LAKCHIRI IMANE
L1 MATHEMATIQUES
INSA HDF

TP LANGAGE C

RAPPORT

2025

SOMMAIRE :

page

1-Introduction :

Le projet que nous présentons ici vise à développer un programme éducatif interactif développée en langage C, permettant de de renforcer les révisions des mathématiques des élèves du CM1 niveau difficile. L'objectif principal est de proposer une série d'exercices variés : opérations arithmétiques classiques (addition, soustraction, multiplication, divisions), les tables de multiplication, les conversions d'unités, géométrie (volumes, de périmètres, d'aires, etc.). Sous forme de jeu, afin de favoriser l'apprentissage tout en s'amusant.

L'utilisateur peut choisir parmi plusieurs types d'exercices via un menu principal. Chaque exercice pose une question générée aléatoirement, et l'élève dispose de **trois tentatives maximums** pour trouver la bonne réponse. À chaque tentative, le programme indique si la réponse est correcte ou non. Si l'élève réussit dès la première tentative, il reçoit **10 points**, puis **5 points** s'il réussit à la deuxième tentative, et **1 point** à la troisième. En cas d'échec aux trois tentatives, la bonne réponse est affichée, mais aucun point n'est attribué.

L'application inclut également une **gestion complète des scores** : à chaque lancement, le joueur entre son nom, et son score précédent est automatiquement récupéré depuis un fichier texte. À la fin de la session, le nouveau score est enregistré avec la date et l'heure, permettant un suivi personnalisé de la progression. Cette structure encourage les utilisateurs à rejouer pour améliorer leur score tout en consolidant leurs connaissances.

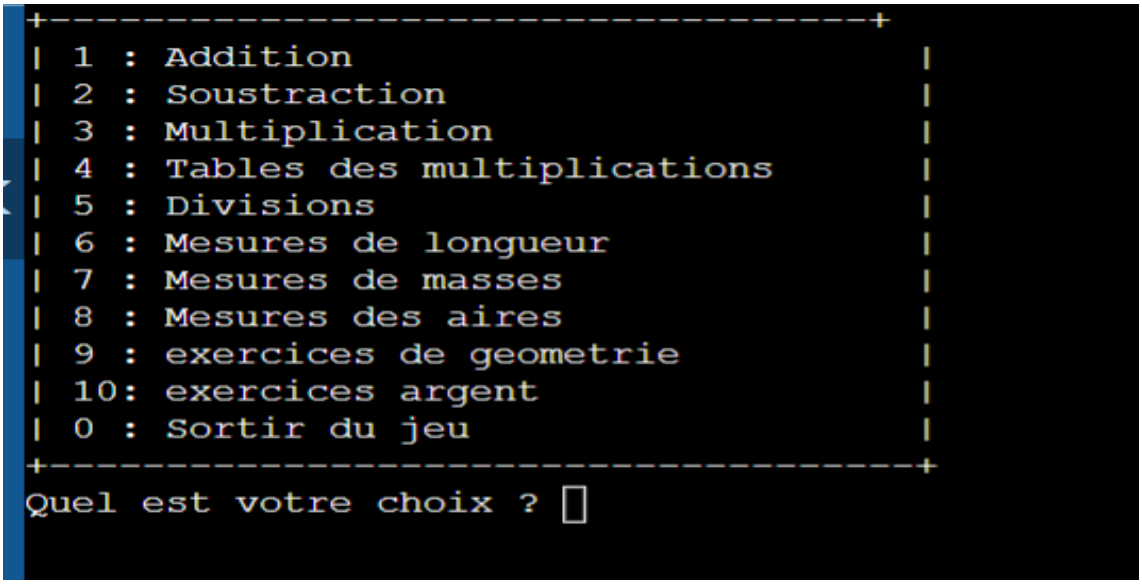
Ce projet combine ainsi apprentissage et évaluation continue des acquis, en s'appuyant sur les fonctionnalités du langage C, tels que les structures conditionnelles, les boucles, les entrées/sorties de fichiers et la génération de nombres aléatoires.

