1. **Tr i par sélection :**
* **Analyse du Programme principal** :
* **Algorithme du Programme principal** :
1. **Début Sélection**
2. **PROC Saisir(n)**
3. **PROC Remplir(T,n)**
4. **PROC Tri\_Selction(T, n)**
5. **PROC Afficher(T, n)**
6. **Fin Sélection**
* **Algorithme de la procédure Saisir** :
1. **DEF PROC Saisir (var n :entier)**
2. [ ] **Répéter**

 Ecrire("saisir la taille d'un tableau=")

 Lire(n)

 **Jusqu’à n>0**

1. **Fin Saisir**
* **Algorithme de la procédure Remplir** :
1. **DEF PROC** Remplir **( var T :tab; n :entier)**
2. [ ] **Pour** i de 1 à n **faire**

 Ecrire("T[",i,"]="),Lire(T[i])

 **FinPour**

1. **Fin Remplir**
* **Algorithme de la procédure Tri\_Selection**:
1. **DEF PROC Tri\_Selection (Var T :tab;n :entier)**
2. [ ] **Pour i de 1 à n-1 faire**

 [pos\_min🡨i] **Pour** j **d**e i+1 **à** n **faire**

 [ ] **Si** ( T[j] < T[pos\_min])**Alors**

 **pos\_min**🡨**j**

 **Finsi**

 **FinPour**

 [ ] **Si** ( pos\_min < > i)**Alors**

 **PROC** Permute(T[i], T[pos\_min])

 **Finsi**

 **FinPour**

1. **FIN Tri\_Selection**

**NOM :** Sélection

**Résultat=PROC Afficher(T, n)**

 **PROC Tri\_Selction(T, n)**

 **PROC Remplir(T,n)**

 **PROC Saisir(n)**

**FIN** Sélection

 **T.D.N.T**

|  |
| --- |
| **Type** |
| tab= tableau de 50 entiers |

**TDO.Globaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| n | Var/entier | La taille  |
| T | Var/tab | tableau à trier |
| Saisir,Remplir,Tri\_Selection,Afficher | procedure |  |

**Analyse de la procédure Saisir** :

**DEF PROC Saisir (var n :entier)**

**Résultat**= [ ]

 **Répéter**

 n=donnée("saisir la taille d'un tableau=")

 **Jusqu’à n>0**

**Fin Saisir**

**Analyse de la procédure Remplir** :

**DEF PROC** Remplir **( var T :tab; n :entier)**

**Résultat=** [ ]

**Pour** i de 1 à n **faire**

 T[i]=donnée("T[",i,"]=")

**FinPour**

**Fin Remplir**

**TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i | Var/entier | compteur |

**Analyse de la procédure Tri\_Selection**:

**DEF PROC Tri\_Selection (var T :tab ;n :entier)**

**Résultat=** [ ] **Pour i de 1 à n-1 faire**

**TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i,j | Var/entier | compteur |
| pos\_min | Var/entier | Position de la petite valeur |
| Permute | procédure |  |

 [pos\_min🡨i] **Pour** j **d**e i+1 **à** n **faire**

 [ ] **Si** ( T[j] < T[pos\_min])**Alors**

 **pos\_min🡨j**

 **Finsi**

 **FinPour**

 [ ] **Si** ( pos\_min < > i)**Alors**

 **PROC** Permute(T[i], T[pos\_min])

 **Finsi**

**FinPour**

**FIN Tri\_Selection**

**Analyse de la procédure Permute** :

* **Algorithme de la procédure Permute** :
1. **DEF PROC** Permute **(var a,b :entier)**
2. aux 🡨a
3. a 🡨b
4. b🡨aux
5. **FIN Permute**

**Algorithme de la procédure Afficher**:

1. **DEF PROC** Afficher **( T :tab ;n :entier)**
2. [ ] **Pour** i de 1 à n **faire**

 Ecrire(T[i],"|")

 **FinPour**

1. **Fin Afficher**

**DEF PROC** Permute **( var a,b :entier)**

**Résultat=** aux 🡨a

 a 🡨b

 b🡨aux

**FIN Permute**

 **TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| aux | Var/entier | Variable intermédiaire |

**Analyse de la procédure Afficher**:

**DEF PROC** Afficher **( T :tab ;n :entier)**

**Résultat=** [ ] **Pour** i de 1 à n **faire**

 Ecrire(T[i],"|") **TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i | Var/entier | compteur |

