Université de Reims Champagne-Ardenne
UFR Sciences Exactes et Naturelles
Licence Informatique
INFO0402: Programmation Orientée Objet

TD N°8

Exercice 4: Liste et Vecteur

On veut dans cette partie manipuler une liste.

1. Remplir une liste avec des valeurs aléatoires uniformes entre 0 et 255.
2. Trier le vecteur par ordre croissant. On proposera deux solutions.
3. Pourquoi la solution de trier la liste peut être quand même plus rapide ?
4. Enlever les éléments dupliqués.
5. Inverser la liste chainée.

Solution:

1. voir code suivant :

```
const size_t nElts = 100;
std::random_device rd;
std::mt19937 gen(rd());
std::uniform_int_distribution<> dis(0, 255);
auto RandGen = [&gen, &dis]() { return dis(gen); };
list<int> l(nElts); // l'allocation de la place est ici
generate_n(l.begin(), nElts,RandGen);
```

2. voir code suivant :

```
// méthode interne de la classe liste
l.sort();
```

```
vector<int> v;
// passage par le tri d'un tableau
copy(l.begin(), l.end(), v.begin());
sort(v.begin(), v.end());
copy(v.begin(), v.end(), l.begin());
// 35% plus rapide
```

3. lorsque les éléments sont triés avec la méthode interne, les éléments ne sont jamais déplacés ni copiés, seuls les pointeurs sont modifiés.

   Ici, le coût de la copie étant faible, les recopies et tri dans le tableau sont plus performants. Mais, dès que la taille de la structure grandit, la tri direct de la structure est plus avantageux.

4. voir code suivant :

```
l.unique();
```

```
// plus rapide
copy(l.begin(), l.end(), v.begin());
unique(v.begin(), v.end());
copy(v.begin(), v.end(), l.begin());
```

   Mêmes remarques qu'à la question précédente.

5. voir code suivant :

```
// méthode interne (pointeur)
l.reverse();
```

```
// à la main
list<int> l2;
copy(l.rbegin(),l.rend(), back_inserter(l2));
swap(l, l2);
```