



COURS et TP DE LANGAGE C++

Chapitre 6

Les tableaux et les chaînes de caractères

Joëlle MAILLEFERT

joelle.maillefert@iut-cachan.u-psud.fr

IUT de CACHAN

Département GEII 2

Les tableaux à plusieurs dimensions:

Tableaux à deux dimensions:

Déclaration: **type nom[dim1][dim2];** Exemples: **int compteur[4][5];**
float nombre[2][10];

Utilisation: Un élément du tableau est repéré par ses indices. En langage C++ les tableaux commencent aux indices 0. Les indices maxima sont donc dim1-1, dim2-1.

Exemples:

```
compteur[2][4] = 5;  
nombre[i][j] = 6.789;  
cout<<compteur[i][j];  
cin>>nombre[i][j];
```

Il n'est pas nécessaire de définir tous les éléments d'un tableau. Les valeurs non initialisées contiennent alors des valeurs quelconques.

Exercice VI 2: Saisir une matrice d'entiers 2x2, calculer et afficher son déterminant.

Tableaux à plus de deux dimensions:

On procède de la même façon en ajoutant les éléments de dimensionnement ou les indices nécessaires.

INITIALISATION DES TABLEAUX

On peut initialiser les tableaux au moment de leur déclaration:

Exemples:

int liste[10] = {1,2,4,8,16,32,64,128,256,528};

float nombre[4] = {2.67,5.98,-8.0,0.09};

int x[2][3] = {{1,5,7},{8,4,3}}; // 2 lignes et 3 colonnes

TABLEAUX ET POINTEURS

En déclarant un tableau, on dispose d'un pointeur (adresse du premier élément du tableau).
Le nom d'un tableau est un pointeur sur le premier élément.

Les tableaux à une dimension:

Les écritures suivantes sont équivalentes:

<code>/* Allocation dynamique pendant l'exécution du programme */ int *tableau = new int[10];</code>	<code>/* Allocation automatique pendant la compilation du programme */ int tableau[10];</code>	déclaration
<code>*tableau</code>	<code>tableau[0]</code>	le 1er élément
<code>*(tableau+i)</code>	<code>tableau[i]</code>	un élément d'indice i
<code>tableau</code>	<code>&tableau[0]</code>	adresse du 1er élément
<code>(tableau + i)</code>	<code>&(tableau[i])</code>	adresse d'un élément i

Il en va de même avec un tableau de réels (float).

Remarques:

- La déclaration d'un tableau entraîne automatiquement la réservation de places en mémoire. C'est aussi le cas si on utilise un pointeur et l'allocation dynamique en exploitant l'opérateur **new**.
- On ne peut pas libérer la place réservée en mémoire pour un tableau créé en allocation automatique (la réservation étant réalisée dans la phase de compilation – en dehors de l'exécution). Par contre, en exploitant l'allocation dynamique via un pointeur, la primitive **delete** libère la mémoire allouée.

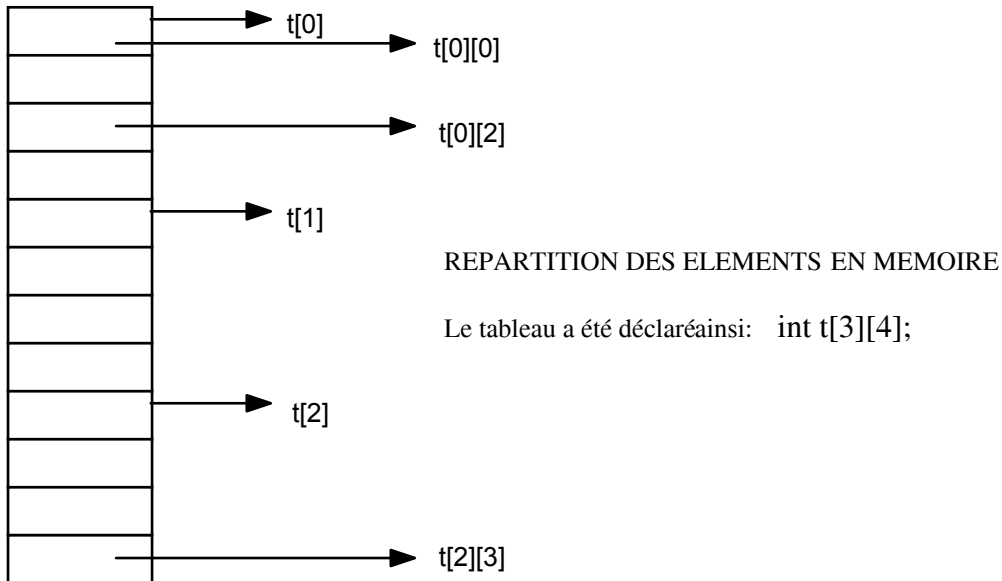
Les tableaux à plusieurs dimensions:

Un tableau à plusieurs dimensions est un pointeur de pointeur.

