```
// SOLUTION DU PROBLEME :

// DEBUT :
//          i             j
//        i = 0         j = i + 1
//          0           0 + 1
//          0             1
// 00 01

// 00 02
// 00 03
// 00 04
// 00 05
// 00 06
// 00 07
// 00 08
// 00 09

//          i             j
//          1           j = i + 1
//          1           1 + 1
//          1             2
// 01 02

// ...

// FIN :    i             j
//          98          j = i + 1
//          98          98 + 1
//          98            99
// 98 99


// IMPLEMENTATION :

#include <unistd.h>

// prototype des fonctions "helper"
```

```
// la fonction ft_write_numbers servira à extraire
// la dizaine et l'unité des nombres à 2 chiffres
// (des nombres i et j)
// calculés par ft_write_numbers
// pour pouvoir faire écrire ces nombres par
// ft_write_number (même fonction que pour l'ex05)
// elle permettra en même temps d'écrire '0' devant
// les nombres à 1 chiffre

void    ft_write_numbers(int n);
void    ft_write_number(int n);

void    ft_print_comb2(void)
{
        // i stockera le 1er nombre à 2 chiffres de la combinaison
        int     i;

        // i stockera le 2ème nombre à 2 chiffres de la combinaison
        int     j;

        // on initialise i à 0
        i = 0;

        // pour faire boucler i (1er nombre) de 0 à 98 :
        // tant que i sera inférieur ou égal à 98
        // (car le maximum du 1er nombre est 98)
        // on incrémentera i de 1 (voir // *)
        // (après avoir traité le 2eme nombre)
        while (i <= 98)
        {
                // j (2eme nombre) sera égal à i + 1
                // donc 1 au départ (voir SOLUTION plus haut)
                j = i + 1;

                // tant que j sera inférieur ou égal à 99
                // (car le maximum du 2eme nombre est 99)
                // on incrémentera j de 1 (voir // **)
                while (j <= 99)
```

```
                {
                        // on extrait les chiffres du nombre i
                        // et on l'écrit
                        ft_write_numbers(i);

                        // on écrit " " après avoir écrit le 1er nombre
                        write(1, " ", 1);

                        // on extrait les chiffres du nombre j
                        // et on l'écrit
                        ft_write_numbers(j);

                        // si la combinaison n'est pas 98 99
                        if (!(i == 98 && j == 99))
                                // on écrit ", " après avoir écrit la combinaison
                                write(1, ", ", 2);
                        // **
                        // on incrémente j de 1
                        j++;
                }
                // *
                // on incrémente i de 1
                i++;
        }
}

// pour extraire la dizaine et l'unité du nombre passé
// en paramètre

// en effectuant la division entière de n par 10
// (avec / 10)
// (division entière car n et 10 sont de type int) :
// le résultat sera le chiffre des dizaines
// exemple : 45 / 10 = 4

// puis le reste de la division entière de n par 10
// (avec modulo (%) de 10) :
// le résultat sera le chiffre des unités
```

```
// exemple : 45 % 10 = 5

void    ft_write_numbers(int n)
{
        // extraction de la dizaine
        // division par 10 et envoi du résultat
        // (le quotient de la division par 10)
        // a ft_write_number
        // pour l'écrire
        ft_write_number(n / 10);

        // extraction de l'unité
        // division par 10 et envoi du reste
        // (le reste de la division)
        // a ft_write_number
        // pour l'écrire
        ft_write_number(n % 10);
}

// voir ex05
void    ft_write_number(int n)
{
        char    c;

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