TP QCM

Programmation C++ Linux

Lycée Diderot, Paris

BTS SNIR 1

TABLE DES MATIERES

TABLE DES ILLUSTRATIONS

INTRODUCTION

Au TP précédent, on s'est intéressé à la création d'un programme soumettant un questionnaire à choix multiple à son utilisateur.

Pour l'heure, le questionnaire est composé d'une seule question et de ses réponses associées. C'est un peu léger. On va donc aborder un type de données permettant le traitement automatisé de plusieurs questions: les tableaux.

ATTENTION

A chaque séance, vous devrez faire une sauvegarde de votre travail sur Github avec Github Desktop. Ne pas le faire vous vaudra un 0.

Ne pas m'appeler pour valider les différents exercices vous vaudra un 0.

FORMAT DECLARATION ET INITIALISATION D'UN TABLEAU

Il vous faut trois choses pour déclarer un tableau:

- Son type
- Son identificateur
- Sa taille

La syntaxe est similaire à celle employée pour déclarer une variable de type simple. Par exemple, pour déclarer un tableau de 15 entiers, on écrira:

```
int tabEntiers[15];
```

Une chose vous est permise à la déclaration: vous pouvez initialiser toutes les valeurs du tableau d'un coup. Le plus simple est de mettre toutes les valeurs à 0, de manière générale. Cela dépend évidemment de ce que contient le tableau. Pour notre exemple, cela donne:

```
int tabEntiers[15]={0};
```

On peut vouloir aussi initialiser avec d'autres valeurs. Il faut donc spécifier chaque valeur pour chaque case du tableau, comme ceci:

```
int tabEntiers[15]={11,34,56,98};
```

Dans l'exemple ci-dessus, on initialise les 4 premières cases du tableau avec les valeurs 11, 34, 56 et 98. Les 11 cases suivantes prennent la valeur 0.

PARCOURS D'UN TABLEAU

L'intérêt d'utiliser un tableau est de regrouper plusieurs variables d'un même type et de pouvoir y accéder de manière plus automatisée.

Par exemple, pour afficher toutes les valeurs d'un tableau, je ne suis pas obligé d'écrire:

```
cout<< tabEntiers[0];
cout<< tabEntiers[1];
...
cout<< tabEntiers[14];
```

Je peux accéder à ces valeurs grâce à une structure itérative, le **for**. Il s'agit d'une boucle qui se répète un nombre de fois fini. Sa syntaxe est :

```
for ([condition de départ];[condition de sortie];[pas])
{
    instructions;
}
```

QUESTIONNEMENT MULTIPLE

On affine le diagramme de séquence du TP précédent afin d'illustrer le comportement que l'on souhaite voir apparaitre ici. A terme, il faut déjà penser qu'il peut y avoir plusieurs questionnaires, même si pour l'instant, il n'y en a qu'un seul.

