MM031 - TP2

Maxime Chupin : chupin@ann.jussieu.fr

20 Janvier 2016

1 Fichiers .hpp et .cpp et la compilation

Dans cet exercice, nous allons mettre en place une architecture type de programme en C++. Les bonnes habitudes en terme d'écriture de programme sont importantes. Nous allons commencé par créer trois fichiers : `main.cpp`, `add.hpp`, et `add.cpp` :

→ le fichier `add.cpp` va contenir une fonction qui fait l'addition de deux nombres entiers,

→ le fichier `add.hpp` va contenir la déclaration de la fonction définie dans `add.cpp` (fichier d'entête ou header file en anglais),

→ le fichier `main.cpp` va contenir le programme final qui va être exécuté.

Exercice 1

1. Créez un fichier `add.cpp` qui contient la définition d'une fonction qui renvoie la somme de deux nombres entiers, appelez-la par exemple `add`.
2. Créez un fichier `add.hpp` qui contient la déclaration de la fonction `add`.

   Remarque : Il est d'usage d'ajouter au début du fichier `add.hpp` les instructions pour le pré-compilateur : `#ifndef add_hpp` et `#define add_hpp`. Et d'ajouter à la fin de ce même fichier, l'instruction `#endif`. Ceci permet au pré-compilateur de ne pas inclure plusieurs fois ce fichier d'entête dans un code. Le fichier ressemble donc à ceci

```
// Fichier add.hpp d'interface du fichier add.cpp
#ifndef add_hpp
#define add_hpp

...

#endif
```

3. Créez un fichier `main.cpp` qui demande à l'utilisateur de taper deux nombres dans le terminal et puis qui affiche la somme de ces deux nombres dans le terminal en utilisant la fonction du fichier `add.cpp`.
4. Compilez tous les *<fichiers>*`.cpp` par la commande `g++ main.cpp add.cpp -o main1` et testez votre programme (tapez `./main1`)
5. Compilez d'abord le fichier `add.cpp` par la commande `g++ -c add.cpp` et puis le fichier `main.cpp` par la commande `g++ -c main.cpp`. Finalement, créez un exécutable par la commande `g++ main.o add.o -o main2`. Testez votre programme (tapez `./main2`).

2 Algorithmes de base sur les tableaux

Le but de cette partie est d'écrire un programme principal qui

→ permet à l'utilisateur de rentrer une taille de tableau,
→ construit un certain tableau (voir l'explication ci dessous),
→ indique son élément maximum,
→ son élément minimum,
→ et le trie.

Vous écrirez toutes les procédures et fonctions [1] nécessaires dans un fichier `functab.cpp`, leur déclaration dans un fichier d'entête `functab.hpp`. Vous les testerez (au fur et à mesure) dans un fichier `maintab.cpp`. Vous ferez une compilation de ces fichiers comme expliqué dans la dernière partie de l'exercice 1, question 5.

Exercice 2 : boite à outils

1. Écrire une procédure qui prend en entrée un tableau t et un entier n, qui alloue au tableau t la taille n, et qui remplie le tableau de la manière suivante : $t[i]$ est égal à $i + 2$ quand i est paire et est égal à i^2 quand i est impaire. (vous pouvez utiliser $i\%2$ pour avoir la parité de i).
2. Écrire une fonction qui prend en entrée un tableau et sa taille, et qui renvoie l'élément maximum du tableau et son indice.
3. De même écrire une fonction renvoyant l'élément minimum du tableau.
4. Écrire une procédure qui prend en entrée un tableau t, et deux indices i et j , et qui échange les deux éléments du tableau correspondant à ces indices.

Exercice 3 : tri par sélection

On s'intéresse ici au problème de tri d'un tableau de N nombres, c'est-à-dire ranger dans l'ordre croissant une suite finie d'entiers initialement placés dans le désordre. Il existe de nombreux algorithmes de tri. Au cours de cette séance, nous allons implémenter le *tri par sélection*.

1. On commence par chercher l'élément maximum du tableau.
2. On intervertie cet élément avec le dernier élément du tableau.
3. On réitère sur le tableau composé des N-1 premiers éléments.
4. On s'arrête lorsque le tableau considéré n'a plus qu'un élément.

Exemple :

89	103	1	10	5	42	→	89	42	1	10	5	103
						→	89	42	1	10	5	103
						→	5	42	1	10	89	103
						→	5	42	1	10	89	103
						→	5	10	1	42	89	103
						→	5	10	1	42	89	103
						→	5	1	10	42	89	103
						→	5	1	10	42	89	103
						→	1	5	10	42	89	103

5. Écrire une procédure qui prend en entrée un tableau sous forme de pointeur sur des entiers et sa taille et qui le trie suivant cet algorithme (en utilisant les procédures ou fonctions créées précédemment).

1. La différence entre *procédure* et *fonction* est ténue. Usuellement on parle de *procédure* lors que la fonction C++ est une routine, c'est-à-dire qu'elle ne retourne pas de valeur (type de sortie `void`). Si par contre elle retourne une valeur alors on parle de fonction.

3 Triangle de Pascal

Exercice 4

1. Écrire deux procédures associant $n!$ à un entier n, la première étant séquentielle et la seconde étant récursive.
2. Écrire une procédure renvoyant la combinaison $\binom{n}{p} = \frac{n!}{p!(n-p)!}$.
3. Écrire deux procédures affichant le triangle de Pascal, l'une utilisant les combinaisons, l'autre la formule récursive $A_{i,j} = A_{i-1,j-1} + A_{i-1,j}$ où i est l'indice de ligne du triangle et j celui de colonne.