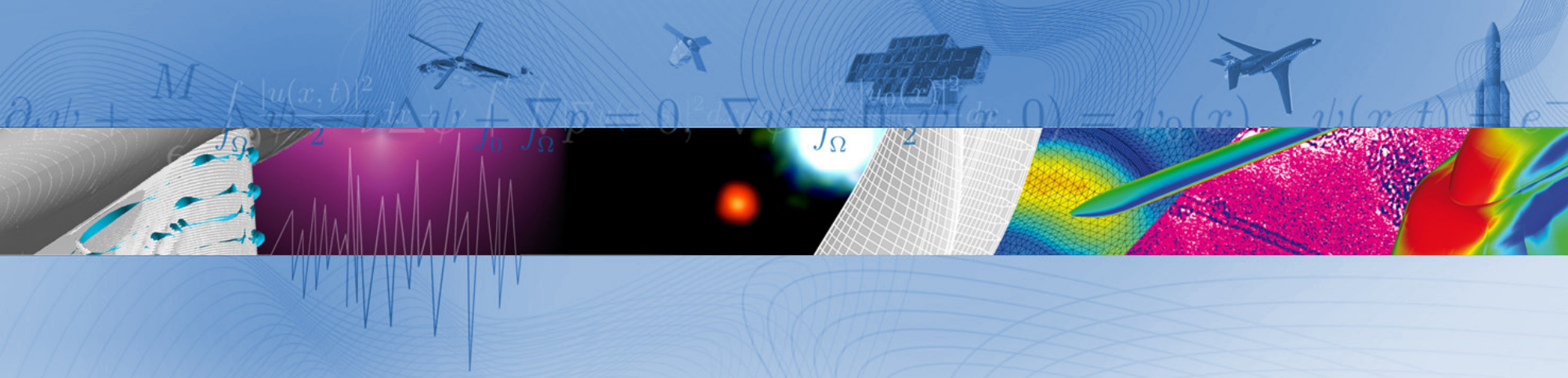

Introduction à la programmation C++

Divers

Alexandre Boulch

aboulch.github.io

ONERA
THE FRENCH AEROSPACE LAB

Séance

Chapitres 11 et 12 du livre.
Seulement quelques exemples : le reste est à lire dans le livre.

Plan de la séance

Les chaînes de caractères

On a déjà utilisé des chaînes de caractères :

```
string s = "toto"; // creation et affectation
char c = s[2]; // troisieme caractere
int l = s.size(); // recuperer la taille
```

Elles sont plus que des tableaux de caractères.

Chaînes de caractères

1. Comparaison : c'est l'ordre lexicographique qui est utilisé et l'ordre ASCII.

```
"a" < "b"    // --> TRUE
"d" > "a"    // --> FALSE
"a" < "ab"   // --> TRUE
"A" < "a"    // --> TRUE
"cat" < "caterpillar"  // --> TRUE
```

Les autres opérateurs de comparaison existent aussi.

2. Recherche

```
size_t i = s.find('h'); // i : indice de h dans s
size_t j = s.find('h',3); // j : indice de h dans s a partir de 3

size_t k = s.find("hop"); // k : indice de la sous-chaine dans s
size_t l = s.find("hop",3); // l : indice dans s a partir de 3
```

Si la recherche n'existe pas, `find` renvoie `string::npos`.

Chaînes de caractères

3. Concaténation

```
string a = "le debut et";
string b = "la fin";

string sum = a + b;
cout << sum << endl; // le debut et la fin
```

Dans le livre

4. Extraction de sous chaînes
5. Intéraction avec l'utilisateur
6. Chaînes au format C

Plan de la séance

Fichiers

Les fichiers se manipulent avec un `stream` qui fonctionne comme `cout` et `cin`.

Inclusion

```
#include<fstream>
using namespace std;
```

Écriture dans un fichier

```
ofstream f("chemin/fichier.txt");
ofstream f2;
f2.open("chemin/fichier.txt");

f << "ligne " << 1 << endl;
f << "ligne 2 ";
f << endl;

f.close();
```

Fichiers

Lecture dans un fichier

```
ifstream g("chemin/fichier");
int i;
double d;
g >> i >> d;
g.close();
```

Tester si le fichier est ouvert

```
ofstream f2;
f2.open("chemin/fichier.txt");
if(! g.is_open()){
    cout << "Erreur" << endl;
    return 1;
}
...
f.close();
```

Autres possibilités avec les fichiers : dans le livre.

Plan de la séance

La STL

Standard **T**emplate **L**ibrary

- Librairie disponible par défaut en C++
- De nombreux modules :
  - classes : chînes de caractères, tableaux, piles ...
  - algorithmes : tri, $n^{\text{ième}}$ éléments ...
  - lecture / écriture

Des modules connus :

- `iostream`
- `string`

vector

Une classe particulièrement intéressante de la STL est la classe `vector`.

- tableau dynamique
- pas de gestion de la mémoire
- interface type tableau

```
#include <vector>

using namespace std;
```

Classe template

La classe `vector` est une classe template, elle peut s'adapter à tous les types de données. (les templates seront évoqués dans les cours suivants).

```
// creation d'un vector
vector<T> tab;

// Exemple :
vector<int> t_int;
vector<double> t_double;
vector<Matrix> t_mat;
vector<float*> t_point;
```

vector : utilisation

Comme un tableau classique :

```
// creation d'un vector d'entier 100 cases
vector<int> t(100);

// acces au cases
for(int i=0; i<100; i++){
    t[i] = i*i;
}
cout << t[5] << endl;

// recuperer la taille
cout << t.size() << endl;
```

où l'on peut récupérer la taille.

vector : utilisation

- Redimensionner :

```
// creation d'un vector d'entier 100 cases
t.resize(1000);
```

Les éléments présents sont gardés.

- Création et remplissage :

```
// creation d'un vector d'entier 100 cases rempli avec 5.6
vector<double> t2(1000, 5.6);
```

vector : utilisation

- Premier élément :

```
cout << *t.begin() << endl;
```

C'est un iterateur (fonctionne un peu comme un pointeur)

- Fin de vector :

```
t.end(); // Attention pointe juste derriere la derniere case
```

vector : utilisation

- Concaténer :

```
vector<int> t1 (10,2);
vector<int> t2 (30, 100);
t2.insert(t2.end(), t1.begin(), t1.end());
```

C'est un iterateur (fonctionne un peu comme un pointeur)

- Trier :

```
#include <algorithm>

...

std::sort(t.begin(), t.end());
```

Plan de la séance

Sur un exemple

Valeurs par défaut

Donner une valeur à un argument dans la plupart des cas.

Référence en type de retour

Pouvoir accéder à un champ directement de puis une méthode.

Opérateur () et []

Faciliter les accès aux champs.

Exercices d'entraînement

Exercice 1

Implémenter une fonction qui prend en argument deux tableaux dynamiques et leur taille respective et qui fait la copie de l'un dans l'autre. (Attention à désallouer et redimentionner correctement)

Exercice 2

Créer une classe Tableau, avec taille et tableau dynamique qui :

- Constructeurs (vide, copie, taille...), destructeur
- Accesseurs (classiques + operateurs[])
- Operateur+ qui crée un nouveau tableau (concaténation)
- Opérateur= et ==

Plan de la séance

Autres

Lire les chapitres 11 et 12.

Pour la semaine prochaine

Envoyer un mail pour dire ce que vous voulez en révision.

A faire avant le TP

Faire l'évaluation sur Educnet du cours.

TP

Finir le Serpent et le Tron : checklist sur le site du cours.