INFO2009-Introduction à l'informatique

Aide à la réussite - Session nº2 - Tableaux

Bertrand Alexis

Université de Liège

2023

Rappel

Un tableau est une collection de variables du même type. Ces éléments sont ordonnés de sorte à ce que chacun est caractérisé par un indice unique au sein du tableau. En C (et dans la majorité des programmes informatiques), le premier indice du tableau de taille ℓ est 0 et le dernier est $\ell - 1$, en sachant que tous les indices sont consécutifs. Un tableau de taille ℓ aura donc les indices $0, 1, 2, \ldots, \ell - 2, \ell - 1$.

Définir un tableau

Afin de définir un tableau unidimensionnel (un vecteur) de taille fixe, nous effectuons l'instruction : `type var[nb_elements];`
Il est également possible d'initialiser un tableau lors de sa définition.

Exemple

```
int var[2] = {1,2};
```

Accéder à un élément du tableau

Afin d'accéder à un élément du tableau, nous devons donner l'indice de ce dernier dans le tableau. On peut faire ceci de cette manière : `var[i]`, où i correspond à l'indice de l'élément qu'on tente d'accéder. Attention que si on tente d'accéder à un indice qui n'est pas dans le tableau, nous obtiendrons une erreur d'accès invalide à la mémoire, aussi appelé une erreur de segmentation.

Accéder à un élément du tableau par un pointeur

Quand on initialise une variable de type tableau, la variable correspond en fait à l'identificateur du tableau et correspond à l'endroit dans la mémoire où se situe le tableau. Grâce à cette représentation, faire l'instruction `var + i` correspond à récupérer l'endroit dans la mémoire où se situe le ième élément du tableau. Dans ce cas, les deux instructions suivantes sont pareilles et renvoient le même résultat, à savoir la valeur du ième élément du tableau : `var[i] == *(var + i)`.

Fonction avec un tableau en argument (1/2)

Afin de passer un tableau en argument, il suffit de mettre `type var[]` comme argument dans la définition de la fonction. Il est alors possible de donner le tableau en argument en appel à la fonction en mettant l'identificateur du tableau. Comme on a vu que `var[i] == *(var + i)`, on peut facilement comprendre que on peut également mettre `type* var` comme argument de la fonction à la place.

Fonction avec un tableau en argument (2/2)

Exemple

```
void f(type *var){
    ...
}

int main(){
    type var[i];
    f(var);
    return EXIT_SUCCESS;
}
```

Types de passage d'arguments

Rappel

Comme vous avez dû le voir dans le cours théorique, les arguments peuvent être passés par valeur ou pointeur.

Étant donné que `var[i] == *(var + i)`, on peut facilement comprendre que donner un tableau en argument correspond à donner son adresse mémoire et que du coup, on effectue toujours un passage par pointeur dans ce cas.

Tableau multidimensionnel

Il n'est pas que possible de créer un vecteur. Nous pouvons créer un tableau avec autant de dimensions qu'on veut. Pour chaque dimension, il suffit de préciser sa taille lors de la définition du tableau : `type var[`ℓ_1`][`ℓ_2`]...`. Accéder à un élément du tableau fonctionne de la même façon : `var[i][j][...]`.

Créer un tableau à taille dynamique

Parfois, nous ne connaissons pas la taille d'un tableau au préalable. Il est donc intéressant d'allouer un tableau dynamiquement. Pour ce faire, il va falloir utiliser une des fonctions `malloc`, `calloc`, ou `realloc`.

Exemple

```
type* var = malloc(nb_element * sizeof(type));
```

Désallouer un tableau à taille dynamique

Attention !

Lorsqu'on alloue dynamiquement une variable en C, il faut obligatoirement libérer l'espace utilisé grâce à la fonction `free(var)` avant la fin du programme, sinon nous risquons des pertes de mémoires. Les pertes de mémoires sont des endroits de la mémoire qui ne seront plus accessibles même après la fin de l'exécution du programme car réservés à des variables du programme.

Note

Certains langages de programmation ont ce qu'on appelle un "garbage collector", il s'agit d'un moyen de récupérer toute la mémoire qui a été allouée au programme et qui n'est pas utilisé afin de libérer l'espace si nécessaire. Ce n'est cependant pas le cas en C. La règle un malloc = un free est alors d'application !

