



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



I. 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.

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

I.1 Sous fonction `afficherMenu()`

Tâche 2. Créer la sous fonction `afficherMenu()`

Tâche 3. 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()`

I.2 Sous fonction `choisirLog()`

Tâche 4. Créer la sous fonction `choisirLog()`

Tâche 5. 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.

I.3 Validation

Appeler l'enseignant pour qu'il valide votre travail



II. 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.

Tâche 6. Rechercher la définition de modularité d'un programme informatique

II.1 QtCreator – Création d'un module

II.1.1 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.

II.1.2 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)
```



II.1.3 Création du module Menu

Tâche 7. Suivre les étapes ci-après pour créer le module Menu

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

II.1.4 Programmation modulaire

Tâche 8. 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

II.2 Validation

Appeler l'enseignant pour qu'il valide votre travail



III. 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.

III.1 Documenter voter code

Tâche 9. Suivre le tutoriel [Documentation des fichiers du projet](#) pour documenter vos fichiers

Tâche 10. Suivre le tutoriel [Documentation 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.

III.2 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.

Tâche 11. Suivre le tutoriel [Doxyfile](#) 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

III.3 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

Tâche 12. Suivre le tutoriel [Générer automatiquement la documentation avec le pipeline Gitlab CI/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

IV. Tests de validation

Le document « Gestion_Log_DD.V.odt » contient les tests à réaliser pour vérifier les exigences du projet Gestion-log.

Tâche 13. Effectuer le test ESS01 et compléter le Dossier de Validation DDV

V. 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)

VI. 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.