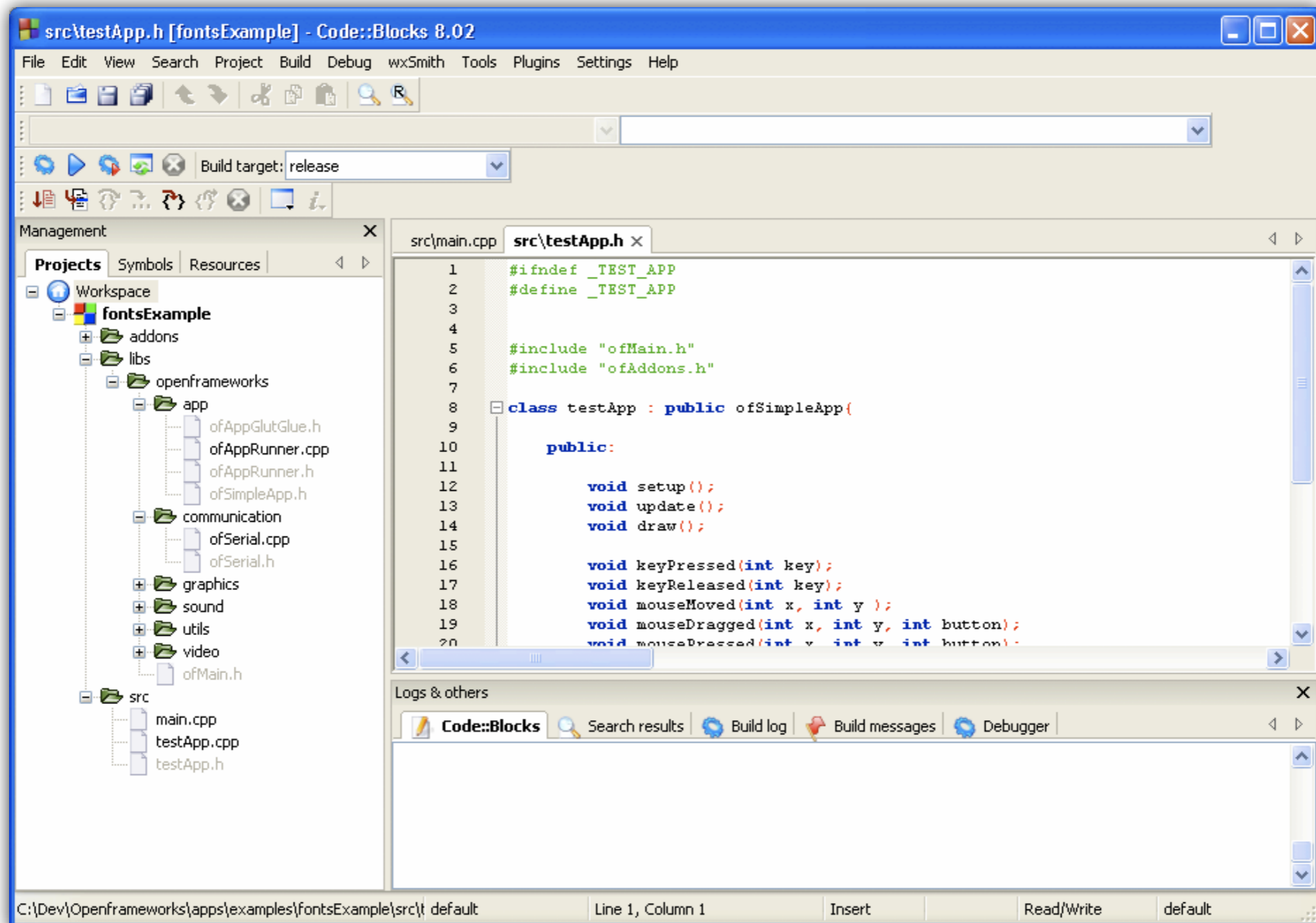
# OF : IDE

- Installation
  - **XCode** sur Mac (Tiger / Leopard)
  - **CodeBlocks** sur Windows (XP / Vista)

# OF : IDE (XCode)



# OF : IDE (CodeBlocks)



# OF : IDE

- Integrated development environment
- **Gestion des fichiers sources :**
  - Edition (coloration syntaxique, complétion)
  - Création (à partir de templates)
  - Exploration des définitions de classes

