**L3 INFO**
**INFO505 – TD et TP – 2024-2025**
**TD/TP de Programmation en C : Création d'un Jeu « Sauvons nos montagnes et la Planète »**

## Préambule :

L'idée est d'exploiter les séances de TD (en partie) et TP restantes pour progressivement construire un jeu en suivant une approche modulaire. L'objectif principal reste bien sûr **d'améliorer vos compétences en Programmation C.**

**Concepts pédagogiques abordés (TP3) :**

- les structures (struct), les listes chaînées, la gestion dynamique de mémoire
- la documentation du code (selon la norme Doxygen) et sa modularité introduite à travers la création de bibliothèques
- les modes de gestion des erreurs et les tests unitaires effectués
- l'exploitation des directives de pré-compilation pour optimiser le code
- la modélisation qui devra permettre de facilement faire évoluer le jeu (les règles du jeu et les méthodes de calcul et mise a jour des scores) pour le rendre de plus en plus complet et ludique au fil des séances
- l'utilisation de pointeurs de fonctions pour créer par exemple plusieurs version de fonction de calcul des indicateurs

Objectif du jeu

Le but du jeu est de protéger nos montagnes (une planète) contre la pollution et l'épuisement des ressources naturelles. En tant que joueur, vous devez prendre des décisions écologiques stratégiques tout en affrontant des événements climatiques ou autres imprévisibles. Votre mission est de maintenir un équilibre entre les ressources naturelles, la pollution et l'énergie disponible pour éviter le « Game Over » de la planète.

Conditions de victoire ou de défaite (il s'agit d'une liste minimale, vous pouvez proposer d'autres idées)

Victoire : Le jeu n'a pas de fin fixe obligatoire, mais on pourra considérer que la planète est sauvée si vous parvenez à la maintenir en équilibre pendant un certain temps… (nombre de tours de jeu, par exemple, 10 tours sans atteindre une situation critique).

Défaite : La partie se termine lorsque l'une des conditions suivantes est remplie : Les ressources naturelles tombent à 0 ou le niveau de pollution atteint MAXPOLLUTION.

Tour de jeu et indicateurs clés (il s'agit d'une liste minimale, vous pouvez proposer d'autres idées)

Chaque tour, l'état de la planète est représenté par trois indicateurs :

- Niveau de pollution : Une pollution élevée est dangereuse pour la survie. Si elle atteint MAXPOLLUTION, la planète est en crise et le jeu se termine.
- Ressources naturelles : Il s'agit de la quantité de ressources disponibles sur la planète. Si elles tombent à 0, la planète ne peut plus subvenir aux besoins de sa population, et le jeu est perdu.
- Énergie : Les Elements écologiques que vous entreprenez consomment de l'énergie. Vous devrez veiller à utiliser votre énergie avec prudence, car certaines Actions sont coûteuses.

Les Elements écologiques

À chaque tour, vous aurez la possibilité de choisir une Action écologique pour aider la planète. Les Actions sont disponibles sous forme de liste chainée, et vous pourrez en sélectionner une par tour. Chaque Action a un impact sur la pollution, les ressources et l'énergie, ... Par exemple :

- Planter des arbres : réduit la pollution tout en augmentant légèrement les ressources naturelles, mais elle consomme de l'énergie.
- Transition énergétique (amélioration des équipements ?) : réduit significativement la pollution, mais nécessite beaucoup d'énergie.
- Réduction des déchets (du tourisme ?) : diminue la pollution de manière modérée avec une faible consommation d'énergie.

Chaque Element que vous choisissez modifie l'état de la planète. Vous devrez donc adapter votre stratégie en fonction des besoins actuels de la planète.

Les événements climatiques aléatoires (positifs ou négatifs)

Outre le Actions écologiques, des Evénements climatiques aléatoires surviennent à chaque tour. Ces événements peuvent avoir un impact positif ou négatif sur la planète. Par exemple :

- Incendie : Augmente la pollution et réduit les ressources.
- Tempête : Augmente la pollution et réduit modérément les ressources.
- Sécheresse : Réduit drastiquement les ressources naturelles.
- Pluies abondantes : Réduit la pollution et augmente les ressources naturelles.
- RAS

Ces événements sont imprévisibles et ajoutent une couche de difficulté au jeu, obligeant à réagir rapidement pour éviter une catastrophe environnementale.

Tour de jeu / stratégie (version minimale à vous de proposer mieux)

À chaque tour, les étapes suivantes se déroulent :

- Un événement climatique aléatoire est appliqué à la planète, modifiant ses indicateurs.
- Le joueur choisit une Action écologique parmi celles disponible sur la planete.
- L'Element sélectionnée est appliquée, modifiant les niveaux de pollution, de ressources et d'énergie via une fonction de mise à jour des indicateurs qu'il devra être facile de faire évoluer.
- L'état de la planète est vérifié pour voir si une des conditions de Game Over est atteinte.

