MOOC Init. Prog. C++
Exercices semaine 6

Exercice 18 : Générateur automatique de lettres (fonctions et chaînes de caractères, niveau 1)

Cet exercice correspond à l'exercice n°16 (pages 54 et 217)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de cet exercice est d'écrire un programme nommé `lettre.cc`, qui constituera un générateur automatique (très simpliste) de lettres.

1. Écrivez tout d'abord une fonction `genereLettre` (sans argument, et sans valeur de retour). Cette fonction devra simplement produire la sortie suivante à l'écran : (ne vous occupez pas de la mise en évidence)

   ```
   Bonjour chère Mireille,
   Je vous écris à propos de votre cours.
   Il faudrait que nous nous voyons le 18/12 pour en discuter.
   Donnez-moi vite de vos nouvelles !
   Amicalement, John.
   ```

   Invoquez (appelez) simplement la fonction `genereLettre` depuis la fonction principale *main*, compilez votre programme et assurez vous de son fonctionnement correct.

2. Modifiez maintenant la fonction `genereLettre`, de manière à rendre paramétrables les parties mises en évidence dans le texte précédant. Il vous faudra pour cela passer un certain nombre d'arguments à la fonction, de manière à spécifier :
   - La formulation adaptée pour l'entrée en matière (`chère` pour un destinataire féminin, et `cher` pour un destinataire masculin). Ce choix devra être fait en fonction de la valeur d'un argument booléen (*masculin* ou *feminin*) ou énuméré (*sexe*).
   - Le nom du destinataire, nommé par exemple *destinataire*
   - Le sujet de la lettre (paramètre nommé *sujet*)
   - La date du rendez-vous sous forme de deux paramètres entiers, l'un pour le jour et l'autre pour le mois
   - La formule de politesse (paramètre *politesse*)
   - et le nom de l'auteur (*auteur*).

   La fonction aura donc 7 paramètres.

   Invoquez la fonction au moins deux fois de suite depuis le `main`, en paramétrant les appels de sorte à produire la lettre précédente, et la réponse ci-dessous :

   ```
   Bonjour cher John,
   Je vous écris à propos de votre demande de rendez-vous.
   Il faudrait que nous nous voyons le 16/12 pour en discuter.
   ```

```
Donnez-moi vite de vos nouvelles !
Sincèrement, Mireille.
```

Note : On ne vous demande *pas* de faire saisir les différents arguments au clavier par l'utilisateur, juste d'appeler deux fois la fonction avec les bons arguments dans le `main`.

Exercice 19 : nombres complexes (structures, niveau 1)

Cet exercice correspond à l'exercice n°22 (pages 60 et 225)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Le but de ce programme est d'effectuer les manipulations élémentaires sur les nombres complexes : addition, soustraction, multiplication et division.

Dans le fichier `complexes.cc`, définissez une structure `Complexe` représentant un nombre complexe comme deux `double` (forme cartésienne).

Ensuite, prototypez puis définissez une procédure `affiche` qui prend un nombre complexe en argument et l'affiche.

Dans le `main()`, déclarez et initialisez un nombre complexe. Affichez-le. Compilez et exécutez votre programme pour vérifier que tout fonctionne comme prévu jusqu'ici.

Prototypez puis définissez une fonction `addition` qui prend deux nombres complexes en argument et retourne leur somme.

Testez votre fonction dans le `main()`.

Terminez l'exercice en écrivant puis testant les fonctions `soustraction`, `multiplication` et `division`.

Rappel : la multiplication de `z=(x,y)` par `z'=(x',y')` est le nombre complexe `z*z'=(x*x'-y*y', x*y'+y*x')`.
la division de `z=(x,y)` par `z'=(x',y')` est le nombre complexe `z/z'=((x*x'+y*y')/(x'*x'+y'*y'), (y*x'-x*y')/(x'*x'+y'*y'))`.

