Université AbdelMalek Essaâdi
Faculté des Sciences et Techniques
Département Génie Informatique
Tanger

Année Universitaire : 2020/2021.
Concepts de Base de POO & POO en C++
(Filière: *LST-GI (***S5**), Gr2 : G21 et G22)

Pr. E. M. EN-NAIMI & Dr. N. EL GHOUCH

Séries de TDs/TPs N° : (1, 2 & 3)

Exercice 1 :

Tester cet exemple et s'il y a des erreurs essayer de les corriger ! :

```
#include <iostream.h>
#include <conio.h>
void main()
{
int i,n=30,*pt;
char *ch="Essayer la  cout !";
float x = 345.678;
cout<<"Salut tout le monde \n";
cout<<ch<<"\n";
cout<<"Bonjour Cher(e)s Etudiants CI-LSI_1\n"<<ch<<"\n";
cout<<"n= "<<n<<"  x= "<<x<<"  pt= "<<pt<<"\n";
getchar() ;
}
```

Exercice 2 :

Tester cet exemple et s'il y a des erreurs essayer de les corriger ! :

```
#include <iostream.h>
#include <conio.h>
void main()  //il faut remplacer void par int (en c++) car la fct main retourne une valeur  entiere
{
int n;
char ch[20],c; float x;
cout<<"Entrer Une Valeur  entiere:";
cin>>n;
cout<<"Entrer une valeur  reelle:";
cin>>x;
cout<<"Entrer  une phrase:";
cin>>tc;
cout<<"Entrer  un caractere:";
cin>>c;
cout<<"Relecture: "<<n<<" "<<x<<" "<<ch<<" "<<c<<"\n";
 getchar() ;
}
```

Exercice 3 :

Quels résultats fournit le programme suivant :

```
#include <iostream>
using namespace std ;
main ()
{
int i, j, n ;
i = 0 ; n = i++ ;
cout << "A : i = " << i << " n = " << n << "\n" ;
i = 10 ; n = ++ i ;
cout << "B : i = " << i << " n = " << n << "\n" ;
i = 20 ; j = 5 ; n = i++ * ++ j ;
cout << "C : i = " << i << " j = " << j << " n = " << n << "\n" ;
i = 15 ; n = i += 3 ;
cout << "D : i = " << i << " n = " << n << "\n" ;
i = 3 ; j = 5 ; n = i *= --j ;
cout << "E : i = " << i << " j = " << j << " n = " << n << "\n" ;
}
```

Exercice 4 :

Quels résultats fournira ce programme :

```
#include <iostream>
using namespace std ;
main()
{
int n=10, p=5, q=10, r ;
r = n == (p = q) ;
cout << "A : n = " << n << " p = " << p << " q = " << q
<< " r = " << r << "\n" ;
n = p = q = 5 ;
n += p += q ;
cout << "B : n = " << n << " p = " << p << " q = " << q << "\n" ;
q = n < p ? n++ : p++ ;
cout << "C : n = " << n << " p = " << p << " q = " << q << "\n" ;
q = n > p ? n++ : p++ ;
cout << "D : n = " << n << " p = " << p << " q = " << q << "\n" ;
}
```

Exercice 5 :

Soient les déclarations suivantes :

```
int n = 5, p = 9 ;
int q ;
float x ;
```

Quelle est la valeur affectée aux différentes variables concernées par chacune des instructions suivantes ?

```
q = n < p ; /* 1 */
q = n == p ; /* 2 */
q = p % n + p > n ; /* 3 */
x = p / n ; /* 4 */
x = (float) p / n ; /* 5 */
x = (p + 0.5) / n ; /* 6 */
x = (int) (p + 0.5) / n ; /* 7 */
q = n * (p > n ? n : p) ; /* 8 */
q = n * (p < n ? n : p) ; /* 9 */
```

Exercice 6 :

Quelles erreurs ont été commises dans chacun des groupes d'instructions suivants :

1.
```
if (a<b) cout << "ascendant"
else cout << "non ascendant" ;
```
2.
```
 int n ;
 ...
 switch (2*n+1)
 { case 1 : cout << "petit" ;
 case n : cout << "moyen" ;
 }
```
3.
```
const int LIMITE=100
 int n ;
 ...
 switch (n)
 { case LIMITE-1 : cout << "un peu moins" ;
 case LIMITE : cout << "juste" ;
 case LIMITE+1 : cout << "un peu plus" ;
 }
```

Exercice 7 :

