UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0601 – 2021/2022
***Resp.* : Cyril Rabat**

TP n°3

Les fichiers bas-niveau (suite)

Le but de ce TP est de tester la lecture/écriture de chaînes de caractères et de structures dans des fichiers binaires. La seconde partie présente l'éditeur qui fait partie du projet.

1 Structures et chaînes de caractères

Dans cet exercice, nous allons tester la sauvegarde et la lecture de structures et de chaînes de caractères dans un fichier binaire.

Questions

1. Écrivez un programme qui prend en argument le nom d'un fichier. S'il n'existe pas, un fichier vide est créé. Le programme demande ensuite à l'utilisateur de saisir une chaîne de caractères. Elle est ajoutée à la fin du fichier. Ensuite, toutes les chaînes de caractères du fichier sont affichées, une par une. La dernière doit correspondre à la chaîne de caractères qui a été ajoutée en dernier.

   Vous devez lire les chaînes de caractères une par une et non les lire toutes en une seule fois.

2. Créez une structure personne contenant un nom et un prénom (une chaîne de taille fixe de 20 caractères), ainsi qu'un âge (un entier). Initialisez la structure de manière statique (directement dans votre code) puis sauvegardez-la dans un fichier. Vérifiez son contenu avec `od`.
3. Maintenant, écrivez un programme qui permet de lire la structure.
4. Modifiez vos deux programmes précédents en utilisant des `char*` pour le nom et le prénom.

2 Éditeur de cartes pour le projet

Le but de cet exercice est de présenter la première partie du projet final. Il s'agit de créer un éditeur de cartes en exploitant l'interface *ncurses* réalisée dans le premier TP et en y ajoutant la gestion des fichiers.

Une carte correspond à une matrice de 40 colonnes par 20 lignes. Elle peut contenir des cases de base : herbe (vert), eau (bleu), sable (jaune) et cailloux (rouge). Sur ces cases, nous pouvons ajouter des éléments : un trésor (symbole `$`), un monstre (symbole `M`), un artefact (symbole `A`) ou un obstacle (symbole `X`).

Il est très important que votre carte respecte les dimensions imposées.

Un monstre est caractérisé par une chaîne de caractères de taille variable, ainsi que de 5 valeurs (des `unsigned short`) : points de vie, armure, force, vitesse de déplacement et vitesse de frappe. Un artefact est lui aussi caractérisé par les mêmes attributs (lorsque l'utilisateur gagne un artefact, les caractéristiques de son personnage sont modifiées en fonction de celles de l'artefact). Un trésor et un obstacle n'ont pas de caractéristiques particulières.

La sauvegarde d'une chaîne de caractères de longueur variable implique que seuls les caractères 'utiles' sont sauvegardés. Cependant, il peut y avoir une taille maximale afin de pouvoir la lire au clavier.

L'éditeur prend un nom de fichier en argument. Si le fichier existe, la carte existante est affichée et l'utilisateur peut la modifier. Sinon, une carte vide (remplie d'herbe) est créée. L'utilisateur peut ensuite sélectionner un outil (cases de base ou éléments) pour le positionner sur la carte en cliquant avec la souris. Il ne peut y avoir qu'un seul élément sur une case qui peut être posé sur une case de base. Lorsqu'un monstre ou un artefact est posé, l'utilisateur doit saisir les attributs au clavier (dans la fenêtre d'informations, par exemple).

Avant de commencer à coder, vous devez impérativement réfléchir à une structure pour votre carte et votre fichier : comment stocker la matrice et comment stocker les attributs des éléments. Créez ensuite les structures associées avec toutes les fonctions/procédures nécessaires. Faites ensuite tous les tests : lire/écrire une carte, modification (modification d'une case, ajout/suppression de monstres/artefacts, etc.). Une fois tous ces tests réussis, passez à l'intégration *ncurses*.

Un peu plus loin

Le but du projet est de développer un jeu qui consiste à déplacer un personnage sur les cartes créées à l'aide de l'éditeur de cartes. Ce personnage récupère les trésors, les artefacts, se bat avec des monstres. Le personnage commence sur une des cartes choisies aléatoirement (elles sont placées dans un répertoire spécifique). Lorsqu'il atteint un bord (haut, bas, droite ou gauche), il passe sur la carte suivante. Si elle n'a pas été visitée, elle est choisie aléatoirement. Si le personnage revient sur une carte déjà visitée, elle est affichée suivant l'état dans lequel il l'a laissé (monstres tués ou blessés, artefacts ramassés).

Les cartes créées par l'éditeur ne sont pas modifiées dans le jeu. Elles doivent donc être recopiées avant d'être utilisées dans le jeu.

L'état du jeu est sauvegardé **dans un fichier unique**. Il contient notamment les cartes visitées. Il est donc nécessaire de réfléchir à des structures et une architecture de fichier permettant de sauvegarder l'état des cartes, de les retrouver, etc.

Sans coder le jeu (les éléments vous seront fournis plus tard), vous pouvez déjà créer les structures en et les fonctions/procédures nécessaires pour gérer les cartes visitées.