

```
// cette fonction itérative (non récursive) renvoie
// un int (fact) correspondant au résultat de l'opération
// factorielle du nombre passé en paramètre (nb)
// si l'argument n'est pas valide
// (nombre inférieur à 0), la fonction doit renvoyer 0
// REMARQUE : cette fonction ne gère pas les int overflow
```

```
// EXEMPLE :
// la factorielle de 5 (notée 5!) est égale à :
// 5 * 4 * 3 * 2 * 1
// ce qui équivaut à
// 5 * 4 * 3 * 2
```

```
int    ft_iterative_factorial(int nb)
{
    // variable pour stocker le résultat de l'opération factorielle
    int    fact;

    // si nb n'est pas valide (si nb < 0)
    if (nb < 0)
        // on retourne 0
        return (0);

    // GESTION DES CAS PARTICULIERS :
    // 0! = 1
    // 1! = 1

    // sinon, si nb est inférieur ou égal à 1
    // (donc si nb est égal à 0 ou à 1)
    else if (nb <= 1)
        // on retourne 1
        return (1);

    // on initialise fact à 1
    fact = 1;

    // tant que nb est supérieur à 1
    // (il sera décrémenté jusqu'à 2)
```

```

// EXEMPLE : pour nb = 5
// nb vaudra 5, puis 4, puis 3, puis 2

// EXPLICATIONS POUR nb = 5 :
// fact = 1
// nb = 5
// 1) fact = fact * nb = 1 * 5 = 5
// 1') nb-- = nb - 1 = 5 - 1 = 4
// fact = 5
// nb = 4
// 2) fact = fact * nb = 5 * 4 = 20
// 2') nb-- = nb - 1 = 4 - 1 = 3
// fact = 20
// nb = 3
// 3) fact = fact * nb = 20 * 3 = 60
// 3') nb-- = nb - 1 = 3 - 1 = 2
// fact = 60
// nb = 2
// 4) fact = fact * nb = 60 * 2 = 120
// 4') nb-- = nb - 1 = 2 - 1 = 1
// FIN DE LA BOUCLE
// fact = 120
// 5! est bien égal à 120
while (nb > 1)
{
    // on met à jour le résultat de fact
    // au fur et à mesure
    fact = fact * nb;

    // puis on décrémente nb de 1
    nb--;
}
// on retourne le résultat
return (fact);
}

// RESULTAT : 120
#include "ft_putnbr.h"

```

```
#include "ft_iterative_factorial.h"
```

```
int    main(void)
{
    int    nb;
    int    fact;

    nb = 5;
    fact = ft_iterative_factorial(nb);
    ft_putnbr(fact);
    return (0);
}
```