Le jeu est une simulation où les choix des joueurs choix ont un impact direct sur la survie de la planète. Les joueurs devront faire face aux défis environnementaux tout en prenant des décisions stratégiques pour éviter la crise écologique.

# Évaluation des TP/TD

**Après les séances 2 et 3 de TP,** vous devrez **rédiger individuellement une fiche de synthèse (une page A4 maximum)** des travaux effectués durant la séance de TP. Cette fiche au format PDF doit être postée sur MOODLE au plus **tard le jour suivant la séance de TP** et devra aborder les points suivants :

- Synthèse du travail réalisé durant la séance,
- Les aspects importants du TP : explications des choix de conception, des procédures utilisées, des spécificités du code

**Après la dernière séance de TD/TP,** vous devrez déposer **un livrable final (1 fichier ZIP)** à déposer sur MOODLE avant la date définie su par l'enseignant) :

- Programme complet intégrant toutes les fonctionnalités développées (code source + makefile).
- Documentations associées
- 1 fiche de synthèse PDF pour la seance de TP 4

**Des échanges oraux individuels** avec l'enseignant devront être réalisés afin de valider la compréhension du code produit et la réalisation de chacun des livrables intermédiaires définis pour chaque séance de TP.

Les éléments rendus seront évalués par le professeur et seront également traités automatiquement par **un logiciel de détection de plagiat** ou de copie entre étudiants.

**Barème**

<u>Fiches de synthèse et rendu final :</u> 0 = Non rendue ou rendue hors délai, 1 = Très insuffisant, 2 = Insuffisant, 3 = Correct, 4 = Très bien, 5 = Excellent. Les principaux critères sont :

- Qualité et originalité (travail personnel, sans plagiat ni copie)
- Qualité de la mise en page
- Analyses personnelles des travaux réalisés

<u>Oraux et travail en classe :</u> 0 = absence injustifiée ou refus de travailler, 1 = Très insuffisant, 2 = Insuffisant, 3 = Correct, 4 = Très bien, 5 = Excellent. Les principaux critères sont :

- Compétences en C = Capacité à expliquer et justifier le choix de toutes les instructions présentes dans le code produits.
- Motivation, sérieux, autonomie

<u>Rendus final :</u> 0 = Non rendue ou rendue hors délai, 1 = Très insuffisant, 2 = Insuffisant, 3 = Correct, 4 = Très bien, 5 = Excellent. Les principaux critères sont :

- **Qualité du code** : Bonne gestion des pointeurs, allocation dynamique correcte, structuration claire du code. Gestion de la mémoire et des erreurs : pas de fuites de mémoire,
- Qualité de la **modularité** du projet proposé et de sa mise en oeuvre via un **makefile**
- **Documentation** : Code bien commenté (de préférence avec Doxygen), **rapport détaillé** expliquant les choix de conception et les difficultés rencontrées.
- **Fonctionnalité des versions successives** : Même si le jeu complet n'a pas pu être réalisée, une version fonctionnelle la plus complète possible devra être fournie

**La note finale sera obtenue par : Note Finale = (2 x Fiche de synthèse) / 2 + 2 x Travail en classe + Rendu final**

**Partie 1 (TD 3 + TP2) : Gestion de Liste Chaînée de structures**

Objectifs :

1. Concevoir la structure de données Element pour représenter une action ou un évènement climatique.
2. Développer les **fonctions de création, gestion (ajout/suppression), et affichage d'une liste chaînée** d'Elements.
3. Effectuer **des tests unitaires** pour valider le fonctionnement de chaque fonction.
4. Installation des outils utiles par exemple doxygen (cf plus loin)
5. Créer une **bibliothèque statique** regroupant ces fonctions pour une réutilisation future.

Instructions :

1. Définir la structure Element (avec au moins les champs nom et type, …) constituant les éléments de la liste chaînée.

2. Implémenter les fonctions de gestion de la liste chainée:

- ajouterElement(Element **liste, const char *nomElement, const char *nomType): Ajoute un nouvel Element en début de liste.
- ajouterElementTail(Element **liste, const char *nomElement, const char *nomType): Ajoute un nouvel Element en fin de liste.
- supprimerElement(Element **liste, const char *nomElement): Supprime un Element de la liste.
- afficherElements(const Element *liste): Affiche tous les Elements dans la liste.
- libererListeElements(Element *liste): Libère toute la mémoire allouée pour la liste.
- rechercherElementNom(…), rechercherElementIndice(…), tirageAleatoireElement(…)

3. Test unitaire :

- Écrivez un programme de test qui utilise ces fonctions pour manipuler une liste chaînée d'Elements.
- Vérifiez l'ajout, la suppression, l'affichage, et la libération correcte de la mémoire.

