Introduction-1

- Un tableau T est dit « **trié en ordre croissant** » si tous les éléments consécutifs du tableau vérifient :

$$T[i-1] <= T[i]$$

- Il est admis qu'un
  - tableau vide est trié
  - tableau ne contenant qu'un seul élément est trié

Introduction-2

- D'où la définition :
  - Un tableau vide (n=0) est ordonné (trié),
  - Un tableau contenant un seul élément (n=1) est ordonné,
  - Un tableau T[1..n], n>1, est ordonné si

    Pour tout i dans [2..n], T[i-1] <=T[i]

Introduction-3

- Tri d'un tableau
  - Soit un vecteur (tableau à une dimension) T[1..n] à valeurs dans un ensemble E de valeurs muni d'une relation d'ordre notée <
  - Trier le vecteur T consiste à construire un vecteur T'[1..n] tel que :
    - T' soit trié,
    - T' et T contiennent les mêmes éléments.
  - Le plus souvent T et T' sont le même vecteur ; T' est construit en permutant entre eux les éléments de T.

Introduction-4

- Tous les algorithmes de tri utilisent une procédure qui permet d'échanger (de permuter) la valeur de deux variables Dans le cas ou les variables sont réelles, la procédure échanger est la suivante :

```
Procedure Echanger(a,b:réel)
Variables       c:réel

Début
  c ←a;
  a ← b;
  b ← c;
FinProcedure
// En langage C Il faut faire un passage par adresse
```

Tri par sélection-1

- Le principe du tri par sélection d'un tableau est d'aller chercher le plus petit élément du vecteur pour le mettre en premier, puis de repartir du second, d'aller chercher le plus petit élément pour le mettre en second etc.
- Au $i^{ème}$ passage, on sélectionne l'élément ayant la plus petite valeur parmi les positions i..n et on l'échange avec T[i].

Tri par sélection-2

4	2	0	5	3	Tableau de départ
0	2	4	5	3	Le plus petit élément est à sa place
0	2	4	5	3	Les 2 plus petits éléments sont à leur place
0	2	3	5	4	Les 3 plus petits éléments sont à leur place
0	2	3	4	5	Les 4 plus petits éléments sont à leur place

Tri par sélection-3

```
Procédure Tri_Selection (Tableau T[n:entier]:réel )
Variables i,j: entiers
Début
  Pour i allant de 0 à n-1
    Pour j allant de i+1 à n-1
            Si (T[j] > T[i])  Alors
                    Echanger(T[i], T[j] );
            FinSi
    FinPour
  FinPour
Fin
```

Tri par remplacement-1

- Cette méthode simple et intuitive est malheureusement très peu performante.
- Elle consiste à construire un tableau Ttrié[1..n] à partir de T[1..n] tel que :

  Ttrié[i-1] <=Ttrié[i] , pour tout i dans [2..n]
- Principe :
  - Identifier le maximum du tableau
  - Rechercher le minimum du tableau T
  - Recopier ce minimum dans Ttrié à la position i
  - Remplacer le minimum du tableau T par le maximum
  - Recommencer pour i+1

Tri par remplacement-2

```
Algorithme tri_remplacement
Variables   i,j,n: Entiers
            Tableau  T[n:entier],Ttrie[n:entier]:réels
             max:réel
Début
  max ← maximum(T);
  i ← 0;
  Tant que (i<n-1 ) Faire
    j ← indice_min(T);
    Ttrie[i] ← T[j];
    T[j] ← max;
    i ← i+1;
  FinTantQue
  Ttrie[n-1] ← max;
Fin
```

```
Fonction maximum(Tableau T[n:entier]): réel
Variables  i :entier
           max :réel
Début
  max ← T[0];
  Pour i allant de 1 à n-1
    Si ( T[i]>max ) alors
      max ← T[i];
    Finsi
  FinPour
  Retourne (max)
Fin
```

