

Chap. 1 & 2 : Introduction

Introduction au C++

Alexandre Boulch



[retour sur innovation](#)

Plan de la séance

Introduction

L'ordinateur

Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

- ▶ Chercheur à l'ONERA : Traitement d'image et apprentissage par ordinateur
- ▶ Email : boulc-ha@imagine.enpc.fr

Site

<https://sites.google.com/view/boulch>

ou directement dans Google : Alexandre Boulch

Les slides du cours seront sur le site

- ▶ le site du cours :
<http://imagine.enpc.fr/~monasse/Info/>
- ▶ 12 séances
- ▶ 8h30 - 11h15 : Cours + TP
- ▶ TP à rendre (en binôme)
- ▶ DM individuels (3 au total)
- ▶ 2 examens sur machine

Pour chaque chapitre :

- ▶ Cours
- ▶ Fiche récapitulative à la fin du chapitre
- ▶ TPs et programmes à la fin du livre

Plan de la séance

Introduction

L'ordinateur

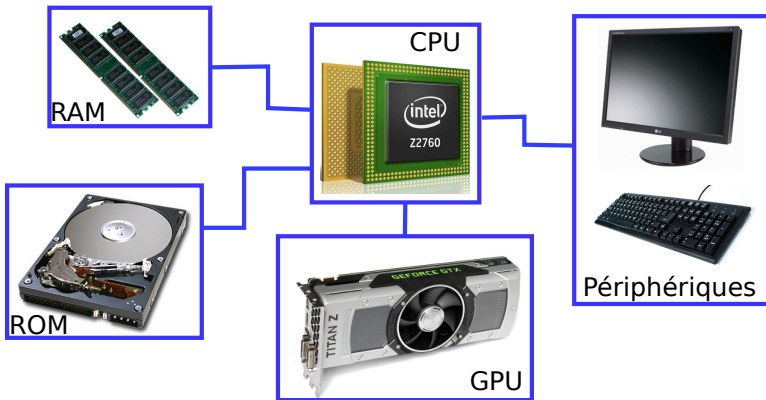
Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

Composants de l'ordinateur



OS (operating system)

C'est le logiciel central de l'ordinateur : il gère les périphériques et les programmes. Il fait travailler le processeur en fonction de la priorité des tâche et s'occupe de la mémoire.



Plan de la séance

Introduction

L'ordinateur

Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

Langages de programmation

Langage

Ensemble des règles d'écriture et des symboles permettant de créer un programme.

De multiples langages

- Langage machine :
assembleur

```
section .data
    helloMsg:
db 'Hello world!',10
    helloSize:
equ $-helloMsg
section .text
    global _start
_start:
    mov eax,4
    mov ebx,1
    mov ecx, helloMsg
    mov edx, helloSize
    int 80h
    mov eax,1
    mov ebx,0
    int 80h
```

Langage

Ensemble des règles d'écriture et des symboles permettant de créer un programme.

De multiples langages

- ▶ Langage machine : assembleur
- ▶ Minimalists : BrainFuck, OokOok ...

```
+++++|>++++>+
++++>+>+<<<<|>
+. >+. +++++. . +++.
>+<+++++
. >. +++ . ----. ---
----. >+. >.
```

Langage

Ensemble des règles d'écriture et des symboles permettant de créer un programme.

De multiples langages

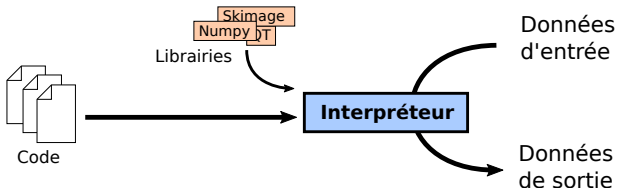
- ▶ Langage machine : assembleur
- ▶ Minimalists : BrainFuck, OokOok ...
- ▶ Utilisables en pratique : C, C++, Python, Java...

```
#include <iostream>
using namespace std;
int main(){

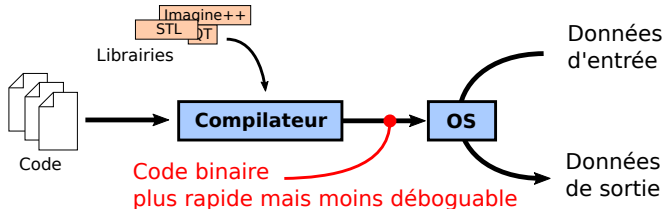
    cout <<"HelloWorld"<<endl;
    return 0;
}
```

