Tableaux

Mercredi 25 Octobre 2017

Michael FRANÇOIS

francois@esiea.fr

esiea
ECOLE D'INGENIEURS
DU MONDE NUMERIQUE

Objectif de ce cours

- Comprendre le concept de tableau.
- Savoir déclarer, remplir et manipuler des tableaux.
- Comprendre les particularités d'un tableau de caractères.
- Voir quelques exemples pratiques.

Intérêt des tableaux

- Jusqu'à présent, on a utilisé des variables déclarées une a une :

```
int varInteger = 2;
double varFloatingPoint = 4.5;
char varCharacter = 'a';
```

- Une limite opérationnelle s'impose quand le nombre de variables à manipuler explose !

Quelques exemple :

• Une image en niveaux de gris prise depuis un APN, c'est 12 millions de valeurs (pixels) indépendantes :

```
int pixel1;
//.... (il manque 11 999 998 lignes)
int pixel12000000;
```

• La température journalière à midi à Paris sur 10 ans :

```
double dayTemp1;
//.... (il manque 3650 lignes)
double dayTemp3652;
```

Comment s'en sortir ?

- En regroupant l'ensemble des variables sous un seul identifiant (nom).

- En distinguant chaque variable par un numéro d'ordre qui est unique dans l'ensemble (comme un ordre chronologique, un classement ou une position).

Définition d'un tableau

Une définition de la notion de tableau pourrait être :

- un nom unique désignant un ensemble de variables indépendantes entre elles ;
- toutes les variables sont du même type ;
- chaque variable est distinguable des autres grâce à son numéro d'ordre dans l'ensemble (comme un ordre chronologique, un classement ou une position).

Les variables qui composent un tableau peuvent être :

- de type élémentaire `char`, `int`, `double`, etc. ;
- de type tableau (une image peut être vue comme un tableau des lignes qui la compose) ;
- de type plus complexe (et pas encore vu) comme les pointeurs ou les structures de données.

Attention à ne pas mélanger des torchons et des serviettes :
TOUTES LES VARIABLES D'UN TABLEAU DOIVENT ÊTRE DU MÊME TYPE ! ! !

Comment déclarer un tableau en C ?

La déclaration d'un tableau en C se fait de cette façon :

```
type name[nbCases];
```

- `type` désigne le type de stockage (`char`, `int`, `double`, `...`) de tous les éléments du tableau ;
- `name` désigne le nom du tableau pour le reste du programme ;
- `nbCases` est un entier qui représente le nombre total de cases du tableau.

Quelques exemples :

- Une image en niveaux de gris prise depuis un APN, c'est 12 millions de valeurs de pixels indépendants :

```
int image[12000000];
```

- La température journalière à midi à Paris sur 10 ans :

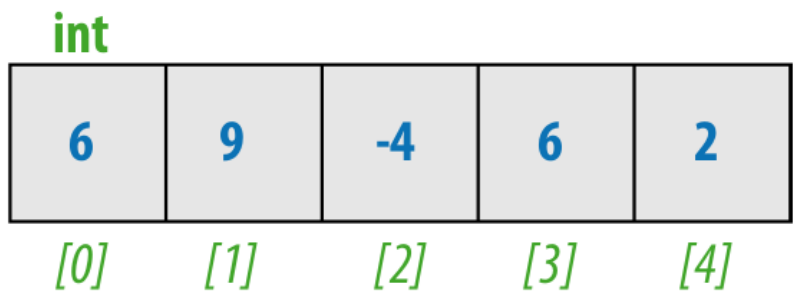
```
double dayTemp[3652];
```

Lors d'une déclaration de tableau :

- il n'y a QUE la réservation de place mémoire ;
- il n'y a PAS d'initialisation de celle-ci (on prend la mémoire dans l'état où elle est, comme avec les variables) ;
- il FAUT donc initialiser les cases du tableau, comme pour n'importe quelle variable.

