Mercredi 23 septembre 2015

Exercice 10. Implémenter
```
Permutation::Permutation(int, const int *);
```

Exercice 11. Définir un opérateur de comparaison `<` entre permutations.

Exercice 12. En prolongeant σ par $\sigma(i) = i$ pour $i > n$, on peut donner un sens au produit de `s1` et `s2`, même si `s1.n != s2.n`. Implémenter cette extension. De même `bool Permutation::next()` pourrait devenir `void Permutation::next()` et faire passer de la dernière permutation de $[1..n]$ à la première de $[1..n+1]$.

Exercice 13. Soit f une fonction continue strictement croissante sur les réels, qui tend vers 0 (resp. 1) quand son argument tend vers $-\infty$ (resp. $+\infty$). Écrire une procédure qui à un tel f et à $x \in]0,1[$ fait correspondre t tel que $f(t) = x$. Décrire une classe qui modélise la notion de générateur de nombres aléatoires. Si f est une loi de probabilité comme ci-dessus, on devrait pouvoir construire un générateur qui suit cette loi. De même, si la loi est un barycentre de masses de Dirac. Pour l'implémentation, on pourra utiliser `rand()`. Voir aussi `srand()` et `RAND_MAX`.