Implémentation du type Vecteur

Objectif

Un vecteur est un ensemble dynamique d'objets d'entiers successifs permettant plusieurs opérations :
- l'obtention de la taille,
- la modification,
- l'insertion à une position
- la suppression,
-la recherche
-tester si le vecteur est vide

L'objectif de ce TP est de fournir une implémentation pour ce type abstraits « Vecteur », en s'aidant de la spécification CASL qui nous a été transmise.

Notre travail a été divisé en cinq étapes :
Étape 1 : Analyse de la spécification Casl du type abstrait (fournie et consultable/éditable grâce au logiciel EMACS)
Étape 2 : Validation de la spécification sous HETS CASL.
Étape 3 : Spécification des opérations du type Vecteur (estVide, modifier, supprimer, ième, ...)
Phase 5 : Implémentation du type « Vecteur », de ses opérations (Vecteur.h, Vecteur.cpp)
Phase 6 : Validation de l'implémentation (main.cpp)

C'est le langage C++ qui a été choisi pour implémenter la spécification Casl vue en cours.

I- Spécification du type abstrait Vecteur :

```
library  libraryVecteur
%%************************************************
%%        TYPE ABSTRAIT DES VECTEURS GENERIQUES
%%   Objet: Specification Algébrique en CASL
%%   Prouvée par: HETS(Isabelle)
%%   Application: manipulation des listes classiques
%%               Date création le 16/04/2005
%%               Auteur: K. OURIACHI, Professeur des universités
%%               UNIVERSITE de PAU/PAYS de l'ADOUR
%%
%%************************************************

%% liste des importations (downloading)
from Basic/Numbers get Nat,Int, Rat

%% specification minimale
spec Vecteur0 [sort Elem] given Int  =
generated type Vecteur[Elem] ::= vecteurVide |
                                 inserer( Vecteur[Elem]; Int; Elem)?

pred
          estVide: Vecteur[Elem]
op
          taille: Vecteur[Elem]  -> Int ;

forall  e1:Elem; i1,k1: Int;v1 :Vecteur[Elem]
      . def inserer(v1,i1,e1) <=> 0< i1 /\ i1 <= taille(v1)+1
%% axiomes concernant le prédicat
      . estVide(vecteurVide)
      . not estVide(inserer(v1,i1,e1))
%% calcul de la taille
      . taille(vecteurVide)= 0
      . taille(inserer(v1,k1,e1)) = taille(v1)+1
end

%%
%%
%%
%%
```

```
%%la spécification minimale Vecteur0 peut enrichie comme suit:
spec Vecteur1 [sort Elem] =
         Vecteur0 [sort Elem]
then
     ops
%% constructeur non générateur
        modifier:Vecteur[Elem]   *  Int * Elem -> ?Vecteur[Elem] ;
     supprimer: Vecteur[Elem]   *  Int -> ?Vecteur[Elem] ;
 %% l'observateur le plus important
     ieme: Vecteur[Elem] * Int ->? Elem
forall e1: Elem; i1,k1:Int; v1:Vecteur[Elem]
%% dom de supprimer
 . def modifier(v1,i1,e1) <=>  0<i1 /\ i1<=taille(v1)
%% dom de supprimer
 . def supprimer(v1,i1) <=>  0<i1 /\ i1<=taille(v1)
 %% dom de ieme
 . def ieme (v1,i1)   <=> 0<i1 /\ i1<= taille(v1)

%% calculer l'élement de rang i
 . 0<i1 /\ i1<=taille(v1)+1 =>  ieme(inserer(v1,i1,e1),i1) = e1
 . i1 < k1/\ k1<=taille(v1)+1 => ieme(inserer(v1,k1,e1), i1) = ieme(v1,i1)
 . k1 <i1 /\ i1<=taille(v1)+1 =>  ieme(inserer(v1,k1,e1),i1) = ieme(v1,i1-1)

%% le constructeur supprimer est défini par induction
 . i1 = k1 => supprimer(inserer(v1,k1,e1),i1) = v1
 . i1 > k1 => supprimer(inserer(v1,k1,e1),i1) = inserer(supprimer(v1,i1-1),k1, e1)
 . i1 < k1 => supprimer(inserer(v1,k1,e1),i1)  = inserer(supprimer(v1,i1),k1-1,e1)

end
```

II- Validation de la spécification :

Il existe des outils pour valider les spécifications formelles.
Il nous a été présenté en cours et TP la commande hets(Heterogeneous Tool Set).
Afin de lancer cette analyse de spécification hets il est nécessaire d'utiliser la commande suivante via un terminal :

```
hets -g libraryVecteur.casl
```

où libraryVecteur.casl consultable sur le logiciel EMACS, est le fichier contenant le composant libraryVecteur.

