TD5 de Programmation C

L1 MIASHS, 2022-2023

Le but de ce TD est de réaliser différents programmes avancés mettant en œuvre tous les concepts vus dans ce cours, permettant d'obtenir de petites applications proposant diverses actions.

Exercice 1 (Jeu du 421) *Vous allez programmer une application permettant de jouer au jeu de dés appelé 421, dans une version simplifiée à un seul joueur. Ce jeu consiste à lancer 3 dés et à marquer des points en fonctions des valeurs obtenues. Pour chaque partie, on peut effectuer jusqu'à 3 lancers de dés, le résultat de la partie étant le cumul des points obtenus sur ces lancers. Le joueur pourra effectuer jusqu'à 10 parties.*

Les activités suivantes vont vous permettre d'implanter progressivement cette application, vous permettant de tester votre programme au fur et à mesure de son écriture.

Activité 1.1 *Commencer par déclarer les principales variables nécessaires pour l'application : nombre de parties jouées (`nbparties`), nombre de lancers réalisés dans la partie courante (`nblancers`), tableau des résultats des parties jouées (`resultats`), tableau représentant un lancer (donc 3 entiers) (`lancer`).*
Puis initialiser à 0 toutes ces informations.
Vous aurez besoin d'autres variables, n'oubliez pas de les déclarer, et de les initialiser si nécessaire.

Activité 1.2 *L'application réalisée bouclera sur plusieurs actions, proposées par le menu suivant :*

```
Actions proposées pour la partie 3 :
1. Commencer une nouvelle partie
2. Lancer les dés
0. Quitter le jeu
Que faire ?
```

Programmer une itération qui affiche ce menu (précisant le numéro de la partie en cours), saisit le choix du joueur, et selon ce choix affiche uniquement (pour l'instant) l'action demandée. Si le choix est de quitter le jeu, l'itération doit s'arrêter et le programme se terminer.

Activité 1.3 *Programmer les instructions à exécuter si l'action 1 est choisie : si le nombre de parties jouées est égal à 10, alors signaler au joueur qu'il a atteint le nombre maximal de parties ; sinon, incrémenter le nombre de parties jouées, réinitialiser le nombre de lancers et prévenir le joueur que la nouvelle partie est prête.*

Activité 1.4 *Programmer les instructions à exécuter si l'action 0 est choisie : si le joueur a effectué au moins une partie, lui afficher le nombre de points obtenu à chaque partie jouée ainsi que le nombre moyen de points.*

Activité 1.5 *Programmer les instructions à exécuter si l'action 2 est choisie consiste à réaliser le lancer des dés, et calculer le nombre points obtenus.*

Activité 1.5.1 *Lancer les dés consiste à choisir une valeur entre 1 et 6 pour chaque dé. Pour cela vous utiliserez la fonction `rand` de la bibliothèque `stdlib.h` déjà vue lors de plusieurs TD)*[1]*. Ne pas oublier d'initialiser le tirage aléatoire au tout début de votre programme à l'aide des fonctions `srand` et `time` des bibliothèques `stdlib.h` et `time.h`.*
Les valeurs obtenues pour chaque dé doivent être rangées dans le tableau `lancer` par ordre décroissant, cela vous aidera à calculer les points obtenus.
Afficher au joueur les valeurs des dés et incrémenter le nombre de lancers effectués.

1. Rappel : `rand()` permet d'obtenir un nombre au hasard ; `rand()%6` permet d'obtenir un nombre entre 0 et 5.

Activité 1.5.2 *Calculer le nombre de points obtenus par ce lancer et l'ajouter au résultat de la partie. Pour cela, voici les combinaisons qui rapportent des points :*

- *4, 2 et 1 : 10 points (le fameux 421).*
- *1, 1 et 1 : 7 points (triple 1).*
- *x, 1 et 1 : x points (x purs).*
- *x, x et x : x points (triple).*
- *x, x-1 et x-2 : 2 points (tierce).*
- *sinon : 1 point.*

Effectuer les tests de combinaison dans l'ordre ci-dessus, afin que le joueur ait un maximum de points si son lancer correspond à plusieurs combinaisons (exemple : 1, 1 et 1).
Afficher le nombre de points obtenus au joueur (avec un message le félicitant, ou pas...)

