C language

Contenu

★ Introduction du projet :

Ce projet a pour objectif de concevoir et développer un programme en langage C, destiné à la révision des notions mathématiques de niveau CM1. Il prend la forme d'un outil ludique et interactif, conçu pour renforcer les acquis scolaires à travers une série de mini-jeux éducatifs. Le contenu s'inspire principalement du site championmath.free.fr, conformément aux consignes du projet.

L'approche pédagogique adoptée repose sur l'apprentissage par le jeu : chaque mini-jeu traite d'une notion précise du programme (additions posées, multiplications, fractions, etc.) et propose une difficulté progressive. Le niveau visé est volontairement élevé — au moins CM1 difficile — afin de maintenir un bon niveau d'exigence et de développer les compétences des élèves dans la résolution de problèmes complexes.

La réalisation de ce projet s'appuie sur les fondamentaux de la programmation en langage C, notamment la gestion des entrées/sorties, les structures conditionnelles, les boucles, les fonctions et la manipulation de fichiers. Pour mener à bien ce travail, j'ai mobilisé mes connaissances acquises auparavant et en cours, complétées par des recherches dans la documentation ainsi que par l'usage de l'intelligence artificielle, notamment pour comprendre certaines fonctions à l'aide d'exemples concrets comme strtok() que je ne connaissais pas auparavant.

En complément des attentes initiales, j'ai également développé une interface graphique dynamique en GTK, proposée en tant que fonctionnalité bonus dans le cadre du second projet.

★ Fonctionnalitées implémentées :

- **Extrait de la logique de la fonction MiseAjourpoints():**

```c
if (name && prenom && date && points_str) {
    NbPoints = atoi(points_str); //On convertir la chaine en nombre de points


    if (strcmp(name, NomUser) == 0 && strcmp(prenom, PnomUser) == 0) {
        NbPoints += PointsUser;
        char date_nouvelle[30];
        strcpy(date_nouvelle, date_actu());

        // Écrire dans le fichier temporaire avec les nouvelles informations
        fprintf(F2, "%s|%s|%s|%d\n", name, prenom, date_nouvelle, NbPoints);
    } else {
        // Écrire la ligne originale dans le fichier temporaire
        fprintf(F2, "%s|%s|%s|%d\n", name, prenom, date, NbPoints);
    }
}
```

Explications :

Dans cet extrait de code, la fonction `miseAjourpoint()` a pour objectif de mettre à jour le nombre de points d'un joueur trouvé dans le fichier de scores, tout en mettant à jour la date de sa dernière partie. Le processus commence par une comparaison entre le nom et le prénom de l'utilisateur connecté et les enregistrements du fichier afin de localiser le joueur concerné.

Une fois l'utilisateur identifié, la fonction met à jour son score avec la nouvelle valeur et récupère la date actuelle grâce à la fonction `date_actu()`, afin de l'afficher comme la date de la dernière partie du joueur.

Pour garantir l'intégrité des données, au lieu d'écrire directement dans le fichier source, le programme crée un fichier temporaire. Ce fichier temporaire contient les informations suivantes : pour le joueur identifié, son nom, prénom, le nouveau score et la nouvelle date sont enregistrés. Les autres joueurs, quant à eux, ont leurs informations copiées telles quelles dans le fichier temporaire.

Une fois que toutes les lignes ont été traitées, le fichier temporaire remplace le fichier original. Cela permet de conserver toutes les informations à jour, tout en assurant que les modifications sont appliquées de manière efficace sans risque de perte ou de duplication.

- **Extrait de code concernant l'inscription du joueur :**

```c
printf("Etes vous inscrit ? (Oui/Non) : ");
scanf("%s",reponse);
if(strcmp(reponse , "Oui")==0){
    if(VerificationJoueurExiste(NameUser , SurnameUser) == false){
        CreationJoueur(NameUser,SurnameUser,date_actu(),0);
    }else{
        printf("Heureux de vous revoir ! \n");
    }

}else{
    if(VerificationJoueurExiste(NameUser , SurnameUser) == false){
        CreationJoueur(NameUser,SurnameUser,date_actu(),0);
    }
}
```

Explications :

Dans cet extrait de la fonction principale `main()`, il est important de déterminer si l'utilisateur possède déjà un compte. Cependant, il est possible qu'il réponde "Oui" sans être inscrit. Ainsi, plusieurs vérifications sont nécessaires.