Interprété vs compilé

Langage interprété (Python)



Langage compilé (C++)



Pourquoi programmer en C++

- ▶ un langage **complet**, on peut faire du :
 - ▶ haut niveau, programmation simple avec des bibliothèques
 - ▶ bas niveau, programmation proche de l'architecture de la machine (programmeur expérimenté)
 - ▶ savoir programmer en C++ c'est savoir programmer dans tous les langages
- ▶ un des langages **les plus utilisés**
- ▶ toujours en **évolution** / simplification

Plan de la séance

Introduction

L'ordinateur

Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

- ▶ Le Python est un langage interprété, le C++ est un langage compilé
- ▶ Le C++ sera en général plus rapide que le Python (démonstration)
- ▶ Le C++ est beaucoup plus rigide :
 - ▶ La délimitation des blocs se fait par des accolades {} (tabulation en Python)
 - ▶ Chaque instruction se termine par un ;
 - ▶ Les variables vivent dans le bloc où elles ont été créées.

Python vs C++

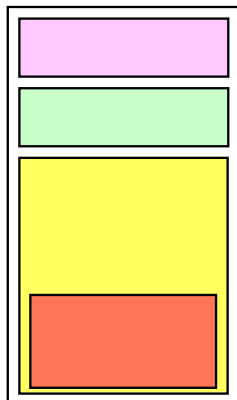
- ▶ Le Python est un langage interprété, le C++ est un langage compilé.
- ▶ Le C++ sera en général plus rapide que le Python
- ▶ Le C++ est beaucoup plus rigide :
 - ▶ La délimitation des blocs se fait par des accolades {} (tabulation en Python)
 - ▶ Chaque instruction se termine par un ;
 - ▶ Les variables vivent dans le bloc où elles ont été créées.

Python

```
for id in xrange(10):  
    a = id  
print a # OK
```

C++

```
int a;  
for(int id=0; id<10; id++){  
    a = id;  
}  
cout << a ;
```



Inclusion des librairies

Définition des noms de domaine

Les fonctions (plus tard)

Le main (fonction particulière)

Hello world

C++

```
// commentaire sur une ligne
#include <iostream>
/* commentaires par
 * bloc */

using namespace std; // utilisation du nom de domaine de la STL

int main()
{
    cout << "HelloWorld"; // ecriture de HelloWorld
    cout << endl; // passage a la ligne

    cout << "HelloWorld" << endl; // les deux en meme temps

    return 0;
}
```

C++

```
#include <iostream>
```

Appeler des libraires donne accès à des fonctions préétablies.

C'est l'équivalent du `import ...` en Python.

`iostream` fait partie de la Standard Template Library (STL).

`iostream` permet de gérer les affichages à l'écran et de récupérer des entrées clavier.

C++

```
using namespace std;
```

En C++, les bibliothèques peuvent avoir un nom de domaine (équivalent du nom de module en Python). Définir le nom de domaine permet de ne pas le répéter avec chaque fonction appelée.

Détails du programme : la fonction main

C++

```
int main(){  
    cout << "HelloWorld";  
    cout << endl;  
    cout << "HelloWorld" << endl;  
    return 0;  
}
```

La fonction **main** est appelée à l'exécution du programme.
Il y a forcément une fonction **main** dans le code.

Détails du programme : la fonction main

C++

```
int main(){ // type : int , nom : main , arguments : rien
```

`int` est le type attendu du résultat de la fonction (ici un entier).

`main` est le nom de la fonction.

Le corps de la fonction (code) est compris entre des accolades.

C++

```
    ...  
    return 0; // 0 valeur de retour lorsque tout s'est bien passe  
}
```

`main` renvoie un entier, ici 0.

