# Programmation Orientée Objet

Les fonctions

**Pr Imane DAOUDI**

**2016-2017**

# Plan

1. Introduction à la programmation orientée objet
2. Généralités
   - Types de base, opérateurs et expressions
   - Les instructions de contrôles
   - Les entrées-sorties
3. Opérateurs sur tableaux, pointeurs et objets
4. Les fonctions
5. Programmation des classes et objets
6. Appels et surcharges
7. L’ héritage
8. Polymorphisme

# 4. Les fonctions

➢ Définitions de fonction

**type nom( liste des paramètres) { corps }**

- type est le type du résultat de la fonction.
- (**void** si il s'agit d'une procédure)
- La liste des paramètres (paramètres formels):

  **type1 p1, ..., typen pn**

- Le **corps** décrit les instructions à effectuer.
- Le corps utilise ses propres variables locales, les éventuelles variables globales et les paramètres formels.
- Si une fonction renvoie un résultat, il doit y avoir (au moins) une instruction **return expr** ;

# 4. Les fonctions

- Exemple

```
int max(int a,int b)
{
        int res=b ;
        if (a>b) res = a ;
        return res ;
}
```

- Appel de fonction

**nom(liste des arguments)**

- Exemple :

```
int k=34, t=5, m ;
m = max(k,2*t+5) ;
```

# 4. Les fonctions

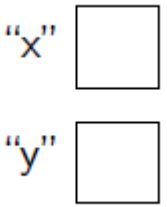
- Exemple

```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```

# 4. Les fonctions

- Exemple

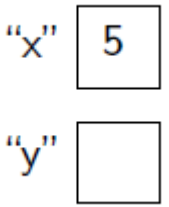
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

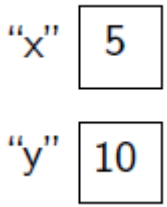
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

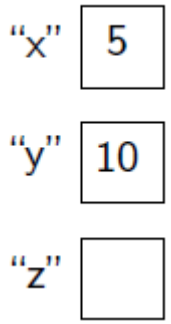
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

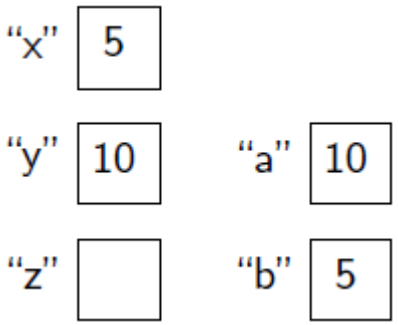
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

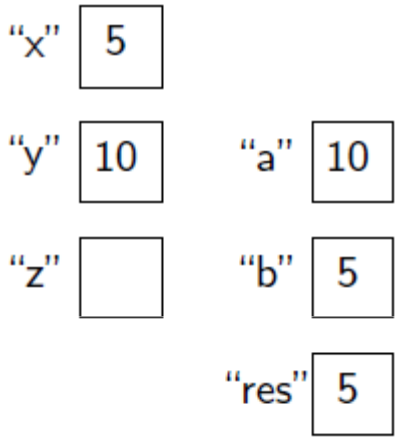
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

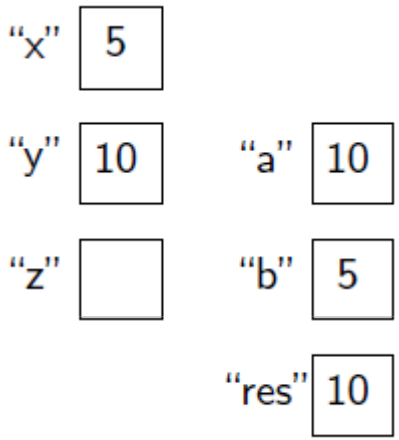
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

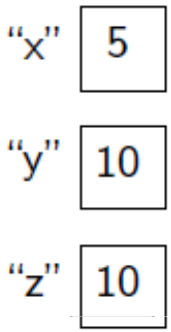
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

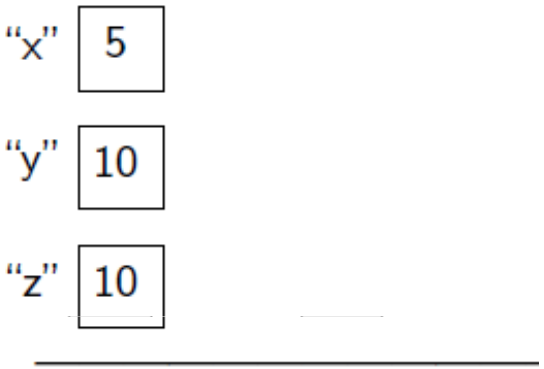
```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```