Soit le programme suivant :

```
#include <iostream>
main()
{ int n ;
cin >> n ;
switch (n)
{ case 0 : cout << "Nul\n" ;
case 1 :
case 2 : cout << "Petit\n" ;
break ;
case 3 :
case 4 :
case 5 : cout << "Moyen\n" ;
default : cout << "Grand\n" ;
}
}
```

Quels résultats affiche-t-il lorsqu'on lui fournit en donnée :

a. 0
b. 1
c. 4
d. 10
e. -5

Exercice 8 :

Soit le petit programme suivant :

```
#include <iostream>
using namespace std ;
main()
{ int i, n, som ;
som = 0 ;
for (i=0 ; i<4 ; i++)
{ cout << "donnez un entier " ;
cin >> n ;
som += n ;
}
cout << "Somme : " << som ;
}
```

Écrire un programme réalisant exactement la même chose, en employant, à la place de l'instruction for :

a. une instruction while,
b. une instruction do ... while.

Exercice 9 :

Quels résultats fournit le programme suivant :

```
#include <iostream>
using namespace std ;
main()
{ int n=0 ;
do
        { if (n%2==0) { cout << n << " est pair\n" ;
n += 3 ;
continue ; }
if (n%3==0) { cout << n << " est multiple de 3\n" ;
n += 5 ;       }
if (n%5==0) { cout << n << " est multiple de 5\n" ;
break ;        }
n += 1 ; }
while (1) ;
}
```

Exercice 10 :

Quels résultats fournit le programme suivant :

```
#include <iostream>
using namespace std ;
main()
{ int n, p ;
n=0 ;
while (n<=5) n++ ;
cout << "A : n = " << n << "\n" ;
n=p=0 ;
while (n<=8) n += p++ ;
cout << "B : n = " << n << "\n" ;
n=p=0 ;
while (n<=8) n += ++p ;
cout << "C : n = " << n << "\n" ;
n=p=0 ;
while (p<=5) n+= p++ ;
cout << "D : n = " << n << "\n" ;
n=p=0 ;
while (p<=5) n+= ++p ;
cout << "E : n = " << n << "\n" ;
}
```

Exercice 11 :

Quels résultats fournit le programme suivant :

```
#include <iostream>
using namespace std ;
main()
{ int i, n ;
for (i=0, n=0 ; i<5 ; i++) n++ ;
cout << "A : i = " << i << " n = " << n << "\n" ;
for (i=0, n=0 ; i<5 ; i++, n++) {}
cout << "B : i = " << i << " n = " << n << "\n" ;
for (i=0, n=50 ; n>10 ; i++, n-= i ) {}
cout << "C : i = " << i << " n = " << n << "\n" ;
for (i=0, n=0 ; i<3 ; i++, n+=i,
cout << "D : i = " << i << " n = " << n << "\n" ) ;
cout << "E : i = " << i << " n = " << n << "\n" ;
}
```

Exercice 12 :

Écrire un programme qui calcule les racines carrées de nombres fournis en donnée. Il s'arrêtera lorsqu'on lui fournira la valeur 0.
Il refusera les valeurs négatives. Son exécution se présentera ainsi :

```
donnez un nombre positif : 2
sa racine carrée est : 1.414214e+00
donnez un nombre positif : -1
svp entrer un nombre positif !
donnez un nombre positif : 5
sa racine carrée est : 2.236068e+00
donnez un nombre positif : 0
```

Rappelons que la fonction sqrt fournit la racine carrée (double) de la valeur (double) qu'on lui donne en argument.

Exercice 13 :

Afficher un triangle isocèle formé d'étoiles. La hauteur du triangle (c'est-à-dire le nombre de lignes) sera fourni en donnée, comme dans l'exemple ci-dessous. On s'arrangera pour que la dernière ligne du triangle s'affiche sur le bord gauche de l'écran.
Combien de lignes ? 10

```
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
```

Exercice 14 :

Écrire un programme qui affiche la « table de multiplication » des nombres de 1 à 10, sous la forme suivante :

```
    I  1  2  3  4  5  6  7  8  9  10
-----------------------------------------------
1   I  1  2  3  4  5  6  7  8  9  10
2   I  2  4  6  8  10 12 14 16 18 20
3   I  3  6  9  12 15 18 21 24 27 30
4   I  4  8  12 16 20 24 28 32 36 40
5   I  5  10 15 20 25 30 35 40 45 50
6   I  6  12 18 24 30 36 42 48 54 60
7   I  7  14 21 28 35 42 49 56 63 70
8   I  8  16 24 32 40 48 56 64 72 80
9   I  9  18 27 36 45 54 63 72 81 90
10  I  10 20 30 40 50 60 70 80 90 100
-----------------------------------------------
```

