Programmation Orientée Objet : Structures de Données Abstraites et Bibliothèques C++

Jean-Cédric Chappelier

Faculté I&C

Organisation du travail (semestre)

		MOOC	déc.	cours 1 h Jeudi 8-9	exercices 2 h Jeudi 9-11
1	19.02.26		0	Intro + compil. séparée	
2	26.02.26	1. Intro POO	0	Intro POO	
3	05.03.26	2. Constructeurs/Dest	0	Constructeurs	
4	12.03.26	3. Surcharge des opé	0	Surcharge	
5	19.03.26	4. Héritage	0	Héritage	
6	26.03.26	5. Polymorphisme	0	Polymorphisme 1	
7	02.04.26		1	Polymorphisme 2 / Collections hétérogènes	
-	09.04.26		-	vacances Pâques	
8	16.04.26	6. Héritage multiple	1	Héritage multiple	
9	23.04.26		-	-	Midtem
10	30.04.26	(7. Etude de cas)	-	Templates	
11	07.05.26		-	Structure de données abstraites ; Bibliothèques	
12	14.05.26		-	(Ascension)	
13	21.05.26	(7. Etude de cas)	-	Bibliothèques (fin) + Révisions	
14	28.05.26		-	-	Examen

Objectifs de la leçon d'aujourd'hui

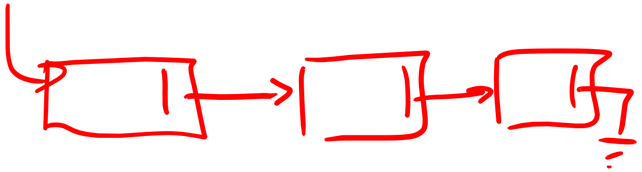
- Concepts fondamentaux
- Étude de cas ?
- Retour sur la série notée

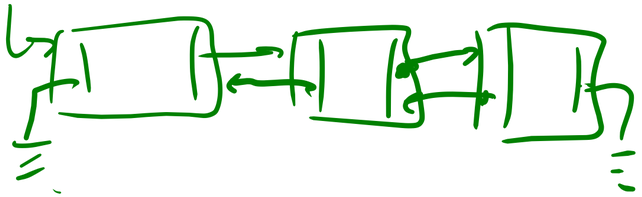
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Points importants (sur 2 semaines)

- comprendre la notion de « structure de donnée abstraite »
- savoir ce qu'est une « liste chaînée » et une « pile »
- (si ce n'est pas encore le cas) notion de pré-déclaration
- notion de « container » C++
- types :
  - `list` et `forward_list`
  - `stack` et `queue`
  - `set` et `unordered_set`
  - `map` et `unordered_map`
- notion d'itérateur
- savoir utiliser `erase()`, `find()` et `sort()`
- ... ...et continuer son apprentissage des bibliothèques C++

  ☞ Conseil : https://en.cppreference.com/





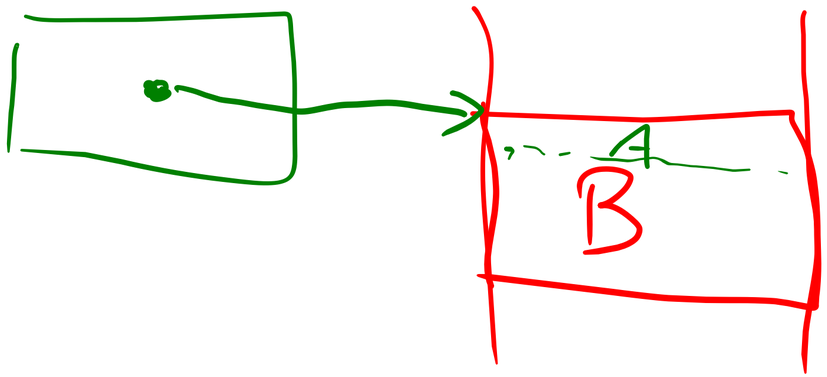
Jean-Cédric Chappelier
& Jamila Sam
EPFL

Etude de cas ?

- reprendre un exemple du cours ?
- questions ?

A
↑
B

unique_ptr<A> copy() = 0;



A x;
B y;
x = y;

Jean-Cédric Chappelier
& Jamila Sam
EPFL

Retours sur le midterm

Les résultats sont globalement très bons.
Base du polymorphisme comprises mais encore quelques difficultés (cf ci-dessous)

« *J'ai aussi l'impression que les étudiant(e)s ont tendance à trop se compliquer la vie ; des solutions plus simples leur auraient évité beaucoup d'erreurs.* »

- gestion mémoire ; compréhention du rôle des pointeurs ; ne pas « en mettre partout »
  - libération de mémoire allouée statiquement
  - `unique_ptr` sans copie
  - utilisation non nécessaire (`Couleur*` ? ? ?)
- manques de ré-utilisation (`Couleur(int)` ; `operator+=(Scene const&)`)
- mauvais « réflexe sans réfléchir » : pourquoi des `ostream` dans

```
dessin << cercle_1 << cercle_2;
```

- compréhention de l'interface/l'encapsulation : méthode `Image::access()`
- destructeur virtuel

Jean-Cédric Chappelier
& Jamila Sam

Quels pointeurs ?

RAPPEL :

utilisation	sur des données	sur des fonctions
référence	références (ou pointeurs à la C)	nom de la fonction
généricité	pointeur à la C ou index dans un tableau	`std::function` ou pointeur à la C
allocation dynamique	smart-pointers surtout `unique_ptr` (ou pointeurs à la C)	—

« *Utilisez des références quand vous pouvez, des pointeurs quand vous devez.* »

Jean-Cédric Chappelier
& Jamila Sam
EPFL