# OF : IDE

- **Compilation**

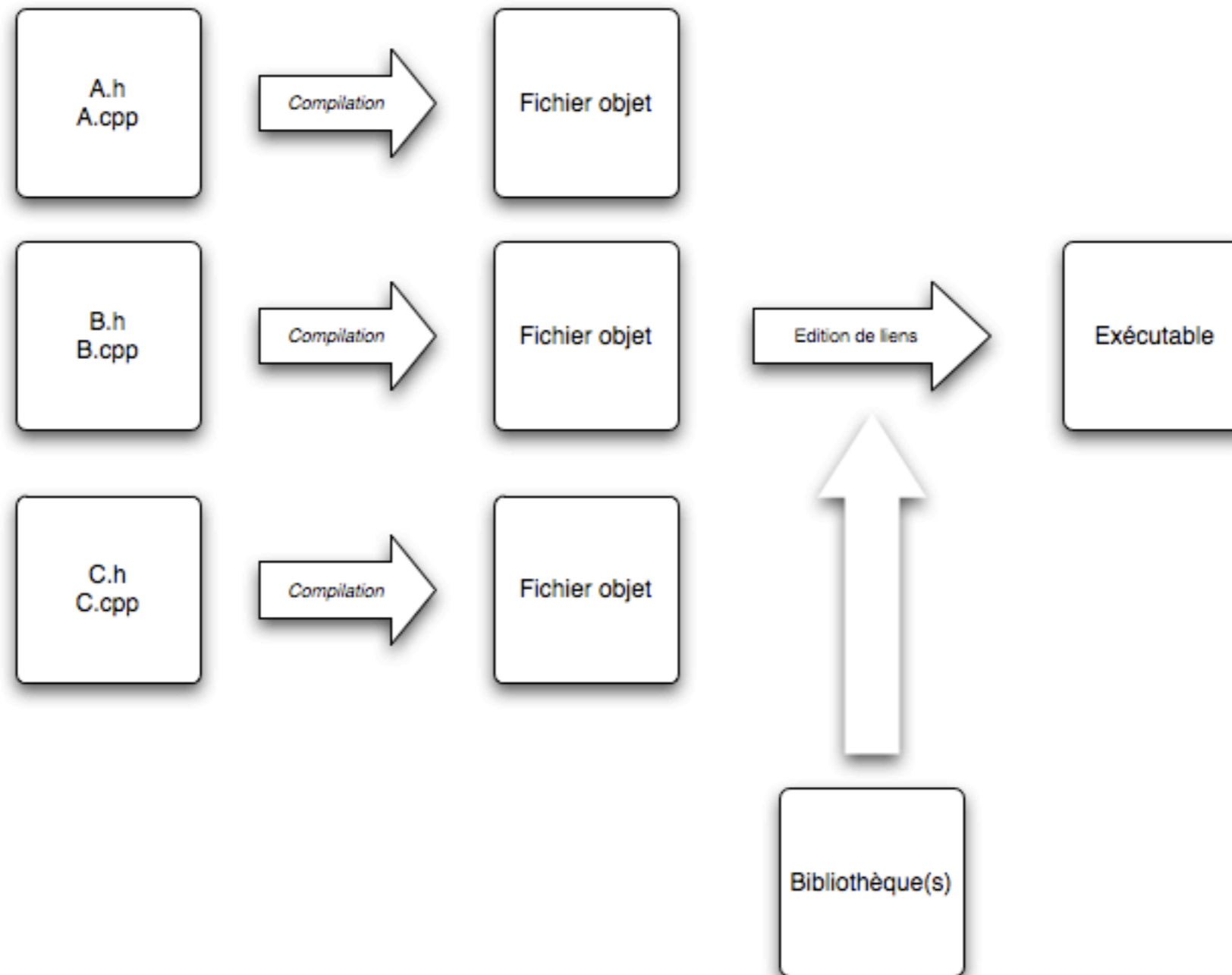
- “*traduction*” des fichiers sources en code machine.

- **Editions de liens**

- liens avec les bibliothèques externes, regroupement des fichiers compilés en un seul fichier (**binary**). Cela peut-être une bibliothèque dynamique, statique ou bien un exécutable.



# OF : IDE - Compilation



# OF : IDE

- **Debugging**

- permet de suivre pas à pas l'exécution d'un programme
- **breakpoint** : point d'arrêt du debugger, permet de scruter l'état du programme (notamment les valeurs de variables par exemple)

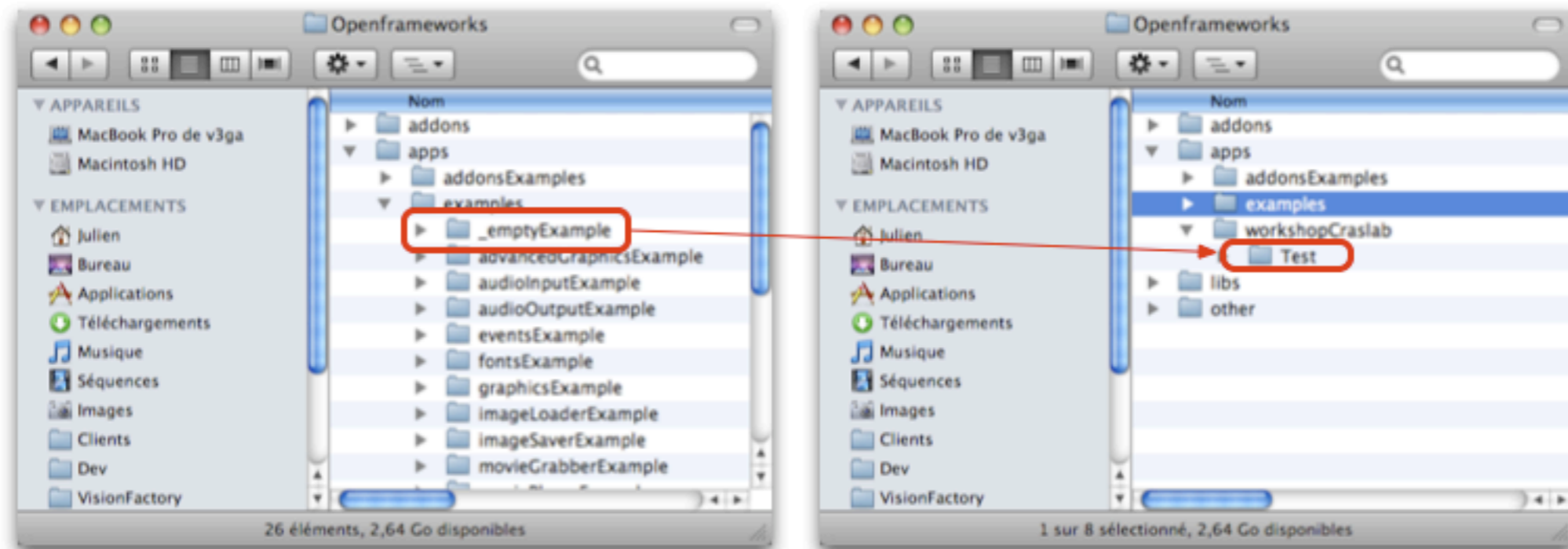
# OF : première compil'