III- Spécification des opérations du type

Il sagit ici de définir les observateurs et les constructeurs
En bleu l'on observe les constructeurs et en vert les observateurs

```
spec Vecteur0 [sort Elem] given Int  =
generated type Vecteur[Elem] ::= vecteurVide |
                                 inserer( Vecteur[Elem]; Int; Elem)?

pred
           estVide: Vecteur[Elem]
op
           taille: Vecteur[Elem]  -> Int ;

forall  e1:Elem; i1,k1: Int;v1 :Vecteur[Elem]
      . def inserer(v1,i1,e1) <=> 0< i1 /\ i1 <= taille(v1)+1
%% axiomes concernant le prédicat
      . estVide(vecteurVide)
      . not estVide(inserer(v1,i1,e1))
%% calcul de la taille
      . taille(vecteurVide)= 0
      . taille(inserer(v1,k1,e1)) = taille(v1)+1



spec Vecteur1 [sort Elem] =
         Vecteur0 [sort Elem]
then
     ops
%% constructeur non générateur
        modifier:Vecteur[Elem]   *  Int * Elem -> ?Vecteur[Elem] ;
     supprimer: Vecteur[Elem]   *  Int -> ?Vecteur[Elem] ;
 %% l'observateur le plus important
     ieme: Vecteur[Elem] * Int ->? Elem
forall e1: Elem; i1,k1:Int; v1:Vecteur[Elem]
%% dom de supprimer
 . def modifier(v1,i1,e1) <=>  0<i1 /\ i1<=taille(v1)
%% dom de supprimer
 . def supprimer(v1,i1) <=>  0<i1 /\ i1<=taille(v1)
 %% dom de ieme
 . def ieme (v1,i1)   <=> 0<i1 /\ i1<= taille(v1)

%% calculer l'élement de rang i
 . 0<i1 /\ i1<=taille(v1)+1 =>  ieme(inserer(v1,i1,e1),i1) = e1
 . i1 < k1/\ k1<=taille(v1)+1 => ieme(inserer(v1,k1,e1), i1) = ieme(v1,i1)
 . k1 <i1 /\ i1<=taille(v1)+1 =>  ieme(inserer(v1,k1,e1),i1) = ieme(v1,i1-1)

%% le constructeur supprimer est défini par induction
 . i1 = k1 => supprimer(inserer(v1,k1,e1),i1) = v1
 . i1 > k1 => supprimer(inserer(v1,k1,e1),i1) = inserer(supprimer(v1,i1-1),k1, e1)
 . i1 < k1 => supprimer(inserer(v1,k1,e1),i1)  = inserer(supprimer(v1,i1),k1-1,e1)
```

IV- Implémentation du type abstrait vecteur

On crée le fichier **vecteur.h** qui comprend :
- la définition d'une classe libraryVecteur.
- la déclaration, dans cette classe, des constructeurs et des observateurs du type.

vecteur.h

```h
#define TAILLE_MAX 100
#include <iostream>
#include <cstdlib>

struct Vecteur{
  int elements[TAILLE_MAX];
  int nbElements;
};

// Définition des Méthodes

// Retourne "true" ou "false", permet de savoir si le Vecteur est instancié ou est vide
bool estVide(Vecteur unVecteur);
//VALIDE

// Permet d'afficher le contenu d'un Vecteur donné en paramètre
void afficherVecteur(Vecteur unVecteur);
//VALIDE

// Retourne la longueur du vecteur donné en paramètre (nombre d'elements qu'il contient)
int obtenirTaille(Vecteur unVecteur);
//VALIDE

// Ajouter un element à une position donnée dans un Vecteur
Vecteur insererAUnePosition(Vecteur unVecteur, int unePosition , int unElement);
//VALIDE

// Modifie à une position dans un Vecteur un Element en le remplacant par l'element en paramètre
Vecteur modifier(Vecteur unVecteur, int unePosition, int unElement);
//VALIDE

// Supprime un element dans un Vecteur à une position donnée en paramètre
Vecteur supprimer(Vecteur unVecteur, int unePosition);
//VALIDE

// Recherche dans un Vecteur à une position donnée l'élément stocké
int ieme(Vecteur unVecteur, int unePosition);
//VALIDE
```

