```
// SOLUTION DU PROBLEME :

// DEBUT :
//          i             j              k
//        i = 0      j = i + 1       k = j + 1
//          0           0 + 1           1 + 1
//          0             1               2
// 012

// 013
// 014
// 015
// 016
// 017
// 018
// 019

//          i             j              k
//          0             2           k = j + 1
//          0             2             2 + 1
//          0             2               3
// 023

// ...
// 029

//          i             j              k
//          0             3           k = j + 1
//          0             3             3 + 1
//          0             3               4
// 034
// ...
// 039
// ...
// 045
// ...
```

```
// FIN :    i             j               k
//        i = 7         j = 8           k = 9
// 789

// IMPLEMENTATION :

#include <unistd.h>

// prototype de la fonction ft_write_number
// (voir plus bas)
// à indiquer pour que les autres fonctions
// du fichier ft_print_comb.c
// (ici, la fonction ft_print_comb seulement)
// aient connaissance de cette fonction
// et puisse l'appeler
// et l'exécuter correctement
// car la fonction ft_write_number
// est définie dans notre fichier APRES
// que l'on ait défini la fonction
// qui l'appelle (ft_print_comb)

// une autre solution serait de définir
// ft_write_number avant ft_print_comb
// mais cela nuierait à la lisibilité du code

// cette fonction ne retourne rien et prend
// un entier (type int) en paramètre, nommé n
void    ft_write_number(int n);

void    ft_print_comb(void)
{
        // on définit 3 variables de type int : i j et k
        // i stockera le premier chiffre de la combinaison
        int     i;

        // j stockera le deuxième chiffre de la combinaison
        int     j;
```

```
// k stockera le troisième chiffre de la combinaison
int     k;

// on initialise i à 0
i = 0;

// pour faire boucler i (1er chiffre) de 0 à 7 :
// tant que i sera inférieur ou égal à 7
// (car le maximum du 1er chiffre est 7)
// on incrémentera i de 1 (voir // *)
// (après avoir traité le 2eme chiffre)
while (i <= 7)
{
        // j (2eme chiffre) sera égal à i + 1
        // donc 1 au départ (voir SOLUTION plus haut)
        j = i + 1;

        // tant que j sera inférieur ou égal à 8
        // (car le maximum du 2eme chiffre est 8)
        // on incrémentera j de 1 (voir // **)
        // (après avoir traité le 3eme chiffre)
        while (j <= 8)
        {
                // k (3eme chiffre) sera égal à j + 1
                // donc 2 au départ (voir SOLUTION plus haut)
                k = j + 1;

                // tant que k sera inférieur ou égal à 9
                // (car le maximum du 3eme chiffre est 9)
                // on incrémentera k de 1 (voir // ***)
                // (après avoir écrit les 3 chiffres)
                while (k <= 9)
                {
                        // on écrit les 3 chiffres les uns à la suite des autres
                        // en faisant appel à notre fonction ft_write_number
                        // en lui passant successivement les paramètres i, j et k
                        // (nos 3 chiffres)
                        // (voir plus bas)
```

```
                                ft_write_number(i);
                                ft_write_number(j);
                                ft_write_number(k);

                                // si la combinaison n'est pas 789
                                if (!(i == 7 && j == 8 && k == 9))
                                        // on écrit ", " après avoir écrit la combinaison
                                        write(1, ", ", 2);
                                // ***
                                // on incrémente k de 1
                                k++;
                        }
                        // **
                        // on incrémente j de 1
                        j++;
                }
                // *
                // on incrémente i de 1
                i++;
        }
}

// cette fonction ne retourne rien
// mais prend un entier en paramètre
// qu'il stockera dans la variable n
// (qui sera ainsi automatiquement déclarée et initialisée
// à la valeur passée en paramètre de la fonction
// lors de son appel)

// pour pouvoir écrire le chiffre passé en paramètre
// il faut tout d'abord le convertir en caractère
// sinon, la fonction write écrira le caractère
// correspondant à ce code décimal (Dec) dans la table ASCII !
// (taper man ascii dans le terminal pour afficher la table ascii)
// LES CARACTERES DE CODE Dec DE 0 A 9 NE CORRESPONDENT PAS
// AU CHIFFRES DE 0 A 9 !
// on ajoute donc le caractère '0'
// (ou plutôt le code décimal correspondant au caractère '0', 48)
```

```
// au chiffre passé en paramètre

// cette méthode fonctionne car :
// '0' = 48
// '1' = 49
// '2' = 50
// '3' = 51
// '4' = 52
// '5' = 53
// '6' = 54
// '7' = 55
// '8' = 56
// '9' = 57

// par exemple, avec le chiffre 5, on aura :
// 5 + 48 = 53
// ce qui correspond bien au code décimal
// du caractère '5'

// REMARQUE :
// l'int n sera converti en char
// lors de son affectation à c
// car c est de type char
// c sera donc bien un caractère
// (correspondant au code décimal de l'int calculé)
// avant de passer en paramètre de write
void    ft_write_number(int n)
{
        char    c;

        c = n + '0';
        write(1, &c, 1);
}
```