- ouverture d'un projet (*apps/examples/*)
  - fichier ".xcodeproj" sur Mac
  - fichier ".cbp" sur PC
- compile and run
- console

# OF : nouveau projet

- Création d'un dossier "propre" pour la durée de l'atelier (ex : *apps/workshopCraslab/* )
- copie du dossier *apps/examples/\_emptyExample/* dans notre dossier *apps/workshopCraslab/*
- Renommer le nom du dossier *\_emptyExample* en *NomDeMonNouveauProjet*

# OF : nouveau projet



# OF : point d'entrée

- fichier principal : **main.cpp**
- fonction **main()**
  - appelée par le système au démarrage de l'application
  - set-up **OpenGL**, création de la fenêtre
  - instance d'application (**testApp**)

# OF : instance d'appli

- **testApp.h** et **testApp.cpp**
- interface d'évènements (souris + clavier)
- interface de la classe
  - `setup()`
  - `update()`
  - `draw()`

# OF : app.setup()

- fonction appelée une seule fois lors de la création de l'instance de l'application
- lieu d'initialisation de nos variables
- définir le framerate (nombre d'images dessinées par seconde) :  
**ofSetFrameRate()**



# OF : `app.update()`

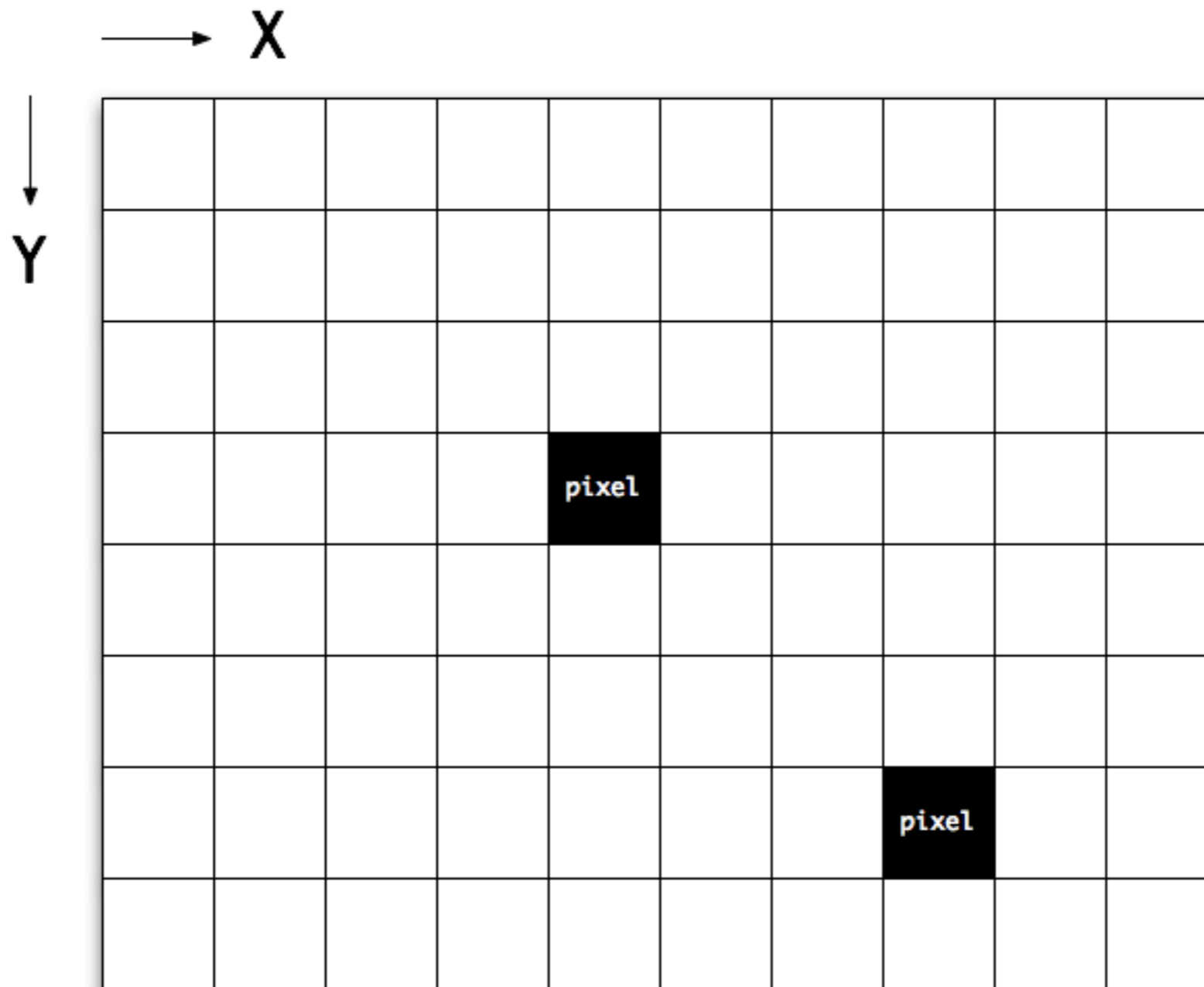
- mise à jour des variables, si celles-ci dépendent du temps (ex : pour animer un objet dans l'espace)
- appelée juste avant `app.draw()`

# OF : `app.draw()`

- fonction principale de dessin
- les commandes de dessin sont placées ici, les variables viennent d'être mises à jour dans le **`update()`** et peuvent être utilisées.

