Cours n° B.2

Tableaux, structures, chaînes de caractères

Tableaux

- ☞ Un tableau est une zone mémoire contigue contenant plusieurs valeurs d'un même type
- ☞ La taille d'un tableau est fixée lors de la définition de celui-ci.
  Elle ne peut pas être modifiée.
- ☞ Les tableaux sont toujours indicés par des valeurs entières de 0 à `taille - 1`
- ☞ **Il n'y a pas de marqueur de fin de tableau** (donc pas de vérification automatique de dépassement ni à la compilation ni à l'exécution)

Exemple de définitions :

`int tent[10]` ..... tableau de 10 entiers
`char tcar[8]` ..... tableau de 8 caractères

Exemples d'indexations :

`tent[3] = 1` ..... 4^{e} case du tableau
`tcar[6+1] = 'a'` ..... dernière case du tableau

```c
#include <stdio.h>

int main (int argc, char ** argv)
{
  int t[10];

  int i;

  for (i = 0; i < 10; i += 1)
    {
      t[i] = 65 + i;
    }

  /* Correct */
  for (i = 0; i < 10; i ++)
    {
      printf ("%d ", t[i]);
    }
  printf("\n");

  /* Correct ? */
  for (i = 0; i <= 10; i ++)
    {
      printf ("%c ", t[i]);
    }
  printf("\n");

  return 0;
}
```

Les structures

Il est possible de créer des objets composites par l'entremise des **structures**.
Déclaration :

```
struct nom_de_la_structure
{
    type_du_membre_1 nom_du_membre_1;
    ...
    type_du_membre_n nom_du_membre_n;
};
```

La structure permet d'associer les membres divers, éventuellement de types différents :

```
struct personne
{
    char nom[8];
    int numero;
};
```

ou encore

```
struct point
{
    int x;
    int y;
};
```

Opérations sur les structures

Les seules opérations autorisées sur les structures sont :

- ☞ copie ou affectation d'une structure (permettant d'utiliser une structure comme paramètre ou retour d'une fonction) ;
- ☞ récupération de l'adresse avec l'opérateur & ;
- ☞ accès aux membres.

Attention :

- ☞ il n'est pas possible de comparer (avec l'opérateur ==) des structures !
- ☞ il est possible d'initialiser une structure par une liste de valeurs.

Retour sur les structures

```
struct point p1,p2;
```
déclaration de variables

```
struct point origine = {0, 0};
```
initialisation

```
p1.x = 2;
```
accès aux membres

```
p1.y = 3;
```

```
p2 = p1;
```
affectation

```
printf("%d %d\n",p2.x,p2.y);
```
affichage 2 3

```
if (p1 == p2)
   printf("egal\n");
```
Interdit !!!

Chaînes de caractères

- ☞ On définit les chaînes de caractères par une **convention** :
  - ☛ Les chaînes de caractères sont des tableaux de caractères
  - ☛ Un caractère est un **entier** défini sur un octet dans le code ASCII. Cet entier représente l'indice du caractère représenté dans la table ASCII
  - ☛ Le marqueur de fin est le caractère `'\0'`, code ASCII 0

- ☞ Les fonctions d'entrées/sorties utilisent cette convention

- ☞ Les constantes de type chaînes de caractères sont exprimés entre guillemets : `"bonjour"`

Exemple

```c
int sont_egales(const char chaine1[], const char chaine2[])
{
    int i = 0;
    while (chaine1[i] == chaine2[i] && chaine1[i] != '\0')
        i++;
    if (chaine1[i] == chaine2[i])
        return 1; /* chaines gales */
    return 0; /* chaines diffrentes */
}
```

Extrait de table ASCII

Dec.	Hex.	Caractère	Dec.	Hex.	Caractère
65	41	A	66	42	B
67	43	C	68	44	D
69	45	E	70	46	F
71	47	G	72	48	H
73	49	I	74	4A	J
75	4B	K	76	4C	L
77	4D	M	78	4E	N
79	4F	O	80	50	P
81	51	Q	82	52	R
83	53	S	84	54	T
85	55	U	86	56	V
87	57	W	88	58	X
89	59	Y	90	5A	Z

Les définitions de type

Il est possible de nommer des types par l'instruction :

```
typedef un_type nom_de_type;
```

```
typedef int entier;
```

```
entier i = 0;
```

```
typedef char * chaine;
```

```
chaine c = "aaa";
```

```
typedef struct point point;
```

```
point p1;
```

```
typedef struct personne etudiant;
```

```
etudiant e;
```

```
typedef struct {
   point haut;
   point bas;
} rectangle;
```

```
rectangle r; r.haut.x = 2; ...
```