LANGAGE C – FICHE 3

Tableaux

**Classification des types de variables**

Les variables peuvent être de différents types, classés comme suit.

Types de variables

Types simples

Types composés

Booléens (bool)

Caractères (char)

Entiers

(short, int, long)

Flottants

(float, double)

Homogènes

Hétérogènes

Tableaux

Enregistrements

**Accès et organisation en mémoire d’un tableau**

Un tableau est une collection ordonnée d’éléments de même type dont la position en mémoire est repérée par un indice. L’indice du premier élément est 0.

Le stockage en mémoire se fait de manière contigüe. L’accès à un élément se fait grâce aux indices.

0

1

2

3

4

5

6

7

8

9

t[3]

Indices

Tableau t

Le tableau est caractérisé par ses dimensions (ou tailles), notée(s) |t|, à savoir le nombre d’éléments qu’il contient.

Le tableau représenté ci-dessus est de taille 10 et l’espace en mémoire nécessaire à son stockage vaut si les éléments stockés sont de type int.

L’accès à un élément se fait en nommant le tableau puis en indiquant l’indice de l’élément entre crochets : t [3] est le quatrième élément du tableau, donc situé à l’indice 3.

Un tableau peut avoir plusieurs dimensions. Le stockage en mémoire se fait alors suivant une organisation à plat et l’accès à un élément se fait à l’aide de plusieurs indices, chacun étant entre crochets.

[ [ ]

[ ]

[ ]

[ ] ]

Tableau t

Stockage en mémoire

[ [ ]

[ ]

[ ]

[ ] ]

t[0] : vecteur 0 ou élément à l’indice 0

t[1] : vecteur 1 ou élément à l’indice 1

t[2] : vecteur 2 ou élément à l’indice 2

t[3] : vecteur 3 ou élément à l’indice 3

Tableau à deux dimensions

t [ i ][ j ] : accès à l’élément d’indice j dans le vecteur i

Un tableau est une structure mutable, à savoir que la valeur associée à un élément du tableau peut être modifiée.

**Déclaration d’un tableau**

La déclaration d’un tableau (ici, tableau à 2 dimensions) se fait grâce à la commande :

type nom\_tableau [dimension1] [dimension2] ;

Le type déclaré correspond au type des éléments qui seront stockés.

La place en mémoire est alors réservée et les valeurs ne sont pas affectées. Les affectations se font alors grâce à une boucle.

Il est possible d’attribuer des valeurs aux éléments dès la déclaration :

|  |  |
| --- | --- |
| **Déclaration** | **Effet** |
| int nom\_tableau [2] = {4} ; | Création d’un tableau nommé nom\_tableau de 2 entiers (type int) ; le premier vaut 4. Le second sera nul. |
| int nom\_tableau [2] = {5, 3} ; | Création d’un tableau de 2 entiers nommé nom\_tableau, valant respectivement 5 et 3.  Lors de la déclaration, les éléments sont séparés par une virgule. |
| int nom\_tableau [2][2] = {{5,3},{1,1}} ; | Création d’un tableau à 2 dimensions. Les valeurs sont spécifiées entre accolades. |

**Chaînes de caractères**

Pour manipuler une chaîne de caractères, la bibliothèque string doit être importée au préalable : # include <string.h>. Elle permet d’avoir accès à certaines fonctions (nombre de caractères, copie, concaténation, …).

Types char (%c)

char : %c 🠖 codé sur 1 octet.

Déclaration : char lettre = ‘a’ ; (attention : guillemets simples)

Affichage et saisie d’un caractère :

|  |  |
| --- | --- |
| printf (« %c », lettre) ; | Affiche la lettre a |
| printf (« %d », lettre) ; | Affiche le code ASCII de la lettre a, soit 97 |
| scanf(« %c », &lettre) ; | Saisie de la lettre |
| lettre = getchar() | Saisie de la lettre |

Acquisition de plusieurs char (%c)

Lorsque l’on souhaite réaliser l’acquisition successive de plusieurs caractères, le fait de valider (touche entrée) l’acquisition est perçue comme un caractère. La séquence d’échappement \n est alors stockée dans la mémoire tampon, en attente de la prochaine acquisition.