Vecteur.cpp

Fonction EstVide

vecteur.cpp vecteur.h

```cpp
#include "vecteur.h"
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
#include <cstdlib>
#define TAILLE_MAX 100

using namespace std;

// Retourne "true" ou "false", permet de savoir si le Vecteur est instancié ou est vide
bool estVide(Vecteur unVecteur){
  if(unVecteur.nbElements == 0){
    return(true);
  }
  else{
    return(false);
  }
}
```

Procédure afficherVecteur

```
20  // Permet d'afficher le contenu d'un Vecteur donné en paramètre
21  void afficherVecteur(Vecteur unVecteur){
22    int i;
23    cout << "" << endl;
24    for(i = 0; i < unVecteur.nbElements; i++){
25      cout << unVecteur.elements[i] << endl;
26    }
27    cout << "" << endl;
28  }
```

fonction obtenirTaille

```
31  // Retourne la longueur du vecteur donné en paramètre (nombre d'elements qu'il contient)
32  int obtenirTaille(Vecteur unVecteur){
33    /*cout<< "Le vecteur contient : "<< unVecteur.nbElements << endl;
34    cout<< " "<< endl;*/
35    return unVecteur.nbElements;
36  }
```

fonction insererAUneposition

```
38  // Ajouter un element à une position donnée dans un Vecteur
39  Vecteur insererAUnePosition(Vecteur unVecteur,int unePosition, int unElement){
40    int i;
41
42    if((unVecteur.nbElements < TAILLE_MAX) && (unePosition > 0) && (unePosition <= unVecteur.nbElements+1)){
43      for(i = unVecteur.nbElements; i >= unePosition; i--){
44        unVecteur.elements[i] = unVecteur.elements[i-1];  //Boucle qui permute les valeurs du Vecteur [i] = [i-1]
45      }
46
47      unVecteur.elements[unePosition-1] = unElement; // Stockage de la nouvelle valeur
48      unVecteur.nbElements = unVecteur.nbElements+1; // Augmentation de une case du vecteur
49
50      return(unVecteur);  //Retourne le vecteur modifié
51    }
52    else{
53      cout << "Erreur dans le programme lors de l'insertion à une position !" << endl;
54      return(unVecteur);
55    }
56  }
```

fonction modifier

```
58  // Modifie à une position dans un Vecteur un Element en le remplacant par l'element en paramètre
59  Vecteur modifier(Vecteur unVecteur, int unePosition, int unElement){
60    int laTaille = obtenirTaille(unVecteur);
61    if ((unePosition>=1) && (unePosition<= laTaille)){
62      unVecteur.elements[unePosition-1]=unElement;
63    }
64    return(unVecteur);
65  }
```

fonction supprimer

```
67  // Supprime un element dans un Vecteur à une position donnée en paramètre
68  Vecteur supprimer(Vecteur unVecteur, int unePosition){
69    int i;
70
71    //Si l'élément à supprimer est à la première position
72    for(i = unePosition-1 ; i != obtenirTaille(unVecteur)  ; i++){
73      unVecteur.elements[i] = unVecteur.elements[i+1];
74    }
75    unVecteur.nbElements = unVecteur.nbElements-1;
76    return(unVecteur);
77  }
```

fonction ième

```
79  // Recherche dans un Vecteur à une position donnée l'élément stocké
80  int ieme(Vecteur unVecteur, int unePosition){
81    if((unePosition > 0) && (unePosition <= unVecteur.nbElements)){
82      return(unVecteur.elements[unePosition-1]);
83    }
84    else{
85      cout << "Position invalide, erreur dans le programme." << endl;
86      return(0);
87    }
88  }
```

