Cégep de Sainte-Foy	**Programme des Techniques de l'informatique** **Programmation de jeux vidéo** Département des Techniques d'intégration multimédia	Automne **2017**

Algorithmes et structures de données II

420-V30-SF	Travail pratique 1	Vecteur d'entiers

Objectifs pédagogiques

Ce travail vise à familiariser l'étudiante ou l'étudiant avec les objectifs suivants :

- Exploiter un langage de programmation structurée (016S)
- Produire des algorithmes (016W)
- Organiser et exploiter des données (0170)

De même, il permet à l'étudiante ou à l'étudiant d'appliquer les standards de conception et de programmation.

Mise en contexte et mandat

Ce travail a pour but premier d'implanter un vecteur d'entiers se comportant comme le vecteur générique de la librairie standard (SL) du langage C++.

L'étudiant devra, suite à la compréhension de la documentation de la SL, définir une représentation interne de la structure de données.

Finalement, dans un environnement C++, l'étudiant devra :

1. Implanter un vecteur dynamique
2. S'assurer de la qualité en déterminant les cas limites
3. S'assurer de la qualité par des tests d'utilisation.

Code de base

Quatre fichiers sont fournis avec cet énoncé:

IntVector.h :
: Fichier de spécification de la classe IntVector. Contient la représentation interne et la déclaration des méthodes.

IntVector.cpp :
: Fichier contenant l'implantation des méthodes de la classe IntVector.

main.cpp :
: Fichier démontrant le fonctionnement de la méthode d'affichage du vecteur. Base pour tester la structure de données.

IntVectorAffichage.cpp :
: Implantation de la méthode afficher. Elle est fournie pour vous aider dans mise au point du vecteur. Vous pouvez la modifier selon vos besoins.

Dans le code qui vous est fourni, les données sont contenues dans un tableau statique. Il est conseillé de faire un maximum de méthodes avant de rendre la structure de données dynamique.

Le type `size_type` est un entier non signé. Il est utile pour représenter le nombre d'éléments, la capacité et les indices d'une donnée dans le IntVector. Vous pouvez l'utiliser à l'extérieur de la classe comme suit :

```
IntVector ::size_type <nom_de_variable>;
```

Pour l'instant le corps des méthodes dans IntVector.cpp ne sont que des substituts qui ne font rien ou retournent des valeurs quelconques. Vous devez les rendre fonctionnellement équivalente à celles de la SL.

Consignes

1. Inscrire votre nom et matricule dans l'en-tête des fichiers livrable.
2. Ne pas modifier aucune signature de méthode du IntVector.
3. La représentation interne de base est un tableau statique. Vous devez remettre une classe qui peut **croitre dynamiquement**.
4. Votre classe ne doit pas laisser fuir de la mémoire.
5. Votre représentation interne ne doit pas être l'une des classes de la SL (ou autres librairies).
6. Un bonus de 5% sera alloué a toute personne qui ajoutera l'opérateur[].

Critères d'évaluations

- Respect des consignes
- Réallocation uniquement lorsque nécessaire
- Exactitude des résultats
- Respect de la complexité algorithmique (notation O)
- La solution peut se faire en une quarantaine de lignes (excluant en-têtes de méthode et accolades)

Conseils

- Les méthodes dans le .cpp sont en ordre de difficulté, essayez de les faire dans cet ordre.
- Ne travaillez que sur une seule méthode à la fois. Passez à la prochaine que lorsque vous avez complètement testé votre implantation.
- Une fois votre classe dynamique, re-testez l'ensemble des méthodes
- Si vous avez des questions contactez le professeur, ne restez pas bloquer.

Contexte de réalisation et démarche de développement

Ce travail doit être réalisé **individuellement**.

Livrable :

- IntVector.h
- IntVector.cpp
- readme.txt (seulement si vous en ressentez le besion)

Conditions de remise :

- La structure de données doit compiler et être fonctionnelle (Tout travail mérite salaire. Le correcteur se paye en points… jusqu'à ce qu'il démissionne)
- Remise via LÉA
- Date limite : Samedi 23 septembre 2017 en fin de journée

Bon travail!