2-Fonctionnalités implémentées :

Fonctionnalité 1 : Menu principal interactif

Le **menu principal interactif** repose sur une boucle `do...while`, qui permet de maintenir l'utilisateur dans l'interface principale tant qu'il ne décide pas explicitement de quitter le programme. À chaque itération, le menu est affiché, l'utilisateur entre son choix, et celui-ci est traité grâce à une structure `switch`, particulièrement adaptée pour gérer plusieurs cas distincts de manière lisible. Chaque option du menu correspond à `un case`, redirigeant l'élève vers un mini-jeu ou une fonctionnalité précise (calculs, conversions, géométrie, etc.). Cette combinaison `do...while` + `switch` garantit une navigation, sans répétition inutile du code, et rend l'interface faire face aux mauvaises saisies grâce à une gestion des cas par défaut. C'est un choix technique à la fois simple, efficace pour gérer ce cas.

```c
        switch (choix) {
            case 1: addition(score); break;
            case 2: soustraction(score); break;
            case 3: multiplication(score); break;
            case 4: tablemultiplication(score); break;
            case 5: divisions_(score); break;
            case 6: mesures_longueur(score); break;
            case 7: mesures_masse(score);break;
            case 8:mesures_aires(score);break;
            case 9:exercice_geometrie(score);break;
            case 10:exo_argent(score);break;
            case 0: printf("Merci de votre visite !\n"); break;
            default: printf("Choix invalide. Essayez encore.\n"); break;
        }
    } while (choix != 0);
```



Fonctionnalité 2 : Génération aléatoire des questions

La **génération aléatoire** a été largement exploitée dans ce projet pour rendre les exercices dynamiques et variés. Grâce à l'utilisation conjointe de **`srand(time(NULL))` et `rand()`.**

Fonctionnalité 3 : compter les scores en utilisant pointeur

Le motif principal derrière l'utilisation d'un pointeur pour le score est d'assurer une **mise à jour directe** de la variable, sans avoir à la renvoyer ou à passer par des retours de fonction. En C, lorsqu'une variable est passée sans pointeur, sa copie est utilisée : toute modification reste locale à la fonction. Avec un pointeur, en revanche, on **agit directement sur l'adresse mémoire** du score initial, ce qui garantit que chaque modification affecte tout le programme. Cela permet une meilleure gestion du score à travers les différents mini-jeux, tout en gardant un code clair, et en évitant les doublons.

Fonctionnalité 4: fonctions pour compter le score selon les tentatives

La fonction `reponsessai()` joue un rôle clé dans la gestion des réponses de l'utilisateur aux exercices de calcul. Elle centralise **la logique des tentatives** et **du système de score**, permettant d'éviter la répétition de code dans chaque mini-jeu. Son but est de :

- **Afficher l'opération à résoudre** (par exemple `5 + 3 = ?`) c'est à dire poser la question en précisant l'opération.
- **Lire la réponse de l'utilisateur**,
- **Vérifier la justesse de cette réponse**, c'est à dire comparer la réponse avec le résultat de l'opération.
- **Attribuer les points** selon le nombre de tentatives (10, 5, 1 point),
- **Afficher la bonne réponse** en cas d'échec après 3 essais,
- **Mettre à jour le score via un pointeur**, ce qui permet de **modifier directement la variable score** globale sans avoir à gérer de valeurs de retour ou variables intermédiaires.

L'appel à la fonction `reponsessai()` dans les différents programmes du jeu (opérations arithmétiques, conversions, etc.) permet de centraliser la gestion des tentatives et du calcul des scores. Cette fonction prend en paramètre les valeurs de l'exercice, la bonne réponse attendue, ainsi qu'un pointeur vers la variable de score, afin de permettre sa mise à jour directe depuis la fonction. **Cette technique permet d'éviter la redondance de blocs similaires pour chaque type d'exercice .

```c
// fonction pour les tentatives et compter les points
void reponsessai(int a, int b, char operation, int resultat, int *score) {
    int reponse, tentatives = 0, points = 0;
    printf("\n%d %c %d = ?\n", a, operation , b);
    while (tentatives < 3) {
        printf("Tentative %d : ", tentatives + 1);
        scanf("%d", &reponse);

        if (reponse == resultat) {
            if (tentatives == 0) points = 10;
            else if (tentatives == 1) points = 5;
            else points = 1;

            printf("Bravo !\n");
            break;
        } else {
            printf("Incorrect.\n");
            tentatives++;
        }
    }

    if (tentatives == 3) {
        printf("La bonne réponse était : %d\n", resultat);
    }

    *score += points;
    printf("Score actuel : %d\n", *score);
}
```

