Chapitre 5

Les tableaux

I. Tableaux à une dimension

Définition

Un tableau T est une variable structurée formée d'un nombre entier N de variables simples de même type, qui sont appelées les composantes du tableau. Le nombre de composantes N est alors la dimension du tableau.

On dit encore que T est un vecteur de dimension N.

Utilité

- Un tableau est une structure de données constituée d'un nombre fini d'éléments de même type.
- Lorsque plusieurs données de même type, généralement destinées au même traitement doivent être accessibles le long d'un programme, on propose d'utiliser la structure d'un tableau.

Composantes

Nom : identificateur d'un tableau.

Type_élément : Les éléments d'un tableau sont caractérisés par leur type (entier, réel, caractère,...).

Indice : Tout type dont les éléments possèdent un successeur (les types scalaires), généralement de type entier.

1.1 Déclaration et Mémorisation

Déclaration

<type simple> <Nom tableau> [<dimension>];

Exemples :

int A[25];

float B[100];

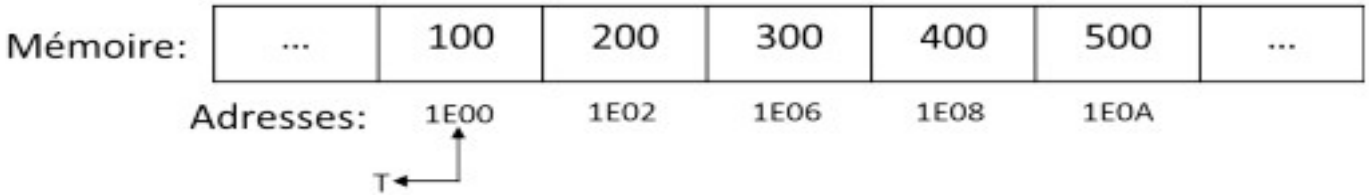
char D[30];

Mémorisation

En C, le nom d'un tableau est le représentant de l'adresse du premier élément du tableau.

Exemple

int T[5] = { 100,200,300,400,500};



Si un tableau est formé de N composantes et si une composante a besoin de M octets en mémoire alors le tableau occupera N*M octets.

2. Initialisation et réservation automatique

Initialisation

Exemple

int A[5] = { 10,20,30,40,50} ;

float B[4]= { -1.05, 3.33 , 87 e -5,-12.3 E4} ;

int C[10] = {0,1,1,0,0,0,0,1,0,1};

Remarque : Si la liste ne contient pas assez de valeurs pour toutes les composantes, les composantes restantes sont initialisées par zéro.

Réservation automatique

Si la dimension n'est pas indiquée explicitement lors de l'initialisation, alors l'ordinateur réserve automatiquement le nombre d'octets nécessaires.

Exemples

int A[ ] = {10,20,30,40,50 };

- réservation de 5*sizeof(int) octets (10 octets)

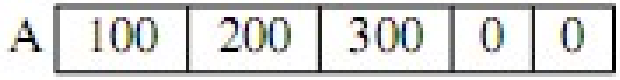
float B[ ] ={-1.05 , 3.33 , 87 e-5, -12.3 E4 };

- réservation de 4*sizeof(float) octets (16 octets)

Exemple

int A[ ] = {100,200,300,400,500} ;

int A[5] = {100,200,300};



int A[3] = {100,200,300,400,500};

ERREUR

3. Accès aux composantes d'un tableau

Considérons un tableau T de dimension N.

- L'accès au premier élément du tableau se fait par T[0]
- L'accès au dernier élément du tableau se fait par T[N-1]

Exemple

int T[5] = { 100,200,300,400,500} ;

Nom: T	100	200	300	400	500
Indice	0	1	2	3	4
Contenu	T[0]	T[1]	T[2]	T[3]	T[4]

4. Chargement d'un tableau

```
void CHARGEMENT(int T[ ], int N)
{
   int i ;
   for (i=0 ; i<N ; i++)
   {
         printf("T[%d] : ",i) ;
         scanf("%d" , &T[i]) ;
   }
}
```

5. Affichage du contenu d'un tableau

```
void AFFICHER(int T[ ],int N)
{
    int i;
    for(i=0; i<N;i++)
            printf("%d\t",T[i]);
}
```

Affichage des adresses des éléments d'un tableau

```
#include<stdio.h>
void main()
{
   int T[5] = {100,200,300};
   int i;
   for(i=0; i<5; i++)
         printf("%d\t",T[i]) ;
   printf("\n");
   for (i=0; i<5; i++)
         printf("%x\t", &T[i]) ;
}
```

Résultat :

```
100 200 300 0 0
21e8 21ea 21ec 21ee 21f0
```

6. Méthodes de tri dans un tableau

A. Tri par sélection (par minimum)

Principe :

Le principe de cette méthode est simple. Elle consiste à :

- Chercher l'indice du plus petit élément du tableau T[0..n-1] et permuter l'élément correspondant avec l'élément d'indice 0 ;
- Chercher l'indice du plus petit élément du tableau T[2..n-1] et permuter l'élément correspondant avec l'élément d'indice 1 ;
- ...
- Chercher l'indice du plus petit élément du tableau T[n-2..n-1] et permuter l'élément correspondant avec l'élément d'indice n-2 ;

```
int INDICE_MIN (int T[ ],int d,int N)
{
    int posmin=d, j;
    for(j=d+1; j<N;j++)
            if(T[j]<T[posmin] )
                posmin=j;
return posmin;
}
```

```
void TRI_SELECTION (int T[ ],int N)
{
   int i, posmin, aux;
   for(i=0; i<N-1;i++)
  {
     posmin= INDICE_MIN(T,i,N);
     if(posmin!=i)
     {
            aux=T[i];
            T[i]=T[posmin];
            T[posmin]=aux;
      }
  }
}
```

