Tableaux, Chaînes de caractère, Entrée standard

ENSIIE FISA IAP 2019-2020

Si vous avez déjà fait les exercices suivants pendant le premier TP, vous pouvez les passer.

Exercice 1 — *Appartenance*

Créer un programme `tableau.c` contenant une fonction qui, connaissant un entier `size`, un tableau `tab` contenant `size` entiers et un entier `m`, renvoie 1 si `m` appartient au tableau `tab`.

Exercice 2 — *Dichotomie*

Dans le fichier `tableau .c`, ajoutez une fonction qui, connaissant un entier `size`, un tableau `tab` contenant `size` entiers triés et un entier `m`, effectue une recherche dichotomique de `m` dans le tableau.

Exercice 3 — *Passage par valeur et passage par référence, comment renvoyer un tableau ?*

Qu'affiche le programme suivant ?

```
1       void f(int x){
2       x = 1;
3       }
4       void g(int t[]){
5       t[0] = 1;
6       }
7       int main(void){
8       int x = 3;
9       f(x);
10      int t[1];
11      t[0] = 3;
12      g(t);
13      printf("%d\n", x);
14      printf("%d\n", t[0]);
15      }
```

Dans le fichier `tableau.c`, ajouter une fonction qui, connaissant un entier `size` et deux tableaux `tab` et `tab2` contenant `size` entiers, écrase le contenu de `tab2` avec le contenu `tab` inversé.

Dans le fichier `tableau.c`, ajouter une fonction qui, connaissant un entier `size`, un tableau `tab` contenant `size` entiers, écrase `tab` avec son propre contenu inversé.

Remarque : En C, renvoyer un tableau est complexe. On va donc éviter de le faire.

Exercice 4 — *Duplicat*

Créer un programme `duplicat.c` qui contient une fonction prenant en entrée trois chaînes de caractères `c`, `s` et `t` et modifie `c` pour qu'il contienne `s` puis `t`. Créer ensuite une seconde fonction, utilisant la première, prenant en entrée deux chaînes `c` et `s` et un entier `n` et modifie `c` pour qu'elle contienne `n` fois de suite la chaîne `s`.

Exercice 5 — *affichage*

Créer un programme `printf2.c` qui contient une fonction prenant en entrée une chaîne de caractères `s` et qui, à l'aide d'une boucle `for` et de la fonction `strlen`, affiche chaque caractère de `s` dans l'ordre sur des lignes distinctes. Faire une seconde fonction qui a le même comportement, mais qui utilise une boucle `while` et qui n'utilise pas la fonction `strlen`.

Exercice 6 — *Oublions le \0*

Créons une chaîne à partir de rien ainsi :

```
1       char s1[10];
2       s1[0] = 'a'
3       s1[1] = 'b'
4       s1[2] = 'c'
5       s1[3] = 'd'
6       printf("%s\n", s1);
7
```

Que se passe-t-il ? A votre avis pourquoi, comment peut-on y remédier ?

Exercice 7 — *palindrome*

Créer un programme `palindrome.c` qui contient deux fonctions. La première reçoit une chaîne de caractères et vérifie si cette chaîne est un palindrome. Si c'est le cas elle renvoie 1 et sinon elle renvoie 0. La seconde reçoit deux chaînes de caractères `s` et `p`. La fonction modifie `p` pour qu'elle devienne un palindrome en mettant à la suite `s` et la chaîne `s` renversée.

Exercice 8 — *Chaîne entière*

Créer un programme `isnumber.c` qui contient une fonction qui, connaissant une chaîne de caractère, vérifie si cette chaîne est un entier (donc constitué uniquement de numéros). Vous pouvez utiliser la fonction `isdigit(char c)` qui renvoie 1 si `c` est un chiffre et 0 sinon.

Pour pouvoir utiliser `isdigit`, vous devez ajouter `#include <ctype.h>` au début du fichier.

Exercice 9 — *Regroupement* Ecrire une fonction qui prend 2 tableaux `t1` et `t2` de `n` entiers, avec, pour tout $\texttt{i} \leq \texttt{n}$, $0 \leq \texttt{t1[i]} \leq \texttt{n}$. Cette fonction doit remplir `t2[j]` avec le nombre de fois que `j` apparaît dans le tableau `t1`.