NB : pour les tableaux de petite taille, il est possible de déclarer et d'initialiser les valeurs des cases du tableau en même temps (si elles sont connues d'avance).

```
int tab[5] = {6,9,-4,6,2};
```

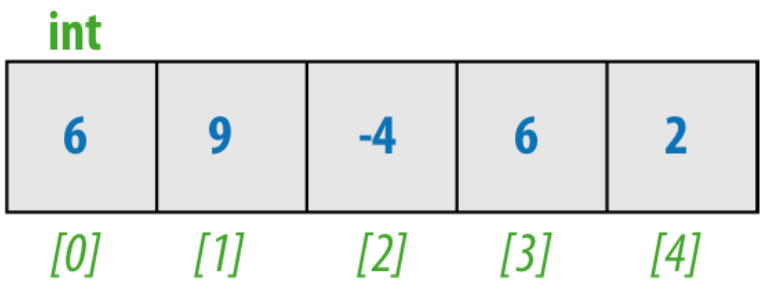


Accès aux valeurs d'un tableau

Dans la partie "actions" d'une fonction :

- l'accès à la valeur en lecture ou en écriture d'une case du tableau se fait en utilisant l'opérateur [] ;
- l'indice (le numéro d'ordre) des cases du tableau débute toujours à 0.

```
array[0] = 6;
array[1] = 9;
array[2] = -4;
array[3] = 6;
array[4] = 2;
```



On considère le programme suivant :

```c
#include <stdio.h>

int main(){
  /* partie declarative */
  int array[3];
  int numCase;
  /* partie actions */
  array[0] = 6; // acces en ecriture
  array[1] = 9;
  array[2] = -4;
  for (numCase = 0; numCase < 3; numCase++){
   printf("array[%d] = %d\n", numCase, array[numCase]); // acces en lecture
  }
  return 0;
}
```

Résultat de l'affichage :

```
array[0] = 6
array[1] = 9
array[2] = -4
```

Passage d'un tableau comme argument d'une fonction

• Comme n'importe quelle variable, un tableau peut être passé en argument à une fonction.

```
void initArray(int array[], int uniqueValue){
   int numCase;
   for (numCase = 0; numCase < 5; numCase++){
     array[numCase] = uniqueValue;
   }
}
int main(){
  int myArray[5];
  initArray(myArray, 0);
  return 0;
}
```

• Le problème essentiel de `initArray` est que la fonction ne pourra être utilisée que sur des tableaux de 5 cases car la taille est fixée dans le corps de la fonction.

• Une solution est de passer la taille du tableau en paramètre pour que la fonction puisse être utilisée quelque que soit le nombre de cases du tableau.

```c
void initTab(int nbCases, int array[], int uniqueValue){
   int numCase;
   for (numCase = 0; numCase < nbCases; numCase++){
     array[numCase] = uniqueValue;
   }
}
int main(){
  int myArray[5];
  initTab(5, myArray, 0);
  return 0;
}
```

Exercice : compléter ce code par la fonction `showArray` qui affiche sur une même ligne un tableau et passe ensuite à la ligne.

Solution :

```c
#include <stdio.h>
void showArray(int nbCases, int array[]){
  int numCase;
  for(numCase = 0; numCase < nbCases; numCase++){
    printf("%d ", array[numCase]);
  }
  printf("\n");
}
void initArray(int nbCases, int array[], int uniqueValue){
   int numCase;
   for(numCase = 0; numCase < nbCases; numCase++){
     array[numCase] = uniqueValue;
   }
}
int main(){
  int myArray[5];
  initArray(5, myArray,0);
  showArray(5, myArray);
  return 0;
}
```

```
0 0 0 0 0
```

Tableau de caractères

• Un caractère est une lettre de l'alphabet (minuscules, majuscules), un chiffre (0...1), une ponctuation, ... bref, un symbole du clavier.

• Problème : la mémoire ne peut stocker que des valeurs numériques ?

• Solution : faire correspondre de manière unique (bijection) un caractère à un nombre !!!

• Cette table de transcodage s'appelle la table **ASCII** :
(**A**merican **S**tandard **C**ode for **I**nformation **I**nterchange)
(Code américain normalisé pour l'échange d'information).

• La table ASCII est normalisée entre les valeurs 0 et 127. Elle est codée sur 7 bits même si un octet est utilisé. Pour cela le bit de poids fort est mis à 1.

• Les accents ou les autres alphabets ne sont pas gérés. Pour cela, d'autres tables ont été crées comme l'UNICODE (UTF-8) par exemple. Elles sont codées sur plus d'octets, .

Table ascii (à partir du caractère 32) :

Dec	Car	Dec	Car	Dec	Car	Dec	Car	Dec	Car	Dec	Car
32	space	48	0	64	@	80	P	96	‘	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	\$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	’	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	del

• Pour ne pas avoir à connaître la table ASCII par cœur, on peut utiliser l'opérateur ' pour convertir un caractère en nombre ASCII.

