TP 5 : Fonctions et tableaux

WIMS.
20 minutes de contrôle sur WIMS.

Exercice 1.

(1) Télécharger l'archive Semaine5.zip, et l'extraire dans votre répertoire Info111.

(2) Configurer Code : :Blocks pour compiler le C++ selon le standard ISO C++ de 2011 :

```
Settings -> Compiler -> Global compiler settings -> Compiler Flags :
Have g++ follow the C++11 ISO C++ language standard
```

(3) Vérifier que le fichier fourni min−max.cpp compile.

(4) Compléter la documentation et le code dans le fichier fourni min−max.cpp en cherchant toutes les indications « À faire ». Vérifier que tous les tests passent.

(5) Même chose pour le fichier fourni tableaux.cpp.

(6) ♣ Transformer les tests manuels en tests automatiques (en utilisant ASSERT).

Exercice 2.

(1) Ouvrir le fichier fibonacci.cpp et l'enregistrer sous le nom : fibonacci−tableau.cpp (cela vous permet de garder la version fournie fibonacci.cpp dont vous aurez besoin ensuite, tout en pouvant modifier le fichier fibonacci−tableau.cpp. Variante ♣ : le faire avec la commande `cp`).

(2) Modifier les premières lignes du fichier fibonacci−tableau.cpp pour ajouter l'inclusion de la bibliothèque `vector` :

```
#include <iostream>
#include <vector>
using namespace std;
```

(3) Implanter la version "tableau" de la fonction `fibonacci` vue en TD.

(4) Vérifier (compilation ET exécution) que votre fonction renvoie bien la valeur attendue, et en particulier que les tests automatiques passent.

(5) De même, à partir du fichier fibonacci.cpp créer de nouveaux fichiers fibonacci−variables.cpp et fibonacci−recursif.cpp, et y implanter les deux autres fonctions fibonacci vues en TD. À chaque fois, vérifier avec les tests automatiques que l'on obtient les résultats attendus.

Exercice ♣ 3.

(1) Implanter une fonction `bool contient(vector<int> tab, int x)` retournant vrai si le tableau `tab` contient l'élément `x` et faux sinon.

(2) Implanter une nouvelle fonction
`vector<int> intersection(vector<int> tab1, vector<int> tab2>)`
retournant un vecteur des éléments communs de `tab1` et `tab2`.

Exercice ♣ 4 (Code morse)**.**
Rappel : les chaînes de caractères se manipulent comme les tableaux.

(1) Consulter le code de `morse.cpp`.

(2) La fonction `string morseCaractere(char a)` utilise une instruction `switch` que nous n'avons pas encore croisé. À quoi aurait ressemblé cette fonction si elle avait été écrite avec des if ?

(3) Ajouter une fonction `string morseChaine(string a)`, avec documentation et tests.

(4) Que fait la fonction `main` ?

(5) Renommer chacune des deux fonctions `morseCaractere` et `morseChaine` en `morse`. Que constate-t-on ?

Exercice ♣ 5 (Python, Euler !)**.**
Pour des explications, voir les deux derniers exercices de la feuille de TP 2.