Exemple

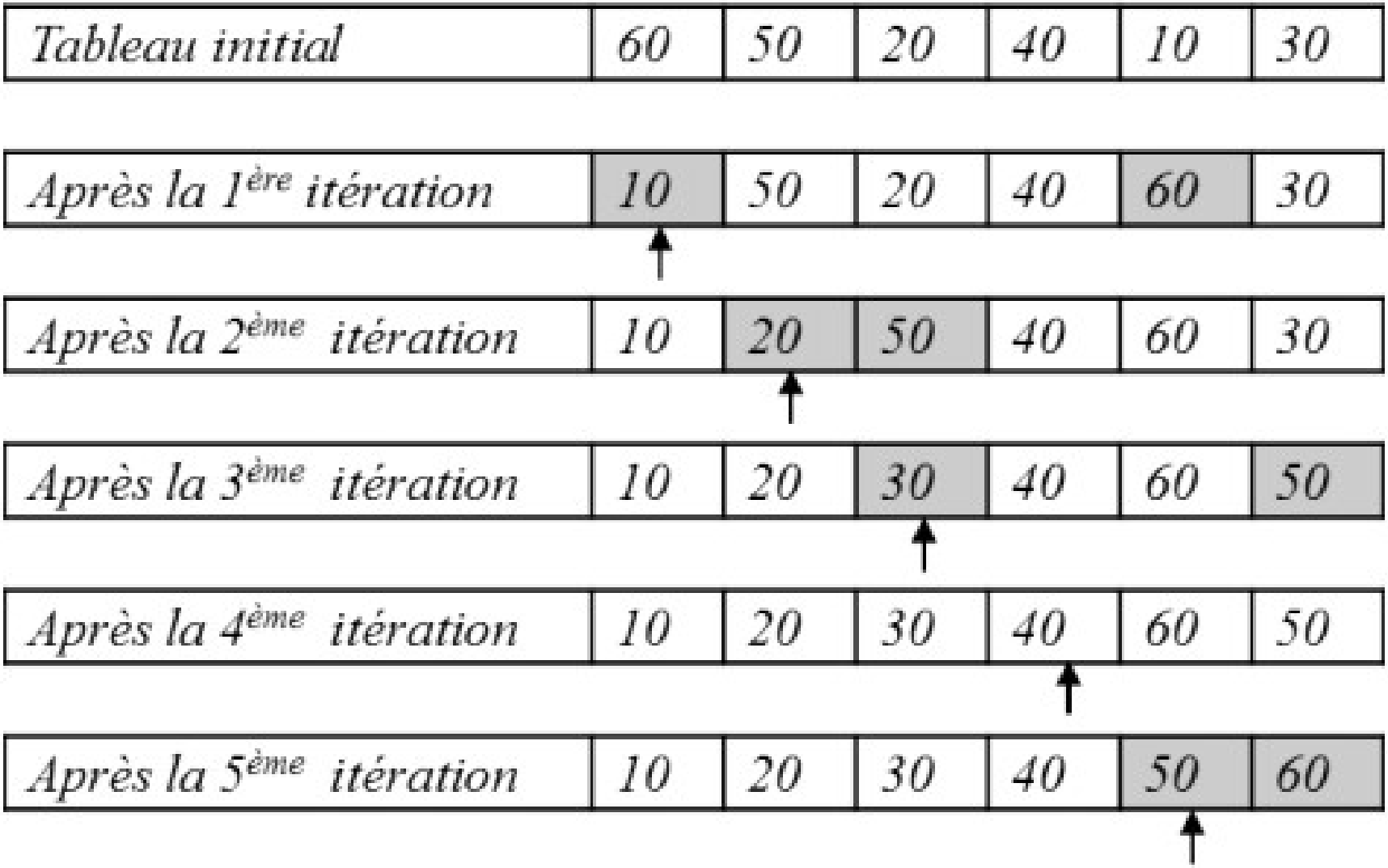


Tableau initial 60 50 20 40 10 30
Après la 1ère itération 10 50 20 40 60 30
Après la 2ème itération 10 20 50 40 60 30
Après la 3ème itération 10 20 30 40 60 50
Après la 4ème itération 10 20 30 40 60 50
Après la 5ème itération 10 20 30 40 50 60

B. Tri à bulles

Principe

Cet algorithme porte le nom de tri bulle car, petit à petit, les plus grands éléments du tableau remontent, par le jeu des permutations, enfin de tableau.

La méthode de tri à bulles consiste à répéter le traitement suivant :

- Parcourir les éléments du tableau de 0 à (n-2) ; si l'élément i est supérieur à l'élément (i+1) , alors on les permute.
- Le programme s'arrête lorsque aucune permutation n'est réalisable après un parcours complet du tableau

```
void TRI_BULLES(int T[],int N)
{
    int i, ok, aux;
    do
    {
            ok=0;
            for(i=0; i<N-1;i++)
               if(T[i]>T[i+1])
              {
                  aux=T[i];
                  T[i]=T[i+1];
                  T[i+1]=aux;
                  ok=1;
               }
            N--;
    }while(N && ok);
}
```

Exemple

Tableau initial	*50*	*30*	*20*	*40*	*10*
1ère étape	*30*	*50*	*20*	*40*	*10*
	30	*20*	*50*	*40*	*10*
	30	*20*	*40*	*50*	*10*
	30	*20*	*40*	*10*	*50*
2ème étape	*20*	*30*	*40*	*10*	*50*
	20	*30*	*10*	*40*	*50*
3ème étape	*20*	*10*	*30*	*40*	*50*
4ème étape	*10*	*20*	*30*	*40*	*50*