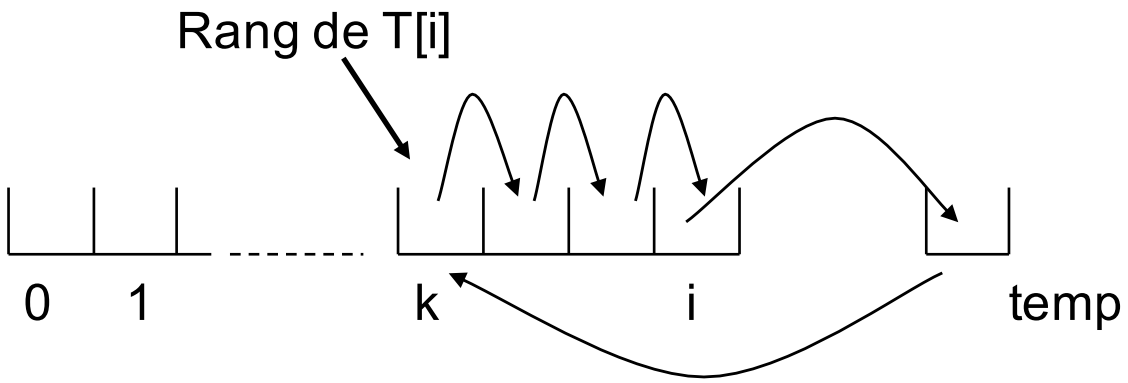
Tri par remplacement-3

- Pour chaque élément rangé dans le tableau T trié, il faut parcourir tout le tableau T et non une partie du tableau T
- Nécessite un 2ème tableau, or si le nombre d'éléments à trier est important, cet algorithme requiert donc un espace mémoire double.

Tri par insertion-1

- Cette méthode de tri insère (au $i^{ème}$ passage) le $i^{ème}$ élément T[i] à la bonne place parmi T[1],T[2]...T[i-1].
- Après l'étape i, tous les éléments entre la première et la $i^{ème}$ position sont triés.
- Il existe plusieurs méthode de tri par insertion selon le principe qui est utilisé pour rechercher le rang de l'élément à insérer parmi les éléments du début de la liste déjà triés

