AIR 2ème années

TP de langage C++
2022-2023

Basé sur les TPs de Maude Manouvrier Merci à elle !.

1 Rappels de C : A rendre à la fin du TP

Exercice 1 *:* **Comparaison de phrases**
Ecrire une fonction qui prend cinq phrases et vérifie si au moins deux phrases sont identiques.

Exercice 2 *:* **Verlan** *Ecrire un programme qui prend cinq phrases et inverse l'ordre des caractères de chaque phrase. Par exemple :*
Allons enfants de la Patrie,
Le jour de gloire est arrivé !
Contre nous de la tyrannie
L'étendard sanglant est levé,
L'étendard sanglant est levé,
sera transformé en :
eirtaP al ed stnafne snollA
évirra tse eriolg ed ruoj eL
einnaryt al ed suon ertnoC
ével tse tnalgnas dradneté'L
ével tse tnalgnas dradneté'L

Vous avez fait un pas vers la cryptographie ! ! ! ! !

Exercice 3 *:* **Palindrome**
Selon la définition dans wikipédia : "un palindrome de lettres, est une figure de style désignant un texte ou un mot dont l'ordre des lettres reste le même qu'on le lise de gauche à droite ou de droite à gauche, comme dans la phrase « ésope reste ici et se repose » ou encore « La mariée ira mal ". écrire un programme qui prend en argument un mot et affiche si le mot est ou pas un palindrome.

Exercice 4 *:* **Somme de 2 matrices.** *Créer 2 matrices mat1 et mat2. Calculer et afficher la matrice mat3 qui est la somme de mat1 et mat2. Rappel : Pour 3 matrices A, B et C, telles que A = B +C, on a :*

$$A_{i,j} = B_{i,j} + C_{i,j}$$

où $A_{i,j}$ est l'élément de A situé sur la ligne i et la colonne j.

Exercice 5 *:* **Trier un tableau de nombre** *Ecrire une fonction qui prend en argument un tableau de 10 nombres et les classes par ordre croissant puis décroissant*

2 Premiers pas en C++

Exercice 6 *:* **Manipulation des entrées-sorties en C++**
écrire un programme permettant à l'utilisateur de saisir un entier `e` *et un réel* `r` *et affichant le produit de* `e` *par* `r`*.*

Exercice 7 *:* **Définition de fonction**
Reprendre l'exercice précédent en calculant le produit à l'intérieur d'une fonction `float produit (int, float)`*.*

Exercice 8 *:* `const` **avec des pointeurs**
Pour chacune des lignes de code suivantes, noter celles qui sont légales et celles qui sont illégales et indiquer ce que fait chaque instruction :

```
int * p;
++(*p);
++p;
int * const cp;
++(*cp);
++cp;
const int * pc;
++(*pc);
++pc;
const int * const cpc;
++(*cpc);
++cpc;
```

Exercice 9 *:* **Manipulation de référence**
Soit le programme suivant :

```
int main()
{
 int i;
 int & j = i;
 int k;

 cout << "Saisir un entier \n"; cin >> i;
 cout << "Valeur de j : " << j << "\n";

 j=j+2;
 cout << "Valeur de j : " << j << "\n";

 cout << "Saisir un entier \n"; cin >> k;
 int & n=k;
 j=n;
 cout << "Valeur de j : " << j << "\n";
 cout << "Valeur de i : " << i << "\n";

 return 0;
}
```

1. *Sans tester sur machine : Indiquer quelles vont être les différentes valeurs affichées.*
2. *Vérifier le résultat sur machine.*

Exercice 10 *:* **Manipulation de référence**
Que font les deux programmes suivants ?

```
int main()
{
        int n = 33;
        int& r = n;
        cout << "n = " << n << ", r= " << r << endl;
        --n;
        cout << "n = " << n << ", r= " << r << endl;
        r *= 2;
        cout << "n = " << n << ", r= " << r << endl;
}

int main()
{
        int n = 33;
        int& r = n;
        cout << "&n = " << &n << ", &r= " << &r << endl;
}
```

Sachant que `n` *est stocké à l'adresse* `0x3fffd14`*.*

Exercice 11 *:* **Passage de paramètres par référence**
Reprendre l'exercice 7 en modifiant la fonction `produit` *de telle sorte qu'elle affecte le résultat du produit à un troisième argument* `p` *de la fonction.*

Exercice 12 *:* **Passage de paramètres par valeur vs. par référence**
Sans tester sur machine dans un premier temps, indiquer où se situe l'erreur dans le programme suivant :

```
//Ce programme contient une erreur
#include <iostream>

void triple(double& num);

int main()
{ double d=7.0;
  triple(&d);
  cout<<d;
  return 0;
}

void triple(double& num)
{ num = 3 * num; }
```

Exercice 13 *:* **Passage de paramètres par valeur vs. par référence**
écrire une fonction permettant d'échanger le contenu de 2 variables de type `string` *de*

deux manière différentes : (1) en transmettant l'adresse des variables concernées et (2) en utilisant la transmission par référence.

Exercice 14 *:* **Sur-définition ou surcharge de fonction**

écrire les fonctions suivantes (sans utiliser la fonction `pow` *:*

- `float puissance(int)` *: fonction retournant le carré de l'entier passé en paramètre de la fonction.*
- `float puissance(float)` *: fonction retournant le carré du réel passé en paramètre de la fonction.*
- `float pruissance(int i,int j)` *: fonction retournant la valeur de* `i` *à la puissance* `j`*.*
- `float pruissance(float f, int i)` *: fonction retournant la valeur de* `f` *à la puissance* `i`*.*

On suppose que les variables suivantes sont définies dans le programme principal :

```
int i, j;
float f;
char c;
```

Pour chacun des appels suivants :

1. *Indiquer (sans tester sur machine) si l'appel est correct et, si oui, indiquer quelle fonction va être effectivement appelée et quelle conversion, s'il y a lieu, va être mise en place.*
2. *Vérifier le résultat de la première question sur machine.*

   (a) `puissance(i);`

   (b) `puissance(f);`

   (c) `puissance(c);`

   (d) `puissance(i,j);`

   (e) `puissance(f,i);`

Exercice 15 *:* **Allocation et désallocation mémoire**

écrire un programme demandant à l'utilisateur de saisir un nombre `nb` *d'éléments dans un tableau d'entiers et créant dynamiquement un tableau contenant exactement* `nb` *entiers dont les valeurs varient de 0 à* `nb-1`*.*

Exercice 16 *:* **Manipulation de pointeurs et d'adresse**

- *écrire une fonction* `location` *qui utilise des pointeurs pour chercher l'adresse d'un entier donné dans un tableau donné ? Si l'entier est trouvé, la fonction retourne son adresse ; dans le cas contraire, elle retourne NULL.*
- *Tester la fonction avec le programme* `main` *suivant :*

```
int main()
{ int a[8]={22, 33, 44, 55, 66, 77, 88, 99}, *p, n;
  do {  cin >> n;
        if(p=location(a,8,n)) cout << p << ", " << *p << endl;
          else cout << n << " non trouv\'e" << endl;
     }
  while (n>0);
}
```

