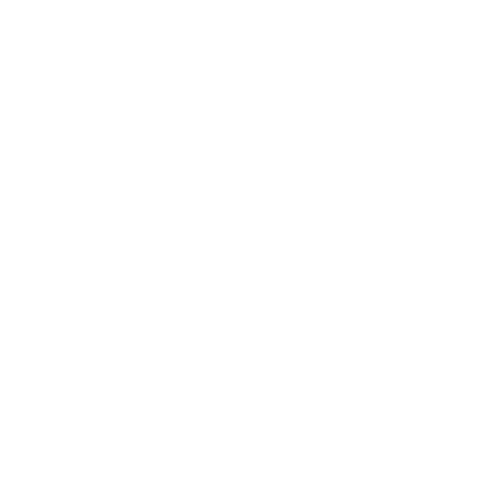
****

**CAHIER DES CHARGES**

**Programmation impérative**

****

**INTRODUCTION**

Nom du projet : Programmation impérative

Projet N° : 02

Date : 07/10 - 25/10

Classe : GTECH-1

**Modalités**

Travail individuel

**Contexte et Description du Projet**

Brève description du contexte dans lequel le projet s’inscrit et détails du projet :

Le projet consiste à développer un **jeu de Démineur en C++**, jouable dans la **console Windows**.  
 Le joueur interagit avec une **grille de cases**, certaines contenant des mines.

**Objectif :** révéler toutes les cases sans mines.  
 Lorsqu’une case est révélée :

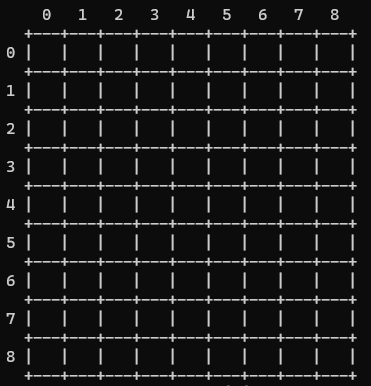
* elle affiche soit un chiffre indiquant le nombre de mines adjacentes,
* soit une mine (ce qui met fin à la partie).

Le joueur peut **marquer les cases suspectes** à l’aide de drapeaux, afin d’éviter les erreurs de révélation.

**Specs**

Vous devez implémenter dans votre application :

#### **1.Interface Console**



#### **Affichage**

* La grille est composée de cases organisées en lignes et colonnes.
* Chaque case peut afficher :
  + **espace vide** : case non révélée
  + **1** à **8** : nombre de mines adjacentes
  + **\*** : mine révélée
  + **F** : drapeau (case marquée)

L’écran est régulièrement rafraîchi à l’aide de **system("cls")**.

#### 

#### 

#### **Commandes**

* Le joueur saisit les **coordonnées** d’une case pour la révéler.
* Il peut **placer ou retirer un drapeau** sur une case suspecte.
* Les saisies sont **validées** : si une entrée est incorrecte, un message d’erreur s’affiche et le joueur est invité à réessayer.

### **2. États du jeu**

#### **Initialisation**

* **Taille de la grille :**
  + Mode **Facile** : 9×9
  + Mode **Normal** : 16×16
  + Mode **Difficile** : 30×16
  + Mode **Personnalisé**
* **Placement des mines :**
  + **15 %** des cases contiennent une mine.
  + La première case révélée est **toujours sans mine**.

#### **États possibles d’une case**

* **Non révélée** : la case n’a pas encore été jouée.
* **Révélée** : son contenu (chiffre ou mine) est affiché.
* **Marquée** : un drapeau indique une suspicion de mine.

#### **Fin de partie**

* **Victoire :** toutes les cases sans mine ont été révélées.
* **Défaite :** une mine est révélée → toutes les mines sont affichées, un message de défaite apparaît.
* Le joueur peut choisir de **rejouer une partie**.

#### **3.Logique de jeu**

#### **Révélation des cases**

* Si la case contient **une mine** → fin immédiate de la partie.
* Si la case **n’a pas de mine** :
  + Elle affiche le **nombre de mines adjacentes** (8 directions).
  + Si aucune mine n’est adjacente, les **cases voisines vides** sont automatiquement révélées de manière récursive.

#### **Marquage des mines**

* Le joueur peut placer un drapeau (**F**) sur une case non révélée.
* Une case marquée ne peut pas être révélée.  
  Le joueur peut retirer un drapeau à tout moment.

