TD 4 : Classes
Algorithmie
Master EEA 1ère année 2006

1. Introduction

1.1. Edition du fichier

Afin d'écrire votre script C++, vous pouvez utiliser n'importe quel éditeur de texte, cependant xemacs ou nedit paraissent être un bon choix. On lance l'éditeur à partir d'une console (« nedit & »).
Le fichier aura une extension .cpp.

1.2. Compilation

Ouvrez un terminal et placez-vous dans le répertoire dans lequel vous avez enregistré votre script C. Afin de compiler votre script, utiliser la commande :
g++ nom_du_fichier.cpp -o nom_executable

Ce qui donne, par exemple, si votre script du exo1 se nomme exo1.cpp et que vous voulez que votre exécutable se nomme exo1:
g++ exo1.cpp -o exo1

Le compilateur génère alors un fichier exo1 que vous pouvez exécuter en utilisant la commande ./exo1

1.3. En tête de fichier

L'en-tête du fichier d'extension « .cpp » commencera par :
#include <iostream>
using namespace std;

La première ligne fait appel à la librairie relative aux fonctions d'entrées sorties.
La seconde ligne définit au compilateur que l'on va utiliser les librairies standard du langage C++.

2. Exercice 1

```
//Fonction utilisant la transmission par valeur
void echange(int x, int y)
{
int temp=y ;
y=x ;
x=temp ;
}
int main()
{
int x=3, y=4 ;
cout << " Avant l'appel de la fonction echange »<<endl;
```

```
cout << " x= " << x << ", y= "<<y <<endl;                    (1)
echange(x,y);
cout << " Après l'appel de la fonction echange »<<endl;
cout << " x= " << x << ", y= "<<y <<endl;                    (2)
}
```

1. Qu'est ce que ce programme va afficher ?
2. Réécrire la fonction précédente en passant les arguments de la fonction par référence.
3. Que va t il s'afficher en (2) ?

L'appel de fonction utilisant la transmission par référence est ambiguë parce qu'il est identique à la transmission par valeur (qui lui ne peut pas changer la valeur)

3. Exercice 2

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

const double MEASURE_MAX = 10;                               (1)
const double MEASURE_MIN = 0;                                (2)

class CDevice {
private:
        int Id;
        double meas;
public:
        CDevice ();
        CDevice (int id);
        bool setValue(double val);
        CDevice operator+ (const CDevice &a);
};

CDevice::CDevice (){
        Id = 1;
        meas = 0;
}

CDevice::CDevice (int Id) {
        this->Id = Id;
        this->meas = 0;
}

bool CDevice::setValue(int val) {
        if ((val < MEASURE_MIN) || (val > MEASURE_MAX))
                return false;
        this->meas = (double) val;
        return true;
}

int main(int argc, char* argv[])
{
        CDevice dev1;                                                (3)
```

```
        CDevice dev2 = CDevice(2);                                  (4)
        …
}
```

1. Qu'est ce que la ligne (1) et (2) déclare ? Quelle est sa particularité ?
2. Comment appelle-t-on la méthode CDevice() (3) ? Qu'a-t-elle de particulier ?
3. Comment appelle-t-on la méthode CDevice (int Id) (4) ?
4. Qu'est ce que this ? Est-il un pointeur ou une variable ?
5. Quelle est la particularité d'un paramètre d'une classe déclaré en statique ?
6. Comment implémenter un membre nbDevice pour connaître le nombre de device existant ?
7. Qu'est ce qu'un destructeur ? Comment l'implémente-t-on ?
8. Comment l'utiliser pour connaître le nombre de device ?

4. Exercice 3

```
#include "stdafx.h"
#include <iostream>
using namespace std;

class CVect {
private:
        int nbElt;
        double *adr;
public:
        CVect(int nb);
        ~CVect ();
        void affiche();
};

CVect::CVect(int nb) {
        nbElt  = nb;
        adr = new double[nb];
        cout<<"constructeur"<<endl;
}

CVect::~CVect () {
        delete adr;
        cout<<"destruction"<<endl ;
}

void CVect::affiche() {
        cout<<"hello"<<endl ;
}
int main(int argc, char* argv[])
{
        CVect a(5);
        CVect *b = new CVect(3);

        delete b;
        a.affiche();
```

```
        return 0;
}
```

1. Qu'est ce que ce programme va afficher ?

5. Exercice 4

Cet exercice est la suite du TD précédent.
Le but de cet exercice est de manipuler une classe CComplexe (z = a+ib).

1. Déclarer la classe CComplexe dans un fichier « complexe.h » constituée de ses paramètres a et b, ainsi que de son constructeur par défaut.
2. Créer un fichier « complexe.cpp » dans lequel vous implémentez le constructeur par défaut. On affichera à l'écran dans cette fonction la valeur d'un moins un des paramètres de la classe.
3. Créer un fichier « exo4.cpp » contenant la fonction main dans lequel on déclare un objet z de la classe CComplexe.
   La ligne de commande pour compiler et générer l'exécutable est :
   g++ exo4.cpp complexe.cpp –o exo4
4. Ajouter un constructeur permettant d'initialiser les valeurs de a et b (pouvant être différents de 0) en les passants comme paramètres.
5. Créez une méthode statique (ayant comme paramètre un objet CComplexe) qui affiche à l'écran un nombre complexe sous la forme partie imaginaire et partie réelle.
6. Définir une fonction qui calcule la somme de deux nombres complexes.
7. Définir une fonction qui détermine le produit de deux nombres complexes.