3 Premières notions de classes et d'objets

Exercice 17 *:* **Notions de classe**
écrire une classe `Gadget` *gérant un attribut privée permettant de comptabiliser le nombre d'objets de la classe. Construire et exécuter un programme créant trois objets instances de la classe* `Gadget` *et affichant à chaque fois le nombre courant d'objets.*

Exercice 18 *:* **Appels des constructeurs et destructeurs**

1. *Définir une classe* `Chaine` *ayant un attribut de type* `char *` *et permettant d'avoir le programme principal suivant :*

```
int main()
{ Chaine c("Bonjour");
 cout << "La chaine est \"" << c.afficher() << "\"\n";
}
```

2. *Ajouter, dans le constructeur et le destructeur de la classe* `Chaine`*, des commandes permettant d'afficher à l'écran l'appel implicite de ces méthodes.*
3. *Définir un constructeur par recopie* `Chaine(const Chaine &)`*.*
4. *Sans tester sur machine, indiquer quels vont être les différents appels aux constructeur, constructeur par recopie et destructeur du programme suivant :*

```
int main()
{ Chaine c1("Bonjour"), c2=c1;
 cout << "La chaine c1 a pour valeur \"" << c1.afficher() << "\"\n";
 cout << "La chaine c2 a pour valeur \"" << c2.afficher() << "\"\n";
 return 0;
}
```

5. *Vérifier le résultat de la question précédente sur machine.*

Exercice 19 *:* **Appel de constructeur et passage de paramètre**

1. *Ajouter, à la classe* `Chaine` *de l'exercice 18, une méthode* `bool egal(Chaine c)` *vérifiant l'égalité de l'objet courant avec l'objet* `c`*.*
2. *Ajouter les lignes suivantes au programme principal de l'exercice 18 et indiquer (sans tester sur machine dans un premier temps, puis en vérifiant sur machine par la suite), quels vont être les différents appels aux méthodes de la classe.*

```
if(c1.egal(c2)) cout << "Egalit\'e des chaines \n";
 else cout << "In\'egalit\'e des chaines \n";
```

Exercice 20 *:* **Définition d'une classe** `Pile`
Définir une classe `Pile` *de telle sorte qu'une pile soit définie par un tableau d'entiers* `contenu` *de taille* `max` *(une constante fixée) et par un entier* `sommet` *correspondant au sommet de la pile.*
Par exemple, si la constante `max` *a été définie avec la valeur 10, la pile de sommet* `4` *et de contenu* `3, 2, 1, 5` *sera représentée par un objet de la classe* `Pile` *tel que* `sommet=4` *et :*

- `contenu`$=\{$`3,2,1,5,0,0,0,0,0,0`$\}$*, si l'entier* `0` *ne fait pas partie des entiers pouvant être empilés,*
- *ou* `contenu`$=\{$`3,2,1,5,-1,23,0,13,4,5`$\}$*, avec l'entier* `-1` *indiquant la fin de la pile.*

La classe doit avoir les méthodes suivantes :

- *Un constructeur qui initialise la pile à vide.*
- *Une méthode* `estVide` *retournant* `true` *si la pile est vide et* `false` *sinon.*
- *Une méthode* `SommetDePile` *retournant le sommet de la pile.*
- *Une méthode* `Empiler` *empilant un entier passé en paramètre.*
- *Une méthode* `Depiler` *supprimant le sommet de la pile.*
- *Une méthode* `Afficher` *affichant les éléments de la pile.*

Exercice 21 *:* **Définition d'une classe** `Fraction`

1. *Définir une classe* `Fraction` *ayant deux attributs de type entier privés* `num` *et* `den` *représentant respectivement le numérateur et le dénominateur d'une fraction.*
2. *Définir eux constructeurs pour cette classe :*
   (a) *Le premier constructeur, sans paramètre, construit une fraction dont le numérateur est 0 et le dénominateur est 1.*
   (b) *Le deuxième constructeur construit une fraction à partir du numérateur et du dénominateur transmis en paramètre.*
3. *Définir les méthodes suivantes :*
   (a) `int Signe()` *: retournant 1 si la fraction est positive ou nulle et -1 sinon.*
   (b) `void Afficher()` *: affichant la fraction sous la forme "num/den".*
   (c) `float Valeur()` *: retournant la valeur décimale de la fraction.*
   (d) `void Simplifier()` *: qui simplifie la fraction. Cette méthode nécessite l'écriture d'une fonction récursive* `int pgcd (int a, int b)` *qui calcule le pgcd de deux entiers* `a` *et* `b`*.*
   (e) `Fraction Inverse()` *: retournant la fraction inverse.*
4. *Définir les fonction suivantes :*
   (a) `int compare(Fraction f, Fraction g)` *: retournant 1 si* $f \not\subset g$*, 0 si les deux fractions sont égales et -1 si* $f \not\supset g$*.*
   (b) `Fraction somme(Fraction f, Fraction g)` *: retournant la somme de deux fractions* `f` *et* `g`*.*
   (c) `Fraction difference(Fraction f, Fraction g)` *: retournant la difference de deux fractions* `f` *et* `g`*.*
   (d) `Fraction multiplication(Fraction f, Fraction g)` *: retournant le produit de deux fractions* `f` *et* `g`*.*
   (e) `Fraction division(Fraction f, Fraction g)` *: retournant le quotient de deux fractions* `f` *et* `g`*.*
5. *Reprendre les fonctions définies précédemment et ajouter à la classe* `Fraction` *les opérateurs* `==`*,* `+`*,* `-`*,* `*` *et* $\backslash$*.*

Exercice 22 *:* **Surcharge d'opérateur**

1. *Reprendre la classe de l'exercice 19 en ajoutant une méthode surchargeant l'opérateur* `=`*.*
2. *Ajouter les lignes suivantes au programme principal de l'exercice précédent et indiquer (sans tester sur machine dans un premier temps, puis en vérifiant sur machine par la suite), quels vont être les différents appels aux méthodes de la classe.*

```
Chaine c3;
c3=c2
cout << "La chaine c3 a pour valeur \"" << c3.afficher() << "\"\n";
```

Exercice 23 *:* **Surcharge d'opérateurs**

1. *Remplacer la méthode* `egal` *de l'exercice 22, par la surcharge de l'opérateur* `==`*.*
2. *Ajouter à la classe* `Chaine` *une méthode surchargeant l'opérateur* `+`*.*
3. *Ajouter à la classe* `Chaine` *une méthode surchargeant l'opérateur* `[]`*.*

