Permutation de deux entiers

```
Procédure saisir_entier(binf, bsup : entier ,message : chaîne ,@ x : entier)
Début
   Répéter
      Ecrire (message)
      Lire (x)
   Jusqu'à (x >= binf et x <= bsup)
Fin
```

```
Procédure permut(@ n: entier, @m: entier)
  Début
    aux ← n
    n ← m
    m← aux
Fin
```

TDo Locaux	
aux	Entier

```
Algorithme permutation                (Algorithme principal)
Début
   saisir_entier(1, 100, "donner un 1er entier", n)
   Ecrire("Avant permutation" ,n,m)
   saisir_entier(1, 100, "donner un 2eme entier", m)
   permut(n,m)
   Ecrire("Après permutation" ,n,m)
Fin
```

TDo globaux	
n,m	Entier
Saisir_entier	Procédure qui permet de saisir un entier
permut	Procédure qui permetde permuter deux entiers

```python
def saisir_entier(binf,bsup,message):
    global x
    x=str(bsup+1) #Une valeur qui n'est pas valide pur assurer l'entrée à la boucle
    while not(x.isdecimal() and int(x) in range(binf,bsup+1)):
        x=input(message)
    x=int(x)
def permut():
    global n,m
    aux=n
    n=m
    m=aux
saisir_entier(1,100,"donner n")
n=x
saisir_entier(1,100,"donner m")
m=x
print("avant",n,m)
permut()
print("après",n,m)
```

Tri_bulle

```
Procédure saisir_entier(binf, bsup : entier ,message : chaîne ,@ x : entier)
Début
   Répéter
      Ecrire (message)
      Lire (x)
   Jusqu'à (x >= binf et x <= bsup)
Fin
```

```
Procédure permut(@ t : tab, i : entier)
  Début
    aux ← t[i]
    t[i] ← t[i+1]
    t[i+1] ← aux
Fin
```

TDo Locaux	
aux	Entier (compteur de la boucle)

```
Procédure remplir_tableau(@t : tab ,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      msg ← "donner l'élément n°" +i
      saisir_entier(0, 100, msg, t[i])
   FinPour
Fin
```

TDo Locaux	
i	entier
msg	Chaine de caractères
Saisir_entier	Procédure qui permet de saisir un entier

```
Procédure afficher_tableau(t : tab,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      Ecrire t[i]
   FinPour
Fin_
```

TDo Locaux	
i	entier

```
Procédure tri_bulle(@ t : tab , taille : entier)
Début
   répéter
      echange ← faux
      Pour i de 0 à taille - 2 Faire
         Si t[i] > t[i+1] Alors
            permut(t, i)
            echange ← vrai
         FinSi
      FinPour
```

```
    taille ← taille - 1
  jusqu à (echange = faux)
Fin
```

TDo Locaux	
i	Entier
echange	Booléen
permut	Procédure qui permet de permuter deux entiers

```
Algorithme bulle                (Algorithme principal)
Début
   saisir_entier(1, 100, "donner la taille du tableau", n)
   remplir_tableau(t, n)
   Ecrire("Avant tri" )
   afficher_tableau(t, n)
   tri_bulle(t, n)
  Ecrire("Après tri" )
  afficher_tableau(t, n)
Fin
```

TDo globaux	
n	Entier
t	tab
Saisir_entier	Procédure qui permet de saisir un entier
Remplir_tableau	Procédure qui permet de remplir un tableau
Afficher_tableau	Procédure qui permet d'afficher un tableau
Tri_bulle	Procédure qui permet de trier un tableau

Tableau de déclaration des nouveaux types
Tab= tableau d'entiers

```
from numpy import*
def saisir_entier(binf,bsup,message):
    global x
    x=str(bsup+1) #Une valeur qui n'est pas valide pur assurer l'entrée à la boucle
    while not(x.isdecimal() and int(x) in range(binf,bsup+1)):
        x=input(message)
    x=int(x)
def permut(t,i):
    aux=t[i]
    t[i]=t[i+1]
    t[i+1]=aux
def remplir_tableau(t,taille):
    for i in range(taille):
        msg="donner l'élément n°"+str(i)
        saisir_entier(0,100,msg)
        v=x
        t[i]=int(v)
def afficher_tableau(t,taille):
    for i in range(taille):
        print(t[i],end=" ")
    print()
def tri_bulle(t,taille):
    echange=True
    while (echange==True):
        echange=False
        for i in range(0,taille-1):
            if t[i]>t[i+1]:
                permut(t,i)
                echange=True
        taille=taille-1
saisir_entier(1,100,"donner la taille du tableau")
n=x
t=array([int]*n)
remplir_tableau(t,n)
afficher_tableau(t,n)
tri_bulle(t,n)
afficher_tableau(t,n)

```

Recherche_dichotomique

```
Procédure saisir_entier(binf, bsup : entier ,message : chaîne ,@ x : entier)
Début
   Répéter
      Ecrire (message)
      Lire (x)
   Jusqu'à (x >= binf et x <= bsup)
Fin
```

```
Procédure remplir_tableau(@t : tab ,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      msg ← "donner l'élément n°" +i
      saisir_entier(0, 100, msg, t[i])
   FinPour
Fin
```

TDo Locaux	
i	entier
msg	Chaine de caractères
Saisir_entier	Procédure qui permet de saisir un entier

```
Procédure afficher_tableau(t : tab,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      Ecrire t[i]
   FinPour
Fin_
```

TDo Locaux	
i	entier

```
fonction recherche_dichotomique(t : tab, taille, m : entier) : booléen
Début
   d ← 0
   f ← taille - 1
   répéter
      mil ← (d + f) div 2
      Si m < t[mil] Alors
         f ← mil - 1
      Sinon
         d ← mil + 1
```

```
    FinSi
  Jusqu' à (t[mil] =m) ou (d > f)
  return (t[mil] = m)
Fin
```