# OF : `app.draw()`

Repère 2D : pixels, coordonnées

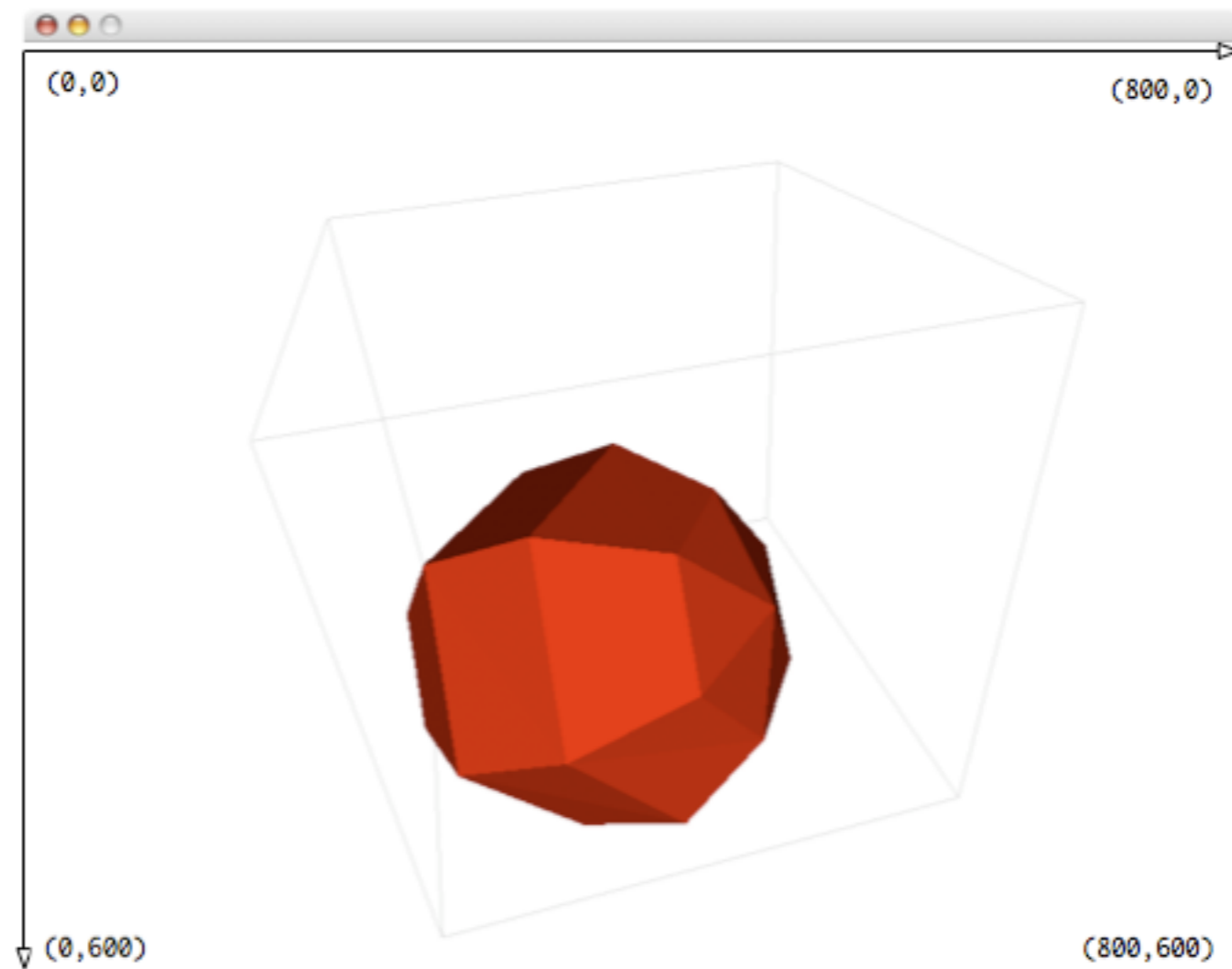


pixel = "picture element"



# OF : `app.draw()`

## Repère 2D



# OF : C++

## Les variables

- Type
- Nom de la variable
  - ne doit pas commencer par un caractère spécial ( '?', '\_ ', espace, chiffres,... )
- Valeur
  - affectation avec l'opérateur =