4 Héritage simple

Exercice 24 *:*

1. *écrire une classe* `Personne` *contenant :*
   - *4 attributs : nom, prénom, age et sexe,*
   - *Un constructeur prenant en paramètre le nom, le prénom, l'age et le sexe de la personne,*
   - *Une méthode* `Get` *pour chacun des attributs,*
   - *Une méthode* `Affiche` *qui affiche les informations sur la personne.*
2. *écrire une classe* `Etudiant` *dérivant publiquement de* `Personne` *et possédant en plus un attribut* `Note`*.*
3. *Surcharger la méthode Afficher définie dans la classe* `Personne`*.*
4. *Indiquer sans machine, quels vont être les appels aux différentes méthodes des classes du programme suivant. Puis tester le sur machine.*

```
int main()
{   Personne *p= new Personne("Gamotte","Albert",34,'M');
    p->Affiche();

    Etudiant e("Gamotte","Albert",34,'M',13);
    e.Affiche();
}
```

Exercice 25 *:*

1. *Créer une classe* `Vect` *permettant de manipuler des "vecteurs dynamiques" d'entiers dont la déclaration est la suivante :*

```
class Vect
{
  int nbElements; // Nombre d'\'el\'ements du vecteur
  int * Adresse;  // Adresse de la zone dynamique contenant les \'el\'ements
  public:
    Vect(int); // Constructeur (avec en param\'etre la taille du vecteur)
    ~Vect();   // Destructeur
    int& operator[] (int); // Acc\'es \'a un \'el\'ement du vecteur par son indice

};
```

2. *Ajouter à la classe précédente un constructeur par copie et un opérateur d'affectation.*

3. *Créer une classe dérivée de* `Vect`*, appelée* `VectB`*, dans laquelle il est possible de fixer les bornes inférieure et supérieure des indices, lesquelles seront transmises au constructeur de la classe. Par exemple* `VectB v(1,5);` *créera un vecteur dynamique dont les éléments seront indicés de 1 à 5 (et non plus de 0 à 5 comme par défaut).*

Exercice 26 *:*

1. *Définir une classe* `Point` *permettant de gérer des points à 2 dimensions.*
2. *Définir une classe* `Segment`*, un segment étant composé de deux points. Cette classe devra contenir une méthode* `norme` *permettant de calculer la norme du segment et un opérateur d'égalité.*
3. *Définir une classe* `Vecteur`*, un vecteur étant un segment orienté. Rédéfnir l'opérateur d'égalité pour cette classe.*

Exercice 27 *: Reprendre l'exercice 24 et tester le avec le programme principal suivant :*

```
int main()
{   Personne *p= new Personne("Gamotte","Albert",34,'M');
    p->Afficher();
    delete p;

    p= new Etudiant("Gamotte","Albert",34,'M',13);
    p->Afficher();
    delete p;
}
```

1. *Que remarquez-vous ?*
2. *Apporter les modifications nécessaires à la classe* `Personne` *pour obtenir le fonctionnement voulu.*

5 Conversion et opérateur de conversion

Exercice 28 *:*
Soit le fichier Exo23.h suivant :

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

class A
{ int x ;
  public:
   A(int i =0) {x=i;} // constructeur
   operator int() // op\'erateur de conversion d'un objet, instance de A, en int
    { cout << "**Appel de int() pour l'objet d'attribut "<< x << endl;
      return x;
    }
};
```

Et le fichier Exo23.cpp suivant :

```
#include "Exo23.h"

void fct (double v)
{ cout << "$$ Appel de la fonction avec comme argument :" << v << endl;}

int main()
{ A obj(1);
  int entier1;
  double reel1, reel2;

  entier1= obj + 1.75; // Instruction 1
  cout << "entier1= " << entier1 << endl;

  reel1= obj + 1.75; // Instruction 2
  cout << "reel1= " << reel1 << endl;

  reel2= obj; // Instruction 3
  cout << "reel2= " << reel2 << endl ;

  fct(obj); // Instruction 4
}
```

1. *Sans tester sur machine, indiquer quelles méthodes et quelles conversions vont être effectuées pour chaque instruction (numérotées de 1 à 4) du programme principal.*
2. *Tester sur machine pour vérifier.*
3. *Ajouter un opérateur de conversion d'un objet de la classe* `A` *en* `double`*.*
4. *Quelles conversions explicites devez-vous faire dans le programme* `main` *pour que le programme compile et que les bons opérateurs de conversion soient appelés ?*

Exercice 29 *:*
Soit le fichier Exo24.h suivant :

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

class B
{ int b1;
  public :
   // constructeur sans argument
   B() {cout << " Passage dans B::B(); " << endl;}
   // constructeur \‘a un argument
   B(int b) {b1=b; cout << "Passage dans B::B(int); " << endl;}
   int GetB() {return b1;}
};

class C
{ int c1;
  public :
   // Constructeur sans argument
```

```
   C() { cout << "Passage dans C::C(); " << endl; }
   // Constructeur \‘a un argument de type entier
   C(int c) {c1=c; cout << " Passage dans C::C(int); " << endl;}
   // Constructeur \‘a un argument de type instance de la classe B
   C(B obj) { c1=obj.GetB(); cout << "Passage dans C::C(B obj)" << endl;}

   // Surd\’efinition de l’op\’erateur =
   C& operator = (const C& obj)
     { c1=obj.c1;
       cout << "Passage dans C& operator = (const C& obj)" << endl;
       return *this;
     }
   // Surd\’efinition de l’op\’erateur + par une fonction amie
   friend C& operator+(const C&,const C&);
};
```

Soit le fichier Exo24.cpp suivant :

```
#include "Exo24.h"

C& operator+(const C& obj1,const C& obj2)
{ C* obj3 = new C(obj1.c1+ obj2.c1);
  cout << "Passage dans operator+(C obj1,C obj2)" << endl;
  return *obj3;
}

int main()
{ B obj2B1, obj2B2, obj2B3;
  C obj2C1, obj2C2, obj2C3;
  obj2C1=obj2B1;  // Instruction 1
  obj2B1=obj2C1;  // Instruction 2
  obj2C1=obj2C2+ obj2C3; // Instruction 3
  obj2C1=obj2B2+ obj2B3; // Instruction 4
  obj2B2+obj2B3; // Instruction 5
  obj2B1=obj2B2+ obj2B3; // Instruction 6
}
```

1. *Sans tester sur machine, indiquer quelles instructions vont compiler ou non. Lorsqu'une instruction comile, indiquer les conversions faites et les appels de méthodes.*
2. *Tester le résultat sur machine.*
3. *Que faut-il faire pour obtenir le même résultat, sans définir, dans la classe* `C`*, de constructeur* `C(B)` *?*

6 Héritage public, privé et protégé

Exercice 30 *:*
Soit une classe de Base dont la définition est la suivante :

```
class Base
```

```
{ int att1;
  public:
   Base(int i=0) {att1=i;}
   void afficher()
    {cout << "Objet de la classe de Base d'attribut 1 = " << att1 << endl;}
   int GetAtt1() {return att1;}
};
```

