

# Solution

## Note :

❖ Exercices 'plus' à la fin du document sont optionnels à faire or ce sont de même type tu TP basé sur ses deux exercices 2 et 3.

## Exercise 1

```
#include <stdio.h>
void main()
{
    int a, b, signe_de_b=1, i, result=0;

    printf("donner deux entier pour calculer leur produit: \n");
    scanf("%d%d", &a, &b);

    if (b<0)
    {
        b = -b;
        signe_de_b = -1;
    }

    for ( i = 0; i < b; i++)
        result += a;

    if (signe_de_b == -1)
        result = -result;

    printf(" le produit de ces deux entier est : %d ", result);
}
```

Prendre b toujours comme une valeur positive afin de mettre i<b dans 'for' après.

Ajouter a dans result (initialisé par 0) b fois, ici c.à.d. result = a+a+...a (b fois) = a \* b. Et donnant un signe résultat selon 'a' seulement car b est absolue.

Affectation de résultat selon le signe de b

Exemple d'exécution :

```
donner deux entier pour calculer leur produit:
5
5
le produit de ces deux entier est : 25
```

```
donner deux entier pour calculer leur produit:
5
-5
le produit de ces deux entier est : -25
```

```
donner deux entier pour calculer leur produit:
-5
5
le produit de ces deux entier est : -25
```

```
donner deux entier pour calculer leur produit:
-5
-5
le produit de ces deux entier est : 25
```

## Exercise 2

```
#include <stdio.h>
```

```
void main()
{
    int N, M, i, j;
    printf("Donner la longueur N puis largeur M pour afficher ton rectangle : \n");
    scanf("%d%d", &N, &M);

    for (i = 1; i <= M; i++)
    {
        for (j = 1; j <= N; j++)
        {
            if (i == 1 || i == M)
                printf("*");
            else
            {
                if (j == 1 || j == N)
                    printf("*");
                else
                    printf(" ");
            }

            printf("\n");
        }
    }
}
```

Soit i pour le ligne et j pour la colonne  
**(\*Voir note↓)**

Pour tout j entre 1 et N et ligne est '1' ou 'M' remplir tout case (1, j) et (N, j) par \*

Pour tout i différent que 1 ou N remplir (i, 1) et (1, N) par \*

Pour tout autre case (i, j) remplir par ' '.

Sauter une ligne quand l'affichage de chaque ligne se termine

Exemple d'exécution :

```
Donner la longueur N puis largeur M pour afficher ton rectangle :
```

```
20
5
*****
*               *
*               *
*               *
*****
```

```
Donner la longueur N puis largeur M pour afficher ton rectangle :
```

```
60
6
*****
*                               *
*                               *
*                               *
*                               *
*****
```

### ➡ **\*Note :**

- ❖ L'affichage se fait ligne par ligne et non pas par colonne, on peut imaginer un curseur qui se marche et peut afficher quel que soit caractère et sauter des lignes mais non pas retourner sur son chemin.
- ❖ La boucle externe est liée alors au nombre de ligne (largeur) de 1 jusqu'à M, avant de sauter une ligne : une autre boucle interne liée au nombre de colonne (longueur) de 1 jusqu'à N c'est afficher les caractères (sous conditions et relations entre i et j et M et N) puis à la fin sauter une ligne pour entrer de nouveau dans la boucle externe ....

### Exercice 3

```
#include <stdio.h>
```

```
void main()
{
    int N, i, j;
    printf("donner N pour afficher : \n");
    scanf("%d", &N);

    for (i = 1; i <= N; i++)
    {
        for (j = 1; j <= i; j++)
            printf("%d", i);

        printf("\n");
    }
}
```

- ❖ Soit i pour le ligne et j pour la colonne
- ❖ On peut imaginer un droite  $y=x$  ou  $i=j$ , puis pour tout  $j \leq i$  on affiche la valeur de ligne qu'est i
- ❖ Sauter une ligne quand l'affichage de la colonne se termine

Exemple d'exécution :

```
donner N pour afficher :
9
1
22
333
4444
55555
666666
7777777
88888888
999999999
```

```
donner N pour afficher :
15
1
22
333
4444
55555
666666
7777777
88888888
999999999
101010101010101010
11111111111111111111
12121212121212121212
13131313131313131313
14141414141414141414
15151515151515151515
```

//Exercice plus base sur l'exercice 2 et 3 est de dessiner une maison de dimension  $M \times N$

---

## Exercice 4

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int n, n_initial, nb_chif=0, som_chif=0;
```

```
    printf("donner n pour tester si elle est divisible par 9 ou non : \n");  
    scanf("%d", &n);
```