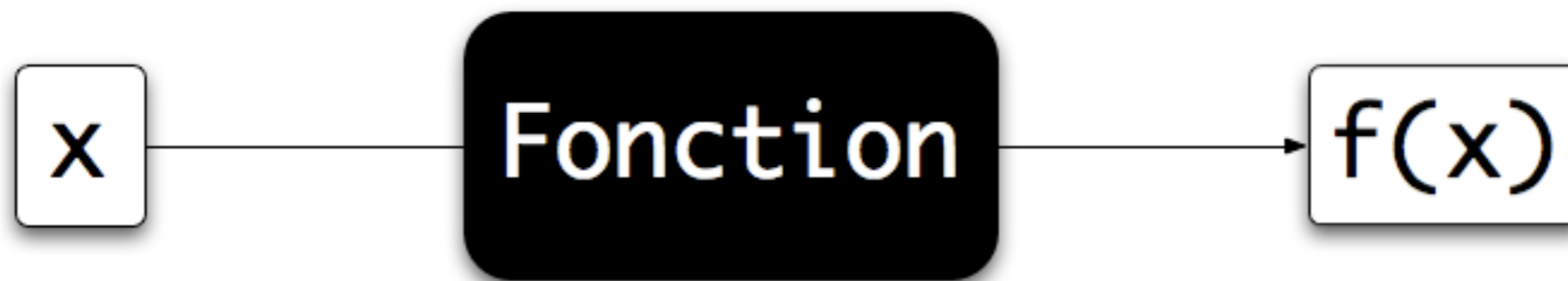
# OF : C++

Variables : types de base

- Type des variables de base
  - *int, boolean, float, char*
- Librairie standard
  - *std::string* (ou *string*)