1. *Dériver une classe* `Derivee` *de la classe de* `Base`*.*
2. *Sans tester sur machine dans un premier temps, puis en testant sur machine par la suite :*
   * *En dérivant la classe* `Derivee` *de manière publique :*
     (a) *Peut-on executer le programme main suivant ?*

```
int main()
{ Base obj;
  Derivee obj1;
  cout << obj.GetAtt1();
  cout << obj1.GetAtt1();
}
```

     (b) *Peut-on accéder à la méthode* `GetAttr1()` *ou directement à* `att1` *dans la surcharge de la méthode afficher de la classe dérivée ?*
   * *Même question en dérivant la classe* `Derivee` *de manière privée.*
   * *Même question en dérivant la classe* `Derivee` *de manière protégée.*

7 Manipulation de pointeurs

Exercice 31 *:*
Implémenter une liste chainée d'entiers définie par les deux classes suivantes :

```
// Classe cellule contenant une cellule de la liste chain\'ee
class Cellule
{ friend class ListeChainee;
  int valeur;
  Cellule* suivant;
  public :
   Cellule(int v=0) {valeur =v; suivant =NULL;}; // constructeur
   ~Cellule() {delete suivant;} // destructeur
   Cellule(const Cellule & c) // copy constructeur
   void afficher();
   int GetValeur();
   Cellule* GetSuivant();
   Cellule operator=(const Cellule& c);
   bool operator==(const Cellule &c);
};

class ListeChainee
{ Cellule *premier, *dernier, *courant;
  public:
```

```
   ListeChainee();  // conctructeur sans argument
   ListeChainee(int v);// Constructeur avec un argument
   ~ListeChainee();
   ListeChainee(const ListeChainee&);

   // M\'ethode d\'eplaçant le pointeur courant sur l'\'el\'ement suivant
   Cellule* Suivant() {courant=courant->suivant; return courant;}
   void AfficherListe() const;
   // M\'ethode indiquant si la liste est vied ou non
   bool ListeVide() const;
   // M\'ethode d'insertion d'une cellule de valeur v apr\‘es l'\'el\'ement courant
   void InsererApres(int v);
   // M\'ethode d'insertion d'une nouvelle cellule apr\‘es l'\'el\'ement courant
   void InsererApres(Cellule*);

   ListeChainee& operator=(const ListeChainee &) ;
   ListeChainee& operator+(const ListeChainee &);
   // Op\'erateur d'acc\‘es \‘a l'\'el\'ement de position i
   Cellule *operator[](int i);
};
```

Les exercices 27, 28 et 30 à 33 sont fortement inspirés de ceux de l'ouvrage "Exercices en langage C++" de Claude Delannoy, Eyrolles, 2004.

8 Héritage multiple

Exercice 32 *:*
Soit le fichier ***Exo27.h*** *suivant :*

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

class A
{ int n;
  float x;
  public :
   A(int p=2)
    { n=p; x=1;cout << "** Construction d'un objet A :" << n << " " << x << endl;}
};

class B
{ int n;
  float y;
  public :
   B(float v=0.0)
    { n=1; y=v; cout << "** Construction d'un objet B :" << n << " " << y << endl;}
};

class C : public B, public A
```

```
{ int n;
  int p;
  public :
   C(int n1=1, int n2=2, int n3=3, float v=0.0) : A(n1), B(v)
    { n=n3; p=n1+n2;
      cout << "** Construction d'un objet C :" << n << " " << p << endl;
    }
};
```

Soit le programme principal suivant :

```
int main()
{ C c1;
  C c2(10,11,12,5.0);
}
```

1. *Sans tester sur machine, indiquer ce que va afficher ce programme.*
2. *Vérifier en testant sur machine.*
3. *Mêmes questions avec la première ligne du constructeur de la classe `C` définie de la manière suivante (le corps restant inchangé) :*
   (a) `C(int n1=1, int n2=2, int n3=3, float v=0.0) : B(v)`
   (b) `C(int n1=1, int n2=2, int n3=3, float v=0.0)`

Exercice 33 *:*
Soit le fichier `Exo28.h` suivant :

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

class A
{ int na;
  public :
   A(int nn1=1) { na=nn1; cout << "** Construction d'un objet A : " << na << endl;}
   int GetNa() {return na;}
};

class B : public A
{ float nb;
  public :
   B(float xx=0.0)
    { nb=xx; cout << "** Construction d'un objet B : " << nb << endl;}
};

class C : public A
{ int nc;
  public :
   C(int nn=2) : A(2*nn+1) { nc=nn; cout << "** Construction d'un objet C : " << nc << endl;}
};
```

```
class D : public B, public C
{ int nd;
  public :
   D(int n1, int n2, float x) : B(x), C(n1)
    { nd=n2; cout << "** Construction d'un objet D : " << nd << endl;}
   int GetNaViaB() {return B::GetNa();}
   int GetNaViaC() {return C::GetNa();}
};
```

Soit le programme principal suivant :

```
#include "Exo28.h"

int main()
{ D d (10,20,5.0);
  cout << "Attribut na de A via B : " << d.GetNaViaB() << endl;
  cout << "Attribut na de A via C : " << d.GetNaViaC() << endl;
}
```

1. *Sans tester sur machine, indiquer ce que va afficher ce programme.*
2. *Vérifier en testant sur machine.*
3. *Que faut-il modifier pour que l'attribut* `na` *de la classe* `A` *ne soit pas dédoublé ?*

9 Méthodes virtuelles, fonctions virtuelles pures et classe abstraite

Exercice 34 *:*

1. *Reprendre l'exercice 31 en modifiant le type de l'attribut* `valeur` *de la classe* `Cellule` *en* `void*` *(i.e. pointeur sur un objet quelconque) et en modifiant en conséquence les méthodes* `Cellule::GetValeur()` *et* `Cellule::afficher()`*.*
2. *Modifier le constructeur avec un argument de la classe* `ListeChainee`*, ainsi que la méthode* `ListeChainee::AfficherListe()` *et le destructeur de la classe que vous rendrez virtuels.*
3. *Définir une classe* `Point` *ayant deux attributs entiers* `x` *et* `y`*, un constructeur et une méthode* `affiche()`*.*
4. *Définir une classe* `ListeDePoints`*, dérivée de la classe* `ListeChainee` *où l'attribut* `valeur` *est redéfini comme étant de type Point*.*

Exercice 35 *:*

1. *Définir une classe* `Base` *abstraite ayant une fonction virtuelle pure* `Afficher()`*.*
2. *Définir une classe* `Ens_Heter` *permettant de manipuler un ensemble d'éléments hétérogènes (de différents types), le type de éléments de l'ensemble étant défini comme objet instance d'une classe* `Base`*.*
3. *Ajouter les méthodes suivantes à la classe* `Ens_Heter` *:*
   (a) `void Ajouter(Base&); // Ajoute d'un élément dans l'ensemble à la fin de l'ensemble`

(b) `bool Appartient(Base&); // Teste l'appartenance d'un élément à l'ensemble`

(c) `int GetNbElem() return nbElem; // Retourne le nombre d'éléments dans l'ensemble`

(d) `const Base& Suivant(); // Retroune l'élement courant et déplace la position courante d'un élément` - *Cette méthode pourra retourner un objet bidon constant lorsque la position courante correspond à celle du dernier élément.*

(e) `bool Fin() return (courant==nbElem); // Indique si on a atteint le dernier élément`

(f) `void Afficher(); // Affiche les valeurs des éléments de la liste`

4. *Définir une classe* `Point` *et une classe* `Chaine` *ayant juste un constructeur et une méthode* `Afficher()`.