Tri par insertion-2



- Principe de l'algorithme :
  - Pour i allant de 1 à n-1

    déplacer T[i] vers le début du tableau jusqu'à

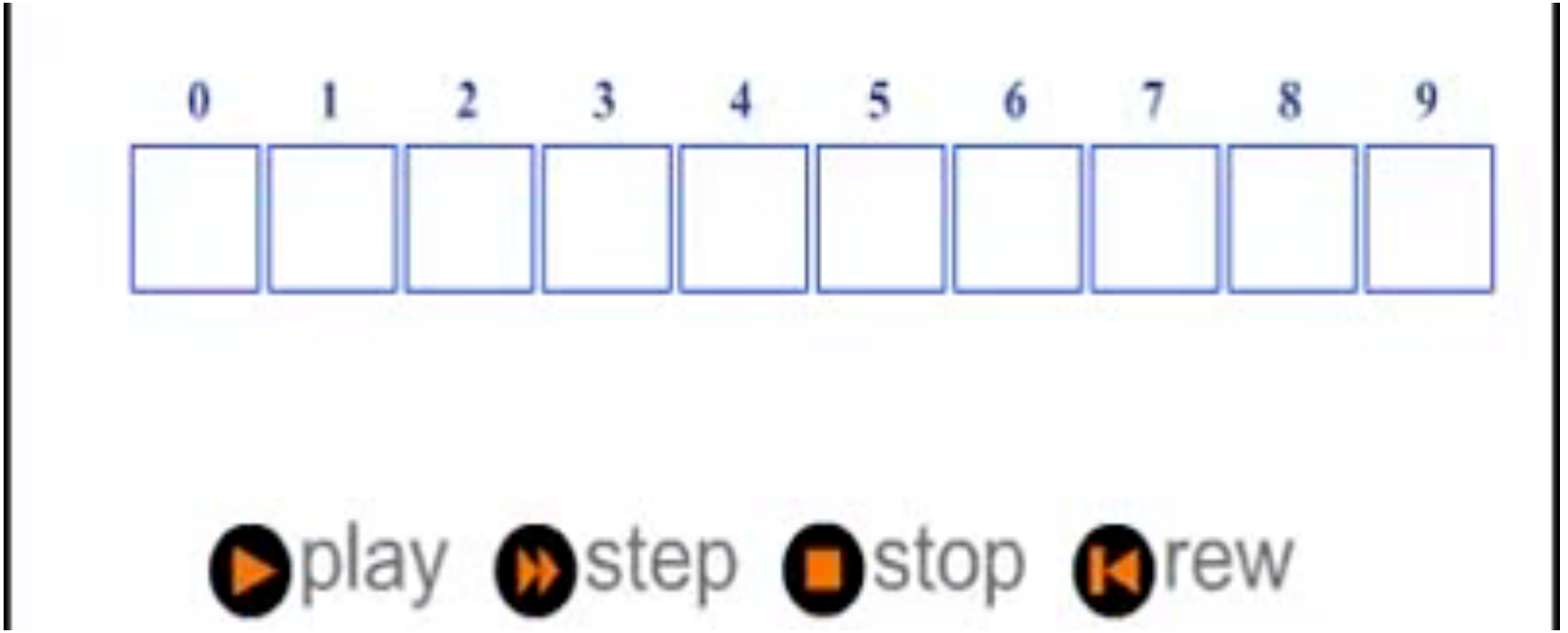
    la position j<=i telle que

    T[j] < T[k] pour j<=k<i et (ou bien T[j]>=T[j-1] ou bien j=1).

Tri par insertion-3

4	2	0	5	3	Vecteur de départ
2	4	0	5	3	Les cellules 1 à 2 sont triées
0	2	4	5	3	Les cellules 1 à 3 sont triées
0	2	4	5	3	Les cellules 1 à 4 sont triées
0	2	3	4	5	Les cellules 1 à 5 sont triées

Tri par insertion-4



Tri par insertion-5

```
Procédure tri_insertion (tableau T[n: entier]:réel)
Variables i,j: Entiers
Début
  Pour i allant de 1 à n-1
    j ← i-1;
    TantQue ( j > =0 et T[j] > T[j+1] ) Faire
      Echanger(T [j+1], T[j] );
      j ← j-1;
    FinTantQue
  FinPour
Fin
```

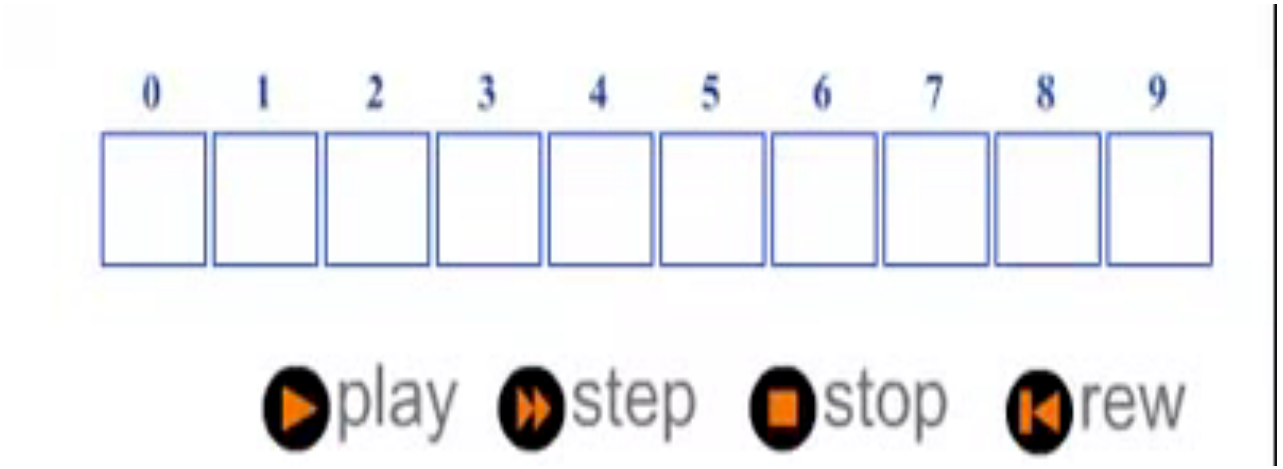
Tri à bulles-1

- Le principe du tri à bulles *(bubble sor*t) est de comparer deux à deux les éléments e_1 et e_2 consécutifs d'un tableau et d'effecteur une permutation si $e_1 > e_2$.
- On continue de trier jusqu'à ce qu'il n'y ait plus de permutation.

Tri à bulles-2

4	2	0	5	3	Vecteur de départ
4	0	2	3	5	Fin du premier passage
0	2	3	4	5	Fin du deuxième et dernier passage

Tri à bulles-3



Tri à bulles-4

```
Procédure Tri_bulles (Tableau  T[n:entier] : réel)
Variables i,j: Entiers
Début
  Pour i allant de n-1 à 1 pas -1
    Pour j allant de 0 à i-1

      Si (T[j] > T[j+1])  Alors
        Echanger(T[j],T[j+1]);
      FinSi
    FinPour
  FinPour
Fin
```