Exemple: `int t[3][4];` t est un pointeur de 3 tableaux de 4 éléments ou bien de 3 lignes à 4 éléments.

Les écritures suivantes sont équivalentes:

t[0]	&t[0][0]	t	adresse du 1er élément
t[1]	&t[1][0]		adresse du 1er élément de la 2e ligne
t[i]	&t[i][0]		adresse du 1er élément de la ième ligne
t[i]+1	&(t[i][0])+1		adresse du 1er élément de la ième +1 ligne



Exercice VI 3:

Un programme contient la déclaration suivante:

```
int tab[10] = {4,12,53,19,11,60,24,12,89,19};
```

Compléter ce programme de sorte d'afficher les adresses des éléments du tableau.

Exercice VI 4:

Un programme contient la déclaration suivante:

```
int tab[20] = {4,-2,-23,4,34,-67,8,9,-10,11, 4,12,-53,19,11,-60,24,12,89,19};
```

Compléter ce programme de sorte d'afficher les éléments du tableau avec la présentation suivante:

4	-2	-23	4	34
-67	8	9	-10	11
4	12	-53	19	11
-60	24	12	89	19

LES CHAINES DE CARACTERES

En langage C++, les chaînes de caractères sont des **tableaux de caractères**. Leur manipulation est donc analogue à celle d'un tableau à une dimension:

Déclaration: **char nom[dim];** ou bien **char *nom=new char[dim];**

Exemple: **char texte[20];** ou bien **char *texte=new char[20];**
Il faut toujours prévoir une place de plus pour une chaîne de caractères. En effet, en C ou en C++, une chaîne de caractère se termine toujours par le caractère **NUL** ('\0'). Ce caractère permet de détecter la fin de la chaîne lorsque l'on écrit des programmes de traitement.

Affichage à l'écran:

On utilise l'opérateur **cout** :
char texte[10] = « BONJOUR »;
cout<<"VOICI LE TEXTE:"<<texte;

Saisie:

On utilise l'opérateur **cin** :
char texte[10];
cout<<"ENTRER UN TEXTE: ";
cin>> texte;

Remarque: **cin** ne permet pas la saisie d'une chaîne comportant des espaces : les caractères saisis à partir de l'espace ne sont pas pris en compte (l'espace est un délimiteur au même titre que LF).

A l'issue de la saisie d'une chaîne de caractères, le compilateur ajoute '\0' en mémoire après le dernier caractère.

Exercice VI 5:

Saisir une chaîne de caractères, afficher les éléments de la chaîne caractère par caractère.

Exercice VI 6:

Saisir une chaîne de caractères. Afficher le nombre de lettres e de cette chaîne.

Fonctions permettant la manipulation des chaînes :

Les bibliothèques fournies avec les compilateurs contiennent de nombreuses fonctions de traitement des chaînes de caractères. En BORLAND C++, elles appartiennent aux bibliothèques **string.h** ou **stdlib.h**. En voici quelques exemples:

Générales (string.h):

Nom : **strcat**

Prototype : **void *strcat(char *chaine1, char *chaine2);**

Fonctionnement : concatène les 2 chaînes, résultat dans chaine1, renvoie l'adresse de chaine1.

Exemple d'utilisation :

```
char texte1[30] = "BONJOUR "; // remarquer l'espace après le R
char texte2[20] = "LES AMIS";
strcat(texte1, texte2);
// texte2 est inchangée, texte1 vaut maintenant "BONJOUR LES AMIS"
```

Nom : **strlen**

Prototype : **int strlen(char *chaine);**

Fonctionnement : envoie la longueur de la chaîne ('\0' non comptabilisé).

Exemple d'utilisation :

```
char texte1[30] = "BONJOUR";
int L = strlen(texte1);
cout<< "longueur de la chaîne : "<<L; // L vaut 7
```

Nom: **strrev**

Prototype : **void *strrev(char *chaine);**

Fonctionnement : inverse la chaîne et, renvoie l'adresse de la chaîne inversée.

Exemple d'utilisation :

```
char texte1[10] = "BONJOUR";
strrev(texte1); // texte1 vaut maintenant "RUOJNOB"
```

Comparaison (string.h):

Nom : **strcmp**

Prototype : **int strcmp(char *chaine1, char *chaine2);**

Fonctionnement : renvoie un nombre:

- positif si chaine1 est supérieure à chaine2 (au sens de l'ordre alphabétique)
- négatif si chaîne1 est inférieure à chaine2
- nul si les chaînes sont identiques.