5. *Tester vos classe avec le programme principal suivant :*

```
int main()
{ Point p(1,3);
  Chaine ch("Bonjour");
  Ens_Heter e;

  cout << "Nombre d'\'el\'em\'ents dans la liste : " << e.GetNbElem() << endl;
  cout << "Contenu de l'ensemble : " << endl; e.Afficher();

  e.Ajouter(p);
  cout << "Nombre d'\'el\'em\'ents dans la liste : " << e.GetNbElem() << endl;
  cout << "Contenu de l'ensemble : " << endl; e.Afficher();

  e.Ajouter(ch);
  cout << "Nombre d'\'el\'em\'ents dans la liste : " << e.GetNbElem() << endl;
  cout << "Contenu de l'ensemble : " << endl; e.Afficher();
}
```

10 Patrons de fonctions

Exercice 36 *:*
Définir un patron de fonctions, prenant en paramètre un tableau de type quelconque et un entier représentant le nombre d'éléments dans le tableau, et permettant d'exécuter le programme principal suivant :

```
int main()
{ int ti[] ={3,5,2,1};
  float tf[] ={2.5,3.2,1.8};
  char tc[] = {'a', 'e', 'i', 'o', 'u'};

  cout << somme(ti,4) << endl;
  cout << somme(tf,3) << endl;
  cout << somme(tc,4) << endl;
}
```

11 Patrons de classes

Exercice 37 *:*
Définir un patron de classes représentant des "vecteurs dynamiques" d'élements de type quelconque et permettant d'exécuter le programme suivant :

```
int main()
{ Vect<int> vi(5);
  int i;
  for(i=0;i<5;i++) vi[i]=i;
  for(i=0;i<5;i++) cout << "vi[" << i <<"]=" << vi[i] << endl;

  Vect<double> vd(2);
  vd[0]=1.2;vd[1]=4.4;
  cout << "vd[0]=" << vd[0] << endl;
  cout << "vd[1]=" << vd[1] << endl;
}
```

12 Gestion des exceptions

Exercice 38 *:*
Reprendre l'exercice 37 et ajouter le contrôle des exceptions permettant de gérer les mauvais indices lors de l'appel de `v[i]` *avec* `i>v.nbElements` *ou les mauvais paramètres passés au constructeur tels que* `Vect v(n);` *avec* `n<=0`*.*

13 Surdéfinition des opérateur << et >>

Exercice 39 *:*
Reprendre l'exercice 18 sur les chaînes de caractères et redéfinir les opérateurs `<<` *et* `>>` *pour cette classe.*

Exercice 40 *:*
Reprendre la classe `Vect` *de l'exercice 25 sur les vecteurs d'entiers dynamiques et redéfinir les opérateurs* `<<` *et* `>>` *pour cette classe.*

14 Sujet d'examen de Janvier 2006

Durée : 1h30 - Aucun document autorisé. Le barème indiqué est approximatif (sur 45). Les exercices 36 à 39 sont repris et adaptés de "C++ - Testez-vous !" de A. Zerdouk, Ellipses, 2004.

Exercice 41 *(5 points soit 11%)*
Parmi les instructions du programme ci-dessous, indiquer celles qui compilent de celles qui ne compilent pas. Vous pouvez uniquement préciser le numéro de la ligne. Justifier.

```
1.    #include <iostream>
2.
3.    int main()
4.     { int tab[] = {1,2,3};
5.       const int *ptab1 = tab;
6.       const int * const ptab2 = tab;
7.       const int ctab[] = {4,5,6};
8.
9.       ptab1[1]=-1;
10.      ptab2[1]=-1;
11.      ctab[1]=-1;
12.
13.       for (int i=0; i<3; i++)
14.       { ptab1++;
15.         ptab2++;
16.       }
17.     }
```

Exercice 42 *(5 points soit 11%)*
Qu'affiche le programme suivant ? Justifier.

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

class A
{ int id;
  public :
   A(int i) : id(i) { cout << "A::A(int) : " << id << endl; }
   A& operator=(const A& obj)
    { if(this!=&obj) id = obj.id;
      cout << "A::operator=(const A&) : " << id << endl;
      return *this;
    }
   ~A() { cout << "A::~A() : " << id << endl; }
};

int main()
{ A obj1(1);
  A obj2=obj1;
}
```

Exercice 43 *(5 points soit 11%)*
Qu'affiche le programme suivant ? Justifier.

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

class A
{ static int nbObjets;
  int id;
  public :
   A() { id=nbObjets; nbObjets++; }
   A(const A& obj) : id(obj.id) { nbObjets++; }
   void afficher()
      { cout << "id : " << id << " - nbObjets : " << nbObjets << endl; }
   ~A() {}
};

void MaFonction(A obj) { obj.afficher(); }
int A::nb = 0;
int main() { A obj; MaFonction(obj); }
```

Exercice 44 *(3 points soit 7%)*
Expliquer pourquoi ce programme ne compile pas et proposer 3 corrections possibles. Justifier.

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

class A
{ int id;
  public :
   A(int i) : id(i) {}
   ~A() {}
};

class B : public A
{ public :
   B() {}
   ~B() {}
};

int main() {}
```

Exercice 45 *(3 points soit 7%)*
Soit la classe `Vecteur`*, définie de la manière suivante :*

```
class Vecteur
{ int nbElements;
  int * Adresse;
  public :
  Vecteur(int taille)
    { if (taille <= 0) { nbElements=0; Adresse = NULL; }
       else { nbElements=taille; Adresse = new int [nbElements]; }
    }
  ~Vecteur() {delete Adresse;}
  int GetNbElements() {return nbElements;}
}
```

1. *Sans les implémenter mais en précisant leur signature, indiquer quelles sont les méthodes et/ou fonctions amies nécessaires pour que le programme principal suivant puisse s'exécuter.*

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

int main()
{ Vecteur v(3) ;
  for(int i=0;i<v.GetNbElements();i++) v[i]=i;
  cout << "Affichage de v : " << v ;
}
```

2. *Comment pouvez-vous interdire à des programmeurs, utilisant cette classe, de réaliser une affectation entre deux objets de la classe* `Vecteur` *(i.e. de réaliser une instruction du type* `v1=v2;`*, les variables* `v1` *et* `v2`*, instances de la classe* `Vecteur`*, ayant été préalablement déclarées).*

Exercice 46 *(10 points soit 22%)*
Soit la classe suivante :

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

class A
{ protected :
   int id;
  public :
   A(int i=0) : id(i) { cout << "A::A(int)" << endl; }
   ~A() { cout << "A::~A()" << endl; }
   int GetId() {return id;}
   void afficher() { cout << "id : " << id << endl; }
};
```

1. *Si l'opérateur* `+` *a été redéfini par une fonction amie de la classe* `A`*, dont le corps est le suivant :*

```
A operator+(const A& obj1, const A& obj2)
 { A obj ; obj.id=obj1.id+obj2.id ; return obj; }
```

Indiquer ce qu'affiche le programme principal suivant. Justifier.