 **FinPour**

**Fin Afficher**

 **Traduction Pascal :**

 **end;**

**begin**

**for i:=1 to n-1 do**

 **begin**

 **posmin:=i;**

 **for j:=i+1 to n do**

 **begin**

 **if(t[j]<t[posmin])then**

 **posmin:=j;**

 **end;**

 **if posmin <>i then**

 **permute(t[i],t[posmin]);**

 **end;**

**end;**

**procedure Afficher(t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write(t[i],' | ');**

 **end;**

**end;**

**{\*\*\*\*\*\*\*\*\*\*\* programme principal\*\*\*\*\*\*\*\*}**

**begin**

 **Saisir(n);**

 **Remplir(t,n);**

 **Tri\_Selection(t,n);**

 **Afficher(t,n);**

**end.**

**program Selection;**

**uses wincrt;**

**type tab=array[1..50] of integer;**

**var n:integer;**

 **t:tab;**

**procedure Saisir(var n:integer);**

**begin**

**repeat**

 **write('saisir la taille d''un tableau= ');**

 **readln(n);**

**until n>0;**

**end;**

**procedure Remplir(var t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write('t[',i,']=');**

 **readln(t[i]);**

**end;**

**end;**

**procedure Tri\_Selection(var t:tab;n:integer);**

**var i,j,posmin:integer;**

 **procedure permute(var a,b:integer);**

 **var aux:integer;**

 **begin**

 **aux:=a;**

 **a:=b;**

 **b:=aux;**

1. **Tr i à Bulles :**
* **Algorithme de la procédure Tri\_Bulles**:
1. **DEF PROC Tri\_Bulles (var T :tab ; n :entier)**
2. **Répéter**

 [ Echange🡨faux] **Pour** i de 1 à n-1 **faire**

 [ ] **Si** ( T[i] > T[i+1])**Alors**

 **PROC** Permute(T[i], T[i+1])

 Echange 🡨 vrai

 **FinSi**

 **FinPour**

 **n**🡨**n-1**

 **Jusqu'à (**Echange = Faux) ou (n=1)

1. **Fin Tri\_Bulles**
* **Analyse de la procédure Tri\_Bulles**:

**DEF PROC Tri\_Bulles (var T :tab ; n :entier)**

**Résultat=** [ ]

**Répéter**

[ Echange🡨faux] **Pour** i de 1 à n-1 **faire**

 [ ] **Si** ( T[i] > T[i+1])**Alors**

 **PROC** Permute(T[i], T[i+1])

 Echange 🡨 vrai

 **FinSi**

 **FinPour**

 **n**🡨**n-1**

**Jusqu'à (**Echange = Faux) ou (n=1)

**Fin Tri\_Bulles**

**TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i | Var/entier | compteur |
| Echange | Var/booléen | Variable booléenne |
| Permute | procédure |  |

 **Traduction Pascal :**

 **a:=b;**

 **b:=aux;**

 **end;**

**begin**

**repeat**

 **echange:=false;**

 **for i:=1 to n-1 do**

 **if(t[i]>t[i+1])then**

 **begin**

 **permute(t[i],t[i+1]);**

 **echange:=true;**

 **end;**

**n:=n-1;**

**until(echange=false) or (n=1);**

**end;**

**procedure Afficher(t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write(t[i],' | ');**

 **end;**

**end;**

**begin**

 **Saisir(n);**

 **Remplir(t,n);**

 **Tri\_Bulles(t,n);**

 **Afficher(t,n);**

**end.**

**program Bulles;**

**uses wincrt;**

**type tab=array[1..50] of integer;**

**var n:integer ; t:tab;**

**procedure Saisir(var n:integer);**

**begin**

**repeat**

 **write('saisir la taille d''un tableau= ');**

 **readln(n);**

**until n>0;**

**end;**

**procedure Remplir(var t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write('t[',i,']='); readln(t[i]);**

**end;**

**end;**

**procedure Tri\_Bulles(var t:tab;n:integer);**

**var i:integer;echange:boolean;**

 **procedure permute(var a,b:integer);**

 **var aux:integer;**

 **begin**

 **aux:=a;**

1. **Tr i par Insertion :**
* **Algorithme de la procédure Tri\_Insertion**:
1. **DEF PROC Tri\_Insertion (var T :tab ;n :entier)**
2. [ ] **Pour** i de 2 à n **faire**

