Objectifs

* Réaliser l’itération 3 de l’application Gestion\_Log
  + Améliorer l’interaction utilisateur
  + Améliorer la modularité de la programmation
  + Générer la documentation automatiquement
  + Valider les exigences du CDC (ESS01, REQ014) → DevOps

Contenu technique

* Créer un module C++ (.h et .cpp)
* Programmer deux sous fonctions

Durée 3h

# Améliorer la programmation : Création de fonctions

Notre programme commence à être plus complexe et nécessite de regrouper les fonctionnalités dans des modules dédiés.

Nous allons avant tout découper le programme en deux sous fonctions.

1. Regarder le DDC pour déterminer les deux sous fonctions à créer, leur rôle et leur prototype (signature)

## Sous fonction afficherMenu()

1. Créer la sous fonction afficherMenu()
2. Re factoriser votre programme pour appeler cette sous fonction

Résultats attendus :

* la fonction afficherMenu() est créée en respectant le prototype proposé dans le DDC.
* Les différents choix possibles de menu sont affichés en faisant appel à la sous fonction afficherMenu()

## Sous fonction **choisirLog**()

1. Créer la sous fonction choisirLog()
2. Re factoriser votre programme pour appeler cette sous fonction

Résultats attendus :

* la fonction choisirLog() est créée en respectant le prototype proposé dans le DDC.
* Les logs sont correctement affichés grâce à cette sous fonction.

## Validation

**Appeler l’enseignant pour qu’il valide votre travail**

# Améliorer la programmation : Création d’un module

Pour encore améliorer le programme, il est nécessaire de créer un module qui contiendra les deux sous fonctions.

1. Rechercher la définition de modularité d’un programme informatique

## QtCreator – Création d’un module

### Composition d’un module

En C++, un module est composé de deux fichiers :

* Le fichier .h contient les déclarations des fonctions
* Le fichier .cpp contient les définitions des fonctions

Ces deux fichiers .h et .cpp seront rangés dans un répertoire du nom du module.

### CMakeList.txt

Le fichier CmakeList.txt indique au compilateur quels fichiers sont à inclure dans votre projet pour construire l’exécutable.

Il faut ajouter à ce fichier chaque nouveau module pour qu’il apparaisse dans votre arborescence de projet et utilisé pour construire l’exécutable.

Le fichier CmakeList.txt fournit avec le modèle de projet pré-inclus des variables et il suffit de le modifier comme ci-après pour que votre module soit utilisé.

Les modifications que vous devez apporter à ce fichier sont en **gras**.

#1/3-Ajouter les fichiers sources de votre projet

set(SRCS

**Menu/menu.cpp**

)

#2/3-Ajouter les fichiers headers de votre projet

set(HEADERS

**Menu/menu.h**

)

#3/3-Ajouter les noms des dossiers de vos modules

**include\_directories(Menu)**

### Création du module Menu

1. Suivre les étapes ci-après pour créer le module Menu
2. Créer un nouveau dossier appelé Menu dans votre projet
3. Créer un fichier menu.h (nouveau fichier C/C++ Header File) dans le dossier Menu
4. Créer un fichier menu.cpp (nouveau fichier C/C++ Source File) dans le dossier Menu
5. Modifier le CMakeList.txt pour ajouter votre module
6. Déplacer les deux fonctions afficherMenu() et choisirLog() dans le fichier menu.cpp
7. Déclarer les prototypes des fonctions dans menu.h
8. Inclure votre module (#include "menu.h**"**) dans votre fichier main.cpp

### Programmation modulaire

1. Vérifier que votre programme fonctionne correctement avec ce nouveau module

Résultat attendu :

* Le programme fonctionne correctement avec les sous fonctions placées dans le module Menu

## Validation

**Appeler l’enseignant pour qu’il valide votre travail**

# Documentation

Vous allez générer la documentation html de votre projet à partir des balises doxygen de documentation de votre projet.

Ensuite, cette documentation sera générée automatiquement grâce au pipeline CI et disponible sur Gitlab.

## Documenter voter code

1. Suivre le tutoriel [Document](https://www2.ciel-kastler.fr/docs/Programmation/documentation.html" \l "documentation-des-fichiers-du-projet)ation des fichiers du projet pour documenter vos fichiers
2. Suivre le tutoriel [Document](https://www2.ciel-kastler.fr/docs/Programmation/documentation.html" \l "documentation-des-fonctions-ou-méthodes)ation des fonctions ou méthodes pour documenter vos fonctions

Résultats attendus :

* Les 3 fichiers (main.cpp, menu.h, menu.cpp) sont documentés.
* Les 3 fonctions (main(), afficherMenu() et choisirLog() ) sont documentés.

## Générer la documentation HTML de votre projet

Un fichier Doxyfile préconfiguré est fourni avec le modèle de projet.

Il faut quand même le personnaliser en modifiant le nom par défaut du projet et en choisissant un logo pour celui-ci.

1. Suivre le tutoriel [Doxyfile](../../ProjetGestionLogEduscol2025/3-Iterations/Suivre%20le%20tutoriel%20Documentation%20des%20fonctions%20ou%20méthodes%20pour%20documenter%20vos%20fonctions) pour générer la documentation html de votre projet sur votre poste.

Résultats attendus :

* La documentation html du projet est générée
* Les fichiers et fonctions documentés apparaissent dans la documentation html
* Le nom du projet a été personnalisé et un logo a été choisi

## Générer automatiquement la documentation avec le pipeline Gitlab CI/CD

Un fichier .gitlab-ci.yml est proposé avec le modèle de projet.

Vous allez ajouter une étape au pipeline pour générer automatiquement la documentation lors de chaque commit.

La documentation du projet sera accessible dans l’espace pages de Gitlab

1. Suivre le tutoriel [Générer automatiquement la documentation avec le pipeline Gitlab CI/CD](../../ProjetGestionLogEduscol2025/3-Iterations/Générer%20automatiquement%20la%20documentation%20avec%20le%20pipeline%20Gitlab%20CI/CD)  pour ajouter l’étape de génération de la documentation à votre pipeline

Résultats attendus :

* Une étape supplémentaire (deploy) est exécutée à chaque commit sur le pipeline CI
* La documentation est accessible via le menu pages

# Tests de validation

Le document « Gestion\_Log\_DDV.odt » contient les tests à réaliser pour vérifier les exigences du projet Gestion-log.

1. Effectuer le test ESS01 et compléter le Dossier de Validation DDV

# Outils **DevOps**

Afin de finaliser votre travail et avant de livrer celui-ci, n’oubliez pas :

* D’indenter correctement votre code
* De documenter le code et les fonctions
* De sauvegarder votre travail sur Gitlab et de créer un (tag v3.0 correspondant à la fin de l’itération3)

# Livrable

Sur le moodle de la section

→ Le dossier de validation avec ESS01 complété

→ La documentation générée localement

→ La documentation générée grâce au pipeline (lien sur le readme du projet )

Sur le serveur GIT de la section

→ Le projet **<VOTRE\_NOM>\_Gestion\_log** avec le tag correspondant à la fin de l’itération3.