```
int main()
{ A obj, obj2(2), obj3(1);
  obj = obj3+obj2;
  obj3.afficher();
  obj.afficher();
}
```

2. *Même question que précédemment, si l'opérateur* `+` *a été redéfini par une méthode de la classe* `A`*, dont le corps est le suivant :*

```
A& A::operator+(const A& obj)
 { this->id+=obj.id; return *this; }
```

3. *Si on ajoute la définition de classe suivante :*

```
class B : private A
{ char c;
  public :
   B(char cara, int i=0) : A(i) { c=cara; }
   ~B() {}
   B(const B & obj) { c=obj.c; }
   void afficher() { cout << "objet :" << c << id << endl; }
};
```

   (a) *Que va afficher le programme suivant ? Justifier.*

```
int main()
{ B obj('a',1);
  B obj1=obj;
  obj1.afficher();
}
```

   (b) *Même question que précédemment, mais en supposant qu'on a redéfini la constructeur par recopie de la classe* `A` *de la manière ci-dessous. Justifier.*

```
A::A(const A& obj)
 { if(this!=&obj) id=obj.id; return *this; }
```

   (c) *Que faut-il faire pour que l'objet* `obj1` *soit correctement initialisé ? Justifier.*

4. *Le programme suivant compile-t-il ? Justifier.*

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

int main()
 { B obj('a');
   cout << obj.GetId() ;
 }
```

Exercice 47 *(14 points soit 31%)*
Soit le programme suivant :

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

class A
{ int id1;
  public :
   A(int i=0) : id1(i) { cout << "A::A(int)" << endl; }
   ~A() { cout << "A::~A()" << endl;}
   A(const A & obj) { id1=obj.id1; cout << "A:A(const A &)" << endl; }
   void afficher() { cout << "id1 : " << id1 << endl; }
   int GetId1() {return id1;}
};

class B : public A
{ int id2;
  public :
   B(int i=0, int j=0) : A(i), id2(j) { cout << "B::B(int,int)" << endl; }
   ~B() { cout << "B::~B()" << endl; }
   void afficher() { A::afficher() ; cout << "id2 : " << id2 << endl; }
};

int main()
{ B obj;
  A *p =&obj;
  p->afficher();
}
```

1. *Que va afficher ce programme ? Justifier.*

2. *Que faut-il faire, sans modifier le programme* ***main****, pour que l'affichage se fasse correctement. Justifier.*

3. *On suppose qu'on a défini le patron de fonctions suivant :*

```
template <typename T> T Maximum(T var1, T var2)
  { if(var1>var2) return var1; else return var2; }
```

   (a) *Proposer, sans les implémenter, 2 solutions pour que l'on puisse exécuter le programme suivant. Expliquer.*

```
int main()
{ A obj1, obj2(3);
  Maximum(obj1,obj2).afficher();
}
```

   (b) *Quelles que soient les solutions proposées à la question précédente, que va afficher le programme principal précédent ? Justifier.*

15 Sujet d'examen de Juillet 2009

Durée : 1h30 - Aucun document autorisé. Le barème indiqué est approximatif (sur 80).

On supposera que tous les programmes du sujet débutent par les instructions suivantes :

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

Exercice 48 *(13 points – 16%)*

1. *Quelles sont les méthodes que toute classe C++ a par défaut ?*
2. *En prenant l'exemple de la classe* `C`*, définie ci-dessous, redefinissez chacune de ces méthodes en faisant en sorte qu'elles fassent exactement ce qu'elles font par défaut.*

```
class C
 { int * tab;
   int nb;
 };
```

Exercice 49 *(27 points – 34%)*
Soit le programme suivant :

```
class A
{ public:
   A() { cout << "A()" << endl; }
   A(int b) { cout << "A(int b)" << endl; }
   A(const A& obj) { cout << "A(const A& obj)" << endl; }
   A& operator=(const A& obj) { cout << "A operator=(const A& obj)" << endl; }
   ~A() { cout << "~A()" << endl; }
   void afficher() {cout << "Affichage" << endl; }
};

void f1(A obj) { cout << "f1" << endl; }

void f2(A& obj) { cout << "f2" << endl; }

A f3(A obj) { A obj1; cout << "f3" << endl; return obj1; }

int main()
{
  A o1, o2(2), o3=o1, *p1, *p2;   // ligne 1
  p1 = &o1;                       // ligne 2
  p2 = new A(2);                  // ligne 3
  delete p2;                      // ligne 4
  p2 = p1;                        // ligne 5
  o2 = o3;                        // ligne 6
  f1(o1);                         // ligne 7
  f2(o2);                         // ligne 8
  f3(o1).afficher();              // ligne 9
}
```

Indiquer quels vont être les différents affichages. Vous préciserez clairement quel est l'affichage associé à chacune des instructions du programme (vous pouvez utiliser les numéros des lignes mis en commentaire dans le ***main****).*

Exercice 50 *(4 points – 5%)*
Soit le programme suivant :

```
class A
{ int a ;
  public:
   A(int a) { cout << "A( " << a << " )" << endl; this->a=a; }
};

int main()
{ A obj1(1);
  A obj2;
}
```

Le programme compile-t-il ?

- *Si oui, indiquez ce qu'il va afficher.*
- *Si non, indiquez deux solutions permettant que le programme compile sans avoir à changer le* ***main****.*

Exercice 51 *(13 points – 16%)*
Soient les classes suivantes :

```
class A
{ int a;
  public:
    A(int i) : a(i) { cout << "A::A(int i=" << i << ")" << endl; }
    ~A() { cout << "A::~A()" << endl; }
    void afficher() { cout << "a = " << a << endl;}
};

class B : public A
{ int b;
   public:
     B(int i, int j) : A(i), b(j)
       { cout << "B::B(int i=" << i << ", int j=" << j << ")" << endl; }
    ~B() { cout << "B::~B()" << endl; }
    void afficher() { A::afficher(); cout << "b = " << b << endl;}
};
```

1. *Qu'affiche chacune des instructions du programme ci-dessous ? (Vous pouvez utiliser les numéros des lignes mis en commentaire dans le* ***main****.)*