```
    n_initial = n;
```

```
    while (n!=0) // même que while(n) car  
    // while(0) où 0 est le condition faux  
    {  
        som_chif += n % 10;  
        n /= 10;  
        nb_chif++;  
    }
```

- ❖ Addition à « some\_chif » le reste de n par 10. Ex sur 4537 : 453.7 (ajouter 7)
- ❖ Division entière par 10 et ignorant le reste ex : 4537 /10 = 453
- ❖ Addition au nb de chiffre 1
- ❖ Répétition jusqu'à annulation de n

```
    if (som_chif % 9 == 0)
```

```
        printf("%d est compose de %d chiffres.\nLa somme de ses  
chiffres est : %d \nIl est divisible par 9", n_initial, nb_chif,  
som_chif);
```

```
    else
```

```
        printf("%d est compose de %d chiffres.\nLa somme de  
ses chiffres est : %d \nIl n'est pas divisible par 9", n_initial,  
nb_chif, som_chif);
```

Affichage  
clair.

Exemple d'exécution :

```
donner n pour tester si elle est divisible par 9 ou non :  
57883  
57883 est compose de 5 chiffres.  
La somme de ses chiffres est : 31  
Il n'est pas divisible par 9
```

```
donner n pour tester si elle est divisible par 9 ou non :  
45360  
45360 est compose de 5 chiffres.  
La somme de ses chiffres est : 18  
Il est divisible par 9
```

## Exercice 5

```
#include <stdio.h>
#include <math.h>
```

### Part 1:

```
void main()
{
    int n, i, j;
    double Un_1, Un, factn, puis;

    printf("donner le premier element U_1 puis n pour calculer U_n\n");
    scanf("%lf%d", &Un_1, &n);

    printf("pour U_%d egale a %0.01f, U_%d est egale a ", 1, Un_1, n);

    for (i = 2; i <= n; i++)
    {
        factn = 1; puis = 1;
        for (j = 1; j <= i; j++)
        {
            factn *= j;
            puis *= Un_1;
        }

        Un = puis / factn;

        Un_1 = Un;

        printf("\t %0.201f\n", Un);
    }
}
```

Saisir des données  $U_1$  et  $n$  afin de calculer  $U_n$  (optionnel)

Calcul de  $n!$  L'affecter dans  $factn$

Calcul de  $U_n$

- Afin de calculer  $U_{n+1}$  on a besoin de la valeur de  $U_n$  on le met dans  $U_{n-1}$
- Alors que  $U_{n+1}$  sera désigné par  $U_n$

Affichage de résultat

Exemple d'exécution :

```
donner le premier element U_1 puis n pour calculer U_n de la suite :
1
3
pour U_1 egale a 1, U_3 est egale a 0.0208333333333333218
```

```
donner le premier element U_1 puis n pour calculer U_n de la suite :
2.5
6
pour U_1 egale a 3, U_6 est egale a
10693070606605176667363082240.00000000000000000000
```

/\*note que la convergence de cette suite depend de  $U_1$  : pour tout  $U_1 \geq 0$  et  $U_1 < 2.2902$   
 $U_n \longrightarrow 0$ .  
Tandis que pour tout  $U_1 > 2.2903$  :  $U_n \longrightarrow +\infty$   
\*/

**Part 2:** (même que part 1 mais au lieu d’affichage à la fin du boucle >> affichage dans le boucle)

```
void main()
{
    int n, i, j;
    double Un_1, Un, factn=1;

    printf("donner le premier element U_1 puis n pour calculer
    jusqu'a U_n de la suite : \n ");
    scanf("%lf%d", &Un_1, &n);

    printf("pour U_%d egale a %0.0lf, on a\n", 1, Un_1, n);

    for (i = 2; i <= n; i++)
    {
        factn *= i;
        Un = powf(Un_1, i) / factn;
        Un_1 = Un;
        printf("U_%d= \t %0.20lf\n", i, Un);
    }
}
```

} (\*Voir note↓)

Exemple d’exécution :

