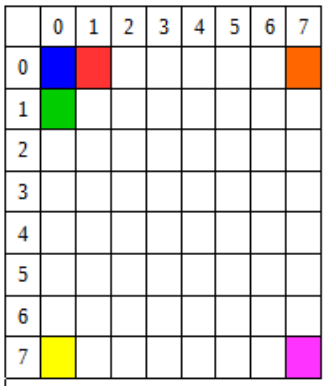
Les boucles définies :

Exercice n°1 : Prise en main des Leds de la carte Sense-hat.

Complétez le programme suivant pour obtenir la figure ci-dessous :

```
1
2   #include <stdio.h>
3   #include <stdlib.h>
4   #include <senseHat.h>
5
6   int main()
7   {
8       InitialiserLeds();
9
10      Allumer(0,0,BLEU);
11      Allumer(0,1,ROUGE);
12      Allumer(1,0,VERT);
13
14
15      return 0;
16  }
17
```

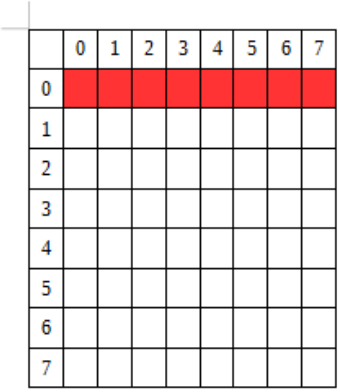


Exercice n°2 : Les boucles définies

On souhaite allumer la première ligne en rouge.

Plusieurs solutions :

- Écrire 8 fois *Allumer(0,...,ROUGE)* en complétant le paramètre manquant par 0, 1, 2,…., 7. Cette solution n'est pas toujours très efficace surtout lorsqu'il y a beaucoup de répétition à effectuer.
- Utiliser une boucle, ici on connaît le nombre d'itérations à réaliser : On peut donc utiliser une ***boucle définie***.



Algorithme	
début InitialiserLeds() pour colonne allant de 0 à 7 Allumer(0,colonne,ROUGE) finPour fin	Lexique des variables : colonne entier

Après avoir testé ce programme, modifiez le pour allumer la première colonne en rouge.
Comment modifier le programme pour allumer une led sur deux ?
Toujours en utilisant une boucle définie, comment allumer uniquement les leds de la ligne 2 à la ligne 5 ?

Exercice n°3 : Les boucles imbriquées

On souhaite maintenant remplir la matrice de leds en bleu.

Pour cela on peut retenir l'idée : ***Pour chaque ligne allumer la led de chaque colonne en bleu.***

Algorithme	
début InitialiserLeds() pour ligne allant de 0 à 7 pour colonne allant de 0 à 7 Allumer(ligne,colonne,BLEU) finPour finPour fin	Lexique des variables : colonne entier ligne entier

Codez cette algorithme en langage C dans un nouveau projet nommé ***exercice3***.

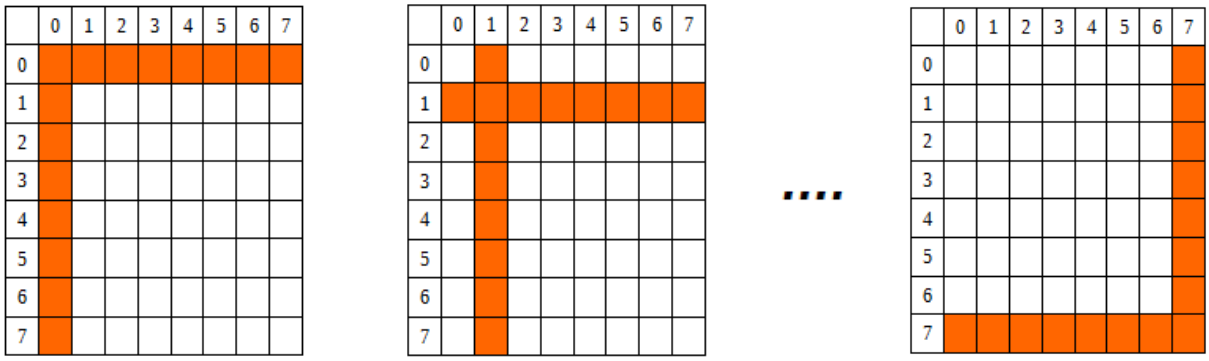
On souhaite maintenant que l'affichage se fasse plus lentement pour bien voir le défilement de chaque ligne. Il est nécessaire de temporisé entre deux lignes.