```
int main()
{ A *obj = new A(3);   // ligne 1
  obj->afficher();     // ligne 2
  delete obj;          // ligne 3

  obj= new B(4,5);     // ligne 4
```

```
        obj->afficher();     // ligne 5
        delete obj;          // ligne 6
      }
```

2. *Rappeler la définition de la liaison dynamique.*
3. *Indiquer si la liaison dynamique est mise en œuvre dans le programme précédent. Si oui, expliquer pourquoi. Si non, indiquer ce qu'il faut modifier pour qu'elle le soit et quelles seront alors les modifications au niveau de l'affichage du programme principal.*

Exercice 52 *(9 points – 11%)*
En reprenant la classe `A` *de l'exercice précédent et en ajoutant la redéfinition de l'opérateur* `<<` *par une fonction amie de la classe* `A` *de la forme :*

```
friend ostream& operator << (ostream& os, const A& obj)
 { os << "a=" << obj.a << endl;
   return os;
 }
```

Soit la classe `TabDeA` *définie de la manière suivante :*

```
class TabDeA
 { A** tab;
   int nbElements;

   public:
    TabDeA()
     { tab=NULL;
       nbElements=0;
     }

    TabDeA(int taille) : nbElements(taille)
     { tab= new A*[nbElements];
       for (int i=0; i< nbElements ; i++) tab[i] = new A(i);
     }

    void erase()
     { for (int i=0; i< nbElements ; i++) delete tab[i];
       delete tab;
       tab=NULL;
       nbElements=0;
     }

    ~TabDeA()
     { cout << "TabDeA::~TabDeA() " << endl;
       erase();
     }

    void Afficher() const
     { if (tab!=NULL)
        for (int i=0; i< nbElements ; i++) cout << (*tab[i]) ;
     }
};
```

1. *Le programme ci-dessous compile mais crée une erreur (segmentation fault) à l'exécution. Quelle en est d'après-vous la cause ? Expliquer.*

```
int main()
{  TabDeA obj(4);
   obj.Afficher();

   TabDeA *obj1 = new TabDeA(obj);
   obj1->Afficher();
   delete obj1;

   obj.Afficher();
}
```

2. *Quelle(s) méthode(s) faut-il définir ou redéfinir pour que le programme précédent s'exécute correctement ? - Vous donnerez le corps de la (ou les) méthode(s) nécessaire(s).*

Exercice 53 *(14 points – 18%)*
On suppose que l'on a défini la classe suivante :

```
class Erreur : public exception
{ int num;

  public:
   Erreur(int i) : num(i) {}

   const char * what() const throw()
    { if(num==1) return ("Mauvais indice!!\n");
        else if(num==2) return ("Mauvaise taille de vecteur!!\n");
          else return "Exception inconnue!\n";
    }
};
```

Définir un patron (template) de classes représentant des "vecteurs dynamiques" et permettant d'exécuter le programme suivant :

```
int main()
{
 try {
       int taille, indice;

       cout << "Taille du vecteur:" << endl;
       cin >> taille;

       Vect<int> vi(taille);
       int i;
       for(i=0;i<taille;i++) vi[i]=i;
       cout << "Indice de l'\'el\'ement \`a afficher:" << endl;
       cin >> indice;
       cout << "vi[" << indice << "]=" << vi[indice] << endl;

       Vect<char> vc(taille);
       for(char j='a', i=0; i<taille ; i++, j++) vc[i]=j;
       cout << "Indice de l'\'el\'ement \`a afficher:" << endl;
       cin >> indice;
       cout << "vc[" << indice << "]=" << vc[indice] << endl;
     }

  catch(Erreur e)
   { cout << e.what() << endl; exit(1);}

  return 0;
}
```

Rappel : un patron de classe se définit par `template <class T> class Vect`.

16 Sujet d'examen d'Avril 2012

On supposera que tous les programmes C++ du sujet dèbutent par les instructions suivantes :

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

Exercice 54 *(20 points)*

1. *Quelles sont les mèthodes que toute classe C++ a par dèfaut ?*
2. *En prenant l'exemple de la classe* `C`*, dèfinie ci-dessous, redèfinissez chacune de ces mèthodes en faisant en sorte qu'elles fassent exactement ce qu'elles font par dèfaut.*

```
class C
 { int * tab;
   int nb;
 };
```

3. *Quel(s) problème(s) peuvent poser ces mèthodes par dèfaut ? Donnez des exemples.*
4. *Redèfinissez ces mèthodes de telle sorte que les problèmes èvoquès prècèdemment n'apparaissent plus.*

Exercice 55 *(5 points)*
Expliquer pourquoi ce programme ne compile pas et proposer 3 corrections possibles. Justifier.

```
class A
{ int id;
  public :
   A(int i) : id(i) {}
   ~A() {}
};

class B : public A
{ public :
   B() {}
   ~B() {}
};

int main() {}
```

Exercice 56 *(20 points)*
Soit le programme suivant :

```
class A
{ public:
   A() { cout << "A()" << endl; }
   A(int b) { cout << "A(int b)" << endl; }
   A(const A& obj) { cout << "A(const A& obj)" << endl; }
   A& operator=(const A& obj) { cout << "A operator=(const A& obj)" << endl; }
   ~A() { cout << "~A()" << endl; }
   void afficher() {cout << "Affichage" << endl; }
};

void f1(A obj) { cout << "f1" << endl; }

void f2(A& obj) { cout << "f2" << endl; }

A f3(A obj) { A obj1; cout << "f3" << endl; return obj1; }

int main()
{
  A o1, o2(2), o3=o1, *p1, *p2;   // ligne 1
  p1 = &o1;                       // ligne 2
  p2 = new A(2);                  // ligne 3
  delete p2;                      // ligne 4
  p2 = p1;                        // ligne 5
  o2 = o3;                        // ligne 6
  f1(o1);                         // ligne 7
  f2(o2);                         // ligne 8
  f3(o1).afficher();              // ligne 9
}
```

Indiquer quels vont être les diffèrents affichages et pourquoi ils ont lieu. Vous prèciserez clairement quel est l'affichage associè à chacune des instructions du programme (vous pouvez utiliser les numèros des lignes mis en commentaire dans le ***main****).*

Exercice 57 *(10 points)*
Soient les classe ***A*** *et* ***B*** *dèfinies de la manière suivante :*

```
class A
{ int a;
  public:
   A(int i=0) : a(i) { cout << "A::A(int i=" << i << ")" << endl; }
   ~A() { cout << "A::~A()" << endl;}
   int GetA() const { return a; }
   friend ostream& operator << (ostream& os, const A& obj)
    { os << "a=" << obj.a << endl;
      return os;
    }
};
```