```
donner le premier element U_1 puis n pour calculer jusqu'a U_n de la suite :
1
8
pour U_1 egale a 1, on a
U_2= 0.50000000000000000000
U_3= 0.020833333333333333218
U_4= 0.000000000784917197905
U_5= 0.00000000000000000000
U_6= 0.00000000000000000000
U_7= 0.00000000000000000000
U_8= 0.00000000000000000000
```

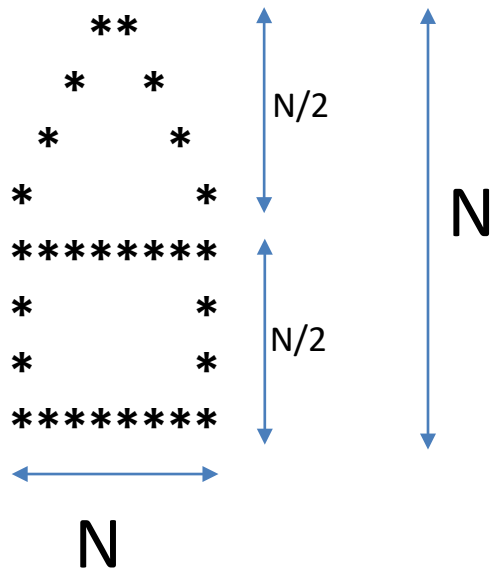
```
donner le premier element U_1 puis n pour calculer jusqu'a U_n de la suite :
3
8
pour U_1 egale a 3, on a
U_2= 4.50000000000000000000
U_3= 15.18750000000000000000
U_4= 2216.83789062500000000000
U_5= 446157730943249.06250000000000000000
U_6= inf
U_7= inf
U_8= inf
```

➡ **\*Note expert :**

- ❖ On peut mettre la même solution déjà écrit dans **Part 1**, mais juste pour changer un peu les idées et optimiser la solution : notons que dans **Part 1** on calcul à chaque fois  $n!$  de nouveau tandis que si on a  $(n-1)!$  il suffit de le multiplier par  $n$  pour obtenir  $n!$  ;
- ❖ A ne pas oublier que la fonction `powf()` nécessite la librairie `<math.h>`.

### Exercise 1 Plus

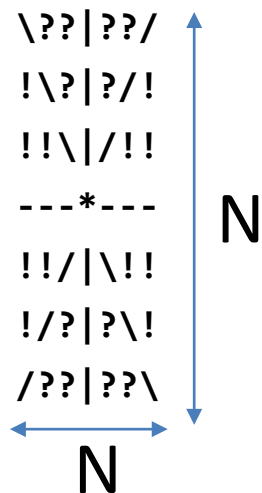
Ecrire un programme qui lit un entiers positifs  $N$  pair désignant la dimension d'une maison et affiche son dessin comme indique la figure si dessous respectant sa symétrie axiale :



---

### Exercise 2 Plus

Ecrire un programme qui lit un entiers positifs  $N$  désignant la dimension  $N$  impair et affiche le dessin comme indique la figure si dessous respectant ses symétries axiaux et central :



## Solution

### *Exercise 1 Plus*

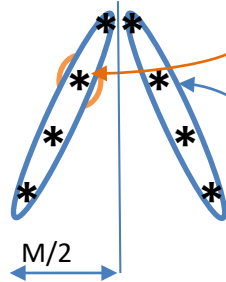
Analyse : soit deux boucles à deux variable  $i$  pour le ligne et  $j$  pour le colonne (initialises tous les deux par 1) et on a  $N = \text{longueur} = \text{largeur}$ . Quel que soit le dessin : à trouver les relations mathématiques existant entre  $i$ ,  $j$ ,  $N$  et  $M$  (largeur si existe) pour quel que soit  $i$  et  $j$  dans le domaine précise du dessin.

Tout d'abord remarquons que cette maison est constituée de 2 parties l'une qui est rectangle (même que l'exercice 2) et sur laquelle un cône (semblable à l'exercice 3) :

Partie simple : (même que ex.2)

```
*****
*      *
*      *
*****
```

Partie a analyse :



Presque c'est un demi dessin alors on la même largeur tandis que demi longueur.  
Remarquons que ici :  $j = M/2 - i + 1$ .  
Test On a  $i = 2, j = 3, M/2 = 4$ .  
 $J = 4 - 2 + 1 = 3$ . Vrai pour 4 points

$j = M/2 + i$  vrai pour les autres.  
(Dessin symétrique)

Ou voici ce tableau pour un mieux vue et calcul :

$i \backslash j$	1	2	3	4	5	6	7	8
1				*	*			
2			*			*		
3		*					*	
4	*							*

D'où le code de la solution sera clair :

(Voir la page suivante)



```

#include <stdio.h>
void main()
{
    int N, M, i, j;
    printf("donner la dimension N pour dessiner ta maison : \n");
    scanf("%d", &N);
    M = N;
    for (i = 1; i <= N / 2; i++)
    {
        for (j = 1; j <= M; j++)
            if (j == M / 2 - i + 1 || j == M / 2 + i)
                printf("*");

            else
                printf(" ");

        printf("\n");
    }

    for (i = 1; i <= N / 2; i++)
    {
        for (j = 1; j <= M; j++)
            if (i == 1 || i == N / 2)
                printf("*");

            else
                if (j == 1 || j == M)
                    printf("*");
                else
                    printf(" ");

        printf("\n");
    }
}

```

Affichage du cône :  
Pour tout j sous ces conditions trouvées afficher « \* » tout autre afficher « ».

(Même que ex.2)  
Faire attention aux limites des boucles.

Exemple d'exécution :

donner la dimension N pour dessiner ta maison :

8