Exercice 15 :

Soient les déclarations (C++) suivantes :

```
int fct (int) ; // fonction I
int fct (float) ; // fonction II
void fct (int, float) ; // fonction III
void fct (float, int) ; // fonction IV
int n, p ;
float x, y ;
char c ;
double z ;
```

Les appels suivants sont-ils corrects et, si oui, quelles seront les fonctions effectivement appelées et les conversions éventuellement mises en place ?

a. fct (n) ;
b. fct (x) ;
c. fct (n, x) ;
d. fct (x, n) ;
e. fct (c) ;
f. fct (n, p) ;
g. fct (n, c) ;
h. fct (n, z) ;
i. fct (z, z) ;

Exercice 16 :

1) Écrire plus simplement en C++ les instructions suivantes, en utilisant les opérateurs **new** et **delete** :

```
int * adi ;
double * add ;
.....
adi = malloc (sizeof (int) ) ;
add = malloc (sizeof (double) * 100 ) ;
```

2) Écrire plus simplement en C++, en utilisant les spécificités de ce langage, les instructions C suivantes :

```
double * adtab ;
int nval ;
.....
printf ("combien de valeurs ? ") ;
scanf ("%d", &nval) ;
adtab = malloc (sizeof (double) * nval) ;
```

3) Déclarer un tableau monodimensionnel, appelé Tab1, de 10 réels, puis calculer et afficher leur moyenne ?

Exercice 17 :

a. Transformer le programme suivant pour que la fonction fct devienne une fonction en ligne.

```
#include <iostream>
using namespace std ;
main()
{
int fct (char, int) ; // déclaration (prototype) de fct
int n = 150, p ;
char c = 's' ;
p = fct ( c , n) ;
cout << "fct (\'" << c << "\', " << n << ") vaut : " << p ;
}
int fct (char c, int n) // définition de fct
{
int res ;
if (c == 'a') res = n + c ;
else if (c == 's') res = n - c ;
else res = n * c ;
return res ;
}
```

b. Comment faudrait-il procéder si l'on souhaitait que la fonction fct soit compilée séparément ?

Exercice 18 : (*Exercice Facultatif* !)

I) Réaliser une classe point permettant de manipuler un point d'un plan. On prévoira :
• un constructeur recevant en arguments les coordonnées (float) d'un point ;
• une fonction membre deplace effectuant une translation définie par ses deux arguments (float) ;
• une fonction membre affiche se contentant d'afficher les coordonnées cartésiennes du point.
Les coordonnées du point seront des membres données privés.
On écrira séparément :
• un fichier source constituant la déclaration de la classe ;
• un fichier source correspondant à sa définition.
Écrire, par ailleurs, un petit programme d'essai (main) déclarant un point, l'affichant, le déplaçant et l'affichant à nouveau.

II) Réaliser une classe point, analogue à la précédente, mais ne comportant pas de fonction affiche. Pour respecter le principe d'encapsulation des données, prévoir deux fonctions membre publiques (nommées abscisse et ordonnée) fournissant en retour l'abscisse et l'ordonnée d'un point. Adapter le petit programme d'essai précédent pour qu'il fonctionne avec cette nouvelle classe.

III) Ajouter à la classe précédente (comportant un constructeur et trois fonctions membre deplace, abscisse et ordonnee) de nouvelles fonctions membre :
• homothetie qui effectue une homothétie dont le rapport est fourni en argument;
• rotation qui effectue une rotation dont l'angle est fourni en argument;
• rho et theta qui fournissent en retour les coordonnées polaires du point.

IV) Modifier la classe point précédente, de manière que les données (privées) soient maintenant les coordonnées polaires d'un point, et non plus ses coordonnées cartésiennes. On évitera de modifier la déclaration des membres publics, de sorte que l'interface de la classe (ce qui est visible pour l'utilisateur) ne change pas.

V) Soit la classe point créée dans l'exercice (I), ci-dessus , dont la déclaration était la suivante :

```
class point
{ float x, y ;
public :
point (float, float) ;
void deplace (float, float) ;
void affiche () ;
}
```

Adapter cette classe, de manière que la fonction membre affiche fournisse, en plus des coordonnées du point, le nombre d'objets de type point.