TP7 POO - C++

gilles.scarella@univ-cotedazur.fr, simon.girel@univ-cotedazur.fr

1 Surcharge de fonctions

Dans cet exercice, dans le fichier *myMin.cpp*, on va écrire la fonction ***myMin*** sous au moins trois versions différentes

- La première version de la fonction ***myMin*** doit prendre 3 *int* comme arguments d'entrée et renvoyer un *int* correspondant au plus petit argument.
- La deuxième version de la fonction ***myMin*** doit prendre 3 *double* comme arguments d'entrée et renvoyer un *double* correspondant au plus petit argument.
- La troisième version de la fonction ***myMin*** ne prend qu'un seul argument d'entrée: un tableau de 3 *int* et doit renvoyer un *int* correspondant au plus petit argument.
- Toujours dans le fichier *myMin.cpp*, écrire une fonction *main* qui teste les trois fonctions. Enfin, définir un tableau de 3 *double* et tester ***myMin*** pour ce tableau. Que constatez-vous? Proposer un correctif. Inversement, pour quelle version de *myMin* et quel exemple aurait pu-t-on se passer tout en conservant un résultat identique (du point de vue mathématique)?

2 Calcul des moments

En statistiques, le moment d'ordre k d'un vecteur x de $\mathbb{R}^n$ est défini de la manière suivante:

$$m_k = \frac{1}{n}\sum_{i=0}^{n-1}(x_i - \bar{x})^k \quad \text{où } \bar{x} \text{ est la moyenne de } x$$

On demande ici d'écrire une fonction C++ calculant la quantité m_k, d'une manière générique quand k, x et n sont quelconques, et de la tester.
Le code de cet exercice sera écrit dans le fichier *moment.cpp*.

Pour cela,

- Ecrire une fonction ***moyenne*** à deux arguments, qui prend comme arguments un pointeur sur *double* x et un entier n et qui renvoie un *double*, qui correspond à la moyenne des éléments du tableau x de taille n.

- Ecrire une fonction ***moment*** à trois arguments, qui prend comme arguments un entier k, un pointeur sur *double* x et un entier n et qui renvoie le moment d'ordre k du tableau x de taille n.
- Dans la fonction *main* de *moment.cpp*, tester les deux exemples suivants:

  Pour $x = (6, -2.1, -0.5)$, on doit trouver $m_2 \simeq 12.2689$
  Pour $x = (1.1, 2.1, -3.1, 6.1, -1.1)$, on doit trouver $m_4 \simeq 195.132$

3 Tableaux et pointeurs sur doubles

Le code de cet exercice sera écrit dans le fichier *exo3.cpp*.

- Ecrire une fonction ***create_array*** qui prend un entier n en paramètre et renvoie un pointeur sur *double*. Cette fonction ne fait rien d'autre qu'allouer la mémoire nécessaire à un tableau de n *double* et renvoie le pointeur correspondant. La tester dans une fonction *main*.
- Ecrire une fonction ***get_kbd_number*** qui ne prend aucun paramètre, qui affiche le message "Entrer une valeur réelle" et qui renvoie un *double* que l'utilisateur aura entré au clavier. La tester dans la fonction *main* précédente.
- Ecrire une fonction ***get_rand_number*** qui ne prend aucun paramètre et qui renvoie un nombre aléatoire de type *double* compris entre 0 et 1. On pourra consulter la page : `http://www.cplusplus.com/reference/cstdlib/rand`

  On évitera une division entière en convertissant le numérateur (ou le dénominateur) d'un entier à un nombre flottant.

  La tester dans la fonction *main* précédente.
- Ecrire une fonction ***init_array_kbd*** qui prend un tableau de *double* en paramètre ainsi qu'un entier n. Chaque élément de ce tableau sera initialisé par l'utilisateur qui entrera les valeurs au cours de l'exécution du programme. On utilisera la fonction *get_kbd_number*.
- Ecrire une fonction ***affiche*** qui prend un tableau de *double* en paramètre ainsi qu'un entier n et qui ne renvoie rien. Cette fonction affiche à l'écran les éléments du tableau.
- Tester ***init_array_kbd*** dans la fonction *main* précédente. On attribuera une valeur à n et le tableau de *double* en paramètre doit être déclaré dans *main* et correctement alloué en utilisant par exemple *create_array*. On affichera le contenu du tableau en utilisant *affiche*.

- Ecrire une fonction ***init_ array_ rand*** qui prend un tableau de *double* en paramètre ainsi qu'un entier n. Chaque élément de ce tableau sera initialisé avec un nombre aléatoire. On utilisera les fonctions *get_ rand_ number* et *affiche*.

On se rend bien compte que le code écrit précédemment est redondant et pourrait être amélioré en le rendant plus générique.

- Pour ce faire, écrire une fonction ***apply*** qui prend 3 paramètres: un tableau de *double*, un entier n, ainsi qu'un pointeur sur une fonction qui ne prend rien en paramètre et qui renvoie un *double*. On appellera alors la fonction pointée pour chaque élément du tableau. Le paramètre n représente quant à lui le nombre d'éléments du tableau.

- Ecrire une fonction *main* qui fait appel aux différentes briques mises en place : *get_ kbd_ number*, *get_ rand_ number* et *init_ array*.

4 Chaînes de caractères avec *string*

Le code de cet exercice sera écrit dans le fichier *exo4.cpp*.

Le but de cet exercice est la manipulation de chaînes de caractères.

- Dans la fonction *main*, faites afficher le message "Entrer une chaîne de caractères". A l'exécution du programme, l'utilisateur devra entrer une chaîne de caractères s (supposée pas trop longue et sans blancs). Dans le code, la chaîne de caractères sera lue par *cin* et affectée à une variable de type *string*

- Dans le *main*, faites ensuite afficher la chaîne de caractères entrée, puis le nombre de caractères de la chaîne entrée.

  Indication: toute variable de type string possède la méthode *.size()* renvoyant son nombre de caractères.

- Faites ensuite afficher la chaîne s doublée (on utilisera l'opérateur d'addition classique)

FACULTATIF

- Mêmes questions qu'avant en remplaçant le type *string* par un tableau de 20 *char*. On pourra utiliser les fonctions *strlen* et *strcat* (voir aussi sur `https://www.cplusplus.com/`)