# OF : C++

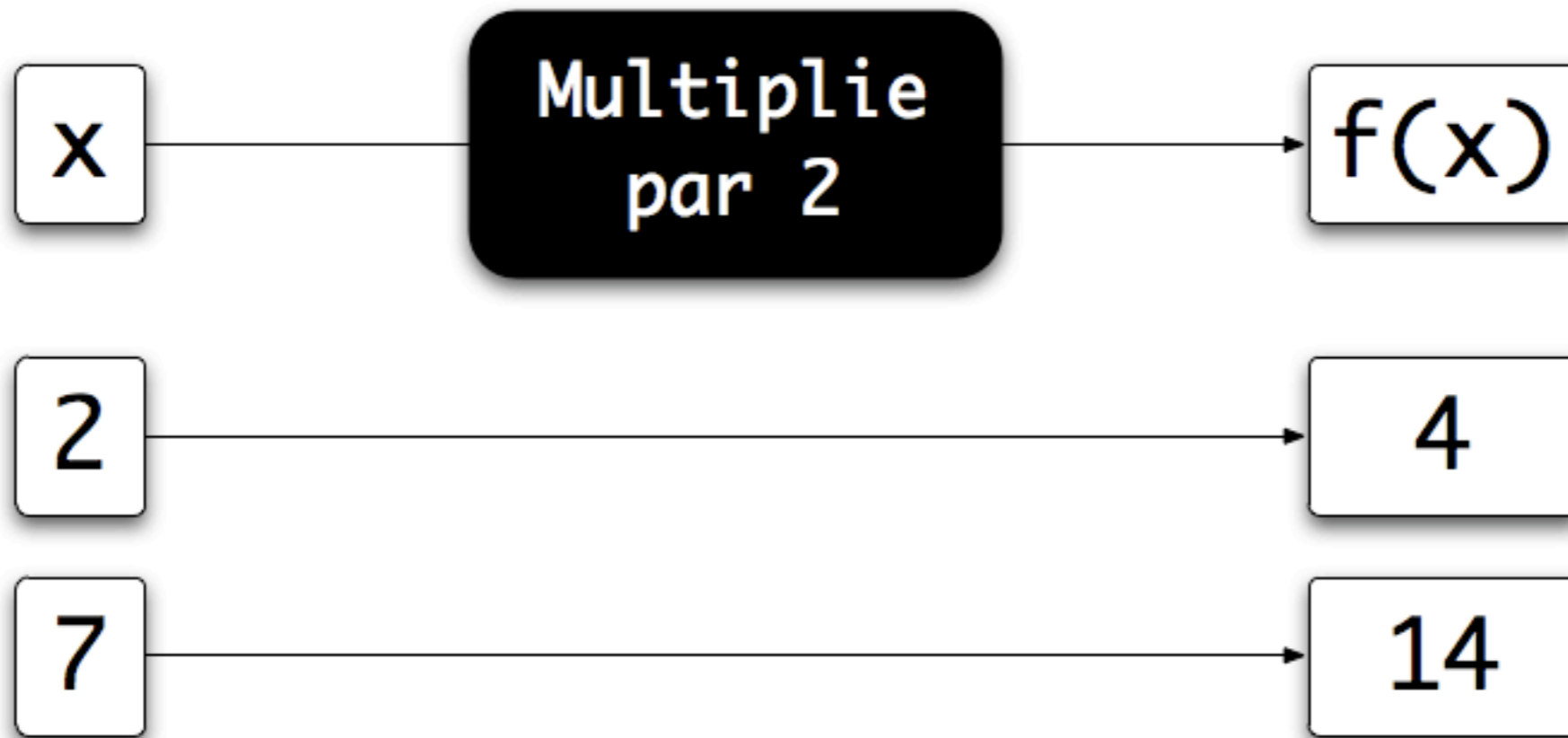
## Les fonctions





# OF : C++

## Les fonctions



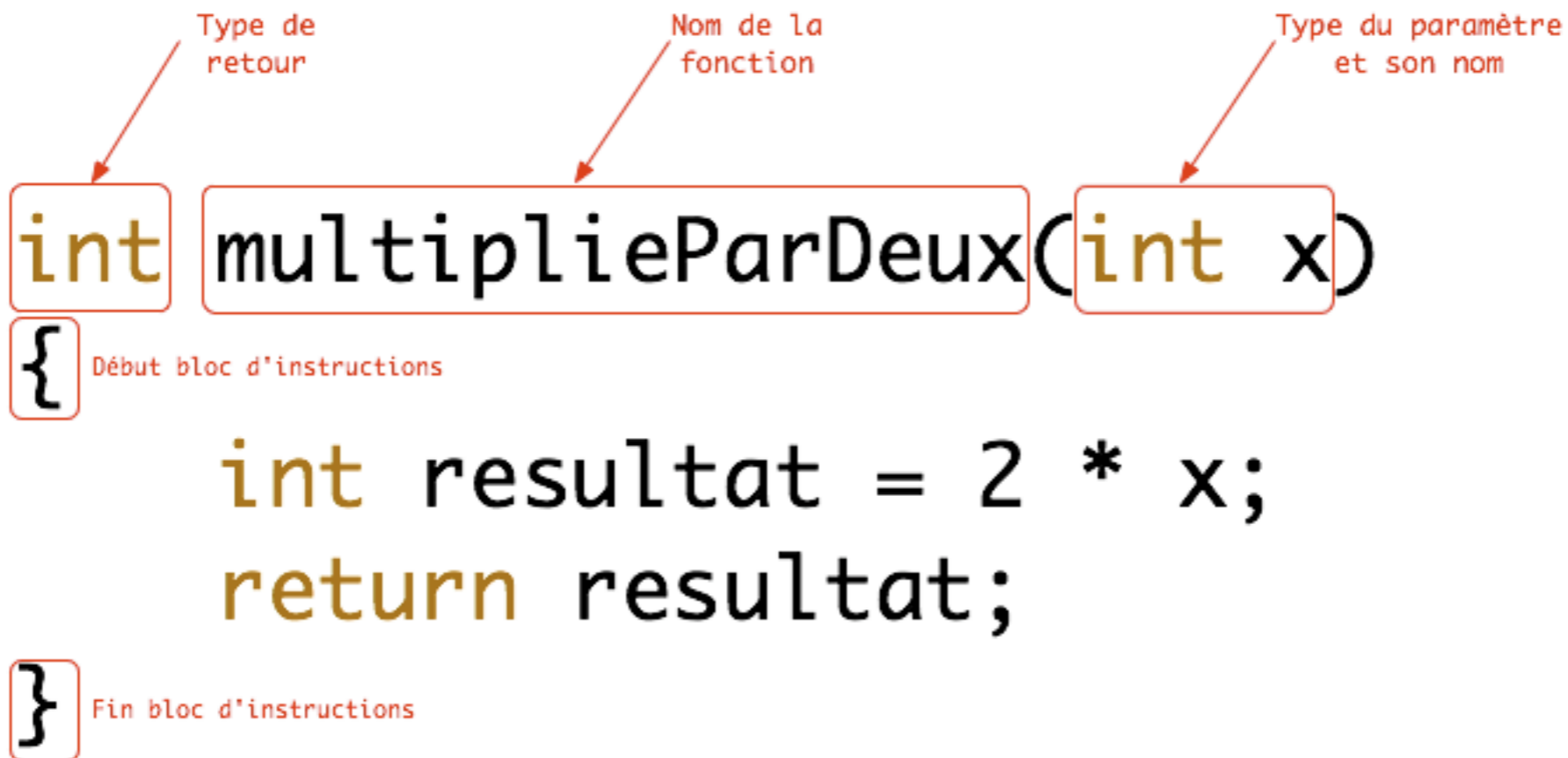
# OF : C++

## Les fonctions



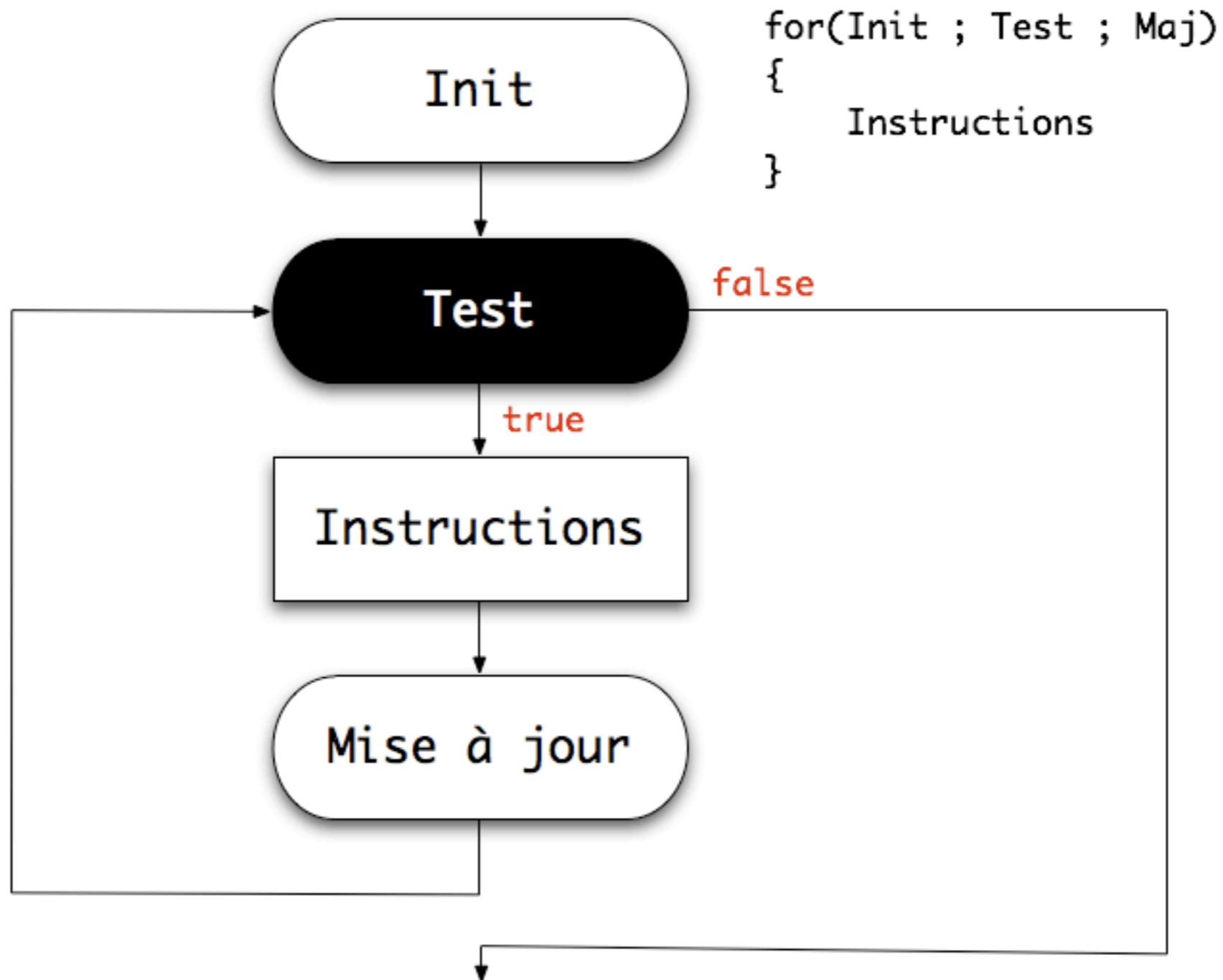
# OF : C++

## Les fonctions : syntaxe



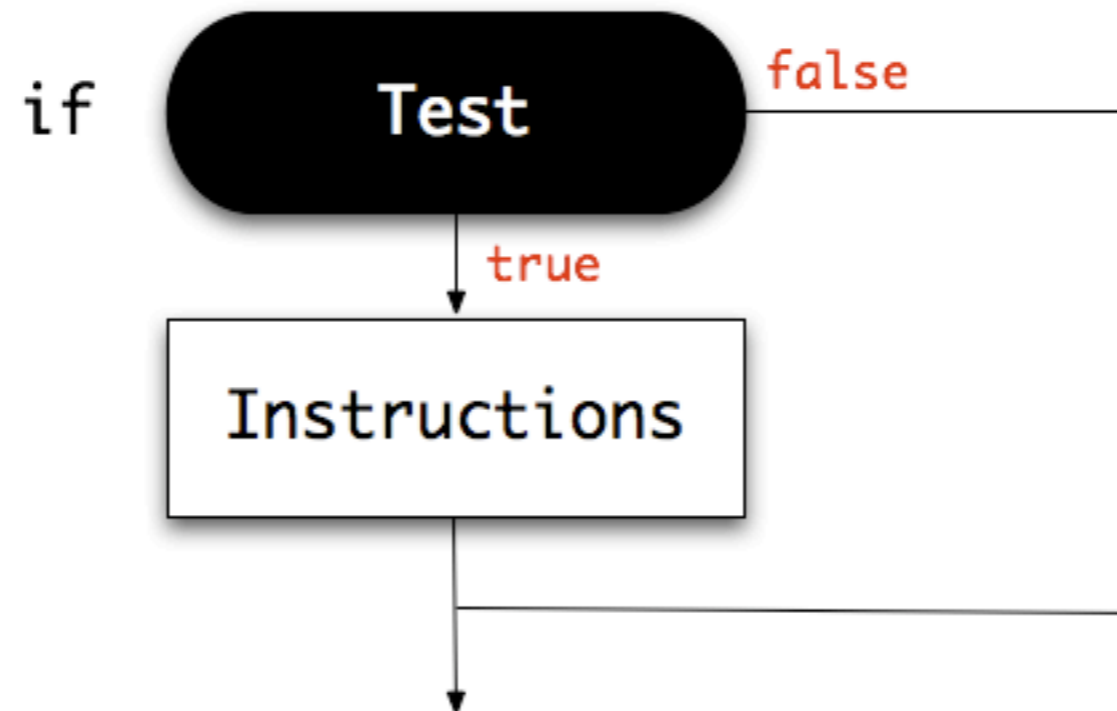
# OF : C++

## Les boucles : for



# OF : C++

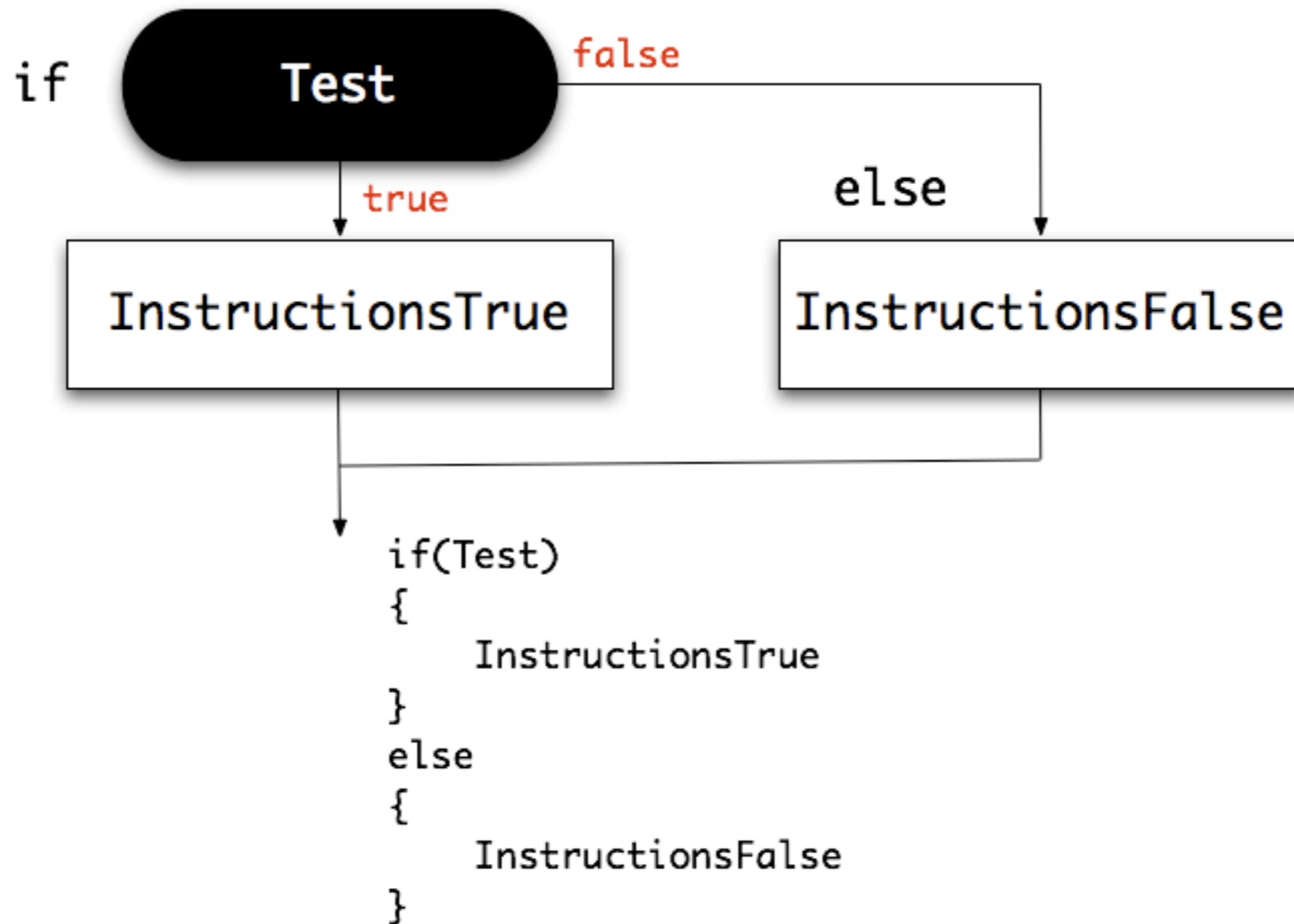
## Les conditions : if



```
if(Test)
{
    Instructions
}
```

# OF : C++

Les conditions : if / else



# OF : C++

## Classe : déclaration .h

```
#ifndef _BALL_H_  
#define _BALL_H_
```

Macro pour empêcher la duplication des définitions de symbole

```
class Ball  
{  
public:  
    Ball      (); // Constructeur  
    Ball      (float x, float y); // Constructeur  
    ~Ball     (); // Destructeur  
  
    // Position  