Exercice 10 — *Permutations* Ecrire une fonction qui connaissant un tableau de n entier, l'entier n et un entier k renvoie le nombre de permutations qu'il faut effectuer pour que le tableau soit séparé en deux parties : à gauche tous les entiers inférieurs à k et à droite tous les entiers supérieurs à k.

Exercice 11 — *Envergure* Ecrire une fonction qui prend 1 tableau d'entiers en entrée et renvoie l'écart maximum existant entre deux copies d'un même entier dans le tableau. Si aucun entier n'est dupliqué dans le tableau, la fonction renvoie -1. Par exemple avec le tableau {`4, 1, 9, 5, 9, 2, 5`} la réponse est 3 car l'écart entre les chiffres 5 est 3.

Exercice 12 — *Join* Ecrire un programme `join.c` qui prend en entrée $c > 1$ arguments et crée et affiche une chaîne de caractère. Cette chaîne contient dans l'ordre tous les arguments exceptés le premier, et ce premier argument est inséré entre chaque paire d'arguments.

```
>  machin:~ ./join -- abc youpi 42 trololo
>  abc--youpi--42--trololo
```

Exercice 13 — *Split* Ecrire un programme `split.c` qui prend en entrée 2 arguments c et s et qui retire de s toute occurrence de c et affiche chaque morceau de s restant sur des lignes séparées.

```
>  machin:~ ./split ; mmh;champagne;;yep;champagne
>  mmh
>  champagne
>
>  yep
>  champagne
```

Vous pouvez gérer les cas ambigus comme vous le souhaitez.
(Par exemple `./split --- hey----je---coupe-----où?`)

Exercice 14 — *On recode printf*

Ecrire un programme `printf2.c` qui prend en entrée une chaîne contenant des lettres, des chiffres, et les formats `%d` et `%s` et d'autres arguments.

Le programme doit se comporter comme si on avait appelé printf avec les arguments dans cet ordre. Il doit afficher une erreur si le format ne correspond pas aux arguments.

Exercice 15 — *Courses de chevaux*

On dispose de 4 tableaux contenant chacun $n < 100$ éléments. Ces éléments sont tous décrits dans le fichier annexe `chevaux-i.txt` ; avec i entre 1 et 5.

La première ligne contient deux entiers c et p, le nombre de chevaux et le nombre de parieurs. Les c lignes suivantes contiennent 2 éléments : le nom du cheval et son classement. Les p lignes suivantes contiennent 3 éléments : le nom du parieur, le numéro du cheval (entre 0 et $c-1$) sur lequel il a parié, et le montant qu'il a parié.

1. On passera le fichier en entrée standard lors de l'appel au programme. Ecrire une fonction qui prend 5 tableaux en entrée, deux tableau de `char* chevaux` et `parieurs` et trois tableaux d'entiers `classement`, `chevalPari`, `montantPari` ; et qui les rempli avec les informations contenues dans le fichier `chevaux.txt`.
2. Ecrire une fonction qui affiche les noms des chevaux dans l'ordre du classement.
3. Ecrire une fonction qui calcule la cote d'un cheval. Pour un cheval, la cote est la somme des montants pariés sur les autres chevaux divisé par le montant parié sur ce cheval.

   Un parieur qui n'a pas parié sur le premier cheval perd tous son argent. Un parieur qui a parié correctement gagne sa mise plus le montant de sa mise multiplié par sa cote.
4. Ecrire une fonction qui affiche, pour chaque parieur, son nom et la différence d'argent entre avant et après le pari.
5. **Pour aller plus loin.** On fait plusieurs courses et on souhaite savoir au bout de combien de courses tous les parieurs ont perdu tout leur argent. Chaque parieur démarre avec 100 euros. Pour chaque course, le classement des chevaux est aléatoire, les parieurs parient 10% de leur mise, arrondi à l'entier supérieur, sur un cheval aléatoire. Calculez combien de courses sont nécessaires pour que tous les parieurs aient perdu leur argent.

Exercice 16 — *Bigint*

On veut créer un type d'entier plus grand que le plus grand entier représentable avec un type primitif C.