Fonction recuperer_score()

Cette fonction permet de **retrouver un score précédent** enregistré dans le fichier `scores.txt`. Le mode `'r'` dans la fonction `fopen()` en est utilisé pour **ouvrir un fichier en lecture seule**. Cela signifie que le programme peut **lire** les données contenues dans le fichier, mais **ne peut pas les modifier** . La fonction commence par :

- Lire le nom de l'utilisateur
- On lit chaque ligne avec **`fgets`**.(parcourir le fichier ligne par ligne)
- **`sscanf`** permet d'extraire le nom, la date et le score de la ligne.
- On compare le nom trouvé avec celui saisi **(`strcmp`)** pour identifier le bon élève.
- Le dernier score trouvé avec ce nom est conservé.
- Affichage du message en fonction de la présence ou non d'un score antérieur.
- *score = score_total;
  Le score est mis à jour via pointeur pour être accessible dans le programme principal.

→ Si le fichier n'existe pas, **`fopen()`** retourne **`NULL`**, ce qui permet de gérer proprement l'erreur avec une condition

```c
// RÉCUPÉRER LE SCORE
void recuperer_score(char *nom, int *score) {
    char ligne[200], nom_lu[100], date[100];
    int score_lu, score_total = 0, trouve = 0;

    printf("Entrez votre nom : ");
    scanf("%s", nom);

    FILE *fichier = fopen("scores.txt", "r");
    if (fichier != NULL) {
        while (fgets(ligne, sizeof(ligne), fichier)) {
            if (sscanf(ligne, " %[^,], Date: %[^,], Score: %d", nom_lu, date, &score_lu) == 3) {
                if (strcmp(nom, nom_lu) == 0) {
                    score_total = score_lu;
                    trouve = 1;
                }
            }
        }
        fclose(fichier);
    }

    if (trouve) {
        printf("Bienvenue %s, score précédent : %d\n", nom, score_total);
    } else {
        printf("Bienvenue %s, pas de score précédent trouvé.\n", nom);
    }
    *score = score_total;
```

Fonction ecrire_score()

Une fois la session terminée, cette fonction **archive le score de l'élève avec un horodatage précis** dans le fichier. Le mode **`"a"`** est utilisé pour **ajouter sans écraser les données existantes.** Elle utilise **`time()`** et **`localtime()`** pour générer **une date et une heure** complètes, stockées avec le nom de l'utilisateur et son score à l'aide de `fprintf()`.

- Le mode `"a"` (append) permet d'**ajouter des lignes** à la fin du fichier, sans effacer les scores précédents.

→ On choisit ce mode plutôt que `"w"` (write) pour **conserver l'historique des tentatives**.
time_t t = time(NULL);
*struct tm tm = *localtime(&t);*

- Ces lignes permettent de marquer chaque enregistrement dans le temps, utile pour suivre la progression.
- Enregistre sur une seule ligne le **nom, la date/heure complète, et le score**.
- *fclose(fichier)* permet de fermer le fichier .

```c
// enregistrer mes scores
void ecrire_score(char *nom, int score) {
    FILE *fichier = fopen("scores.txt", "a");  // append au lieu de "w"
    if (fichier == NULL) {
        printf("Erreur ouverture fichier !\n");
        return;
    }

    time_t t = time(NULL);
    struct tm tm = *localtime(&t);

    fprintf(fichier, "%s, Date: %d-%02d-%02d %02d:%02d:%02d, Score: %d\n",
            nom,
            tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday,
            tm.tm_hour, tm.tm_min, tm.tm_sec,
            score);

    fclose(fichier);
}
// d'autres exercices
```