# 4. Les fonctions

- Exemple

```
int max(int a,int b)
{int res=b ;
if (a>b) res = a ;
return res ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
int z = max(y,x) ;
cout<<" z = "<<z ; }
```




z = 10

- Fin

# 4. Les fonctions

- Exemple: Afficher le contenu d'un tableau d'entiers

```
void AfficherTab(vector<int> T)
{
for (int i=0 ; i< T.size() ; i++)
cout << T[i] << " " ;
}
```

# 4. Les fonctions

- Exemple: Saisie d'un tableau d'entiers

```
vector<int> SaisieTab()
{
int taille ;
cout << " Entrer une taille : " ;
cin >> taille ;
vector<int> res(taille,0) ;
for (int i=0 ; i< taille ; i++) {
cout << " val = " ;
cin >> res[i] ;
}
return res ;
}
```

# 4. Les fonctions

- Exemple: Recherche du plus grand élément

```
int Recherche(vector<int> T)
{
if (T.size()==0) {
cout << ‘‘Erreur ! Tableau vide !’’ << endl ;
return -1 ; }
int res=T[0] ;
for (int i=1 ; i<T.size() ;i++)
if (T[i] > res) res=T[i] ;
return res ;
}
```

# 4. Les fonctions

- Exemple: Recherche de l'indice du plus grand élément

```
int RechercheInd(vector<int> T)
{
if (T.size()==0) {
cout << ‘‘Erreur ! Tableau vide !’’ << endl ;
return -1 ; }
int res=0 ;
for (int i=1 ; i<T.size() ;i++)
if (T[i] > T[res]) res=i ;
return res ;
}
```

# 4. Les fonctions

- Passage de paramètres par valeur

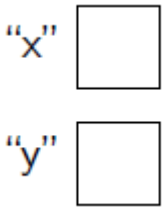
Par défaut, les paramètres d'une fonction sont initialisés par **une copie des valeurs** des paramètres réels.

Modifier la valeur des paramètres formels dans le corps de la fonction **ne change pas la valeur des paramètres réels.**

# 4. Les fonctions

- Exemple: Essai de permutation

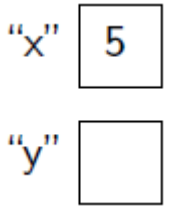
```
void permut(int a, int b)
{
int aux = b ;
b = a ;
 a = aux ;
}

int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ;
}
```



# 4. Les fonctions

- Exemple: Essai de permutation

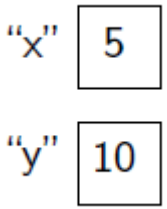
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

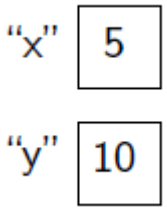
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

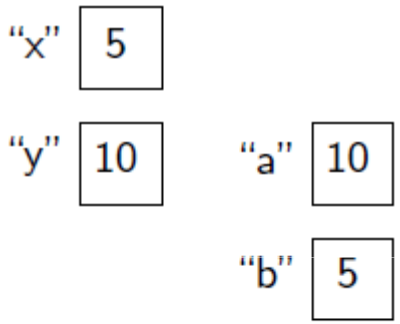
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

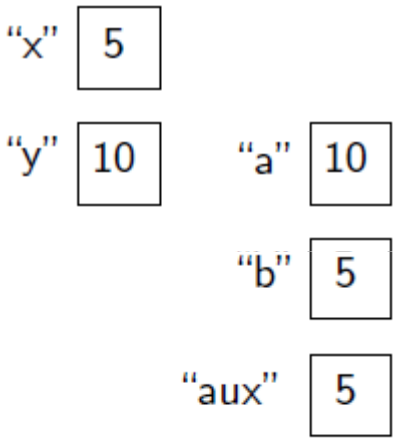
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