Pour acquérir plusieurs caractères, il faut donc entrer les caractères successifs en ne validant qu’après le dernier caractère. Ceci fait, la dernière validation conduit au stockage dans la mémoire tampon de la dernière séquence d’échappement \n.

Une autre solution consiste à vider la mémoire tampon avant une acquisition en utilisant la fonction fflush(stdin). Le stdin correspond à un flux standard entrant, donc en provenance du clavier dans ce cas précis.

L’acquisition d’un caractère peut aussi se faire en utilisant l’instruction getchar() :

|  |  |
| --- | --- |
| char c ;  c = getchar() ; | Stocke le caractère saisi au clavier dans la variable c. |

Le getchar() doit lui aussi être précédé d’un fflush(stdin), pour les mêmes raisons que le scanf.

Attention : C online ne gère pas le fflush(stdin).

Chaîne de caractères : tableau de char (%s)

Une chaîne de caractères est un tableau de char auquel s’ajoute un caractère de terminaison : \0. Les compilateurs actuels gèrent souvent ce caractère de terminaison.

Déclaration :

|  |  |
| --- | --- |
| char chaine [10] ; | chaine peut alors stocker un mot de 10 lettres (la onzième lettre étant le caractère de terminaison \0 est souvent gérée par le compilateur). |
| char chaine [] = " coucou " ; | La chaîne de caractère est créée directement ; le caractère de terminaison est ajouté automatiquement. |

Affichage :

|  |  |
| --- | --- |
| for (i=0 ; i<9 ; i++)  {  printf(" %c ", chaine[ i ] ;  } | Affichage lettre par lettre à l’aide d’une boucle. |
| printf (" %s ", chaine) ; | Affichage de la totalité de la chaîne, sans le caractère de terminaison. |

Acquisition via scanf :

|  |  |
| --- | --- |
| for (i=0 ; i<9 ; i++)  {  fflush(stdin) ;  scanf(" %c ", &chaine[ i ] ;  }  //Caractère de terminaison parfois ajouté automatiquement par le compilateur | Acquisition lettre par lettre à l’aide d’une boucle.  Attention, ne pas oublier le fflush, sinon, il faut éviter de taper sur « entrée » entre chaque lettre : le caractère \n serait comptabilisé comme une lettre !  Ne pas oublier le caractère de terminaison, avec des guillemets simples |
| scanf (" %s ", chaine) ;  // Pas de & devant chaine : un tableau est une adresse | Acquisition de la totalité de la chaîne, sans le caractère de terminaison : il est ajouté automatiquement.  La chaîne saisie au clavier peut être plus courte que celle déclarée.  Les caractères \n ou ̺ (espace) ainsi que les EOF (end of file des fichiers) sont considérés comme des caractères de fin de chaîne. |

Acquisition via gets() :

|  |  |
| --- | --- |
| gets(chaine) ; | Acquisition de la totalité de la chaîne, sans le caractère de terminaison : il est ajouté automatiquement.  La chaîne saisie au clavier peut être plus courte que celle déclarée.  Seul(e) la validation ou le EOF sanctionne la fin de la chaîne de caractères. Les espaces sont pris en compte dans la chaîne. |

Fonctions de manipulation de chaînes de caractères (string.h)

|  |  |
| --- | --- |
| strlen(chaine) | Renvoie un entier correspondant au nombre effectif de caractères dans la chaîne (le caractère de terminaison n’est pas comptabilisé). |
| char chaine [ ] = "test" ;  char copie\_1 [5] ;  char copie\_2 [5] ;  strcpy (copie\_1, chaine) ;  strcpy (copie\_2, chaine, 2) ; | Copie la totalité de chaine dans copie\_1.  Copie les 2 premiers caractères de chaine dans copie\_2. |
| char chaine\_1 = "Bonjour" ;  char chaine\_2 = ", les MP2I ! " ;  strcat (chaine\_1, chaine\_2) ; | chaine\_1 est modifiée pour accueillir la concaténation de chaine\_1 et de chaine\_2.  Le caractère de terminaison de chaine\_1 est supprimé. |