Exemple d'exécution

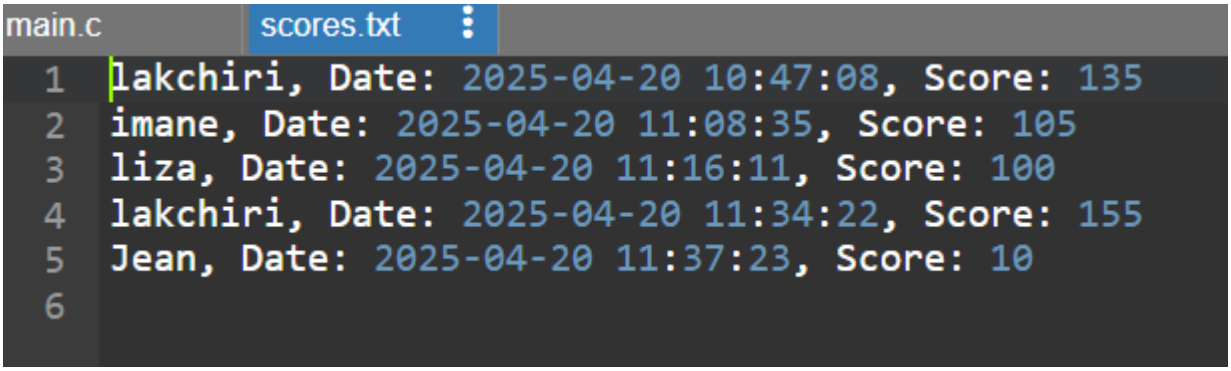
Dans le cas d'un nom d'utilisateur qui existe déjà .

```
Entrez votre nom : lakchiri
Bienvenue lakchiri, score préc dent : 135
Votre score actuel est : 135

+------------------------------------+
```

→ Dans le cas d'un nouvel utilisateur

```
Entrez votre nom : Jean
Bienvenue Jean, pas de score pr cédent trouvé.
Votre score actuel est : 0
```



3- Difficultés rencontrées et solutions apportées :

Au cours du développement de ce projet, plusieurs obstacles techniques et logiques ont été rencontrés :

Problème : Lors de l'enregistrement des scores, le fichier était ouvert en mode `"w"` (write), ce qui écrasait le contenu existant à chaque nouvelle session. Cela entraînait la perte complète des historiques de scores des élèves.
Solution : Le mode d'ouverture a été modifié en `"a"` (append), permettant d'ajouter les nouveaux scores à la fin du fichier sans supprimer les anciens. Ce changement a garanti la persistance des données et le suivi correct de la progression de l'élève.

Problème : La gestion du score entre différentes fonctions causait des incohérences, car les modifications locales dans certaines fonctions n'étaient pas répercutées globalement.

Solution : L'utilisation de pointeurs pour passer le score a permis une mise à jour directe de la variable d'origine, garantissant ainsi un suivi précis et centralisé du score à travers tous les mini-jeux.

Problème : Intégrer un système permettant jusqu'à trois essais par question, avec un score décroissant, complexifiait la logique du programme.
Solution : Une fonction `reponseEssai()` a été développée. Elle centralise la gestion des essais et du score selon le nombre de tentatives, rendant le code plus simplifié.

Autres problèmes :

Parmi les difficultés rencontrées ont concerné plusieurs bugs récurrents, notamment dans la gestion des entrées utilisateur et l'enchaînement des mini-jeux. Des erreurs apparaissaient lorsque l'utilisateur entrait des caractères non prévus ou lorsque le score ne se mettait pas correctement à jour. L'un des principaux défis rencontrés au cours de ce projet a été la gestion du temps. Le sujet du projet ne laissait que peu de marge pour les imprévus, or plusieurs éléments du sujet n'étaient pas clairement définis dès le départ. Certains aspects importants, comme la gestion du score ou la structure des mini-jeux, ont été précisés tardivement, ce qui nous a obligés à revoir l'organisation du code en profondeur, parfois en dernière minute. Cette réorganisation de la logique a généré des perturbations dans le programme, notamment au niveau de l'appel des fonctions et de la mise à jour du score. Un autre obstacle majeur a été la contrainte de temps : le délai imparti pour finaliser le

projet était relativement court pour finaliser le code le tester chaque fois et corriger les erreurs et rédiger le rapport.