1. Créez une structure `struct s_Bigint` qui contient 3 éléments : un entier `int s`, un entier `int p` et un tableau de `N` entiers `int t[N]` où `N` est défini avec le mot-clef `#define` à n'importe quelle valeur entière supérieure à 1000. A l'aide du mot-clef `typedef`, on définiera le type `Bigint` comme étant identique au type `struct s_Bigint`

   Cette structure représente un entier avec `s` $\leq$ `N` chiffres contenus dans le tableau `t` dans l'ordre inverse. L'entier est positif si `p` vaut 1 et négatif sinon. Par exemple si $s = 3$, $p = 1$ et $t = \{3, 4, 1\}$ alors le `Bigint` représente le nombre 143.
2. Ecrire une fonction qui prend en entrée un `Bigint` et affiche le nombre correspondant en console.
   - Attention si `s` $= 0$
   - Attention si t se termine avec des 0
   - Attention si l'entier est 0 mais est indiqué négatif.
3. Ecrire une fonction qui prend en entrée un entier `s` $\leq$ `N` renvoie un Bigint aléatoire avec `s` chiffres.
4. Ecrire une fonction qui prend en entrée deux `Bigint` et renvoie leur somme.

5. Ecrire une fonction qui prend en entrée deux `Bigint` et renvoie leur soustraction.
6. Ecrire une fonction qui prend en entrée deux `Bigint` et renvoie leur multiplication.
7. Ecrire une fonction qui prend en entrée deux `Bigint` et renvoie le quotient de division euclidienne.
8. Ecrire une fonction qui prend en entrée deux `Bigint` et renvoie le reste de leur division euclidienne.
9. Ecrire une fonction qui prend en entrée un entier et renvoie le `Bigint` correspondant à sa factorielle.

Exercice 17 — *Votre premier jeu d'aventure*

On souhaite représenter un territoire de jeu par une matrice 2d composée de cases de différents types (eau, herbe, forêt), sur laquelle un héros peut effectuer des actions et sur laquelle un monstre peut se trouver pour attaquer le héros. Ce héros possède un niveau d'expérience (XP), des points de vie (PV) et un niveau d'attaque (ATK). Le monstre possède des PV et un niveau d'ATK. Le héros démarre avec 0 XP, 1 ATK et 20 PV. Les monstres démarrent avec entre 1 et 5 PV et un ATK entre 1 et 3.

1. Proposer des structures de données pour représenter le jeu.
2. Réaliser une fonction qui affiche l'état du jeu, on prendra soit de représenter le jeu le plus succinctement possible.
3. Réaliser une fonction qui déplace le héros d'une case vers une autre, en utilisant l'entrée standard pour déterminer la direction à suivre. L'entrée standard peut indiquer que le héros ne se déplace pas. Le héros ne peut pas se déplacer sur l'eau. S'il essai, rien ne se passe et la fonction renvoie 1. Sinon elle renvoie 0.
4. Réaliser une fonction qui déplace un monstre d'une case aléatoirement : une chance sur 5 de ne pas bouger et une chance sur 5 d'aller dans une des 4 directions possibles. Un monstre ne peut pas se déplacer sur l'eau. S'il essai, rien ne se passe et la fonction renvoie 1. Sinon elle renvoie 0.
5. Réaliser une fonction qui fait d points de dégâts au héros ou à un monstre. Si ses PV tombent en dessous de 0, la fonction renvoie alors 1, sinon elle renvoie 0.
6. Réaliser une fonction qui fait gagner 1 XP au héros ; si le nombre d'XP du héros dépasse son ATK, elle monte de 1, les XP retombent à 0 et les PV reviennent à 20.
7. Réaliser une fonction initialisant le jeu à partir de l'entrée standard. Celle-ci où sont situés les types de terrains de la carte et l'endroit où sont situés les monstres, leurs PV et leurs ATK, et l'endroit où démarre le héros.
8. Réaliser une fonction qui effectue une itération du jeu :
   - Les monstres se déplacent.
   - Chaque monstre à l'emplacement du héros lui inflige autant de dégât que son ATK. Si le héros n'a plus de PV, il disparaît et la partie s'arrête.
   - Le héros se déplace.
   - Chaque monstre à l'emplacement du héros de voit infliger autant de dégât que l'ATK du héros. Si le monstre n'a plus de PV, il disparaît et le héros gagne 1 XP.
   - Si la carte n'a plus de monstre, le héros gagne la partie.