TDo Locaux	
d, f, mil	Entier

```
Algorithme bulle                    (Algorithme principal)
Début
   saisir_entier(1, 100, "donner la taille du tableau", n)
   remplir_tableau(t, n)
   Ecrire("Avant tri" )
   afficher_tableau(t, n)
   saisir_entier(1, 100, "donner l'élément à rechercher", m)
   si recherche_dichotomique (t, n,m) alors
         Ecrire("L'élément existe dans le tableau" )
  Sinon
         Ecrire("L'élément n'existe pas dans le tableau" )
  finsi
Fin
```

TDo globaux	
n,m	Entier
t	tab
Saisir_entier	Procédure qui permet de saisir un entier
Remplir_tableau	Procédure qui permet de remplir un tableau
Afficher_tableau	Procédure qui permet d'afficher un tableau
recherche_dichotomique	Fonction qui permet de rechercher un entier dans un tableau

Tableau de déclaration des nouveaux types
Tab= tableau d'entiers

```
from numpy import*
def saisir_entier(binf,bsup,message):
    global x
    x=str(bsup+1) #Une valeur qui n'est pas valide pur assurer l'entrée à la boucle
    while not(x.isdecimal() and int(x) in range(binf,bsup+1)):
        x=input(message)
    x=int(x)
def remplir_tableau(t,taille):
    for i in range(taille):
        msg="donner l'élément n°"+str(i)
        saisir_entier(0,100,msg)
        y=x
        t[i]=int(y)
def afficher_tableau(t,taille):
    for i in range(taille):
        print(t[i],end=" ")
    print()

def recherche_dichotomique (t,taille,m):
    d=0
    f=taille-1
    mil=(d+f)//2
    while t[mil]!=m and d<=f:
        if m<t[mil]:
            f=mil-1
        elif m>t[mil]:
            d=mil+1
        mil=(d+f)//2
    return t[mil]==m
saisir_entier(1,100,"donner la taille du tableau")
n=x
t=array([int]*n)
remplir_tableau(t,n)
afficher_tableau(t,n)
saisir_entier(1,100,"donner l'élément à recherche")
m=x
if recherche_dichotomique(t,n,m):
    print("l'élément existe")
else:
    print("l'élément n'existe pas")

```

Recherche_séquentielle

```
Procédure saisir_entier(binf, bsup : entier ,message : chaîne ,@ x : entier)
Début
   Répéter
      Ecrire (message)
      Lire (x)
   Jusqu'à (x >= binf et x <= bsup)
Fin
```

```
Procédure remplir_tableau(@t : tab ,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      msg ← "donner l'élément n°" +i
      saisir_entier(0, 100, msg, t[i])
   FinPour
Fin
```

TDo Locaux	
i	entier
msg	Chaine de caractères
Saisir_entier	Procédure qui permet de saisir un entier

```
Procédure afficher_tableau(t : tab,taille : entier)
Début
   Pour i de 0 à taille - 1 Faire
      Ecrire t[i]
   FinPour
Fin_
```

TDo Locaux	
i	entier

```
fonction recherche_sequentielle(t : tab, taille, m : entier) : booléen
Début
   d ← 0
   f ← taille - 1
   répéter
      mil ← (d + f) div 2
      Si m < t[mil] Alors
         f ← mil - 1
      Sinon
```

```
        d ← mil + 1
      FinSi
   Jusqu' à (t[mil] =m) ou (d > f)
   return (t[mil] = m)
Fin
```

TDo Locaux	
d, f, mil	Entier

```
Algorithme bulle                    (Algorithme principal)
Début
   saisir_entier(1, 100, "donner la taille du tableau", n)
   remplir_tableau(t, n)
   Ecrire("Avant tri" )
   afficher_tableau(t, n)
   saisir_entier(1, 100, "donner l'élément à rechercher", m)
   si recherche_sequentielle (t, n,m) alors
         Ecrire("L'élément existe dans le tableau" )
  Sinon
         Ecrire("L'élément n'existe pas dans le tableau" )
  finsi
Fin
```

TDo globaux	
n,m	Entier
t	tab
Saisir_entier	Procédure qui permet de saisir un entier
Remplir_tableau	Procédure qui permet de remplir un tableau
Afficher_tableau	Procédure qui permet d'afficher un tableau
recherche_sequentielle	Fonction qui permet de rechercher un entier dans un tableau

Tableau de déclaration des nouveaux types
Tab= tableau d'entiers

```
from numpy import*
def saisir_entier(binf,bsup,message):
    global x
    x=str(bsup+1) #Une valeur qui n'est pas valide pur assurer l'entrée à la boucle
    while not(x.isdecimal() and int(x) in range(binf,bsup+1)):
        x=input(message)
    x=int(x)
def remplir_tableau(t,taille):
    for i in range(taille):
        msg="donner l'élément n°"+str(i)
        saisir_entier(0,100,msg)
        y=x
        t[i]=int(y)
def afficher_tableau(t,taille):
    for i in range(taille):
        print(t[i],end=" ")
    print()
def recherche_sequentielle (t,taille,m):
    i=0
    while t[i]!=m and i<taille-1:
        i=i+1
    return t[i]==m
saisir_entier(1,100,"donner la taille du tableau")
n=x
t=array([int]*n)
remplir_tableau(t,n)
afficher_tableau(t,n)
print()
saisir_entier(1,100,"donner l'élément à rechercher")
m=x
if recherche_sequentielle(t,n,m):
    print("l'élément existe")
else:
    print("l'élément n'existe pas")
```