    float     _x,_y; // Membres publics  
  
    // Maj  
    void      update   (); // Méthode publique  
  
    // Dessin  
    void      draw     (); // Méthode publique  
};
```

```
#endif
```

# OF : C++

## Classe : déclaration

- “public”
  - accessibilité des membres depuis d'autres instances classes.
- “private”
  - accessibilité des membres seulement depuis des instances de la classe.
- “protected”
  - accessibilité des membres seulement depuis des instances de la classe.



# OF : C++

## Classe : implémentation .cpp

```
#include "Ball.h" // inclusion des définitions

Ball::Ball()
{
    this._x = 0.0f;
    this._y = 0.0f;
}

Ball::Ball(float x, float y)
{
    this._x = x;
    this._y = y;
}

void Ball::update()
{
    // Implémentation de la méthode
    // ...
}

void Ball::draw()
{
    // Implémentation de la méthode
    // ...
}
```

# OF : C++

## Impression sur la console

- `printf("n=%d", nombreEntier);`
- `printf("f=%3.2f", nombreAVirgule);`
- `printf("s=%s", chaineChars.c_str());`

# Quelques liens connexes

- Demandez le programme  
[http://interstices.info/jcms/c\\_42286/demandez-le-programme](http://interstices.info/jcms/c_42286/demandez-le-programme)
- Algorithmes, mode d'emploi  
[http://interstices.info/jcms/c\\_42224/algorithmes-mode-demploi](http://interstices.info/jcms/c_42224/algorithmes-mode-demploi)
- Chronologie des langages de programmation  
[http://fr.wikipedia.org/wiki/Chronologie\\_des\\_langages\\_de\\_programmation](http://fr.wikipedia.org/wiki/Chronologie_des_langages_de_programmation)
- Cours de Christian Casteyde sur le C++  
<http://casteyde.christian.free.fr/cpp/cours/index.html>