```

    **
   * *
  *   *
 *     *
*****
 *     *
 *     *
*****

```

## Exercise 2 Plus

Analyse : même analogie que l'exercice précédent, prenant les mêmes variables et définition. C'est à trouver une relation entre  $i$   $j$  et  $N$  on a une trace ressemblant à un repère plus précisément une matrice ou on a les propriétés suivantes :

- 4 droites principales celles qui sont entourés comme indique le tableau ci-dessous : si on sait ses conditions alors que les autres conditions des points d'affichage seront faciles puisqu'elles sont limitées entre eux.
- Un point d'intersection qui est l'origine : c'est plus facile de le prendre le premier « if » pour ne pas confondre et mettre les autres dans « if/else » structure imbriqués.

○ Point origine  $y = x = 0$   
Ou  $i = j = E(N/2) + 1$

○  $y = -x$   
Ou  
 $j = i$

Et

○  $y = x$   
Ou  
 $j = N - i + 1$

○  $x = 0$   
Ou  
 $j = E(N/2) + 1$

Et

○  $y = 0$   
Ou  
 $i = E(N/2) + 1$

$i \backslash j$	1	2	3	4	5	6	7
1	\	?	?	/	?	?	/
2	!	\	?	/	?	/	!
3	!	!	\	/	/	!	!
4	-	-	-	*	-	-	-
5	!	!	/	/	\	!	!
6	!	/	?	/	?	\	!
7	/	?	?	/	?	?	\

### Note :

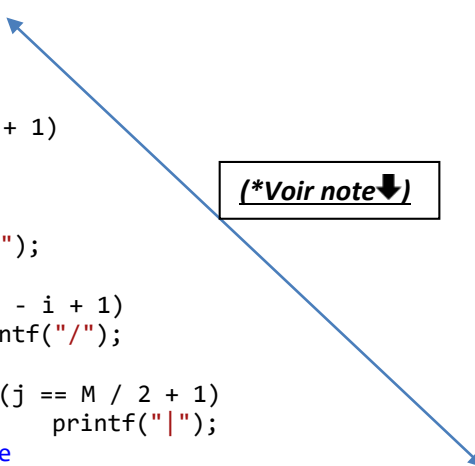
- ❖ «  $i$  » désigne le nombre du ligne et «  $j$  » celle de la colonne
- ❖ «  $E$  » est la partie entière du nombre réel c.a.d. Ici on a  $N$  impair et positive alors  $E(N/2) = N/2 - 0.5$ .  
 $E(7/2) = E(3.5) = 3$ .
- ❖ Mais en informatique en déclarant  $N$  comme un entier int sa division n'est autre que  $E(x/2)$  avec  $x$  positive la variable déclarée.

D'où le code de la solution sera clair :

```
#include <stdio.h>

void main()
{
    int N, M, i, j;
    printf("donner la dimension N pour dessiner l'art : \n");
    scanf("%d", &N);
    M = N;
    for (i = 1; i <= N; i++)
    {
        for (j = 1; j <= M; j++)
            if (i == j && j == M / 2 + 1)
                printf("*");
            else
                if (j == i)
                    printf("\\");
                else
                    if (j == M - i + 1)
                        printf("/");
                    else
                        if (j == M / 2 + 1)
                            printf("|");
                        else
                            if (i == M / 2 + 1)
                                printf("-");
                            else
                                if ((j > i && j < M / 2 + 1) ||
                                    (j > M - i + 1 && j < M / 2 + 1) ||
                                    (j > M - i + 1 && j < M / 2 + 1))
                                    printf("?");
                                else
                                    printf("!");

        printf("\n");
    }
}
```



(\*Voir note↓)

On peut mettre un caractère restant ayant plusieurs ensembles dans else à la fin.

Exemple d'exécution :

```
donner la dimension N pour dessiner l'art :
7
\??|??/
!\?|?/!
!!\|/!!
----*----
!!/\|!!
!/?|?\!
/??|??\
```

#### ➡ \*Note :

- ❖ Structure de « if/else » imbriquées en commençant par les ensembles le plus communs inclus dans des autres ensemble plus grand c.à.d. ces ensembles ont la priorité d'être écrits avant celles qui le contiennent. (On remarque ici 13 ensembles formées par 4 caractères)
- ❖ Lorsque les ensembles des conditions seront indépendants (c.à.d. pas d'intersection dans les ensembles restes) on peut changer l'ordre de « if/else ».
- ❖ Pour **limiter** un domaine utiliser « && » (ex. de l'ensemble « \* » ou l'un de « ? ») et pour l'**union** des domaines au cas de même caractère utiliser « || » (ex. de 4 ensemble de caractère « ? »).