4. Création d'une bibliothèque statique :

- Compilez les fonctions dans une bibliothèque statique (libElements.a) pour une utilisation dans les prochaines séances.

Livrables attendus en fin de TD3/TP2) :

- Code source des fonctions gestion de liste et de tests unitaires.
- Fichier libListeElement.a contenant la bibliothèque statique.
- 1er Makefile et programme principal ( main.c qui lance test_liste_chainee() )

**Partie 2 (TD4 – TP3) : Gestion de la Planete et de ses evolutions**

Objectifs Gestion des Planetes :

1. Concevoir et implémenter la structure Planete + développer des fonctions pour créer, modifier, et afficher une planete, associer dynamiquement des Elements à une planete.
2. **Ajouter des directives de pré-compilation pour optimiser le code et la gestion du code (header-guard, mode debug ou release,…)**
3. **Commenter le code selon la norme doxygen**

Instructions Gestion de la Planete :

1. Définir une structure Planete avec des champs pour le nom, niveauPollution, niveauRessources, niveauEnergie, et des pointeurs vers les listes chaînées d'Actions et de d'Eevenements climatiques, etc
2. Implémenter les fonctions :
   - creationPlanete(const char *nom, int energie, int ressources, int pollution,…):
   - ajouterAction(Planete *p, const char *nomAction, **….**): Ajoute une Action au Planete
   - ajouterEvenement(Planete *p, const char *nomEvenement,….): Ajoute un Evenement a une Planete
   - afficherPlanete(const Planete p): Affiche les caractéristiques d'une planète avec ses Eléments et Actions
   - verifierPlanete(const Planete p): Verifie et Affiche l'etat d'une planete (gestion du game over)
   - modifierPlanete(Planete *p, ...): Permet de modifier certains attributs d une Planete
3. Intégration avec la bibliothèque libElements.a :
   - Utilisez les fonctions de la bibliothèque statique développée lors de la première séance pour gérer les Elements des Planetes

Livrables attendus :

- Code source des fonctions de gestion de la Planete
- Fonction de test pour valider la création, l'affichage, et l'association d'Elements aux Planetes

Objectifs Gestion des tours de jeu :

1. Implémenter les fonctions de sélection aléatoire (ou non) d'actions et d'évènements
2. Développer la logique de jeu et les modes de calcul des évolutions des indices (scores)
3. **Utilisation de pointeur de fonctions pour le calcul des scores (notion de stratégie)**
4. Mettre à jour les attributs de Planete durant chaque tour de jeu
5. Finaliser le jeu en intégrant toutes les fonctionnalités dans la fonction principale (main)

Instructions :

1. Fonctions de gestion du jeu :
   - selectionnerElementAleatoire(Element *liste): Sélectionne un élément aléatoire dans une liste chaînée
   - calculScoreV1(…) : fonctions de calcul des score selon event, action stratégie sélectionnée
   - tourDeJeu(Planete *p, ListeElement *ev, ListeElement *action, …): Simule un tour de jeu, calculant et affichant les Action + Evenement choisis et les évolutions de la planete associées
   - jeuComplet(): Gère une partie complète jusqu'au Game Over
2. Finalisation dans la fonction jeuComplet() :
   - Créez une Planete, ajoutez-lui des Actions et des Evenements climatiques
   - Affichez les informations initiales de la Planete
   - Lancez une partie (full automatique), afficher les évènements et actions et affichez le résultat final
   - Libérez toute la mémoire allouée à la fin du programme

Livrables attendus :

- **Makefile quasi final**
- Programme complet avec la fonction main intégrant toutes les fonctionnalités développées
- Documentation du code

# ANNEXES

**Installation de doxygen sur votre compte debian**

apt-get download doxygen doxygen-gui
dpkg -x doxygen_<version>.deb ~/doxygen_local/
dpkg -x doxygen-gui_<version>.deb ~/doxygen_local/
export PATH=~/doxygen_local/usr/bin:$PATH

doxywizard &
ou en ligne de commande avec Doxyfile → doxygen -g + doxygen Doxyfile

#Mon Doxyfile
# Nom du projet
PROJECT_NAME = "MonProjet"
# Dossiers à documenter
INPUT = ./src ./include
# Générer la documentation HTML
GENERATE_HTML = YES
# Générer un fichier PDF via LaTeX
GENERATE_LATEX = NO
# Activer les graphiques et diagrammes (nécessite Graphviz)
HAVE_DOT = YES