Exercices

1. Écrire un programme permettant de générer un triangle de pascal de la taille donnée en entrée par l'utilisateur. Un triangle de pascal est une matrice où les éléments au-dessus de la diagonale principale sont nuls et chaque élément $a_{i,j}$ en-dessous de la diagonale principale est égal à la somme de l'élément directement au-dessus de lui avec l'élément à la gauche de ce dernier. Pour ce faire, il faut suivre les consignes suivantes :

Exercices

1.1 Récupérer la longueur ℓ donnée en entrée par l'utilisateur dans la fonction main puis créer une matrice de dimension $\ell \times \ell$ et initier tous ses éléments à 0 sauf le premier élément à 1 (pascal[0][0]= 1).

1.2 Créer une fonction permettant de remplir la matrice en suivant les règles de pascal. Cette fonction doit prendre en argument la matrice représentant le triangle de pascal ainsi que sa longueur.

1.3 Créer une fonction permettant d'afficher dans la console la matrice de pascal sous sa vraie forme, prenant en argument la matrice et sa longueur.

Exercices

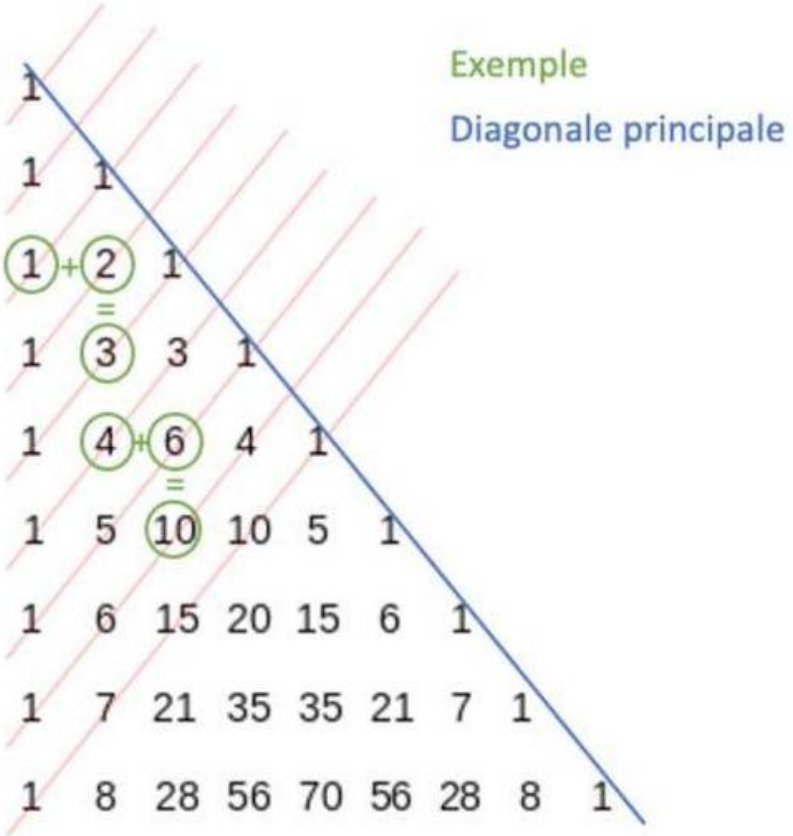

Figure: Triangle de pascal

Exercices

2. Écrire une fonction qui renvoie vrai si la chaine de caractères donnée en entrée est un palindrome et faux sinon.
3. Écrire une fonction qui trie un tableau donné en argument. Dans cet exercice, il n'est pas demandé que l'agorithme soit le plus efficace. Une implémentation naïve est correcte.
4. (Examen d'août 2021) Écrire une procédure prenant en arguments un tableau `t` de nombres réels, le nombre $n \geq 0$ d'éléments de `t`. L'opération effectuée par cette procédure consiste à modifier le contenu du tableau `t` de la façon suivante : si $t_0, t_1, t_2, \ldots$ sont les contenus initiaux des cases `t[0]`,`t[1]`,`t[2]`,... de `t`, alors après l'exécution de la procédure, la première case de `t` doit contenir t_0, la deuxième $t_0 + t_1$, la troisième $t_0 + t_1 + t_2$, et ainsi de suite.

Exercices

5. (Examen d'août 2019) Écrire une fonction prenant en arguments deux tableaux d'entiers t_1 et t_2 ainsi que leur taille, et retournant la longueur de leur plus grand préfixe commun, c'est-à-dire le plus grand entier n tel que $t_1[i] = t_2[i]$ pour tout i tel que $0 \leq i < n$.
6. (Examen de janvier 2022) Écrire une fonction prenant en argument une chaîne de caractères, et retournant la somme de tous les chiffres qu'elle contient. Par exemple, pour la chaîne `"Le 24 janvier 2022"`, cette fonction doit retourner 12. Dans le cas particulier d'une chaîne qui ne contient aucun chiffre, la fonction doit retourner 0.