Cette fonction est utilisée pour classer des chaînes de caractères par ordre alphabétique.

Exemple d'utilisation :

```
char texte1[30] = "BONJOUR ";
char texte2[20] = "LES AMIS";
int n = strcmp(texte1, texte2); // n est positif
```

Copie (string.h):

Nom : strcpy

Prototype : **void *strcpy(char *chaine1, char *chaine2);**

Fonctionnement : recopie chaine2 dans chaine1 et renvoie l'adresse de chaîne1.

Exemple d'utilisation :

```
char texte2[20] = "BONJOUR ";
char texte1[20];
strcpy(texte1, texte2);
// texte2 est inchangée, texte1 vaut maintenant "BONJOUR "
```

Recopie (string.h):

Ces fonctions renvoient l'adresse de l'information recherchée en cas de succès, sinon le pointeur NULL (c'est à dire le pointeur dont la valeur n'a jamais été initialisée).

void *strchr(chaine, caractere); recherche le caractère dans la chaîne.

void *strrchr(chaine, caractere); idem en commençant par la fin.

void *strstr(chaine, sous-chaîne); recherche la sous-chaîne dans la chaîne.

Exemple d'utilisation :

```
char texte1[30] ;
cout<< "SAISIR UN TEXTE :";
cin>>texte1;
if( strchr(texte1, A) != NULL) cout<<"LA LETTRE A EXISTE DANS CE TEXTE ";
else cout<<<< "LA LETTRE A NEXISTE PAS DANS CE TEXTE ";
```

Conversions (stdlib.h):

int atoi(char *chaine); convertit la chaîne en entier

float atof(char *chaine); convertit la chaîne en réel

Exemple d'utilisation :

```
char texte[10]; int n;
cout<<"ENTRER UN TEXTE: "; cin>>texte;
n = atoi(texte) ;
cout<<n;
// affiche 123 si texte vaut "123", affiche 0 si texte vaut "bonjour"
void *itoa(int n, char *chaîne, int base);
// convertit un entier en chaîne:
// base: base dans laquelle est exprimé le nombre,
// cette fonction renvoie l'adresse de la chaîne.
```

Exemple d'utilisation :

```
char texte[10];
int n=12;
itoa(12, texte, 10); // texte vaut "12"
```

Pour tous ces exemples, la notation void signifie que la fonction renvoie un pointeur (l'adresse de l'information recherchée), mais que ce pointeur **n'est pas typé**. On peut ensuite le typer à l'aide de l'opérateur cast.*

Exemple:

```
char *adr;
char texte[10] = "BONJOUR";
adr=(char*) strchr(texte, 'N'); //adr pointe sur l'adresse de la lettre N
```

Exercice VI 7:

L'utilisateur saisit le nom d'un fichier. Le programme vérifie que celui-ci possède l'extension .PAS

Exercice VI 8:

Un oscilloscope à mémoire programmable connecté à un PC renvoie l'information suivante sous forme d'une chaîne de caractères terminée par '\0' au PC:

"CHANNELA 0 10 20 30 40 30 20 10 0 -10 -20 -30 -40 -30 -20 -10 -0"

Afficher sur l'écran la valeur des points vus comme des entiers. On simulera la présence de l'oscilloscope en initialisant une chaîne de caractères char mesures [100].

CORRIGE DES EXERCICES

Exercice VI 1:

```
#include <iostream.h>
#include <math.h>
#include <conio.h>

void main()
{
    float nombre[10], moyenne = 0, ecart_type = 0;
    int i;

    // saisie des nombres

    cout <<"SAISIR 10 NOMBRES SEPARES PAR RETURN :\n";
    for(i=0;i<10;i++)
    {
        cout<<"nombre["<<i<<" ] = ";
        cin >>nombre[i];
    }

    // calcul de la moyenne

    for(i=0;i<10;i++)
    {
        moyenne = moyenne + nombre[i];
    }
    moyenne = moyenne/10;

    // calcul de l'écart type

    for(i=0;i<10;i++)
    {
        ecart_type =
            ecart_type + (nombre[i]-moyenne)*(nombre[i]-moyenne);
    }
    ecart_type = sqrt(ecart_type)/10; // racine

    cout<<"MOYENNE = "<<moyenne<<" ECART_TYPE = "<<ecart_type;

    cout<<"\nPOUR CONTINUER FRAPPER UNE TOUCHE : ";
    getch();
}
```