 **[x**🡨**T[i],** j🡨i ]

 **Tant que** ( j >1) et ( T[j-1] > x ) **faire**

 T[j ]🡨 T[j-1]

 j 🡨 j - 1

 **FinTantQue**

 **T[j ]** 🡨**x**

 **FinPour**

1. **Fin Tri\_Insertion**

 **Analyse de la procédure Tri\_Insertion** :

**DEF PROC Tri\_Insertion (var T :tab ;n :entier)**

**Résultat=** [ ] **Pour** i de 2 à n **faire**

 **[x🡨T[i],** j🡨i ]

 **Tant que** ( j >1) et ( T[j-1] > x ) **faire**

 T[j ]🡨 T[j-1]

 j 🡨 j - 1

 **FinTantQue**

 **T[j ]** 🡨**x**

**FinPour**

**Fin Tri\_Insertion**

**TDO.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i,j | Var/entier | compteur |
| x | Var/entier |  |

 **Traduction Pascal :**

**procedure Tri\_Insertion(var t:tab;n:integer);**

**var i,j,x:integer;**

**begin**

 **for i:=2 to n do**

 **begin**

 **x:=t[i];**

 **j:=i;**

 **while (j>1) and (t[j-1]>x) do**

 **begin**

 **t[j]:=t[j-1];**

 **j:=j-1;**

 **end;**

 **t[j]:=x;**

 **end;**

**end;**

**procedure Afficher(t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write(t[i],' | ');**

 **end;**

**end;**

**{\*\*\*\*\*\*\*\*\*\*\* programme principal\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*}**

**begin**

 **Saisir(n);**

 **Remplir(t,n);**

 **Tri\_Insertion(t,n);**

 **Afficher(t,n);**

**end.**

**program Insertion;**

**uses wincrt;**

**type tab=array[1..50] of integer;**

**var n:integer;**

 **t:tab;**

**procedure Saisir(var n:integer);**

**begin**

**repeat**

 **write('saisir la taille d''un tableau= ');**

 **readln(n);**

**until n>0;**

**end;**

**procedure Remplir(var t:tab;n:integer);**

**var i:integer;**

**begin**

**for i:=1 to n do**

 **begin**

 **write('t[',i,']=');**

 **readln(t[i]);**

**end;**

**end;**

1. Les recherches:
2. La recherche séquentielle :
* **Analyse de la fonction Recherche:**

**DEF FN rech\_seq (T:TAB ;n,v:entier): Booléen**

**Résultat=rech\_seq🡨trouve**

 **[i**🡨**0,trouve🡨faux]Répéter**

 **i**🡨**i+1**

 **si (T[i]=v) alors**

 **trouve🡨vrai**

 **finsi**

 **jusqu'a (trouve=vrai) ou (i=n)**

**Fin rech\_seq**

**T.D.O.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i | Var/entier | Compteur |
| trouve | Var /booléen | Le résultat de la recherche |

* **Algorithme de la fonction recherche:**
1. **DEF FN rech\_seq (T:tab ;n,v:entier): Booléen**
2. **[i**🡨**0,trouve**🡨**faux]Répéter**

 **i**🡨**i+1**

 **si (T[i]=v) alors**

 **trouve🡨vrai**

 **finsi**

 **jusqu'a (trouve=vrai) ou (i=n)**

1. **rech\_seq**🡨**trouve**
2. **Fin rech\_seq**
* **En Pascal:**

**function rech\_seq(T:tab;n,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=true)or (i=n)**

 **rech\_seq:=trouve;**

**end;**

1. La recherche dichotomique :

**Remarque:**

La recherche dichotomique s'applique sur un tableau trié.

* **Analyse de la fonction rech\_dich:**

**DEF FN rech\_dich(T:tab ;n,v:entier): Booléen**

**Résultat=rech\_dich🡨trouve**

**T.D.O.Locaux:**

|  |  |  |
| --- | --- | --- |
| **Objet** | **Nature/type** | **Rôle** |
| i | Var/entier | Compteur |
| trouve | Var /booléen | Le résultat de la recherche |
| a | Var/entier | Le début de l’intervalle de recherche |
| b | Var/entier | La fin de l’intervalle de recherche |
| m | Var/entier | Le milieu de l’intervalle de recherche |

 [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=vrai) OU (a>b)

**FIN rech\_dich**

* **Algorithme de la fonction rech\_dich:**
1. **DEF FN rech\_dich(T:tab ;n,v:entier): Booléen**
2. [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=vrai) OU (a>b)

1. **rech\_dich**🡨**trouve**
2. **Fin rech\_dich**
* **En Pascal:**

**function rech\_dich(T:tab;n,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=true) or (a>b);**

**rech\_dich:=trouve;**

**end;**