Mini-projet 2 (Module et Généricité)

Rappel : Comme pour tous les TP, il faut commencer par faire un « svn update » depuis votre dossier « pim/tp » pour récupérer les fichiers fournis.

1 Cahier des charges

Notre objectif est de définir une structure de données de type *ensemble*. Par exemple, on peut considérer un ensemble d'entiers E_1 composé des éléments 5, 10 et 7 que l'on notera $\{5, 10, 7\}$, comme en mathématiques. Dans un ensemble l'ordre des éléments n'a pas d'importance. Ainsi $\{5, 10, 7\}$ et $\{7, 5, 10\}$ correspondent au même ensemble.

Un ensemble est équipé d'opérations qui permettent de :

1. Savoir si un ensemble est vide ou pas.
2. Obtenir la taille d'un ensemble (le nombre d'éléments qu'il contient).
3. Savoir si un élément est présent dans un ensemble.
4. Ajouter un élément dans un ensemble.
5. Supprimer un élément d'un ensemble.

Il existe aussi un sous-programme qui permet d'initialiser un ensemble (il est vide) et un sous-programme qui permet de détruire un ensemble (libérer les ressources qu'il utilise).

Ce nouveau type *Ensemble* [1] et les sous-programmes associés seront définis dans un module. Pour que ce module soit utilisable dans différents contextes, il faut permettre à ses utilisateurs de choisir le type des éléments de l'ensemble (un ensemble d'entiers, de caractères, de dates, etc.).

Nous souhaitons avoir deux versions de ce module. La première, `Ensembles_Tableau` utilisera un tableau pour stocker les éléments de l'ensemble. La seconde, `Ensembles_Chainage`, utilisera l'allocation dynamique de mémoire [2] pour chaîner linéairement les éléments de l'ensemble.

L'objectif est donc de définir ces modules génériques, leurs programmes de test et un programme les utilisant. La suite précise ce qui est attendu.

E_1 Les modules `Ensembles_Tableau` et `Ensembles_Chainage` doivent fournir les mêmes opérations avec les mêmes signatures. Il pourra cependant y avoir des différences dans les spécifications, en particulier les contrats.

E_2 L'utilisateur de ces modules doit pouvoir choisir le type des éléments de l'ensemble (ensemble d'entiers, de réels, de dates, etc.). Dans le cas du module `Ensembles_Tableau` il doit aussi pouvoir choisir la capacité (capacité de 10, 50, 10000, etc.).

1. On l'appellera `T_Ensemble` en Ada.
2. Les cours, TD et TP auront lieu la semaine du 21.

E_3 Le type `T_Ensemble` sera très privé (`limited private`).

E_4 La taille d'un `Ensemble_Tableau` doit être obtenue en temps constant.

E_5 Pour chaque sous-programme, sa spécification doit être donnée. Les préconditions et post-conditions seront formalisées dans la syntaxe Ada (dans la mesure du possible).

E_6 Une opération `Appliquer_Sur_Tous` sera disponible sur le module *ensembles*. Elle consiste à appliquer une opération sur chaque élément de l'ensemble. Elle pourra être utile par exemple pour afficher l'ensemble : il suffira d'appliquer sur chaque élément de l'ensemble l'opération qui affiche un élément.

E_7 Dans `Ensembles_Chainage`, « supprimer un élément » doit utiliser la récursivité.

E_8 Dans `Ensembles_Chainage`, « ajouter un élément » ne doit pas utiliser la récursivité.

E_9 Les programmes doivent s'exécuter sans que valgrind (ou valkyrie) ne signalent d'erreurs concernant la gestion de la mémoire.

E_{10} Un scénario (programmes `scenario_*.adb`) permettra de tester une utilisation simplifiée des opérations sur les ensembles.

E_{11} Des tests supplémentaires devront être faits dans les fichiers `test_ensembles_*.adb`. On indiquera clairement l'objectif des tests faits.

E_{12} Un programme `nombre_moyen_tirage_*.adb` calculera le nombre moyen de tirages qu'il faut faire pour obtenir tous les nombres d'un intervalle entier Min..Max en utilisant le générateur aléatoire Alea. Le nombre moyen sera calculé sur 100 essais.

2 Conseils

1. Lire complètement le fichier LISEZ-MOI.txt, en particulier les questions qui y sont posées.
2. Commencer par faire la version *tableau* des ensembles et des programmes associés (premier rendu). Une fois cette version tableau testée, on pourra s'intéresser à la version *chainage*. Les spécifications des modules, les programmes de test et d'utilisation seront presque les mêmes. Ici, nous ferons du copier/coller. Des techniques sont possibles pour l'éviter [3] comme engendrer les fichiers à partir d'un modèle (par exemple avec la commande *make* qui sera vue au deuxième semestre en *Langage C*) ou en s'appuyant sur l'héritage, le sous-typage et la liaison dynamique (ces notions sont présentes en Ada mais ne font pas partie du module PIM ; elles seront abordées au deuxième semestre dans le module *Technologie Objet* avec le langage Java).
3. Pousser régulièrement les modifications sur SVN.

3 Livrables

Les squelettes des livrables sont fournis sur le SVN, dossier `pr2`. Les versions intermédiaires et finales devront y être poussées. Aucun fichier n'est à ajouter sur SVN.

3. Le copier/coller est rarement une bonne chose !

L_1 `LISEZ-MOI.txt` : fichier fourni à compléter. Le lire dès le début du projet !

L_2 `*.ads` : la spécification des modules. Les sous-programmes doivent **impérativement** apparaître dans l'ordre du sujet (en commençant toutefois par initialiser et détruire).

L_3 `ensemble_*.adb` : l'implantation des modules. Les sous-programmes apparaîtront **impérativement** dans l'ordre du sujet.

L_4 `scenario_*.adb` : un scénario pour tester de manière simplifiée le module Ensembles. On doit simplement mettre le code correspondant après les commentaires qui ne sont pas suivis d'instructions. On ne fera ni plus, ni moins que ce qui est demandé.

L_5 `test_ensemble_*.adb` : les programmes de tests des modules Ensembles.

L_6 `scenario_*.adb` et `nombre_moyen_tirage_*.adb` : programmes qui utilisent les modules Ensembles.

4 Echéances

— lundi 14 octobre : mise en ligne du sujet sur Moodle et des fichiers fournis sur SVN.

— du 14 au 18 octobre : une séance de TP consacrée à ce projet. Il est conseillé d'avoir lu le sujet du projet avant ce TP.

— vendredi 25 octobre pour rendre la version *tableau* (`*_tableau.ad*`).

— semaine du 4 novembre : l'enseignant de TP pourra faire un retour sur la version rendue. Elle seront prises en compte pour le rendu final.

— jeudi 14 novembre : date limite pour pousser les livrables sur SVN.