Tout d'abord, nous comparons la réponse de l'utilisateur à "Oui" à l'aide de `strcmp()`. Ensuite, la fonction `VerificationJoueurExiste()` est appelée pour vérifier si le joueur est enregistré dans le fichier contenant les données des joueurs (nom, prénom, score, et dernière connexion). Cette fonction prend en paramètre le nom et le prénom de l'utilisateur afin de vérifier son existence dans le jeu.

Si le joueur n'est pas trouvé, nous appelons `CreationJoueur()` pour créer un nouveau compte, en utilisant les informations fournies lors de la connexion. Son score est initialisé à 0, et la date actuelle, récupérée via `dateactu()`, est enregistrée comme première connexion dans le jeu.

Si le joueur est déjà inscrit, un message de bienvenue lui est affiché. Les points ne sont pas affichés automatiquement, mais le joueur peut les consulter à tout moment en appuyant sur la touche dédiée dans le menu de navigation qui lui sera proposé par la suite ou simplement entre chaque partie .

- **Extrait de code concernant la récupération des points du joueur :**

```c
printf("Souhaitez vous récuperer vos points (O/N) ? :");
scanf(" %c", &RecupPoint);

if (RecupPoint == 'O') {
    AfficherPointsJoueur(NameUser,SurnameUser);
}
```

Explications :

L'utilisateur se voit proposer d'afficher ses points. S'il accepte, la fonction `AfficherPointsJoueur()` est appelée. Toutefois, cet appel est volontairement placé après l'enregistrement du joueur, afin de garantir que le nombre minimal de points soit bien attribué au joueur avant l'affichage. Cela permet de s'assurer que, même lors d'une première connexion, le joueur dispose d'un score cohérent à visualiser.

- **Extrait de code concernant la fonction d'affichage des points :**

```c
void AfficherPointsJoueur(char *NomUser, char *PnomUser) {
    int NbPoints;
    if (chercherJoueur(NomUser, PnomUser, &NbPoints)) {
        printf("Vous avez : %d points.\n", NbPoints);
    } else {
        printf("Vos points n'ont pas été trouvés par le système.\n");
    }
}
```

Explications :

Cette fonction peut sembler simple au premier abord, mais elle repose sur la fonction `chercherJoueur()`, qui est un peu plus complexe.

Lors de la première condition, `chercherJoueur()` est appelée pour rechercher le joueur dans le fichier d'enregistrements. Cette fonction parcourt le fichier ligne par ligne à la recherche d'une correspondance entre le nom et le prénom fournis. Pour cela, elle utilise `strtok()` afin de découper chaque information et extraire les celles relatives aux joueurs.

Tant que le joueur n'est pas trouvé, la lecture du fichier continue. Une fois une correspondance identifiée, la fonction extrait les points associés, encore sous forme de chaîne de caractères, puis les convertit en entier à l'aide de `atoi()`.Elle retourne ensuite le score du joueur ou si aucun joueur correspondant n'est trouvé, un message indiquant que ses points n'ont pas pu être récupérés est affiché.

- **Extrait de code de la partie principale du programme :**

```c
switch (choix) {
    case 0:
        printf("Merci de votre visite !\n");
        break;
    case 1:
        MiseAjourPoints(NameUser,SurnameUser,FonctionAddition());
        break;
```

Explications :

Cette partie représente le cœur du projet : l'appel des différentes fonctions en fonction du choix de l'utilisateur. Pour cela, l'utilisation d'un switch-case s'est révélée être la solution la plus adaptée, car elle permet de traiter chaque option de manière distincte et lisible.J'ai volontairement pris qu'un extrait car celui ci est assez évident car il comprends dans une chaque cas différents l'appel d'une fonctions différentes selon le mini jeu qui sera imbriquée dans la fonction `MiseAjourPoints()`.

Par exemple, dans le cas 1, si l'utilisateur choisit cette option, la fonction addition() est appelée à l'intérieur de `MiseAJourPoints()`. Cette dernière est indispensable car elle permet de mettre à jour les points du joueur à la fin du mini-jeu.

J'ai conçu chaque fonction de mini-jeu pour qu'elle retourne un entier, facilitant ainsi son intégration dans la fonction `AfficherPointsJoueur()`. Cette dernière prend en paramètres le nom, le prénom et le score du joueur. Grâce à ce fonctionnement, il est possible d'appeler directement une fonction de mini-jeu dans les paramètres de `MiseAJourPoints()`.

Chaque mini-jeu dispose de sa propre fonction, qui retourne systématiquement un entier représentant le nombre de points gagnés pendant la partie. Une fois la partie terminée, la fonction `MiseAJourPoints()` est appelée pour actualiser le score du joueur et enregistrer la date de sa dernière participation, grâce à l'appel de `dateactu()` intégré dans cette même fonction. À chaque connexion, le joueur a la possibilité de récupérer ses points, soit dès le début de la session, soit entre deux parties via le menu principal.

