Sup Galilée
Mathématiques Appliquées et Calcul Scientifique
Deuxième année
Année universitaire 2018–2019

Langage C++(G4SCCP)
Cours : X. Juvigny
TPs : S. Landier
sam.landier@onera.fr

Travaux pratiques N°2

Les bases du langage C++

https://github.com/Macs1718/Promotion_2020
https://en.cppreference.com/w/
https://stackoverflow.com/

Exercice 6 (autres containers STL : de type « set »)

Créer un fichier exo6.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

Container STL : std::set

a) Définir une fonction fill_set :
- qui prend en paramètre un std::set<int> et un entier (taille souhaitée)
- qui insère successivement N-1 (deux fois) , N-2 (deux fois) ,…0 (deux fois)

b) Exécuter fill_set et imprimer ensuite le contenu de s en le parcourant à l'aide d'une boucle C++11. Qu'observe-t-on ? quels sont donc les deux propriétés principales du container std::set ?

c) Définir une fonction check :
- Qui prend en paramètre un std::set S et un entier K (valeur à vérifier)
- qui retourne vrai si K appartient à S. **Attention : ne pas utiliser une boucle ! trouver via l'interface de std::set la méthode appropriée.**

Container STL : std::unordered_set

d) refaire a) et b) avec ce type, nommer la fonction fill_uset. Quelle(s) est (ou sont les) différence(s) ?

Exercice 7 (autres containers STL : de type « map »)

Créer un fichier exo7.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)

a) définir un alias pour le type std::map<std ::string, int> (par exemple name_to_age)
b) Définir une fonction fill_map1 :
    - qui prend en paramètre un std::map<std::string, int> et un entier N,
    - qui insère successivement, avec l'opérateur [] :
        - (« toto_K », K)
        - (« toto_K », K*K)

    où K varie de N-1 à 0.

    Note : pour former la chaine de caractère « toto_K », utiliser un objet de type std ::ostringstream.

c) Définir une fonction fill_map2 qui fait l'insertion avec la méthode insert de map. ***Note : il faudra employer la fonction std::make_pair.***
d) Créer 2 maps m1 et m2, les remplir respectivement avec les fonctions fill_map1 et fill_map2 et afficher leur contenu à l'aide de boucles C++11. Qu'observe-t-on ? quels sont les propriétés des maps et la différence entre les 2 méthodes de remplissage ?

e) Définir la fonction :

    ostream& operator « (ostream& out, const name_to_age & m)

qui affichera les informations contenus dans les map m1 et m2 avec les instructions :

```
std::cout << m1 << std::endl ;

std::cout << m2 << std::endl ;
```

Exercice 8 (algorithme de comptage des insertions échouées)

Créer un fichier exo8.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream)
Ecrire une fonction qui prend en paramètre un entier N. La fonction doit créer successivement N nombre aléatoires (compris entre 0 et N-1) et les insérer dans un container approprié (interne à la fonction). Il faut alors comptabiliser toutes les insertions qui ont échoué et retourner le nombre.
Note : trouver la méthode appropriée dans l'interface du container.

Le programme doit avoir la sortie suivante :
Entrez un nombre entier positif :
100
Il y a eu 3 insertions échouées.

Note : on utilisera la fonction std::rand et l'operateur % pour générer les valeurs aléatoires et les limiter à l'intervalle [0,N-1].

Exercice 9 (algorithme de tri)

Créer un fichier exo9.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream). Inclure également le header chrono.h téléchargeable ici : https://github.com/Macs1718/Promotion_2020/tree/master/TPs/src

a) Ecrire une fonction de tri *mysort* qui prend en paramètre un tableau et sa taille et réordonne de façon croissante les valeurs du tableau.
b) Initialiser un tableau avec init, appeler la fonction mysort et imprimer les valeurs du tableau.
c) Trier le tableau à l'aide cette fois de l'algorithme std::sort (#include <algorithm>)
d) Inclure le fichier chrono.h et comparer les temps d'exécution des deux algorithmes pour N=10^2, 10^3, 10^4. Qu'observez-vous ?

Note : instancier un chrono, utiliser les méthodes start() et elapsed() pour respectivement lancer le chrono et l'arrêter.

Exercice 10 (opérations booléennes sur container)

Créer un fichier exo10.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream).

Dans la fonction main, effectuez les opérations suivantes :

a) Création et remplissage de 2 vecteurs (std::vector) da taille 100 : v1[i]=100-1-i, v2[i]=2*i
b) Créer un vecteur v3 qui contient l'intersection de v1 et v2 **(utiliser std::set_intersection**)
c) Créer un vecteur v4 qui contient la différence de v1 et v2 (**utiliser std::set_ difference**)

Note :

- **vérifier dans la doc en ligne la condition que doit vérifier v1 et v2 pour être utilisée avec ces fonctions.**
- **Le container de sortie (v3 et v4) est passé en argument via un itérateur.**
- **Il y a mieux à faire que de dimensionner le container de sortie avant appel aux fonctions. En effet, on ne connait pas a priori par exemple le nombre d'éléments communs..cf. std ::back_inserter**

Exercice supplémentaire (algorithme de connectivité)

Créer un fichier exoSupp.*cpp*, y définir une fonction main et inclure le header pour les entrées/sorties C++ (iostream).

Dans cette exercice, il s'agit d'écrire un programme qui lit une séquence de pairs d'entier données en entrée par l'utilisateur (std ::cin >> p >> q) et qui « connecte p à q ». Le programme affiche, après chaque insertion, la chaîne de tous les entiers connectés.
L'utilisateur définit au départ la valeur max N pour les entiers et le programme doit sortir dès lors que l'utilisateur donne la valeur -1 pour p ou q.

Exemple d'interface du programme :

Entrer l'entier max : 10
Donner une paire: 1 2
1 2
Donner une paire: 3 4
3 4
Donner une paire: 1 4
1 2 3 4
Donner une paire: 4 5
1 2 3 4 5
Donner une paire: 7 8
7 8
Donner une paire: 5 8
1 2 3 4 5 7 8
Donner une paire: -1 5
Fin du programme

Note : on maintiendra en interne un tableau dynamique (par ex. un std ::vector) nommé *link* de taille N, qui assure le lien avec la propriété suivante :
p et q sont connectés si et seulement si link[p] == link[q]