UNIVERSITÉ DE SHERBROOKE
DÉPARTEMENT D'INFORMATIQUE

IFT 339

Série d'exercices - Thème #3 : Mécanisme d'abstraction en C++

Exercice 1 :

Définir un classe générique Matrice encapsulant une matrice carrée de taille variable n x n, telle que le type des données stockées dans la matrice, et la taille n de la matrice sont choisis par l'utilisateur. Exemple d'utilisation :

```
Matrice<int , 3> M ; // Matrice de taille 3x3 stockant des entiers.
M.at(0,0) = 5 ; // Affecte la valeur 5 a l'element en position (0,0)
```

Tous les éléments doivent etre stockés dans un seul tableau de taille fixe n*n.

```
template <typename TYPE, unsigned int DIM>
class Matrice {
  private :
  . . .
  public :
  // construction avec parametre
  Matrice(TYPE val);

  // destructeur
  ~Matrice();

  //operateur d'acces
  TYPE& at(size_t i, size_t j);

  //addition de deux matrices de meme taille
  operator+(Matrice<TYPE,DIM>);
}
```

Exercice 2 :
Écrire une fonction qui prend en paramètres un tableau d'entiers (int * tab), la taille du tableau (size_t size), et 2 pointeurs vers des entiers (int* min, int* max). La fonction doit renvoyer dans les entiers pointés par min et max respectivement le plus petit et le plus grand entiers du tableau.

```
void trouve_min_max(int * tab, size_t size, int* min, int* max){
  ...
}
```

Exercice 3 :

Définir un classe générique Polygone à nombre de sommets variables n, telle que le nombre de sommets du polygone est choisi par l'utilisateur. Exemple d'utilisation :

```
Polygone<4> R ; // Polygone a 4 sommets
```

La classe Polygone doit utiliser une classe Point2D définie au préalable. Écrire le constructeur sans paramètre, un constructeur avec paramètre (un tableau de Points), le destructeur s'il y a lieu, et une fonction permettant de calculer le diamètre d'un polygone.

Correction de l'Exercice 2 :

```
#include <iostream>
using namespace std;

// Solution avec passage de parametre par valeur
void trouve_passage_par_valeur(int *tab, size_t size, int *min, int *max)
{

    int mini = tab[0];
    int maxi = tab[0];

    for (size_t i = 0; i < size; i++){
        if (mini > tab[i]){
            mini = tab[i];
        }
        if (maxi < tab[i]){
            maxi = tab[i];
        }
    }

    // passage par valeur : min et max sont des copies de minm et maxm qui
    // sont passes en parametre de la fonction 'trouve_passage_par_valeur'
    // dans le main. min et minm pointent donc au meme endroit, de meme que
    // max et maxm. On peut donc aller modifier la valeur des entiers vers
    // lesquels ils pointent.
    // Notez que si minm et maxm sont egaux a 'nullptr', il est impossible
    // d'aller modifier la valeur des entiers vers lesquels ils pointent,
    // puisque minm et maxm  ne sont pas accessibles dans cette fonction.
    // On ne peut utiliser que l'operateur de dereferencement '*' pour
    // aller modifier les entiers vers lesquels minm et maxm pointent.

    *min = mini;
    *max = maxi;
}
```

```
// Premiere solution avec passage de parametre par reference
// et utilisation de l'operateur d'indirection '*'
void trouve_passage_par_reference1(int *tab, size_t size, int * &min, int * &max)
{
    int mini = tab[0];
    int maxi = tab[0];

    for (size_t i = 0; i < size; i++){
        if (mini > tab[i]){
            mini = tab[i];
        }
        if (maxi < tab[i]){
            maxi = tab[i];
        }
    }

    // passage par reference : min et max sont des alias de minm et maxm qui
    // sont passes en parametre de la fonction 'trouve_passage_par_reference1'
    // dans le main. min et minm constituent donc le meme pointeur, de meme que
    // max et maxm. Cela signifie que minm et maxm sont accessibles dans la
    // fonction en utilisant les alias min et max.
    // Dans ce cas, on peut gerer les cas dans lesquels minm ou maxm = nullptr
    // Ici, on utilise l'operateur de dereferencement '*'.
    if(min == nullptr)
        min =  new int(mini);
    else
       *min = mini;
    if(max == nullptr)
        max =  new int(maxi);
    else
       *max = maxi;
}
```

```
// Seconde solution avec passage de parametre par reference
// et utilisation de l'operateur de dereferencement '&'
void trouve_passage_par_reference2(int *tab, size_t size, int * &min, int * &max)
{
    int mini = 0;
    int maxi = 0;

    for (size_t i = 0; i < size; i++){
        if (tab[mini] > tab[i]){
            mini = i;
        }
        if (tab[maxi] < tab[i]){
            maxi = i;
        }
    }
    // Comme dans la fonction precedente, minm et maxm sont accessibles dans la
    // fonction en utilisant les alias min et max.
    // On peut egalement gerer tous les cas, y comprie celui ou
    // minm ou maxm = nullptr
    // Ici, on utilise l'operateur de dereferencement '&' pour faire pointer
    // minm et maxm vers des cases du tableau.

    min = &tab[mini];
    max = &tab[maxi];
}


int main(int argc, char const *argv[])
{
    int *minm = new int();
    int *maxm = new int();

    // Val min = 0; Val max = 4
    int *tab = new int[4];
    tab[0] = 0;
    tab[1] = 2;
    tab[2] = 4;
    tab[3] = 3;
```

```
trouve_passage_par_valeur(tab, 10, minm, maxm);

cout << "Passage de parametre par valeur (erreur si minm, maxm = nullptr):"<< endl;
cout << "   Minimum = " << *minm << endl;
cout << "   Maximum = " << *maxm << endl;

delete minm;
delete maxm;
minm = nullptr;
maxm = nullptr;

trouve_passage_par_reference1(tab, 10, minm, maxm);

cout << "Passage de parametre par reference 1 avec minm, maxm = nullptr:"<< endl;
cout << "   Minimum = " << *minm << endl;
cout << "   Maximum = " << *maxm << endl;

trouve_passage_par_reference1(tab, 10, minm, maxm);

cout << "Passage de parametre par reference 1 avec minm, maxm != nullptr:"<< endl;
cout << "   Minimum = " << *minm << endl;
cout << "   Maximum = " << *maxm << endl;

delete minm;
delete maxm;
minm = nullptr;
maxm = nullptr;

trouve_passage_par_reference2(tab, 10, minm, maxm);

cout << "Passage de parametre par reference 2 avec minm, maxm = nullptr:"<< endl;
cout << "   Minimum = " << *minm << endl;
cout << "   Maximum = " << *maxm << endl;

trouve_passage_par_reference2(tab, 10, minm, maxm);

cout << "Passage de parametre par reference 2 avec minm, maxm != nullptr:"<< endl;
```

```
    cout << "   Minimum = " << *minm << endl;
    cout << "   Maximum = " << *maxm << endl;

    // Notez qu'apres l'appel a 'trouve_passage_par_reference2', on ne peut pas
    // desallouer les espaces vers lesquels pointent minm et maxm puisqu'ils
    // pointent vers des cases de tableau tab.

    //delete minm;
    //delete maxm;
    delete [] tab;
    return 0;
}
```