> La fonction **sleep**() de la librairie **<unistd.h>** endort le processus appelant pendant le nombre de secondes passé en paramètre à la fonction.
>
> La fonction **usleep**() de la librairie **<unistd.h>** suspend l'exécution du programme appelant durant le nombre de microsecondes passé en paramètre à la fonction.
>
> *La période de sommeil peut être allongée par la charge système, par le temps passé à traiter l'appel de fonction, ou par la granularité des temporisations système.*

Complétez votre programme pour qu'il y est une pose d'une seconde entre chaque ligne.

Modifiez votre programme pour ne faire afficher qu'une seule ligne à la fois, d'abord la ligne 0 puis la 1 puis la 2…, vous pouvez utiliser la fonction ***Effacer()*** de la librairie senseHat ou remettre la ligne à la valeur **NOIR** pour produire cet effet.

Modifiez votre programme pour réaliser la figure suivante :



L'affichage commence par la première figure, puis passe à la suivante ligne 1 et colonne 1 puis ligne 2 colonne 2… jusqu'à l'obtention de la dernière figure. Il est nécessaire de ralentir l'affichage pour voir l'effet visuel.

Comment faire pour la figure inverse ?

Complétez votre programme pour enchaîner les deux figues.

Les boucles indéfinies :

Exercice n°4

Parfois, on ne sait pas à l'avance combien de fois il est nécessaire de recommencer l'itération, on utilise alors une ***boucle indéfinie***.

Exemple : *Faire l'affichage d'une figure tant que l'utilisateur n'a pas appuyer sur le bouton central du joystick.*

Algorithme

```
début
        InitialiserLeds()
        InitialiserJoystick()
        faire
                pour ligne allant de 0 à 7
                        pour colonne allant de 0 à ligne
                                Allumer(ligne,colonne,BLEU)
                                Allumer(colonne,ligne,BLEU)
                        finPour
                        dormir(200ms)
                        Effacer()
                finPour
                touche ← ScannerJoystick()
        tantQue touche ≠ KEY_ENTER
fin
```

Lexique des variables : colonne entier
ligne entier
touche entier

Source en langage C

```c
#include <senseHat.h>
#include <unistd.h>

int main()
{
    int ligne;
    int colonne;
    int touche;
    InitialiserLeds();
    InitialiserJoystick();
    do
    {
        for(ligne = 0 ; ligne <=7 ; ligne++)
        {
            for (colonne = 0 ; colonne <= ligne ; colonne++)
            {
                Allumer(ligne,colonne,BLEU);
                Allumer(colonne,ligne,BLEU);
            }
            usleep(200000);
            Effacer();
        }
        touche = ScannerJoystick();
    } while(touche != KEY_ENTER);
    return 0;
}
```

Dans le programme précédent, la figure disparaît lorsqu'elle est complètement réalisée. On souhaite maintenant pouvoir la figer à tout moment.
La boucle « ***pour ligne allant de 0 à 7 … finPour***» ne peut plus être une boucle définie puisqu'il faut interrompre le traitement sur l'appui du bouton centrale du joystick, l'algorithme devient donc :

Algorithme

```
début
        InitialiserLeds()
        InitialiserJoystick()
        touche ← 0
        faire
                ligne ← 0
☆               tantQue touche ≠ KEY_ENTER et ligne ≤ 7
                        Effacer()
                        pour colonne allant de 0 à ligne
                                Allumer(ligne,colonne,BLEU)
                                Allumer(colonne,ligne,BLEU)
                        finPour
                        dormir(200ms)
                        ligne ←  ligne + 1
                        touche ← ScannerJoystick()
                finTantQue
        tantQue touche ≠ KEY_ENTER
fin
```

Lexique des variables : colonne entier
ligne entier
touche entier

☆ Dans cette algorithme on constate que le **test** pour rester dans la boucle est placé avant la boucle. Il est composé de deux conditions qui doivent être vraie toutes les deux pour poursuivre. En langage C cela va se traduire par :

```
while (touche != KEY_ENTER && ligne <= 7)
{
...
}
```

Attention il n'y a pas de point virgule lorsque le test est placé avant la boucle

Maintenant, si on souhaite faire une boucle infinie, il faut écrire « **while(1) »** en effet 0 représente faux en langage C et différent de 0, ici 1, représente vrai.

Modifiez le programme précédent pour que si on appui sur le bouton central du joystick on fige la figure et si on appuie une seconde fois la figure se redessine.

Pour tester le second appui sur le bouton central, on peut ajouter le code suivant après la sortie de boucle après le premier appui.

```
do
{
    touche = ScannerJoystick();
}while(ligne != 8 && touche != KEY_ENTER );
touche = 0;
```

Pensez également à utiliser une boucle infinie pour que le programme ne s'arrête jamais.

Que se passe t'il : - si on supprime « **ligne != 8** » dans de maintient du **while** ?

- si on condition de maintient du **while** devient :

```
(touche != KEY_ENTER && ligne != 8);
```