```
// TRI PAR SELECTION :

void    ft_swap(int *a, int *b);

// la fonction ft_sort_int_tab prend en paramètres
// un pointeur vers un tableau d'entiers (tab)
// et la taille de ce tableau
// (son nombre d'éléments) (size)
void    ft_sort_int_tab(int *tab, int size)
{
        // on déclare i
        // qui servira d'indice pour parcourir le tableau
        // (jusqu'à son avant-dernier élément)
        int     i;

        // on déclare j
        // qui servira d'indice pour chercher l'élément
        // le plus petit dans la partie non encore triée du tableau
        // (en partant de i + 1)
        int     j;

        // on déclare min_i
        // qui sera utilisé pour stocker l'indice
        // de l'élément le plus petit
        int     min_i;

        // on initialise i à 0 pour commencer le tri
        // du tableau à partir de son premier élément
        // (à partir de l'indice 0)
        i = 0;

        // tant que i est inférieur à size - 1
        // (la boucle s'arrêtera à l'avant-dernier élément
        // car le tableau sera alors complètement trié :
        // le dernier élément sera forcément le plus grand !)
        while (i < size - 1)
        {
                // à chaque tour de boucle, min_i est initialisé à i
```

```
// pour que l'on considère le premier élément du tri
// en cours comme le plus petit élément
// trouvé pour l'instant

// REMARQUE : i sera incrémenté
// à la fin du tour de boucle
// et lors du tour de boucle suivant, min_i
// prendra donc cette valeur incrémentée
min_i = i;

// on initialise j à i + 1
// pour que la comparaison des éléments
// entre eux commence à partir de l'élément
// suivant directement l'élément en position i
j = i + 1;

// on exécute une deuxième boucle
// dans laquelle j sera incrémenté
// pour parcourir tout le tableau jusqu'à la fin
// pour trouver l'élément le plus petit
// parmi les éléments restants)
while (j < size)
{
        // si l'élément à l'indice j est inférieur
        // à l'élément à l'indice min_i
        if (tab[j] < tab[min_i])

                // alors min_i sera mis à jour
                // avec l'indice j
                // (un nouveau minimum sera donc trouvé
                // parmi les éléments restants)
                min_i = j;
        // on incrémente j
        // pour passer à l'élément suivant du tableau
        j++;
}
// une fois le minimum trouvé *
// on appelle la fonction ft_swap
```

```
                // avec l'adresse de l'élément le plus petit trouvé *
                // &tab[min_i]
                // et l'adresse du premier élément *
                // &tab[i]
                // * : parmi les éléments restants du tableau
                ft_swap(&tab[min_i], &tab[i]);

                // on incrémente i
                // pour passer au prochain élément du tableau à trier
                i++;
        }
}

// pour échanger les valeurs de 2 entiers

// cette fonction prend en paramètres 2 pointeurs
// vers des variables de type entier
// et échange leurs valeurs
// (voir ex02)

void    ft_swap(int *a, int *b)
{
        // pour stocker temporairement la valeur
        // pointée par a
        int     temp;

        temp = *a;

        // on remplace la valeur de la variable
        // pointée par a par celle pointée par b
        *a = *b;

        // on remplace la valeur de la variable
        // pointée par b par celle stockée dans temp
        *b = temp;
}
```

```c
// main.c :

#include "ft_sort_int_tab.h"
#include "ft_putnbr.h"
#include <unistd.h>

// pour remplir le tableau de valeurs non triées
void    ft_fill_tab(int *tab);

// pour afficher les éléments du tableau à l'écran
void    ft_print_tab(int *tab, int size);

int     main(void)
{
        // on déclare un tableau de 10 entiers
        int     tab[10];

        // on déclare size, qui représente la taille du tableau
        int     size;

        // on initialise size à 10
        size = 10;

        // on appelle la fonction ft_fill_tab
        // qui remplira le tableau tab
        // avec des valeurs prédéfinies
        ft_fill_tab(tab);

        // on affiche le tableau avant le tri
        ft_print_tab(tab, size);

        // on saute une ligne
        write(1, "\n", 1);

        // on trie les éléments du tableau
        // par ordre croissant
        ft_sort_int_tab(tab, size);
```

```
	// on affiche le tableau après le tri
	ft_print_tab(tab, size);

	// on saute une ligne
	write(1, "\n", 1);
	return (0);
}

// on remplit le tableau avec des valeurs spécifiques
// qui serviront d'exemple pour le tri
void	ft_fill_tab(int *tab)
{
	tab[0] = 3;
	tab[1] = 0;
	tab[2] = 6;
	tab[3] = 4;
	tab[4] = 1;
	tab[5] = 2;
	tab[6] = 0;
	tab[7] = 5;
	tab[8] = 42;
	tab[9] = 9;
}

// on affiche le tableau à l'écran en le parcourant et en exécutant ft_putnbr()
// sur chacun de ses éléments
void	ft_print_tab(int *tab, int size)
{
	int	i;

	i = 0;
	while (i < size)
	{
		ft_putnbr(tab[i]);
		write(1, " ", 1);
		i++;
	}
}
```