## **4. Structure du code**

Afin d’assurer une bonne organisation et une meilleure lisibilité du projet, le programme doit être **découpé en plusieurs fichiers sources** et utiliser des **structures de données adaptées**.

### **Structures de données**

Deux structures principales ::

* **struct Cell** : représente une case individuelle de la grille.  
   Elle contient les informations suivantes :
  + si la case contient une mine ;
  + si elle a été révélée ;
  + si elle est marquée par un drapeau ;
  + le nombre de mines adjacentes.
* **struct Grid** : regroupe l’ensemble des cases et les dimensions de la grille :
  + tableau 2D de Cell ;
  + nombre de lignes et de colonnes ;
  + nombre total de mines.

### **Découpage**

* **main.cpp**
* **grid.cpp** / **grid.h**
* **cell.cpp** / **cell.h** .

Pour les plus avancés uniquement

**Utilisation de** \_**getch() pour déplacer le curseur sur la grille** :

* **Déplacement avec les touches** : Au lieu d'entrer des coordonnées manuellement, le joueur peut utiliser les touches fléchées pour se déplacer sur la grille et sélectionner des cases à révéler ou à marquer.
* **Mise en évidence de la case sélectionnée** : La case actuelle où se trouve le curseur peut être affichée en surbrillance en changeant la couleur de l'arrière-plan.
* Utiliser le code **ANSI** pour afficher de la couleur.
* Utiliser un tableau avec **une dimension** pour représenter la grille.
* Représenter une case avec tous ses états possible en utilisant **uniquement** une variable de type **int** (sans changer la **complexité** du programme)

Informations complémentaires :

# 

# OBJECTIFS PÉDAGOGIQUES ET PROFESSIONNELS DU PROJET

**Savoirs**

* **Apprendre** les bases de la programmation en C++, comprenant la syntaxe, les types de données, et les structures de contrôle.
* **Comprendre** les concepts fondamentaux du C++, tels que les pointeurs, les tableaux, et la gestion de la mémoire.
* **Découvrir** les mécanismes d'interaction avec l'utilisateur via une interface console, comprenant la gestion des entrées et sorties.
* **Connaître** les concepts de base liés à la création d'un jeu, comme la gestion des états et la logique de jeu.

**Savoir-être**

* **Faire preuve** de persévérance face aux défis techniques, en s'engageant à résoudre les problèmes rencontrés.
* **Discuter** et **échanger** avec d'autres étudiants pour partager des idées et des approches, tout en travaillant individuellement sur le projet.
* **Accepter** les erreurs comme une opportunité d'apprentissage et les utiliser pour améliorer sa compréhension de la programmation.
* **Organiser** son travail de manière autonome, en respectant les délais et en planifiant efficacement le temps nécessaire pour chaque étape du projet.

**Savoir-faire** / Compétences

* **Écrire** des programmes en utilisant les structures de données appropriées pour représenter les éléments du jeu, comme la grille et les cases.
* **Gérer** les entrées utilisateur de manière robuste, en vérifiant la validité des données saisies et en traitant les erreurs de manière appropriée.
* **Développer** des algorithmes pour la génération aléatoire de contenu, comme les mines sur la grille, et pour la mise à jour de l'état du jeu en fonction des actions du joueur.
* **Tester** et **déboguer** le code pour s'assurer du bon fonctionnement du programme, même dans des situations imprévues ou des cas extrêmes.

**Ressources**

1. **A savoir/à apprendre** (Ce travail pourra être évalué / noté en début de projet) :

* Les bases de la programmation impérative

1. **A lire/à consulter** :

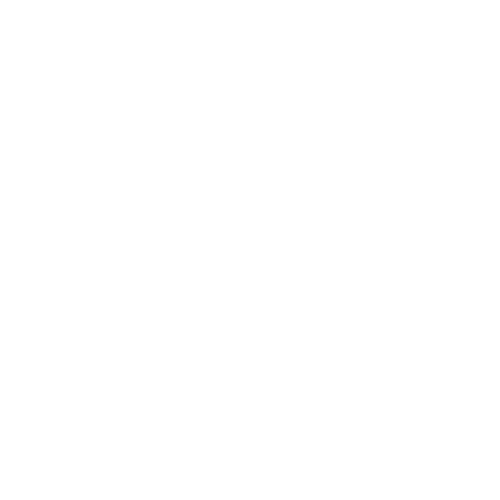
* https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c

1. **A installer/pré-requis techniques** :