```
class B : public A
{ int b;
  public:
  B(int j) : b(j) { cout << "B::B(int j=" << j << ")" << endl; }
  ~B() { cout << "B::~B()" << endl; }
};
```

1. *Qu'affiche le programme suivant :*

```
int main()
{
  B obj(4);
}
```

2. *On souhaite ajouter au programme la fonction ci-dessous :*

```
ostream& operator << (ostream& os, const B& obj)
{ const A* obj1=&obj;
  os << (*obj1) << "b=" << obj.b << endl;
  return os;
}
```

Mais la compilation de cette fonction crèe l'erreur suivante :

```
>g++ exo.cpp
exo.cpp: In function ‘std::ostream& operator<<(std::ostream&, const B&)’:
exo.cpp:18: ‘int B::b’ is private
exo.cpp:28: within this context
```

Proposer 3 solutions permettant de corriger l'erreur.

3. *En supposant qu'on ait ajoutè dans chacune des classes `A` et `B`, la mèthode suivante :*

```
void afficher() const
  { cout << (*this) ; }
```

(a) *Qu'affiche le programme ci-dessous ?*

```
int main()
{ A *obj = new A(3);
  obj->afficher();
  delete obj;

  obj= new B(5);
  obj->afficher();
  delete obj;
}
```

(b) *Rappeler la dèfinition de la liaison dynamique.*

(c) *Indiquer si la liaison dynamique est mise en œuvre dans le programme prècèdent. Si oui, expliquer pourquoi. Si non, indiquer ce qu'il faut modifier pour qu'elle le soit.*

Exercice 58 *(9 points)*
On suppose que l'on a dèfini la classe suivante :

```
class Erreur : public exception
{ int num;

  public:
   Erreur(int i) : num(i) {}

   const char * what() const throw()
    { if(num==1) return ("Mauvais indice!!\n");
        else if(num==2) return ("Mauvaise taille de vecteur!!\n");
          else return "Exception inconnue!\n";
    }
};
```

Dèfinir un patron (template) de classes reprèsentant des "vecteurs dynamiques" et permettant d'exècuter le programme suivant :

```
int main()
{ try { int taille, indice;

        cout << "Taille du vecteur:" << endl;
        cin >> taille;

        Vect<int> vi(taille); int i;
        for(i=0;i<taille;i++) vi[i]=i;
        cout << "Indice de l'\‘el\‘ement \‘a afficher:" << endl;
        cin >> indice;
        cout << "vi[" << indice << "]=" << vi[indice] << endl;

        Vect<char> vc(taille);
        for(char j='a', i=0; i<taille ; i++, j++) vc[i]=j;
        cout << "Indice de l'\‘el\‘ement \‘a afficher:" << endl;
        cin >> indice;
        cout << "vc[" << indice << "]=" << vc[indice] << endl;
     }

  catch(Erreur e)
   { cout << e.what() << endl; exit(1);}

  return 0;
}
```

Rappel : un patron de classe se dèfinit par `template <class T> class Vect`.

Exercice 59 *(11 points)*
On suppose que l'on a dèfini la classe suivante et le programme principal suivants :

```
class MaClasse
{
  int att;

  public:
   MaClasse(int att) {this->att=att;}
   void setAtt(int att) {this->att=att;}
   int getAtt() {return att;}
   void echanger(MaClasse o1);  //ligne 1
   void additionner(MaClasse o1);  //ligne 2
};

void MaClasse::echanger(MaClasse o1) //ligne 3
{
  int temp;
  temp = o1.getAtt();//ligne 4
  o1.setAtt(att); //ligne 5
  att=temp;
}

void MaClasse::additionner(MaClasse o1) //ligne 6
{
  att+=o1.getAtt();//ligne 7
}

int main()
{
  MaClasse o3(3), o4(4);
  o3.echanger(o4); //ligne 8
  cout << "Attribut a de o3 : "<< o3.getAtt() << endl;
  cout << "Attribut a de o4 : "<< o4.getAtt() << endl;
  o3.additionner(o4); //ligne 9
  cout << "Attribut a de o3 : "<< o3.getAtt() << endl;
  cout << "Attribut a de o4 : "<< o4.getAtt() << endl;
}
```

1. *Qu'affiche le programme ci-dessous ? bExpliquer*
2. *Pour chacune des mèthodes pour lesquelles vous pensez que le fonctionnement est incorrect : proposer 2 solutions diffèrentes – Attention : vous n'avez le droit de modifier que les lignes numèrotèes de 1 à 9 (ayant un commentaire de la forme* `//ligne` *).*

Exercice 60 *(7 points)*
En reprenant la classe de l'exercice prècèdent :

1. *Quel est l'èquivalent de la mèthode* `MaClasse(int att)` *si on avait crèer la classe en Python ?*
2. *Quel est l'èquivalent du mot clè C++* `this` *en Python ?*
3. *Quelle mèthode/fonction faut-il redèfinir en C++ et en Python pour permettre d'exècuter l'instruction* `o1+o2`*,* `o1` *et* `o2` *ètant des objets de la classe* `MaClasse` *? Vous prèciserez la signature et le corps de la mèthode ou fonction, dans les deux langages.*
4. *Même question que prècèdemment, pour l'instruction C++* `cout << o1 ;` *et l'instruction Python* `print o1`*.*
5. *Comment faire en python pour que l'attribut de la classe soit privè ?*

Exercice 61 *(13 points)*
Soit le programme[1] *suivant Python :*

```
#! /usr/bin/env python
# -*- coding: Latin-1 -*-

def eleMax(lst, debut =0, fin =-1):
    "renvoie le plus grand \‘el\‘ement de la liste lst"
    if fin == -1:
        fin = len(lst)
    max, i = 0, 0
    while i < len(lst):
        if i >= debut and i <= fin and lst[i] > max:
            max = lst[i]
        i = i + 1
    return max

# test :
serie = [9, 3, 6, 1, 7, 5, 4, 8, 2]
print eleMax(serie)
print eleMax(serie, 2)
print eleMax(serie, 2, 5)
```

1. *Indiquer ce que va afficher le programme.*
2. *Proposer un programme èquivalent en C++.*

Astuce : Le programme C++ suivant[2] *peut vous aider :*

```
int main ()
{
  list<int> liste;  // empty list of ints
  // the iterator constructor can also be used to construct from arrays:
  int myints[] = {16,2,77,29};
  list<int> liste2; (myints, myints + sizeof(myints) / sizeof(int) );
  cout << "size: " << (int) liste2.size() << endl;
```

1. Repris de Apprendre à programmer avec Python de Gèrard Swinnen, 2009
2. Repris de `www.cplusplus.com`

```
cout << "The contents of lists are: ";
for (liste<int>::iterator it = liste.begin(); it != liste.end(); it++)
  cout << *it << " ";
```