ensiie
école nationale supérieure d'informatique pour l'industrie et l'entreprise

TP STL

(noté)

Prérequis

- Référence C++ (& STL) : http://www.cplusplus.com/reference/
- Référence STL : http://www.sgi.com/tech/stl/table_of_contents.html
- Le proxy http à mettre en place (si besoin uniquement) dans votre browser pour accéder à ces sites est 172.30.140.22 : 3128

1. Etude de code

Notez les typedefs au début du main qui permettent de définir la nature des éléments des conteneurs (`Element = double`), le conteneur utilisé (`Container = vector<Element>`), ainsi que la nature de l'output iterator utilisé pour afficher le contenu d'une étendue d'itérateurs sur la console (`Printer = ostream_iterator<Element>`).

Etudiez le patron de fonction

```
template <class ForwardIterator>
bool mysteryCheck(const ForwardIterator & first, const ForwardIterator & last)
{
	…
}
```

définit au début de main.c.

Que fait ce patron de fonction ?

2. Génération de valeurs aléatoires bornées [min … max]

2.1. Créez une classe Foncteur `BoundedRandomGenerator<T>` de type générateur, dont l'opérateur d'appel de fonction (`T operator()(void)`) permet de générer des nombres aléatoires de type `T` compris entre deux bornes (min et max). On pourra utiliser : (voir man 3 random)

- `void srandom(unsigned seed)` pour initialiser le générateur de nombres aléatoires lors de la construction de l'objet. On pourra avantageusement utiliser `time(NULL)` comme seed.
- `long random()` pour générer un nombre aléatoire (plus aléatoire que la fonction `rand()`) compris entre 0 et RAND_MAX. Puis renormalisez ce nombre entre min et max.

2.2. Dans le programme principal (main.cpp), utilisez un algorithme standard avec ce foncteur pour remplir le `vector<double> v` avec des valeurs aléatoires comprises entre -12.3 et +25.7.

2.3. Affichez le contenu de `v` en utilisant un algorithme standard avec l'`ostream_iterator<double> printer` défini au début du main.

3. Calcul de la valeur moyenne et de l'écart type

Créez une classe Foncteur
`class StatFunctor : public unary_function<T, void>`
dont l'opérateur d'appel de fonction (`void operator()(const T &)`) permet d'additionner des valeurs et des valeurs au carré de type `T` en vue de calculer sur une série de n valeurs $X = \{x_1, x_2, \cdots, x_n\}$

- la moyenne $\bar{x} = E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i$: avec une méthode `double mean() const`
- et l'écart type $\sigma = \sqrt{E(X^2) - E(X)^2}$: avec une méthode `double std() const`

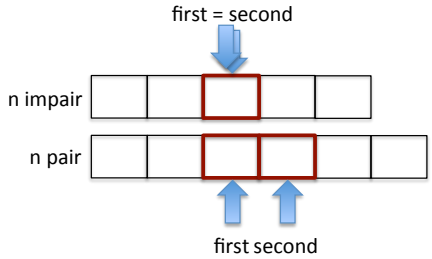
3.1. Appliquez ce foncteur sur l'ensemble des éléments de `v` avec un algorithme standard, puis affichez la moyenne et l'écart type.

4. Recherche de la médiane

Soit E un ensemble de n nombres triés par ordre croissant, la médiane de E est son élément central, ou plus exactement :

$$\begin{cases} \text{si } n \text{ est impair } médiane = \left(\frac{n+1}{2}\right)^{ème} élément \\ \text{si } n \text{ est pair } médiane = \left(\left(\frac{n}{2}\right)^{ème} élément + \left(\frac{n}{2}+1\right)^{ème} élément\right)/2 \end{cases}$$

Nous souhaitons écrire deux algorithmes de recherche de la (ou les) position(s) des éléments constituant la médiane (`median` et `median2`). Pour ce faire, ces algorithmes renverront une paire d'itérateurs indiquant la (ou les) positions où trouver le (ou les) élément(s) constituant la médiane. Dans une étendue d'itérateurs sur des éléments triés on aura alors pour la paire[1] :



Alors que pour une étendue d'itérateurs sur des éléments non triés, first & second peuvent se trouver n'importe où dans l'étendue d'itérateurs.

4.1. Créez une copie triée de `v` dans `vt` en utilisant un algorithme standard. Puis affichez ses éléments.

4.2. Complétez l'algorithme
`pair<RandomAccessIterator, RandomAccessIterator>`
`median(const RandomAccessIterator & first,`
`const RandomAccessIterator & last)` pour trouver une paire d'itérateurs sur les éléments constituants la médiane d'une étendue d'itérateurs [first, last) dont les éléments sont censés être triés (il faudra le vérifier).
Si l'étendue [first, last) n'est pas triée ou si elle est vide on renverra la paire <last, last>.

