

1. Conteneurs & Itérateurs

1. Le conteneur `vector`

1. Question 1

2. Question 2

3. Question 3

Conteneurs & Itérateurs

Au cours de ce TME nous allons passer en revue trois principaux types de conteneurs de la STL, `vector<>`, `list<>` et `map<>`, ainsi que leurs itérateurs. Nous verrons aussi leurs relations avec les fonction de tri.

Pour pouvoir travailler, nous allons utiliser les mots d'un petit texte fourni dans le fichier `GPL_2_text.h`. Le texte vous est fourni sous forme d'un tableau de `char*`. La fin du tableau est indiquée par un pointeur `NULL`.

```
const char* GPL_2_text[] = {
    "GNU", "GENERAL", "PUBLIC", "LICENSE",
    // ...
    NULL
};
```

Le conteneur `vector`

Question 1

Écrire une fonction `vectorBench1()` effectuant les tâches suivantes:

- Charger dans un vecteur de `string` le texte en insérant les nouveaux éléments à la fin.
- Afficher le nombre d'éléments du vecteur.
- Trier les éléments du vecteur.
- Afficher tous les éléments du vecteur. On les affichera sur une seule ligne (ce sera très long).

Compiler et exécuter ce programme. Mesurer sont temps d'exécution grâce à la commande `time` (si votre programme s'appelle `containers`).

```
> time ./containers
```

Remarque: Le temps peut varier légèrement d'une exécution à l'autre en fonction de la charge de la machine. Lancez votre programme plusieurs fois pour avoir un ordre de grandeur.

Question 2

Écrire une fonction `vectorBench2()` idendique à la précédente, mais qui, *au lieu d'insérer les éléments en fin de conteneur*, les insère en tête. Sachant que `vector<>` n'a pas de `push_front()`, comment peut-on faire (simplement).

Mesurer le temps. Conclusion?

Question 3

Écrire une fonction `vectorBench3()`, qui effectue les mêmes traitements que `vectorBench1()` à ceci près que le tri, au lieu d'être effectué une seule fois en fin de fonction sera fait après l'insertion de chaque élément.

Mesurer le temps. Conclusion?