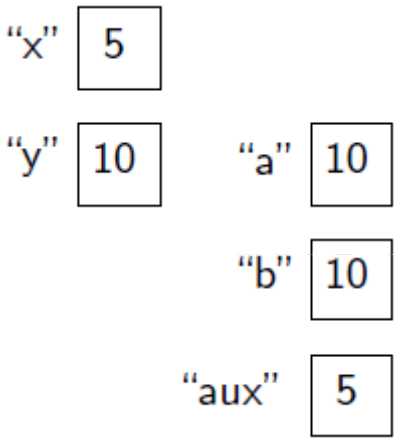
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

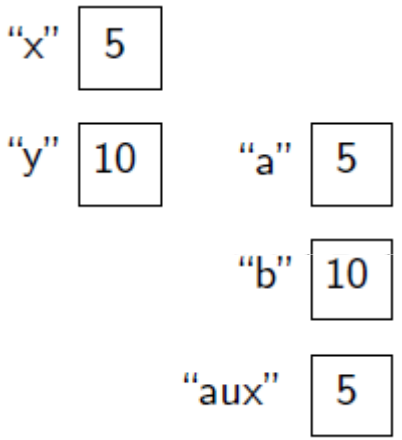
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

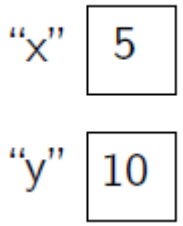
```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Essai de permutation

```
void permut(int a, int b)
{ int aux = b ;
b = a ;
 a = aux ; }
 ...
int main() {
 int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

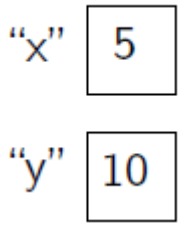
➢ Passage des paramètres par référence

- Pour modifier la valeur d'un paramètre réel dans une fonction, il faut passer ce paramètre par **référence**.
- Une référence sur une variable est un **synonyme** de cette variable, c'est-à-dire une autre manière de désigner le même emplacement de la mémoire.
- On utilise le symbole **&** pour la déclaration d'une référence:
- Dans la liste des paramètres de la définition d'une fonction, **type & pi** déclare le paramètre **pi** comme étant une référence sur le i ème paramètre réel **vi** .

# 4. Les fonctions

- Exemple: Permutation

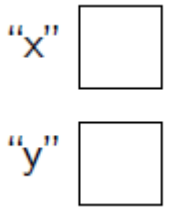
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

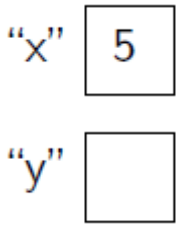
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

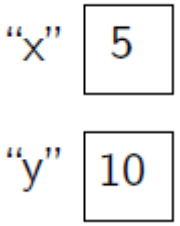
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

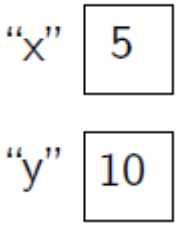
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

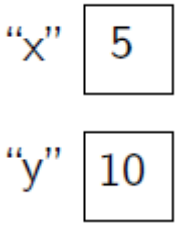
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

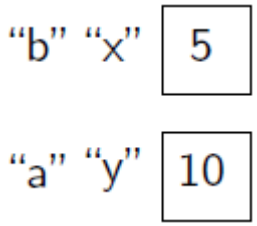
```
void permut(int & a,int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

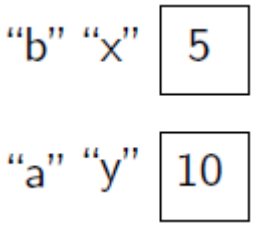
```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```

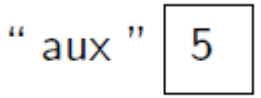


# 4. Les fonctions

- Exemple: Permutation

```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```

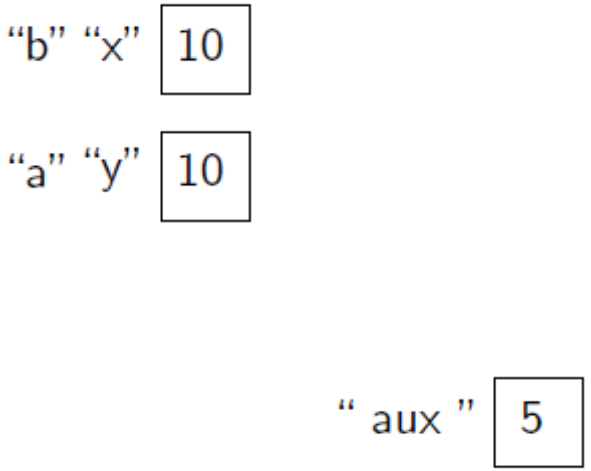


# 4. Les fonctions

- Exemple: Permutation

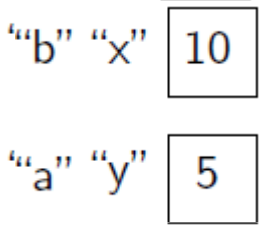
```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```

