Référence/type[1]			☐ TD ☒ AP
Thématique principale	Interfaces		
Thématique(s) secondaire(s)	Spécialisation d'interfaces Fichiers (texte) Algorithmes de la STL (tri) Lambdas		
Durée	2h	***Niveau***[2]	A
Prérequis	✓ Connaissances solides sur la spécialisation ✓ Notion d'interface		
Compétences opérationnelles et niveau cible[3]	✓ Déclarer / Spécifier une classe d'interface (***A***) ✓ Réaliser une interface (***M***) ✓ Spécialiser une interface (***M***) ✓ Accéder à des fichiers (texte) en lecture ou en écriture (***A***) ✓ Identifier un algorithme de la STL adapté à une situation (***A***) ✓ La bibliothèque chrono et les chrono literals du C++ moderne (***A***) ✓ Mettre en place un tri de vecteur d'objets complexes via un algorithme de la STL et des lambdas expressions (***N***, *bonus*)		
Modalités et critères d'évaluation			
Matériel(s), équipement(s), composant(s) nécessaires			
Logiciel(s) nécessaire(s)[4]			
Ressources			

Avant la séance…

Revoir les travaux (cours, AP…) déjà effectués portant sur la spécialisation, notamment si des problèmes de syntaxe sont apparus au cours des séances dédiées.

Faire des recherches afin d'identifier les classes/objets du C++ moderne permettant d'accéder aux fichiers. On se focalisera sur les fichiers texte, que ce soit en écriture ou en lecture.

Faire des recherches sur les algorithmes associés à la STL, identifier notamment quelques algorithmes de tri.

Faire quelques recherches, sans chercher à trop comprendre ce que sont ces objets, sur les lambdas expressions. Essayer de concentrer les recherches sur l'intérêt de ces fonctions lorsqu'elles sont associées à un algorithme de la STL (de tri notamment).

Attention : la notion de lambda est très très complexe, à la fois dans la compréhension de ce que c'est, de la mise en œuvre pratique (syntaxe, notion de capture de contexte…) et même de « à quoi ça sert ce truc ? ». Il s'agira de na pas perdre de temps à trop chercher, sachant que nous ne ferons qu'aborder très rapidement le sujet, et uniquement par le prisme de la critérisation d'un tri.

[1] TD : ½ groupe, apports de cours, mise en pratique guidée – TP : ½ groupe, travail autonome majoritaire.
[2] Le niveau se rapporte à la thématique principale. Il peut ici être entendu comme un indicateur de la difficulté du travail (N-Facile, A-Sans grande difficulté, M-Quelques points complexes, MA-Difficile)
[3] Les niveaux cible correspondent pour chaque compétence opérationnelle au niveau d'acquisition de la celle-ci à l'issue du travail dans son intégralité (en incluant les phases préparatoires et de synthèse).
[4] En plus d'un environnement de développement C++

Travail encadré en séance

Apports de cours

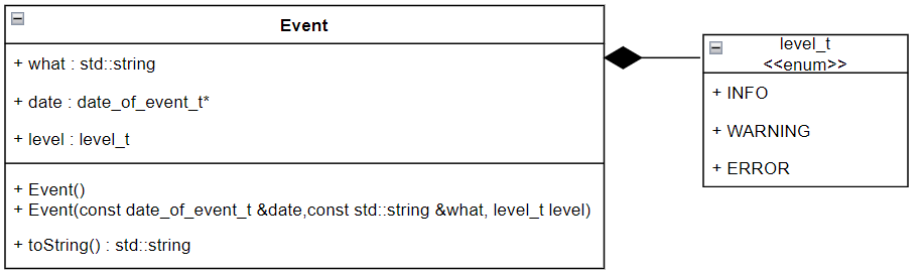
- Avez-vous des questions préliminaires ?

Activités

Présentation

Il s'agit ici de mener à la fois des travaux d'analyse, de réalisation et de tests sur une problématique type datalogger. Le but étant de proposer une architecture de classes permettant de manipuler, de manière uniforme côté API, des objets « datalogger » indépendamment de dispositif réel de rétention des données (Fichier, RAM, EEPROM…).

Notre datalogger, quel qu'il soit, reçoit des objets de type « événements » (Event) et a pour principal rôle d'en assurer le stockage. Une classe totalement validée permet de créer et gérer les événements. Son modèle est fourni ci-dessous, le code est disponible sur Teams.



Travail à réaliser - Analyse

- Réfléchir, proposer et discuter d'une architecture permettant de créer des dataloggers concrets (s'appuyant sur un dispositif de stockage des logs) à partir d'un modèle unifié.
- Identifier (au moins) deux grandes familles de dataloggers en fonction des capacités du dispositif de stockage.
- Modéliser le tout sous la forme d'un diagramme des classes mettant en évidence la notion d'interface.
- Spécifier au mieux ces interfaces.

Travail à réaliser – Implémentation

- Implémenter dans un module logiciel « Datalogger » les interfaces.
- Implémenter dans autant de modules logiciels spécifiques autant de dadaloggers concrets qu'identifiés (objectif = 2).

Synthèse

Bien faire le tri 😊 entre ce qui est vital, important, anecdotique (du moins pour le moment) :

- ***Vital*** : la spécialisation, les méthodes virtuelles (déclaration, définition et utilisation)
- ***Important*** : ce qui tourne autour des fichiers et la notion d'algorithme (savoir que ça existe, savoir trouver celui qui va bien et savoir l'utiliser, même avec l'aide d'une IA générative)
- ***Utile*** : la bibliothèque chrono et les chrono literals
- ***Anecdotique*** : les lambdas

Après la séance…

Revoir les codes proposés comme réponses à la problématique, surtout si les difficultés se sont montrées nombreuses.

Annexes

Annexe 1 : chrono et chrono literals

- ✓ https://en.cppreference.com/w/cpp/chrono
- ✓ https://www.modernescpp.com/index.php/c20-basic-chrono-terminology/
- ✓ https://www.geeksforgeeks.org/chrono-in-c/

Annexe 2 : Fichiers et système de fichiers

- ✓ https://en.cppreference.com/w/cpp/io/basic_ofstream
- ✓ https://en.cppreference.com/w/cpp/filesystem
- ✓ https://www.geeksforgeeks.org/file-system-library-in-cpp-17/
- ✓ https://www.cppstories.com/2024/common-filesystem-cpp20/