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

Année Universitaire : 2024/2025
POO en C++ & JAVA
Filière Ingénieure: LSI, (**S1**)
Pr. E. M. EN-NAIMI

Séries de TDs/TPs N° : (09&10)

Exercice 1:

1) Créer un patron de fonctions permettant de calculer le carré d'une valeur de type quelconque (le résultat possédera le même type). Écrire un petit programme utilisant ce patron.
2) Créer un patron de fonctions permettant de calculer la somme d'un tableau d'éléments de type quelconque, le nombre d'éléments du tableau étant fourni en paramètre (on supposera que l'environnement utilisé accepte les « paramètres expression »). Écrire un petit programme utilisant ce patron.

Exercice 2:

Soit cette définition de patron de fonctions :

```
template <class T, class U> T fct (T a, U b, T c)
{ .....
}
```

Avec les déclarations suivantes :

```
int n, p, q ;
float x ;
char t[20] ;
char c ;
```

Quels sont les appels corrects et, dans ce cas, quels sont les prototypes des fonctions instanciées ?

```
fct (n, p, q) ;      // appel I
fct (n, x, q) ;      // appel II
fct (x, n, q) ;      // appel III
fct (t, n, &c) ;     // appel IV
```

Exercice 3:

Soient les définitions suivantes de patrons de fonctions :

```
template <class T, class U> void fct (T a, U b) { ... }          // patron I
template <class T, class U> void fct (T * a, U b) { ... }        // patron II
template <class T> void fct (T, T, T) { ... }                    // patron III
void fct (int a, float b) { .....}                               // fonction IV
```

Avec ces déclarations :

```
int n, p, q ;
float x, y ;
double z ;
```

Quels sont les appels corrects et, dans ce cas, quels sont les patrons utilisés et les prototypes des fonctions instanciées ?

```
fct (n, p) ;          // appel I
fct (x, y ) ;         // appel II
fct (n, x) ;          // appel III
fct (n, z) ;          // appel IV
fct (&n, p) ;         // appel V
fct (&n, x) ;         // appel VI
fct (&n, &p, &q)      // appel VII
```

Exercice 4:

On a défini le patron de classes suivant :

```
template  <class T>   class point
  { T x, y ; // coordonnees
public :
point (T abs, T ord) { x = abs ; y = ord ; }
void affiche () ;
  } ;
template  <class T>  void point<T> :: affiche () { cout << "Coordonnees : " << x << " " << y << "\n" ; }
```

1) Que se passe-t-il avec ces instructions :

```
point  <char>  p(60, 65) ;
p.affiche () ;
```

2) Comment faut-il modifier la définition de notre patron pour que les instructions précédentes affichent bien :

```
Coordonnees :   60   65
```