TP1: A better C
C++

Objectifs pédagogiques :
- Savoir éditer un programme C++ avec un éditeur dédié, le compiler avec CMake ;
- Utiliser les flux d'entrée/sortie : flux standard (E1) et flux de fichier (E1, E3) ;
- Concevoir et utiliser des patrons (templates) (E2) ;
- Savoir manipuler des collections de la STL : vecteurs (E1, E3, E4, E5) et map (E3) ;

Téléchargez le code fourni depuis la page suivante : `https://mquinson.frama.io/ens-cpp/`
Pour ouvrir le projet dans QtCreator :
— extraire le projet à partir de l'archive fournie ;
— File → Open File or Project... → Go to your project folder → select the file "CMakeLists.txt" → Open → Click on "Configure Project"
Pour ouvrir le projet dans VSCode (avec l'extension CMake Tools) :
— extraire le projet à partir de l'archive fournie ;
— ouvrir le projet dans votre workspace : File → Add Folder to Workspace ... ;
— l'extension demande si elle peut configurer le projet : accepter ;
— si l'extension demande un kit, sélectionner `Non specifié`

★ **Exercice 1: Flots et fichiers** Le tableau suivant contient les notes (entre 0 et 5) d'une classe d'élèves.

```
std::vector<int> notes = {
  3, 1, 3, 1, 2, 2, 5, 5, 5, 2, 3, 2, 2, 1, 1, 1, 2,
  0, 5, 2, 4, 3, 5, 5, 1, 5, 2, 5, 1, 0, 2, 2, 1, 5};
```

▷ **Question 1:** Complétez le programme `ex1-histo.cpp` pour afficher la répartition des notes de la façon suivante pour indiquer que 2 élèves ont 0, 8 élèves ont 1, 10 élèves ont 2, etc. Pour bien aligner les colonnes, vous aurez besoin des manipulateurs IO tels que `std::setw`.

```
 0  1  2  3  4  5
 2  8 10  4  1  9
```

▷ **Question 2:** Modifier votre programme pour faire un histogramme sous forme de bandes horizontales, comme suit :

```
0 : **
1 : ********
2 : **********
3 : ****
4 : *
5 : *********
```

▷ **Question 3:** Modifier votre programme pour lire la liste des notes depuis le fichier fourni `notes.txt`. Attention, les notes de ce fichier sont étalées entre 0 et 20. Notez aussi que par défaut les executables sont générés dans un dossier à coté du dossier contenant les fichiers sources du projet, pensez à cela en écrivant le chemin vers le fichier à ouvrir.

★ **Exercice 2: Patrons** (templates).

Après avoir créé le fichier `ex2-patrons.cpp`, vous aurez besoin de décommenter la ligne `#add_executable(ex2-patrons e` dans le `CMakeLists.txt` pour que le fichier soit compilé.

▷ **Question 1:** Écrivez un patron de fonctions permettant de calculer le carré d'une valeur d'un type quelconque. La réponse est assez semblable au patron de fonction `min` du document associé au cours.

▷ **Question 2:** Écrivez un patron de fonctions permettant de retrouver le plus petit des éléments d'un vecteur, quel que soit le type des éléments contenus dans le vecteur.

★ **Exercice 3: Stylomètre** *(d'après Ali Malik à Stanford)*

On souhaite maintenant réaliser un outil très simple permettant de mesurer la similarité entre deux textes donnés. Il pourra être utilisé pour déterminer l'auteur probable d'un texte anonyme à partir d'un corpus de textes connus. Pour cela, nous allons compter les occurrences d'un certain nombre de mots dans chaque texte pour établir un vecteur de caractéristiques. Par exemple, si on choisi les mots `{"I", "there", "the"}` pour caractériser les textes, on trouve les résultats suivants car le mot "I" n'apparaît qu'une seule fois dans le premier texte tandis qu'il apparaît 4 fois dans le second.

