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*

Analyse de la fonction Recherche:

DEF FN recherche (T:TAB, n: entier, v:entier): Booléen
Résultat=[] si T[i]=v alors recherche← VRAI
sinon recherche← FAUX
finsi
i=[i← 0]Répéter
i← i+1
jusqu'a (T[i]=v) ou (i=n)
FIN recherche

Autre méthode :
Résultat= Recherche ← trouve
trouve=[i ← 0 , trouve ← faux]
répéter
i←i+1
si t[i]=v alors trouve ←vrai
Finsi
jusqu'a (trouve) ou (i = n)
Fin Recherche

T.D.O.Locaux:

Objet	Type/Nature
i	entier

Algorithme de la fonction recherche:

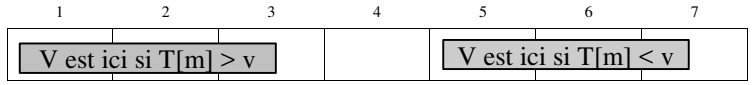
0) DEF FN recherche (T:TAB, n: entier, v:entier): Booléen
1) i← 0
Répéter
i← i+1
jusqu'a (T[i]=v) ou (i=n)
2) si T[i]=v alors recherche← VRAI
sinon recherche← FAUX
3) Fin recherche

Autre méthode :
0) DEF FN recherche(t :tab, n :entier, v :entier) : booléen
1) i←0 , trouve← Faux
2) répéter
i←i+1
si t[i]=v alors trouve ←vrai
Finsi
jusqu'a (trouve) ou (i = n)
3) Recherche ← trouve
4) Fin Recherche

En Pascal:

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

➔ *Recherche dichotomique:*



a=1 m b= n
(milieu de a et b)

Analyse de la fonction recherche_dicho:

DEF FN recherche_dicho (T:TAB,n: entier,v:entier): Booléen
Résultat=recherche_dicho←trouve
trouve=[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)
FIN recherche_dicho

T.D.O.Locaux:

Objet	Type/Nature
Trouve	Booléen
a, b, m	entier

Remarque:
La recherche dichotomique s'applique sur un tableau trié.

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

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;
if 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**
premposmin(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