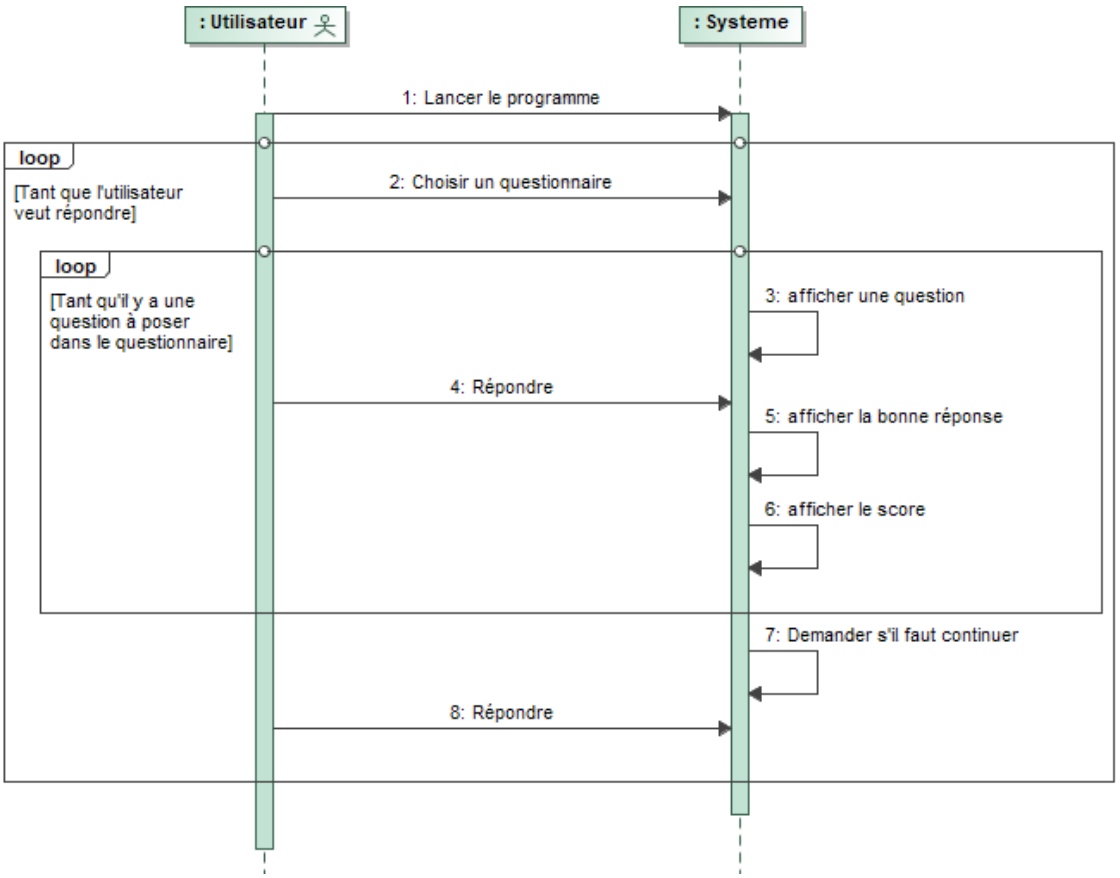

Figure 1: Diagramme de séquence

Votre travail ici va donc consister à gérer un tableau de questions (dès qu'il y a 2 questions, on est bon) avec les réponses associées.

A vous de voir s'il vous faut un ou plusieurs tableaux pour arriver au résultat demandé. Je rappelle, tout est possible, il peut y avoir plusieurs façons de régler le problème. C'est à vous de faire des choix.

MANIPULATIONS DE TABLEAUX

Nous allons laisser de côté pour un temps le QCM et nous occuper à maitriser la notion de tableau de variables.

Nous allons donc faire plusieurs petits exercices à la difficulté croissante.

CHERCHER LA PLUS GRANDE VALEUR D'UN TABLEAU

Vous devez créer un programme permettant de:

- Remplir au clavier par l'utilisateur un tableau de valeurs réelles (un carnet de notes, par exemple)
- Afficher à l'utilisateur la valeur la plus élevée contenue dans le tableau

REMPLIR UN TABLEAU ALEATOIREMENT

Le but est maintenant de remplir un tableau de 9 entiers par des nombres aléatoires.

UTILISATION DES FONCTIONS RAND() ET SRAND()

La fonction **rand()** permet de générer un nombre aléatoire entre 0 et la constante **RAND_MAX**, soit 2147483647. C'est la théorie, en réalité, il s'agit d'une fonction pseudo-aléatoire. Elle donnera la même séquence de nombre à chaque fois que le programme sera lancé. Son algorithme a besoin d'un point de départ (**seed** ou graine) pour générer le nombre aléatoire. En changeant ce point, on peut donc changer le résultat. Si on se débrouille bien, on peut fixer le point de départ en fonction de l'horloge de l'ordinateur: c'est ce que permet la fonction **srand()** (le s est pour **seed**) combinée à la fonction **time()**.

Comment ça marche? Comme ceci:

```
srand(time(NULL));   //initialisation du point de départ
…

m_nombre = rand();   //appel de la fonction rand et
                     //récupération de la valeur aléatoire
                     //dans la variable m_nombre
```

Il n'y a plus qu'à faire le code et remplir le tableau!

Mais avant, il vous faut inclure une librairie de fonctions à votre fichier source afin de pouvoir utiliser ces fonctions. Vous trouverez laquelle en faisant **man srand** dans votre terminal.

EUROMILLIONS

Peut-être avez-vous déjà joué au loto ou à l'euromillions. Il existe pour les joueurs indécis un système permettant de remplir une grille automatiquement et aléatoirement pour le joueur, le système Flash. Nous allons faire notre propre système Flash.

La règle du jeu: il faut cocher 5 numéros (à choisir entre 1 et 50) et 2 étoiles (entre 1 et 11).

Votre programme doit fournir les numéros et les étoiles à jouer à l'utilisateur.

TABLEAUX A PLUSIEURS DIMENSIONS

Jusqu'à présent, nous n'avons utilisé que des tableaux à UNE dimension. Il peut être très intéressant et surtout utile de rajouter des dimensions à tout ça: à l'heure de l'avènement de la 3D dans les salles de cinéma, cela me semble pertinent.

Dans l'absolu, que les tableaux soient de 1, 2, 3 ou n dimensions, ils sont stockés de la même manière dans la RAM: les cases mémoires les unes derrière les autres. Ajouter une dimension à un tableau ne sert qu'au développeur à visualiser les données qu'il doit traiter.

Par exemple, nous serons probablement amenés dans l'année à faire un peu de traitement d'image. Or une image n'est rien d'autre, sur nos écrans qu'un tableau de pixels à 2 dimensions.

SUDOKU

Le but est de fabriquer une grille de Sudoku et de l'afficher. Pour rappel, le Sudoku est un jeu de réflexion basé sur une grille de chiffres. Cette grille se présente ainsi:

8	9	7	2	3	1	5	6	4
2	3	1	5	6	4	8	9	7
5	6	4	8	9	7	2	3	1
7	8	9	1	2	3	4	5	6
1	2	3	4	5	6	7	8	9
4	5	6	7	8	9	1	2	3
9	7	8	3	1	2	6	4	5
3	1	2	6	4	5	9	7	8
6	4	5	9	7	8	3	1	2

On peut la décomposer en 9 tableaux 2 dimensions de 3 par 3. Chaque tableau est rempli par les chiffres de 1 à 9 sans doublons. Le principe du jeu est qu'un chiffre ne peut être répété plusieurs fois sur une même ligne, une même colonne ou un même tableau.

L'objectif: faire une grille basée sur une création en partie aléatoire.