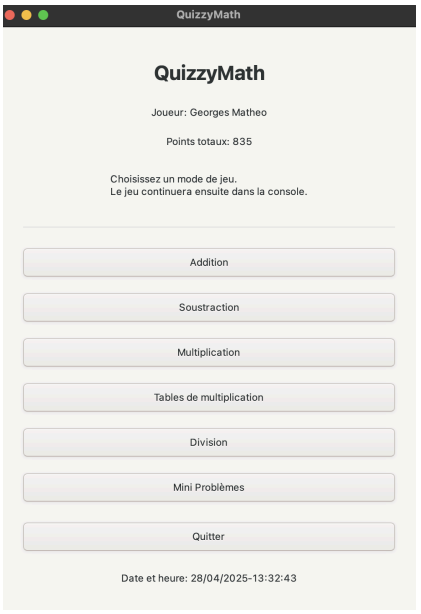
Toutefois , le cas par défaut du menu permet de gérer les saisies invalides.Si l'utilisateur entre un nombre en dehors de l'intervalle [0-8], un message d'erreur s'affiche pour l'inviter à saisir un choix valide.

★ Bonus de fonctionnalitées importées pour l'interface GTK :

```c
// Prépare les données pour le lancement du jeu
gtk_widget_hide(userData->window);

gtk_main_quit();

demarrer_jeu_console(jeuChoisi, userData);
```



Explications :

L'interface du jeu débute par l'affichage d'une fenêtre où l'utilisateur peut choisir son mode de jeu parmi plusieurs options, comme l'addition, la soustraction, ou la multiplication. Une fois qu'un mode de jeu est sélectionné en cliquant sur un bouton, la fonction `on_mode_selected()` est activée pour identifier le choix de l'utilisateur.

Cette fonction compare le texte du bouton cliqué et détermine ainsi quel jeu a été choisi. Ensuite, un message de confirmation apparaît, indiquant à l'utilisateur quel mode de jeu a été sélectionné.

Après la confirmation de l'utilisateur, la fenêtre graphique est fermée, et la fonction `demarrer_jeu_console()` est appelée pour lancer le jeu dans la console. Cela marque la transition de l'interface graphique vers un environnement console, permettant à l'utilisateur de poursuivre son expérience de jeu directement dans le terminal.

★ Difficultés rencontrées et solutions apportées :

- **Problème lié à l'usage de la fonction strtok et des fichiers**

```c
// Extraire les champs
name = strtok(copie, "|");
prenom = strtok(NULL, "|");
date = strtok(NULL, "|");
points_str = strtok(NULL, "|");

if (name && prenom && date && points_str) {
    NbPoints = atoi(points_str); //On convertir la chaine en nombre de points

    if (strcmp(name, NomUser) == 0 && strcmp(prenom, PnomUser) == 0) {
        NbPoints += PointsUser;
        char date_nouvelle[30];
        strcpy(date_nouvelle, date_actu());

        // Écrire dans le fichier temporaire avec les nouvelles informations
        fprintf(F2, "%s|%s|%s|%d\n", name, prenom, date_nouvelle, NbPoints);
    } else {
        // Écrire la ligne originale dans le fichier temporaire
        fprintf(F2, "%s|%s|%s|%d\n", name, prenom, date, NbPoints);
    }
}
```

Explications :

De mon côté, la principale difficulté rencontrée concernait l'utilisation de la fonction `strtok()` pour enregistrer les joueurs dans un fichier en utilisant un séparateur.

J'ai d'abord utilisé la fonction `fscanf()` pour écrire des données formatées sans séparateur, ce qui m'a permis de structurer l'enregistrement de base. Ensuite, après avoir consulté la documentation, j'ai compris l'usage de `strtok()` pour découper une ligne en plusieurs champs. Étant donné que `strtok()` modifie la chaîne d'origine, j'ai utilisé `strdup()` pour en faire une copie.

J'ai initialisé quatre variables pour stocker les informations extraites avec `strtok()`. Lorsqu'un joueur est identifié (nom et prénom correspondants), son score est incrémenté et la date de sa dernière partie est mise à jour grâce à la fonction `date_actu()`, en remplaçant l'ancienne valeur avec `strcpy()`.