• Par exemple, le code suivant :

```
char character;
character = 'A';
printf("Valeur en memoire = %d\n", character);
```

Affichera :

```
Valeur en memoire = 65
```

• Pour l'opération inverse, i.e. passer du nombre stocké en mémoire à l'affichage du caractère, il suffit d'utiliser le descripteur %c dans la fonction `printf`.

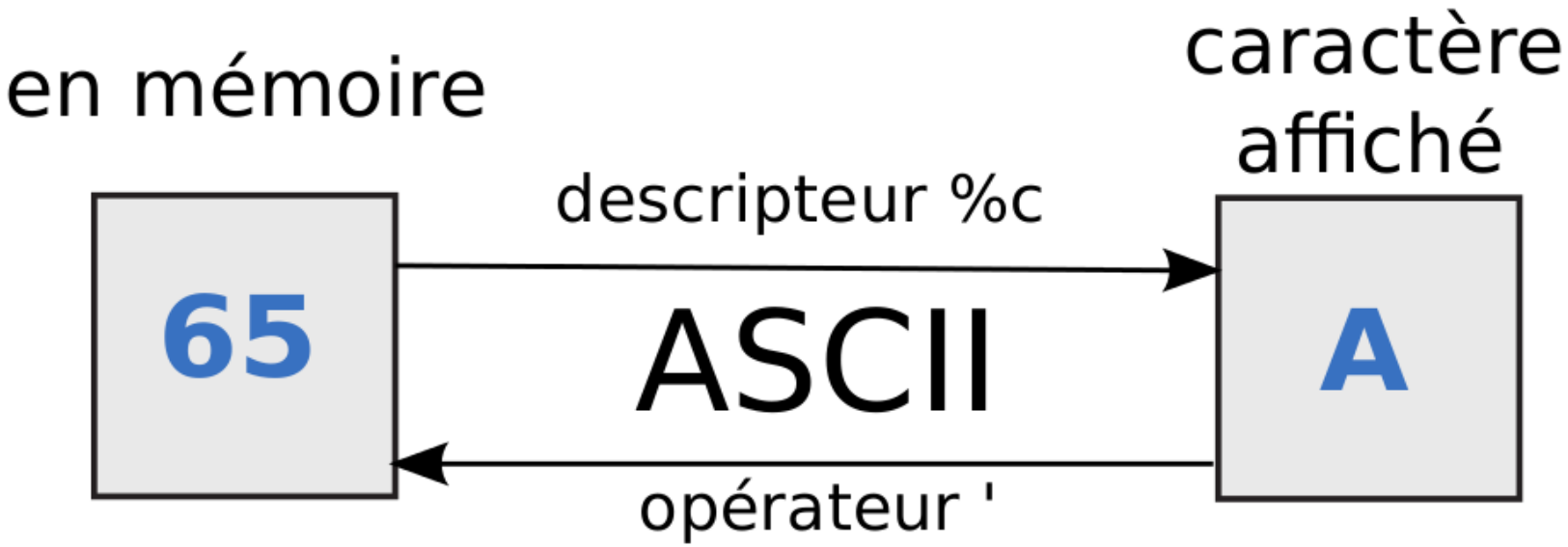
• Par exemple, le code suivant :

```
char character;
character = 'A';
printf("%c vaut en ascii %d\n", character, character);
```

affichera :

```
A vaut en ascii 65
```

En résumé :


en mémoire
caractère affiché
descripteur %c
65
ASCII
A
opérateur '

• Un tableau de caractères en mémoire est tout simplement un tableau d'ENTIERS !!!

• Ce sont de vrais entiers, on peut faire des opérations mathématiques avec.
Par exemple :

```
char character = 'C';
printf("%d - 2 vaut %d\n", character, character - 2);
printf("%c - 2 vaut %c\n", character, character - 2);
```

affichera :

```
67 - 2 vaut 65
C - 2 vaut A
```

• La déclaration, l'accès aux valeurs et le passage en argument de fonction d'un tableau de caractères suit les mêmes règles que pour n'importe quel tableau.

Bibliographie

- L. BEAUDOIN, Introduction à l'algorithmique et au langage C (Tableaux), cours 1A 2016-2017 ESIEA-Paris.
- C. DELANNOY, Langage C, éditions EYROLLES, 4ème tirage 2005.