* Installer Visual Studio 2022
* Installer un client Git (Tortoise Git, Github Desktop …)

**Travail préparatoire**

* Créer un repo Git
* Écrire, compiler et exécuter un programme test

****

**INTERVENANT**

**Nom : Peter Vystavel**

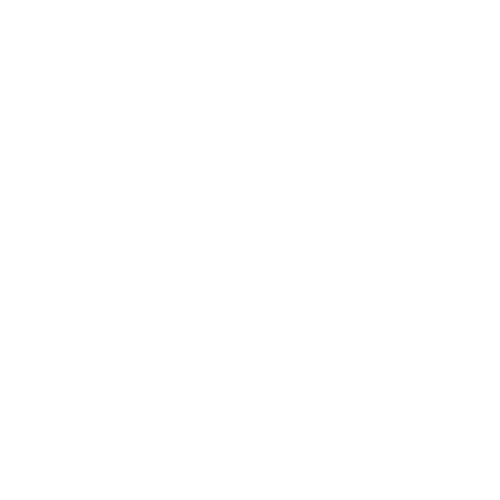
**Titre : Développeur C/C++**

**LinkedIn : https://www.linkedin.com/notifications/?filter=all**

**Biographie**

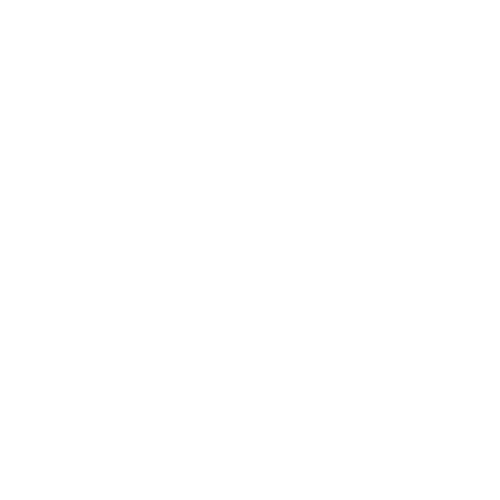
Développeur informatique depuis le début des années 2000, j'ai commencé en autodidacte en C/C++ et ai créé plusieurs petits jeux ainsi qu’un moteur 2D avant d'obtenir un Master en informatique en 2016. J'ai ensuite travaillé comme développeur gameplay chez **Eugène Systems** (*Wargame*, *Steel Division*), puis comme développeur moteur chez **Kylotonn Games** (*WRC*, *V-Rally 4*).

En 2017, j'ai lancé mon premier jeu, **Space Note**, suivi en 2024 de **Space Note 2** sur Android et iOS. Aujourd'hui, j'ai cofondé une l’entreprise **Unlimited Spoons** avec quatre associés, et nous travaillons actuellement sur notre troisième projet, un jeud’action/aventure pour PC et consoles.

****

**ROADMAP**

| **Jalon** | **Livrables attendus** | **Date limite** | **Moyens / formats** |
| --- | --- | --- | --- |
| **1** | **Projet final** | **24/10 à 9h** | **Lien Git** |



**GRILLE D’ÉVALUATION**

L’évaluation est conçue pour être holistique, prenant en compte non seulement le produit final, mais aussi le processus, les compétences acquises et les attitudes démontrées tout au long du projet.

**Barème du projet**

| **Semaine des fondamentaux** | **Semaine(s) du projet** | **Rendu final** | **Soutenance** | **NOTE** |
| --- | --- | --- | --- | --- |
| *Assiduité individuelle* | *Assiduité individuelle* | *Groupe* | *Groupe* |  |
| **/5** | **/5** | **/10** | **-1, 0 ou +1\*** | **/20** |

**Barème de la notation d’assiduité individuelle**

| **0** | Absent |
| --- | --- |
| **1** | Ni implication, ni résultat |
| **2** | Faible implication et faible résultat |
| **3** | Implication sans résultat ou Résultat sans implication |
| **4** | Intermédiaire et résultat conforme au monde professionnel |
| **5** | Implication et résultat d’exception |

**Remarques**

Les rendus doivent impérativement être disponibles pour le jury avant la date butoir depuis le compte GitHub de l’école.

La soutenance est évaluée lors de la présentation finale du projet. Les critères incluent la clarté, la cohérence, et la capacité à articuler et défendre les résultats du projet.

\* la note globale ne peut être ni inférieure à 0 ni supérieure à 20