Exemple d'exécution

```
(1,0) + (0,1) = (1,1)
(0,1) * (0,1) = (-1,0)
(1,1) * (1,1) = (0,2)
(0,2) / (0,1) = (2,0)
(2,-3) / (1,1) = (-0.5,-2.5)
```

Exercice 20 : Questionnaire QCM (structures + vectors, niveau 2)

Cet exercice correspond à l'exercice n°24 (pages 61 et 228)
de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

On cherche ici à faire un programme d'examen sous forme de questionnaire à choix multiple (QCM) où une question est posée et la réponse est à choisir parmi un ensemble de réponses proposées (une seule bonne réponse possible par question).

Dans un programme `qcm.cc`, définissez une structure `QCM` comprenant 3 champs :

1. un champ `question`, chaîne de caractères, qui contiendra la question à poser
2. un champ `reponses` qui sera un tableau de taille variable de chaînes de caractères contenant les réponses proposées
   (c'est un tableau de taille variable car les questions n'auront pas toutes le même nombre de réponses proposées)
3. un champ entier `solution` (entier positif) qui contient le numéro de la bonne réponse (dans le champ `reponses`).

Prototypez puis définissez une procédure `affiche` qui prend un `QCM` en argument et l'affiche. Par exemple :

```
Combien de dents possède un éléphant adulte ?
    1- 32
    2- de 6 à 10
    3- beaucoup
    4- 24
    5- 2
```

Dans le `main()`, créez et initialisez le `QCM` ci-dessus, puis affichez le. Compilez et vérifiez que tout fonctionne correctement jusqu'ici.

Reprenez la fonction `demander_nombre` (avec 2 arguments, point 4 de l'exercice 11), et créez une fonction `poser_question` qui prend un `QCM` en argument, appelle successivement `affiche` et `demander_nombre` et retourne la réponse de l'utilisateur.

Avant de continuer, testez votre programme (affichage de la question et saisie de la réponse).

On cherche maintenant à faire un examen de plusieurs questions. Définissez le type `Examen` comme un tableau dynamique de `QMC`. Créez un `Examen` dans le main(), puis remplissez-le (dans une fonction `creer_examen` c'est mieux !) avec les questions suivantes (code *partiel*) :

C++11

```
// Question 1
{ "Combien de dents possède un éléphant adulte",
  { "32", "de 6 à 10", "beaucoup", "24", "2" },
  2 // réponse
},
```

```
    // Question 2
    { "Laquelle des instructions suivantes est un prototype de foncti
      { "int f(0);"     ,
        "int f(int 0);" ,
        "int f(int i);" ,
        "int f(i);"     },
      3 // réponse
    },

    // Question 3
    { "Qui pose des questions stupides",
      { "le prof. de math",
        "mon copain/ma copine",
        "le prof. de physique",
        "moi",
        "le prof. d'info",
        "personne, il n'y a pas de question stupide",
        "les sondages" } ,
      6 // réponse
    }
```

Pour terminer :

1. Posez les questions une à une ;
2. Comptez le nombre de bonnes réponses
3. Donnez le score à la fin

Exemple d'exécution

```
Combien de dents possède un éléphant adulte ?
    1- 32
    2- de 6 à 10
    3- beaucoup
    4- 24
    5- 2
Entrez un nombre entier compris entre 1 et 5 : 2
Laquelle des instructions suivantes est un prototype de fonction ?
    1- int f(0);
    2- int f(int 0);
    3- int f(int i);
    4- int f(i);
Entrez un nombre entier compris entre 1 et 4 : 3
Qui pose des questions stupides ?
    1- le prof. de math
    2- mon copain/ma copine
    3- le prof. de physique
    4- moi
    5- le prof. d'info
    6- personne, il n'y a pas de question stupide
```

```
    7- les sondages
Entrez un nombre entier compris entre 1 et 7 : 5
Vous avez trouvé 2 bonnes réponses sur 3.
```
