

# Approche « montante » / « par petits bouts »

① Écrire une fonction qui retourne la **somme** des chiffres d'un nombre passé en paramètre.

Par exemple, retourner 7 quand 223 est passé en paramètre ( $2 + 2 + 3 = 7$ ).

② **TESTER** sa fonction !!

③ Écrire ensuite une fonction qui retourne le **produit** des chiffres d'un nombre passé en paramètre.

Par exemple, cette fonction devra retourner 24 quand 423 est passé en paramètre ( $4 \times 2 \times 3 = 24$ ).

④ Utiliser ensuite ces deux fonctions pour écrire une fonction qui **teste** si la somme des chiffres d'un nombre entier passé en paramètre est égale au produit de ces mêmes chiffres.

Par exemple, `somme_produit_egaux(12)` devra renvoyer `false`,  
`somme_produit_egaux(123)` devra renvoyer `true`.

⑤ Utiliser enfin cette fonction pour afficher les 20 nombres recherchés.

# Approche « descendante »

**Supposer** que toutes les fonctions dont on a besoin existent.  
Mais bien les spécifier (prototype) au fur et à mesure de nos besoins.

① Écrire la boucle de recherche des nombres en question

☞ suppose l'existence de `bool somme_produit_egaux(int);`

② Écrire `somme_produit_egaux`

☞ suppose l'existence de `int somme_chiffres(int);`  
et `int produit_chiffres(int);`

③ Écrire `somme_chiffres`

④ Écrire `produit_chiffres`

⑤ Se rendre compte d'une partie commune : **JAMAIS de copié-collé**  
⇒ faire une fonction supplémentaire pour la partie commune

⑥ Penser à écrire des **tests** !