Exercice VI 2:

```
#include <iostream.h>
#include <conio.h>

void main()
{
    int mat[2][2], det;

    // saisie

    cout<<"ENTRER SUCCESSIVEMENT LES VALEURS DEMANDEES : \n";
    cout<<"mat[0][0] = ";
    cin>>mat[0][0];
    cout<<"mat[1][0] = ";
    cin>>mat[1][0];
    cout<<"mat[0][1] = ";
    cin>>mat[0][1];
    cout<<"mat[1][1] = ";
    cin>>mat[1][1];

    // calcul

    det = mat[0][0]*mat[1][1]-mat[1][0]*mat[0][1];

    // affichage
    cout<<"DETERMINANT = "<<det;
    cout<<"\nPOUR CONTINUER FRAPPER UNE TOUCHE: ";
    getch();
}
```

Exercice VI 3:

```
#include <iostream.h>
#include <conio.h>

void main()
{
    int tab[10]={4,12,53,19,11,60,24,12,89,19};
    cout<<"VOICI LES ELEMENTS DU TABLEAU ET LEURS ADRESSES:\n";
    for(int i=0;i<10;i++)
    {
        cout<<"ELEMENT NUMERO "<<i<<"="<<tab[i];
        cout<<" ADRESSE="<<tab+i<<"\n";
    }
    cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE: "; getch();
}
```

Exercice VI 4:

```
#include <iostream.h>
#include <conio.h>

void main()
{
    int tab[20] =
        {4,-2,-23,4,34,-67,8,9,-10,11, 4,12,-53,19,11,-60,24,12,89,19};
    cout<<"VOICI LE TABLEAU:\n\n";
    for(int i=0;i<20;i++)
        if (((i+1)%5)==0)
        {
            cout<<"\t"<<tab[i]<<"\n";
        }
        else
        {
            cout<<"\t"<<tab[i];
        }
    cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE : "; getch();
}
```

Exercice VI 5:

```
#include <iostream.h>
#include <conio.h>

void main()
{
    char i,*phrase = new char[20]; // réserve 20 places

    cout<<"ENTRER UNE PHRASE: ";
    cin>>phrase; // saisie

    cout<<"VOICI LES ELEMENTS DE LA PHRASE:\n";
    for(i=0;phrase[i] != '\0';i++)
    {
        cout<<"LETTRE : "<<phrase[i]<<"\n";
    }

    delete phrase;

    cout<<"\nPOUR CONTINUER FRAPPER UNE TOUCHE : "; getch();
}
```

Exercice VI 6:

```
#include <iostream.h>
#include <conio.h>

void main()
{
    char *phrase = new char[20]; // réserve 20 places
    int compt_e = 0;

    cout<<"ENTRER UNE PHRASE:";
    cin>>phrase; // saisie

    for(int i=0;phrase[i]!='\0';i++)
    {
        if(phrase[i]=='e') compt_e++;
    }
    cout<<"NOMBRE DE e : "<<compt_e;
    delete phrase;
    cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
}
```

Exercice VI 7:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

void main()
{
    char *nom = new char[30];
    char *copie = new char[30];
    int n;
    cout<<"\nNOM DU FICHER (.PAS):"; cin>>nom;
    strcpy(copie,nom);
    strrev(copie); //chaîne inversée
    n = strnicmp("SAP.",copie,4); // n vaut 0 si égalité
    if(n!=0) cout<<"\nLE FICHER N'EST PAS DE TYPE .PAS\n";
    else cout<<"\nBRAVO CA MARCHE\n";
    delete nom; delete copie;
    cout<<"\nPOUR CONTINUER FRAPPER UNE TOUCHE "; getch();
}
```

Exercice VI 8:

```
#include <iostream.h>
#include <conio.h>
#include <stdlib.h>

void main()
{
    char mesures[100] =
        "CHANNELA 0 10 20 30 40 30 20 10 0 -10 -20 -30 -40 -30 -20 -10 0";
    int i,j,val[20],nombre_val=0;
    char temp[4]; // chaîne temporaire

    // recherche des nombres

    for(i=9;mesures[i]!='\0';i++)
    {
        for(j=0;(mesures[i]!=' ') && (mesures[i]!='\0');j++)
        {
            temp[j]=mesures[i];
            i++;
        }
        temp[j] = '\0'; // On borne la chaîne
        // Conversion de la chaîne temporaire en nombre
        val[nombre_val] = atoi(temp);
        nombre_val++;
    }

    // Affichage du résultat

    clrscr();
    for(i=0;i<nombre_val;i++)
    {
        cout<<"val["<<i<<"]="<<val[i]<<"\n";
    }

    cout<<"POUR SORTIR FRAPPER UNE TOUCHE : ";

    getch();
}
```