**Contrôle Continu AP4A – Mardi 23 Avril 2024 - durée 1 heure  
(aucun document autorisé)**

**I** (2 pts) **-** On souhaite écrire une fonction swap en C++, pour permuter le contenu de 2 variables qui peuvent être des objets ou des types élémentaires. Par exemple, on doit pouvoir écrire :

{

int i = 2, j = 3;

swap(i, j);

Point p1(1,2), p2(3,4);

swap(p1, p2);

vecteur<int> v1(3), v2(3);

swap(v1, v2);

}

Quel mécanisme utiliser pour n’écrire qu’une seule fonction swap() pour tout type de donnée et qui fonctionnera pour le programme ci-dessus. En conséquence, donner la/les fonctions swap du programme.

**II.** (4 pts) Considérons la classe Vecteur en C++ tel que réalisé en TP et des variables de type Vecteur. Donner la notation fonctionnelle équivalente des instructions suivantes :

Vecteur a, b, c, d, u, v;

... // les vecteurs sont initialises avec des contenus

a = a \* b + c;

c = (a + b) + c;

v[i] = v[j] = 2.0;

a = b = c = 2;

cout << “Vecteur u : “ << u << “Vecteur v : “ << v << endl;

((a = b) = c) = 2;

if (u != v) u = v;

++u = v++;

**III**. (2 pts) On considère ici la classe Liste telle que codée en récursif :

template <class T>

class Liste {

T val; // tete de liste = element

Liste\* reste; // suivi d'une liste

public: … };

Donner un schéma d’occupation mémoire (pile et tas) pour le programme suivant aux points d’exécution 1) et 2).

{

Liste<float> l;

l.ajouter(3.0, 0);// ajouter

l.ajouter(4.0, 0);

l.ajouter(6.0, l.longueur());

l.ajouter(7.0, 1);

Liste<float>\* pl = new Liste<float>;

pl->ajouter(3.0, 0);// ajouter

pl->ajouter(4.0, 0);

pl->ajouter(6.0, pl->longueur());

pl->ajouter(7.0, 1);

//1)

delete pl;

//2)

}

**IV.** (4 pts) Définissez une classe Personne avec des attributs très généraux comme :

* Le nom
* la date de naissance
* le no de sécurité sociale

Définissez un constructeur prenant en paramètres les trois attributs. Définissez des méthodes qui renvoient les valeurs d’attribut. Surcharger l'operateur ostream& operator<<(ostream&, const Personne&) permettant d’afficher les attributs d’une personne. Donner un exemple de programme (main) illustrant l’utilisation.

**V**. (8 pts) On souhaite définir une sous-classe Iterateur (incluse à l’intérieur) de la classe vecteur respectant l’interface standard des itérateurs de la STL (Standard Template Library) pour parcourir le contenu d’un vecteur. Une ébauche de la classe est la suivante :

template <class T>

class vecteur

{…

public:

class Iterateur {

friend class vecteur;

... pv;

Iterateur(...) : pv(...) {}

public:

bool **operator**!=(Iterateur it) {

return pv != ;

}

Iterateur& **operator**++() {

pv ...;

return ...;

}

Iterateur **operator**++(int i) {

...;

pv ...;

return ...;

}

... **operator**\*() {// operateur de dereferencement \*iter

return ...;

}

};

Iterateur **begin**() {

Iterateur it(...);

return it;

}

Iterateur **end**() {

Iterateur it(...);

return it;

}

pour l’utiliser de manière standard par exemple de la façon suivante :

{

// Test Iterateur standard

vecteur<int> u(3);

u[0] = u[1] = u[2] = 1;

for(vecteur<int>::Iterateur it = u.begin(); it != u.end(); ++it){

\*it = \*it + 2;

}

}

1) Donner le codage complet (interface et implémentation) de la sous-classe Iterateur et des méthodes begin(), end() de la classe vecteur en complétant là où il y a des « … ».

2) Nous n’avons pas écrit la forme de Coplien pour la classe Iterateur, faudrait-il l’ajouter ? en quoi consiste-t-elle ? pourquoi ne pas l’écrire ?