[1] Attention first & second sont ici les membres de la paire. A ne pas confondre avec [first, last) d'une étendue d'itérateurs.

4.3. Utilisez cet algorithme pour trouver la paire d'itérateurs de la médiane de `vt`, puis affichez le ou les éléments constituant la médiane, et le cas échéant sa valeur. Dans la mesure où l'on n'utilisera pas la paire d'itérateurs résultante pour modifier `vt`, on se contentera de `const_iterator`(s) lors de l'appel de `median`.

4.4. Complétez l'algorithme
```
pair<RandomAccessIterator, RandomAccessIterator>
median2(const RandomAccessIterator & first,
        const RandomAccessIterator & last)
```
pour trouver une paire d'itérateurs sur les éléments constituant la médiane d'une étendue d'itérateurs [first, last) dont les éléments ne sont pas triés. L'idée consistera à réutiliser l'algorithme `median` dans `median2` sur une copie locale triée de [first, last) pour en extraire les itérateurs sur la médiane, puis de rechercher les valeurs de ces itérateurs dans le [first, last) initial non trié.

4.5. Utilisez cet algorithme pour trouver la paire d'itérateurs de la médiane de `v`, puis affichez le ou les éléments constituant la médiane, et le cas échéant sa valeur. Dans la mesure où l'on n'utilisera pas la paire d'itérateurs résultante pour modifier `v`, on se contentera de `const_iterator`(s) lors de l'appel de `median2`.

4.6. Vérifiez la propriété $\frac{|\tilde{x}-\bar{x}|}{\sigma} \leq \sqrt{\frac{3}{5}}$ avec $\tilde{x}$: médiane, $\bar{x}$: moyenne & σ : écart type sur `v` ou `vt`.

5. Soustraction de la moyenne de v

Soustrayez à chaque élément de `v` la moyenne de `v` en utilisant uniquement des algorithmes et/ou des foncteurs standard. Puis affichez le `v` modifié.

6. Extraction des éléments positifs et négatifs de v

Copiez les éléments positifs de `v` dans `vp` et les éléments négatifs de `v` dans `vn` en utilisant uniquement des algorithmes et/ou des foncteurs standard. Puis affichez `vp` et `vn`.

7. Changement de nature de conteneur

7.1. Peut on remplacer `vector` par `deque` dans la définition du type de conteneur utilisé au début de la fonction main ? Développez votre réponse en expliquant pourquoi.

7.2. Peut on remplacer `vector` par `list` dans la définition du type de conteneur utilisé au début de la fonction main ? Développez votre réponse en expliquant pourquoi.

8. Changement de nature de contenu

Quelles sont les opérations nécessaires à une classe X pour qu'elle puisse remplacer double dans la définition du type d'éléments Element ?

Annexes

Types internes

Dans un algorithme (au sens de la STL), c'est à dire dans une fonction dont les arguments sont des itérateurs, il est parfois nécessaire de connaître la nature des types utilisés par ces itérateurs : la nature des valeurs, de la différence entre deux itérateurs, etc.

Pour ce faire on peut utiliser les `iterator_traits` relatifs aux itérateurs utilisés.

Par exemple :

```
Iterator algo(const Iterator & first, const Iterator & last)
{
    typedef typename iterator_traits<Iterator>::value_type value_t;
    typedef typename iterator_traits<Iterator>::difference_type diff_t;
    …
    diff_t nbElts = last - first;
    …
}
```

Si le type d'itérateur utilisé dans le programme utilisant cet `algo` n'est pas un type standard reconnu par les `iterator_traits`, cela provoquera une erreur de compilation.

Configuration particulière d'Eclipse

Il se peut que l'indexer d'Eclipse ne soit pas configuré pour reconnaître les types spécifiques à C++11 qui peuvent être utilisés dans le code source. Il risque donc de souligner en rouge les termes qu'il ne reconnaît pas. Ceci n'a **aucune incidence** sur la compilation (qui est dirigée par le Makefile et utilise le standard C++11 grâce à l'option de compilation « -std=c++11 »), mais peut vous gêner dans l'écriture de votre code.

Pour que le C/C++ Development Tooling (CDT) d'Eclipse reconnaisse les instructions C++11, il faut changer la ligne de commande de l'indexer C++ dans la configuration du CDT : Préférences → C/C++ → Build → Settings → Onglet Discovery, sélectionnez la ligne « CDT GCC Built-in Compiler Settings » et rajoutez « -std=c++11 » à la commande affichée en bas, qui devrait maintenant ressembler à ceci :

${COMMAND} ${FLAGS} -E -P -v -dD "${INPUTS}" -std=c++11