V- Validation de l'implémentation

Pour vérifier que l’implémentation est correcte vis-à-vis de la spécification décrite en Casl, il suffit de prouver la satisfiabilité de l’implémentation (main.cpp) de chacune des opérations du type en s’appuyant sur leur spécification.

```
#include "vecteur.h"
#include <iostream>
#include <stdio.h>
#include <cstdlib>

using namespace std;

int main(){
     int i,tailleVecteur;
     Vecteur monVecteur;

     //Déclaration d'un Vecteur vide
     monVecteur.nbElements = 0;

     //Verification de estVide (doit renvoyer true)
     if(estVide(monVecteur) == true){
          cout << "Le vecteur est vide (initialement) , verification correcte de
'estVide()'" <<endl;
               cout<< " "<< endl;
     }
     else{
          cout << "Erreur dans la vérification n°1" << endl;
          cout<< " "<< endl;
     }

     //Le tableau va contenir n elements
     cout << "Combien de cases souhaitez-vous que ce vecteur contienne ?" << endl;
    cin >> monVecteur.nbElements;

     //Vérification :      obtenirTaille()
     cout<< "Appel de 'obtenirTaille()' & vérification . . ."<< endl;
     cout<< "Le vecteur contient : " << monVecteur.nbElements << " elements" << endl;
     cout<< " "<< endl;
     tailleVecteur = obtenirTaille(monVecteur);



     //L'utilisateur va saisir manuellement les 10 valeurs
     for (i=0 ; i < tailleVecteur ; i++){
          cout << "Saisir une valeur pour la case " << i+1 << " ." <<endl;
          cin >> monVecteur.elements[i];
     }



     //Seconde vérification de estVide (doit renvoyer false)
     if(estVide(monVecteur) == true){
          cout << "Erreur dans la vérification n°2" << endl;
          cout << " " << endl;
     }
     else{
          cout << "Le vecteur n'est pas vide (complété précédement), vérification
correcte de 'estVide()'" <<endl;
          cout<< " "<< endl;
     }

     //Affichage
     cout << "Affichage du contenu du Vecteur : " << endl;
     afficherVecteur(monVecteur);
     system("pause");
     system("clear"); //Effacer l'affichage des instructions précédente dans la console

     //Vérification : insererAUnePosition()
```

```
     int maPosition;
     int monElement;

     cout << "Saisissez un nouvel élément à ajouter : " << endl;
    cin >> monElement;

     cout << "A quelle position souhaitez vous insérer ce nouvel élément ?" << endl;
    cin >> maPosition;


     monVecteur = insererAUnePosition(monVecteur, maPosition, monElement);

     //Affichage du Vecteur : Vérification de l'Element inséré
     cout << "Affichage du tableau après insertion" << endl;
     afficherVecteur(monVecteur);
     cout << "Taille : " << obtenirTaille(monVecteur) << endl;
     cout << "" << endl;

     //Vérification : modifier()
     cout << "Saisissez un élément (modification): " << endl;
    cin >> monElement;

     cout << "A quelle position souhaitez vous modifier la valeur par celle saisie
précédement ?" << endl;
    cin >> maPosition;
    cout << "" << endl; // Saut de ligne

    monVecteur = modifier(monVecteur, maPosition, monElement);
    cout << "Affichage du tableau après insertion" << endl;
     afficherVecteur(monVecteur);



     //Vérification de : supprimer()
     cout << "A quelle position souhaitez vous supprimer la valeur ? (verification
supprimer() )" << endl;
    cin >> maPosition;
     monVecteur = supprimer(monVecteur,maPosition);
     afficherVecteur(monVecteur);
     cout<< "Le vecteur contient : "<< monVecteur.nbElements << " elements" << endl;


     //Vérification de : ième()
     cout << "Donnez une position afin d'obtenir sa valeur ? (verification ième() )" <<
endl;
    cin >> maPosition;
     monElement = ieme(monVecteur, maPosition);
     if(monElement != 0){
          cout << "Element obtenu pour la position : " << maPosition << " -> " <<
monElement << endl;
     }
     else{
          cout << "Aucun élement à afficher" << endl;
     }
     return(0);
 }
```

Conclusion

Grâce à cet exercice Nous nous sommes replongés dans les règles que régissent l’implémentation des vecteurs en s'appuyant sur la spécification Casl.
Nous avons appris à lier plusieurs fichiers entre eux pour une compilation finale.