Activité 1.5.3 *Avant de lancer les dés et de calculer les points obtenus, vérifiez qu'une partie est bien commencée (`nbparties` n'est pas à 0), que le jeu n'est pas terminé (nombre maximal de parties pas atteint ou nombre maximal de lancers pas atteint), et que le joueur n'a pas déjà fait ses 3 lancers dans la partie courante.*

Exercice 2 (Construction d'une phrase en anglais) *L'objectif de cet exercice est de travailler avec des chaînes de caractères, pour former une phrase en anglais*[2]*. L'application réalisée bouclera sur diverses actions, proposées par le menu suivant :*

```
Actions proposées :
1. Afficher le texte courant
2. Ajouter un mot à la fin
3. Insérer un espace
4. Supprimer un caractère
5. Mettre une partie du texte en majuscules
6. Mettre une partie du texte en minuscules
0. Quitter l'application
Que faire ?
```

Les étapes suivantes vont vous permettre d'implanter progressivement cette application, vous permettant de tester votre programme au fur et à mesure de son écriture.

Activité 2.1 *Définir dans votre programme une constante représentant la taille maximale du texte, à savoir 50 caractères.*
Déclarer un tableau de caractère de cette taille qui contiendra le texte construit, et un autre qui contiendra un mot à ajouter au texte.
Vous aurez besoin d'autres variables au fur et à mesure de l'écriture de votre programme, comme par exemple une variable contenant la longueur du texte.

Activité 2.2 *Demander à l'utilisateur un premier mot qui permettra d'initialiser le texte. Calculer sa longueur et la stocker dans une variable que vous pourrez utiliser à chaque fois que nécessaire par la suite.*

Activité 2.3 *Écrire le code qui affiche le menu ci-dessus, saisit le choix de l'utilisateur (sous forme d'un chiffre entre 0 et 6), puis selon ce choix effectue le traitement qui consiste (pour l'instant) à simplement afficher l'intitulé de l'action.*
Ajouter une itération autour de tout cela pour répéter l'affichage du menu et l'action choisie tant que l'utilisateur n'a pas choisi de quitter l'application.
Les activités suivantes vont vous permettre de programmer le traitement associé à chacune de ces actions (et qui remplacera donc le simple affichage de l'intitulé de cette action).

2. Travailler avec des mots en anglais est juste une excuse pour ne pas manipuler des mots avec des accents, qui posent parfois problème en C.

Activité 2.4 *Le code correspondant à l'action 1 doit afficher le contenu du texte ainsi que sa longueur.*

Activité 2.5 *Pour l'action 2, demander un mot à l'utilisateur, et si sa longueur ajoutée à la longueur du texte existant ne dépasse par la limite prévue, alors ajouter ce mot au bout du texte grâce à la commande* `strcat`[3] *de la bibliothèque* `string.h`*. Ne pas oublier de modifier la longueur du texte dans la variable la stockant.*

Activité 2.6 *L'action 3 n'est possible que si le texte n'est pas de longueur maximale. Demander à l'utilisateur l'indice dans le texte où insérer cet espace (s'assurer que cet indice est correct), décaler d'une case vers la droite tous les caractères du texte à partir de cet indice, puis placer le caractère ' ' à l'indice indiqué.*
Remarque : pour limiter les erreurs de l'utilisateur, en lui demandant l'indice vous pouvez lui préciser dans quel intervalle de valeurs.
Attention : si votre texte contient 7 caractères, cela signifie que le 8e caractère du tableau est le marqueur de fin '\0' ; il faut donc aussi décaler ce marqueur vers la droite.

Activité 2.7 *Pour l'action 4, demander à l'utilisateur l'indice dans le texte du caractère à supprimer (s'assurer qu'il est correct), décaler d'une case vers la gauche tous les caractères du texte qui suivent cet indice, ainsi que le marqueur de fin.*

Activité 2.8 *Pour l'action 5, demander à l'utilisateur l'indice de début et l'indice de fin pour identifier le morceau de texte à transformer (s'assurer que ces indices sont corrects), puis mettre en majuscule tous les caractères dans cet intervalle grâce à la fonction* `toupper`[4] *de la bibliothèque* `ctype.h`*.*

Activité 2.9 *L'action 6 est identique à la 5, sauf qu'il faut utiliser la fonction* `tolower` *pour mettre un caractère en minuscule.*

3. Rappel : `strcat(s1,s2)` ajoute `s2` à la fin de `s1`, le résultat étant dans `s1`.
4. `toupper(c)` renvoie le caractère `c` passé en majuscule si c'est une lettre, sinon renvoie `c`.