Autres solutions :

Pour surmonter ces difficultés, plusieurs solutions ont été mises en place au fil du développement. Les erreurs liées aux entrées utilisateur ont été résolues par des **tests successifs.** La mise à jour incorrecte du score a été corrigée en centralisant sa gestion via un **pointeur**, nous avons régulièrement **demandé l'aide et l'orientation de notre enseignant**, afin de clarifier certains points techniques et éviter les mauvaises interprétations. Par ailleurs, la réorganisation du code, imposée par les ajustements tardifs, a été facilitée par une structure particulière, divisée en fonctions. Enfin, pour gagner du temps et résoudre rapidement certains blocages, nous avons consulté des pages internet, ce qui nous a permis de comprendre certaines erreurs et d'implémenter des solutions fiables dans un délai raisonnable.

4- Commentaires et Suggestions :

Le projet a bien fonctionné dans l'ensemble, notamment grâce à sa structure et l'utilisation claire des fonctions et des techniques bien adaptées. La gestion des scores et la génération aléatoire ont renforcé l'aspect interactif. Quelques difficultés sont apparues dans la gestion des entrées etl'intégration progressive des fonctionnalités (scores selon les tentatives, l'enregistrement des scores dans fichier). Une meilleure définition des consignes dès le début aurait facilité la planification.

Une amélioration significative serait d'intégrer davantage la gestion des pointeurs et des fichiers dans les travaux dirigés (TD), car ces concepts sont souvent abstraits et théoriques lorsqu'ils sont abordés uniquement en cours magistraux (CM). Bien que nous ayons eu un aperçu de ces notions en CM, leur application pratique était insuffisamment couverte dans les TD. Cela a créé une lacune dans notre maîtrise de ces outils, ce qui a rendu leur utilisation dans le projet plus complexe. Un renforcement de ces compétences à travers des exercices pratiques permettrait une meilleure compréhension et application lors de projets futurs.

Au cours de ce TP, plusieurs apprentissages techniques et méthodologiques ont été réalisés. D'un point de vue technique, j'ai amélioré ma maîtrise des pointeurs en C, notamment dans la gestion dynamique des données, ainsi que la manipulation des fichiers pour la gestion des scores. Ces concepts, bien que théoriquement abordés en cours, ont été appliqués concrètement dans le projet, ce qui m'a permis de mieux comprendre leur utilité et leur gestion en pratique. Ailleurs, j'ai également renforcé ma maîtrise de GitHub qui été utilisé pour principalement pour envoyer le code à notre professeur.

5-Auto-évaluation :

1. Évaluation de mon travail :

Codage : 9/10 Sur le plan du codage, j'ai réussi à mettre en œuvre la plupart des fonctionnalités du projet, en suivant une approche modulaire. Cependant, des difficultés sont apparues lors de l'intégration de certaines fonctionnalités complexes, comme la gestion des pointeurs et des fichiers. Le code fonctionne globalement bien.

Planification : 8/10 La planification a été un élément clé dans la structuration du projet, mais elle n'a pas toujours été optimale, surtout en raison de la clarté limitée des consignes. Certaines parties du code ont dû être réorganisées et adaptées au fur et à mesure.

2. Objectifs pour les futurs projets :

Maitriser les pointeurs et la gestion des fichiers **:** Bien que j'aie utilisé les pointeurs et la gestion des fichiers dans ce projet, j'ai constaté qu'il me reste encore des zones d'ombre dans leur utilisation. Pour les futurs projets, je vais approfondir ma compréhension de ces concepts, particulièrement en ce qui concerne la manipulation de fichiers et la gestion des données à travers des pointeurs.

6- Code et GitHub:

Lien pour dépôt GitHub contenant **code** et fichier **README**

https://github.com/Imane-lakchiri/tp-langage-c-.git

Lien du code

https://onlinegdb.com/ZvBapz4Ew