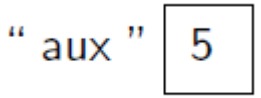


# 4. Les fonctions

- Exemple: Permutation

```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```

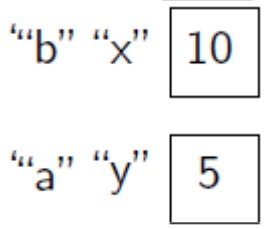


# 4. Les fonctions

- Exemple: Permutation

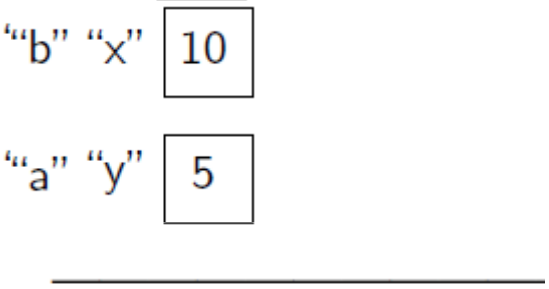
```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```



# 4. Les fonctions

- Exemple: Permutation

```
void permut(int & a, int & b)
{int aux = b ;
b = a ;
a = aux ; }
...
int main() {
int x,y ;
x=5 ;
y=10 ;
permut(y,x) ;
cout<<" x = "<<x ; }
```




x = 10

# 4. Les fonctions

- Passage des paramètres par référence

- Il faut que les paramètres réels soient compatibles avec un passage par référence. . .

  **permut(x,5) n'est pas possible !**

- **Remarque:** le passage par référence est plus efficace que le passage par valeur car il évite la recopie des arguments car seule l'adresse mémoire de l'argument est communiquée à la fonction.

# 4. Les fonctions

- Exercice
- Algorithmes de tri
- Comment trier un tableau d'entiers ?

- **Solution**

✓ rechercher le plus grand élément et le placer à la fin du tableau

✓ recherche le deuxième plus grand et le placer en avant-dernière position

✓ etc.

# 4. Les fonctions

- Exercice
- Algorithmes de tri
- Comment trier un tableau d'entiers **tab**?

- **Solution**
- Pour **IndFin = Tab.size()-1 ... 1** faire
- rechercher l'indice **IMAX** du plus grand élément de **Tab** entre les indices **0** à **IndFin**.
- Permuter les éléments placés en **IMAX** et **IndFin**.

A la première étape, on sélectionne le plus grand élément et on le place à la fin du tableau ; puis on trouve le deuxième plus grand et on le place à l'avant-dernière place etc.

# 4. Les fonctions

- Solution: Fonction de sélection

```
int RechercheInd(vector<int> T, int imax)
{
if (T.size() < imax-1) {
cout << "Erreur ! Tableau trop petit ! " << endl ;
return -1 ; }
int res=0 ;
for (int i=1 ; i<= imax ;i++)
if (T[i] > T[res]) res=i ;
return res ;
}
```

# 4. Les fonctions

- Solution: Le Tri

```
vector<int> Trier(vector<int> T)
{
vector<int> Taux = T ;
int aux ;
for (int i=Taux.size()-1 ; i>0 ; i--) {
aux = RechercheInd(Taux,i) ;
Permuter(Taux[aux],Taux[i]) ;
}
return Taux ;
}
```

# 4. Les fonctions

- Solution: le Tri

- **Tri avec passage par référence**

```
void Trier(vector<int> & T)
{
int aux ;
for (int i=T.size()-1 ; i>0 ; i--) {
aux = RechercheInd(T,i) ;
Permuter(T[aux],T[i]) ;
}
}
```

# Plan

1. Introduction à la programmation orientée objet
2. Généralités
3. Opérateurs sur tableaux, pointeurs et objets
4. Les fonctions
5. Programmation des classes et objets
6. Appels et surcharges
7. L’ héritage
8. Polymorphisme