Vecteur caractéristique {1; 0; 0}

```
Deep into that darkness peering, long I stood
there, wondering, fearing, doubting, dreaming
dreams no mortal ever dared to dream before.

                                -- Edgar Alan Poe
```

Vecteur caractéristique {4; 1; 1}

```
I first met Dean not long after my wife and
I split up. I had just gotten over a serious
illness that I won't bother to talk about,
except that it had something to do with the
miserably weary split-up and my feeling that
everything there was dead.   -- Jack Kerouac
```

Aussi simple qu'elle soit, cette façon de caractériser les textes peut donner d'assez bons résultats, si le vecteur de caractéristiques est bien choisi. On peut ensuite calculer la similarité entre les textes en mesurant l'angle formé entre les vecteurs grâce à la formule suivante : $cos\theta = \frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\times\|\vec{v}\|}$ On rappelle la norme se calcule simplement du produit scalaire avec $\|\vec{u}\| = \sqrt{\vec{u}\cdot\vec{u}}$

▷ **Question 1:** Complétez le code fourni pour cet exercice afin d'appliquer cette méthode, et utilisez votre programme pour déterminer l'auteur du texte `textes/unknown.txt` fourni. Vérifiez votre réponse en cherchant une phrase de ce texte sur internet.

▷ **Question 2:** (optionnelle) Pouvez-vous trouver un meilleur vecteur de mots pour caractériser les textes ? Et si les textes sont en français ?

La première étape est probablement de dénombrer les occurences de chaque mots du texte, sans liste de mots prédéterminées (en utilisant une `std::unordered_map`), puis on analysera à la main les tableaux d'occurences obtenus sur plusieurs textes afin de sélectionner les mots candidats. Différents textes de la litérature sont disponibles sur `https://fr.wikisource.org`.

Dans un second temps, on évaluera différents vecteurs de mots candidats avec la méthode établie à la question précédente pour sélectionner les meilleurs. Envoyez vos meilleurs vecteurs de mots à l'équipe enseignante.

★ **Exercice 4: Vecteurs de taille dynamique** (conseillé mais optionnel).

Ajoutez une ligne dans le fichier `CMakeLists.txt` pour créer un nouvel exécutable pour cette question.

La suite de Golomb (`https://fr.wikipedia.org/wiki/Suite_de_Golomb`) est une suite d'entiers non décroissante où $a(n)$ indique le nombre de fois où n apparaît dans la suite, avec $a(1) = 1$.

Puisque $a(1) = 1$, nous savons que 1 n'apparaît qu'une seule fois dans la suite, ce qui impose à $a(2)$ d'avoir une autre valeur. Comme $a(2)$ est non-décroissante, cette autre valeur doit être plus grande que 1. Puisque $a(2) > 1$, le chiffre 2 apparaît plus d'une fois dans la suite. Donc, $a(2) = 2$. Puisque $a(2) = 2$, le chiffre 2 apparaît deux fois dans la suite, ce qui nous donne la valeur de $a(3)$. Et ainsi de suite pour les valeurs suivantes de la suite.

▷ **Question 1:** Écrivez une fonction calculant le Nième élément de la suite. Vous aurez probablement besoin d'un vecteur d'entiers. On souhaite ne pas avoir à recalculer deux fois la même valeur dans des appels successifs à la fonction.

La méthode std::vector::resize(N) est utile pour agrandir un vecteur et s'assurer qu'il compte au moins N cases : après cet appel on peut utiliser `vect[i]` pour toutes les valeurs de i inférieures à N, alors que `vect[i]` mène à une erreur si on accède à une case du tableau n'existant pas.

★ **Exercice 5: Parcours de vecteurs** (optionnel) On dispose de relevés topologiques sous forme de vecteurs d'entiers. Ils correspondent à l'altitude mesurée tous les kilomètres.

Par exemple, les valeurs `0 0 1 1 2 2 1 1 0 1 2 2 1 1 0` représentent les deux îles suivantes :

```
    /***\       /*\          <-- 2 mètres au dessus du niveau de la mer
  /**********\ /*******\     <-- 1 mètre au dessus du niveau de la mer
**************************   <-- niveau de la mer
| | | | | | | | | | | | | |
0 0 1 1 2 2 1 1 0 1 2 1 1 0   <-- Les données (altitude en chaque point)
```

▷ **Question 1:** Écrivez une fonction prenant un tel vecteur d'entiers en paramètres et calculant le nombre de niveaux d'île distincts. L'exemple précédent est composé de quatre niveaux d'île :

- 1 1 2 2 1 1 (l'île sur la gauche)
- 2 2 (la montagne au sommet de l'île à gauche)
- 1 2 1 1 (l'île de droite)
- 2 (la montagne sur l'île à droite)

Cela peut également être visualisé ainsi : `0 0(1 1(2 2)1 1)0(1(2)1 1)0`

▷ **Question 2:** Écrivez une fonction retournant le nombre de pics présents dans le relevé, c'est-à-dire le nombre de niveaux d'île ne contenant pas de niveaux plus élevé. L'exemple précédent compte deux pics.