La dernière difficulté concerne la mise à jour des données. Pour cela, j'ai utilisé un fichier temporaire dans lequel j'écris les nouvelles données du joueur concerné, tandis que les autres lignes sont simplement copiées.

Une fois ce traitement terminé, le fichier temporaire remplace l'original. Ainsi, le joueur connecté voit son score et sa date mis à jour, tandis que les autres conservent leurs informations intactes.

- **Autre petit problème sur le type d'une variable :**

Au début, j'ai identifié une erreur liée à l'invalidité du pointeur retourné lors de l'obtention de la date et de l'heure actuelles. Cette erreur provenait du fait que la variable contenant ces informations était déclarée localement dans la fonction, ce qui entraînait sa destruction une fois la fonction terminée. Le pointeur devient alors invalide, pouvant provoquer un comportement indéfini lors de son utilisation.

Pour corriger ce problème, j'ai déclaré la variable en tant que `static`, ce qui permet de la faire persister en mémoire même après la fin de l'exécution de la fonction. Ainsi, le pointeur reste valide et la chaîne contenant la date et l'heure actuelle peut être utilisée de manière fiable en dehors de la fonction.

★ Commentaires et suggestions :

- **Commentaire sur ce projet :**

Ce projet en langage C m'a permis de renforcer mes compétences en programmation, notamment à travers la création d'un jeu de mathématiques simple et intuitif. La logique du jeu, bien pensée et facile à mettre en œuvre, m'a permis de me concentrer sur les aspects techniques du projet sans être bloqué par une complexité inutile. J'ai appris à utiliser la fonction `strtok()` pour découper des chaînes de caractères, ce qui m'a été utile pour traiter les informations de manière efficace.

De plus, j'ai exploré la gestion des fichiers temporaires, une technique qui m'a permis de modifier des données dans un fichier sans altérer l'original, un concept clé dans la gestion des données persistantes en C.

Enfin, une autre expérience intéressante fut la création d'une interface graphique avec GTK, ce que je ne savais pas être possible en C, ayant toujours utilisé Tkinter en Python pour la création d'interfaces. Cette découverte m'a permis d'élargir mes connaissances et de mieux comprendre comment les interfaces peuvent être développées dans différents langages.

- **Suggestions sur le projet :**

En ce qui concerne les améliorations possibles, plusieurs idées pourraient être explorées. D'abord, un système de récompenses plus dynamique serait un ajout intéressant, notamment l'intégration de multiplicateurs de points pour les joueurs réguliers ou ceux qui enchaînent des séries de bonnes réponses. Cela permettrait de motiver davantage les utilisateurs et d'ajouter un aspect de progression au jeu.

Ensuite, pour diversifier l'expérience de jeu et s'adapter à différents niveaux de compétence, l'ajout de nouveaux modes de jeu serait pertinent. Par exemple, un mode "difficile" ou "contre-la-montre", où la rapidité des réponses influerait sur le score, offrirait des défis plus stimulants pour les joueurs plus expérimentés.

Enfin, une amélioration possible serait de mettre en place un système de collecte de retours utilisateurs, comme un formulaire (ex. : Google Forms) destiné aux enseignants et aux élèves. Cela permettrait aux développeurs de recueillir des avis précieux pour améliorer le jeu et l'adapter encore mieux aux besoins pédagogiques des utilisateurs.

★ Auto - Evaluation :

Je préfère ne pas trop m'attarder sur la performance réalisée car celle-ci dépend fortement de la motivation et des moyens que se donne la personne pour réaliser un objectif.

Étant donné que j'avais déjà une expérience en programmation en langage C, je ne considérerais pas ce projet comme simple. Cependant, j'ai choisi d'ajouter des éléments supplémentaires, tels que la programmation de l'interface demandée en bonus dans le projet B et l'utilisation de la fonction `strtok()`. J'ai apprécié le temps consacré à ce projet, car j'ai opté pour une approche progressive en réalisant une petite partie chaque jour, plutôt que de tout faire en quelques heures d'affilée.

Pour mes futurs projets, comme je vous l'ai probablement déjà mentionné, j'ai l'intention de réaliser les deux autres projets pendant la période d'inactivité scolaire. Cela me permettra de continuer à promouvoir mes compétences en langage C et d'approfondir mes connaissances.

★ Lien GitHub et structure du code :

Le dépôt comprend plusieurs fichiers : le fichier principal contenant la version demandée du projet, une version plus avancée, un fichier listant les joueurs, et un fichier d'en-tête regroupant toutes les fonctions nécessaires au bon fonctionnement du programme.

Lien Github du code : https://github.com/NoLoveJustFunds/ProjetC