Détails du programme : le code

C++

```
cout << "HelloWorld";  
cout << endl;  
cout << "HelloWorld" << endl;
```

`cout` : mot clé pour écrire à l'écran.

`"HelloWorld"` est une chaîne de caractères.

`endl` : mot clé pour un passage à la ligne.

C++

```
std::cout << "HelloWorld"; // si on utilise pas le nom de domaine de la STL  
std::cout << std::endl;    // ie si pas de using namespace std  
std::cout << "HelloWorld" << std::endl;
```


Quelques règles pour bien démarrer

- ▶ Le code s'écrit **toujours** dans une fonction (le `main` pour l'instant).
- ▶ L'indentation (mise en forme) n'est pas nécessaire mais sera obligatoire dans le cours (lisibilité).
- ▶ Il y a une **unique** fonction `main` par programme.

Plan de la séance

Introduction

L'ordinateur

Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

La librairie Imagine++ est une librairie graphique développée à l'ENPC. Elle contient :

- ▶ des fonctions pour l'affichage graphique (images, dessin. . .)
- ▶ le nécessaire pour l'algèbre linéaire de base (matrices, vecteurs. . .)

Integrated Development Environment

L'Environnement de Développement Intégré est un logiciel qui regroupe des fonctions pour aider le développeur, et ainsi gagner en productivité.

Il existe de nombreux IDE, spécialisés pour un ou plusieurs langages :

- ▶ **C++** : QtCreator, Eclipse, Microsoft Visual Studio, KDevelop, XCode. . .
- ▶ **Python** : Spyder, WingIDE, PyCharm...
- ▶ . . .

Chaque IDE stocke les informations relative à un projet dans un format spécifique :

- ▶ L'emplacement des fichiers de code
- ▶ Les emplacements des librairies

Il est parfois fastidieux de construire un projet.

Pour y remédier on utilise un **moteur de production**.

CMake est **moteur de production**, un logiciel pour tous les OS qui permet de générer les projets pour différents IDEs.

Utilisation

- ▶ Interface utilisateur : Makefiles (Unix), Visual Studio (Windows), Xcode, Eclipse. . .
- ▶ Directement dans l'IDE : QtCreator, KDevelop. . .

Utilisation de fichiers de configuration :

```
CMAKE_MINIMUM_REQUIRED(VERSION 2.6)

#Inclusion des modules (ici Imagine++)
FILE(TO_CMAKE_PATH "$ENV{IMAGINEPP_ROOT}" d)
IF(NOT EXISTS "${d}")
    MESSAGE(FATAL_ERROR "Error: IMAGINEPP_ROOT=" "${d}")
ENDIF(NOT EXISTS "${d}")
SET(CMAKE_MODULE_PATH ${CMAKE_MODULE_PATH} "${d}/CMake")
FIND_PACKAGE(Imagine)

#creation d'un projet
PROJECT(Tp1Supplement)
# un executable
add_executable(Tp1Supplement Tp1Supplement.cpp)
# utilisation de Imagine (partie Graphics)
ImagineUseModules(Tp1Supplement Graphics)
```

Utilisation

On réutilise en modifiant un fichier existant.

Plan de la séance

Introduction

L'ordinateur

Langages de programmation

C++ (vs Python)

IDE, CMake, Imagine

Logiciels et installation

QtCreator est l'IDE que nous utiliserons dans ce cours.

- ▶ Multiplate-forme
(Windows, Ubuntu, OSX)
- ▶ Complet
- ▶ Relativement simple à
utiliser



Utilisation d'une **machine virtuelle**.

- ▶ Un OS complet avec tous les logiciels utilisés. Utilisé pour les démos de cours.
- ▶ Installer Virtual Box.
- ▶ Télécharger l'image de la machine (
<https://sites.google.com/view/boulch/teaching>)

Installations alternatives

Tout installer à la main (site du cours). Moins volumineux.

Pour la semaine prochaine, installation d'Imagine++ sur les portables.

Il y a une séance d'aide pour ceux qui veulent après le cours, à 13h.