

Recherche dans un tableau:

Écrire un programme qui permet de saisir n entiers à mettre dans un tableau T puis une valeur v, puis vérifie si v existe dans T ou non.

➤ Recherche séquentielle

```
0) DEF FN recherche1 (T:TAB,n:entier,v:entier): Booléen
1) i← 0 trouve ←Faux
   Répéter
   i← i+1
   si T[i]=v alors trouve←vrai
   jusqu'a (trouve) ou (i=n)
2) recherche← trouve
3) Fin recherche1
```

T.D.O locaux(Tableau de déclaration des objets locaux)

Objet	Type/Nature	Rôle
i	Entier	
trouve	booléen	

Autre méthode :

```
0) DEF FN recherche2 (T:TAB,n: entier,v:entier): Booléen
1) i← 0
   Répéter
   i← i+1
   jusqu'a (T[i]=v) ou (i=n)
2) si T[i]=v alors recherche← VRAI
   sinon recherche←FAUX
3) Fin recherche2
```

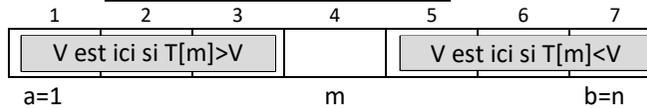
T.D.O locaux(Tableau de déclaration des objets locaux)

Objet	Type/Nature	Rôle
i	Entier	

En Pascal :

```
function recherche1(T:tab;n:integer;v:integer):boolean;
var i:integer; trouve:boolean;
begin
  i:=0; trouve:=false;
  repeat
    i:=i+1;
    if t[i]=v then trouve:=true;
  until (trouve) or (i=n);
  recherche:=trouve;
end;
```

➤ Recherche dichotomique :



Algorithme de la fonction recherche_dicho:

```
0) DEF FN recherche_dicho (T:TAB,n:entier,v:entier): Booléen
1) a← 1, b← n, trouve← FAUX
   répéter
   m← (a+b) DIV 2
   si T[m]=v alors trouve← VRAI
   sinon si T[m]>v alors b← m-1
   sinon a← m+1
   finsi
   jusqu'a (trouve) ou (a>b)
2) recherche_dicho←trouve
3) Fin recherche_dicho
```

T.D.O locaux(Tableau de déclaration des objets locaux)

Objet	Type/Nature	Rôle
a,b,m	Entier	
trouve	booléen	

En Pascal :

```
function recherche_dicho(T:tab;n:integer;v:integer):boolean;
var a,b,m:integer;trouve:boolean;
begin
  a:= 1 ; b:= n; trouve:=false;
  repeat
    m:=(a+b) div 2;
    si T[m]=v then trouve:=true else
      if T[m]>v then b:=m-1 else a:=m+1;
  until (trouve)or (a>b);
  recherche_dicho:=trouve;
end;
```

Tri Sélection :

```
0)DEF Proc Tri_selection (VAR T :TAB ; n :entier)
1)Pour i de 1 à n-1 faire preposmin(T:tab, i:entier, n:entier):entier
   [pos_min←i] Pour j de i+1 à n faire
     Si ( T[j] < T[pos_min])Alors
       pos_min ← j
     Finsi
   FinPour
   Si ( pos_min <> i) Alors Permute(T[i], T[Pos_min])
   Finsi
   FinPour
2)Fin Tri_selection
```

Tri Bulles :

```
0) DEF Proc Tri_Bulles (VAR T :TAB ; n :entier)
1) Répéter
   Echange← faux
   Pour i de 1 à n-1 faire
     Si ( T[i] > T[i+1])Alors Permute(T[i], T[i+1])
     Echange ← vrai
   FinSi
   FinPour
   n ←n-1
   Jusqu'à (Echange = Faux) ou (n=1)
2) Fin Tri_Bulles
```

Tri Insertion :

```
0) DEF Proc Tri_Insertion (VAR T :TAB ; n :entier)
1) Pour i de 2 à n faire
   Tmp ← T[i]
   j ← i
   Decaler(var T:tab, var j, v:entier)
   Tant que ( j >1) et (T[j-1] > Tmp ) faire
     T[j] ←T[j -1]
     j ← j - 1
   FinTantQue
   T[j ] ←Tmp
   FinPour
2) FIN Tri_Insertion
```