Exercices sur les Boucles en C

Challenge 1 : Générateur de Table de Multiplication

Développez un programme qui affiche la table de multiplication complète d'un nombre choisi par l'utilisateur.

Fonctionnalité : Saisir un nombre et afficher ses multiples de 1 à 10.

Exemple d'affichage pour le nombre 4 :

4 × 1 = 4
4 × 2 = 8
4 × 3 = 12
...
4 × 10 = 40

Objectif : Maîtriser la boucle `for` simple et le formatage d'affichage.

Challenge 2 : Calculateur de Factorielle

Créez un programme qui calcule et affiche la factorielle d'un nombre entier positif.

Définition : La factorielle de n (notée n!) est le produit de tous les entiers de 1 à n.

Formule : `n! = 1 × 2 × 3 × ... × n`

Exemple : 5! = 1 × 2 × 3 × 4 × 5 = 120

Note : Gérer le cas particulier où 0! = 1

Objectif : Utiliser une boucle pour des calculs cumulatifs (produit).

Challenge 3 : Sommateur de Nombres Naturels

Développez un programme qui calcule la somme des n premiers nombres naturels.

Formule mathématique : Somme = 1 + 2 + 3 + ... + n

Exemple : Pour n = 4 → Somme = 1 + 2 + 3 + 4 = 10

Vérification : Vous pouvez vérifier avec la formule directe `n(n+1)/2`

Objectif : Utiliser une boucle pour des calculs cumulatifs (addition).

Challenge 4 : Générateur de Nombres Impairs

Créez un programme qui affiche les n premiers nombres impairs.

Principe : Les nombres impairs forment la séquence 1, 3, 5, 7, 9, 11...

Méthode :

- Approche 1 : Générer `2i - 1` pour i de 1 à n
- Approche 2 : Compter et tester la parité de chaque nombre

Exemple : Pour n = 5 → Affichage : 1, 3, 5, 7, 9

Objectif : Comprendre les patterns mathématiques dans les boucles.

Challenge 5 : Calculateur de Puissance (Sans Fonction Pow)

Développez un programme qui calcule une puissance en utilisant uniquement des boucles.

Paramètres d'entrée :

- Base (nombre à élever)
- Exposant (puissance à appliquer)

Méthode : Multiplication répétée → `base × base × ... × base` (exposant fois)

Exemple : $3^4 = 3 \times 3 \times 3 \times 3 = 81$

Cas particuliers à gérer :

- Exposant = 0 → Résultat = 1
- Exposant négatif (optionnel)

Objectif : Implémenter des algorithmes mathématiques avec des boucles.

Challenge 6 : Générateur de Nombres Pairs

Créez un programme qui affiche les n premiers nombres pairs positifs.

Séquence : 2, 4, 6, 8, 10, 12...

Méthode :

- Approche 1 : Générer `2i` pour i de 1 à n
- Approche 2 : Incrémenter par 2 à partir de 2

Exemple : Pour n = 4 → Affichage : 2, 4, 6, 8

Objectif : Maîtriser les patterns de génération de séquences.

Challenge 7 : Inverseur de Nombre (Sans Tableau)

Développez un programme qui inverse les chiffres d'un nombre entier sans utiliser de tableaux.

Principe : Utiliser les opérations modulo (%) et division entière (/) pour extraire les chiffres.

Algorithme :

1. Extraire le dernier chiffre avec `nombre % 10`
2. Construire le nombre inversé
3. Supprimer le dernier chiffre avec `nombre / 10`
4. Répéter jusqu'à ce que le nombre soit nul

Exemple : 12345 → 54321

Objectif : Manipuler les chiffres individuels d'un nombre avec des opérations arithmétiques.

Challenge 8 : Générateur de Suite de Fibonacci

Créez un programme qui génère et affiche les n premiers termes de la suite de Fibonacci.

Définition de la suite :

- F(0) = 0
- F(1) = 1
- F(n) = F(n-1) + F(n-2) pour n ≥ 2

Séquence : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

Méthode : Utiliser deux variables pour stocker les deux termes précédents et calculer le suivant.

Objectif : Implémenter des suites récurrentes avec des boucles.

Challenge 9 : Compteur de Chiffres

Développez un programme qui compte le nombre de chiffres dans un entier positif.

Méthode : Division successive par 10 jusqu'à obtenir 0.

Algorithme :

1. Initialiser un compteur à 0
2. Tant que le nombre > 0 :
    - Incrémenter le compteur
    - Diviser le nombre par 10 (division entière)

Exemple : 12345 → 5 chiffres

Cas particulier : Le nombre 0 a 1 chiffre.

Objectif : Analyser la structure numérique avec des boucles.

Challenge 10 : Sommateur Interactif d'Entiers

Créez un programme qui calcule la somme des n premiers entiers naturels en utilisant une approche itérative.

Note : Ce challenge est similaire au Challenge 3, mais peut être l'occasion d'explorer différentes approches :

- Boucle `for` classique
- Boucle `while`
- Comparaison avec la formule directe `n(n+1)/2`

Exemple : Pour n = 3 → 1 + 2 + 3 = 6

Extension possible : Afficher chaque étape du calcul pour une meilleure visualisation.

Objectif : Renforcer la maîtrise des boucles et des calculs cumulatifs.

Conseils Généraux pour les Boucles

Types de boucles en C :

- `for` : Idéale quand le nombre d'itérations est connu
- `while` : Parfaite pour les conditions d'arrêt dynamiques
- `do-while` : Utilisée quand au moins une exécution est nécessaire

Bonnes pratiques :

- Toujours initialiser les variables d'accumulation
- Vérifier les conditions d'arrêt pour éviter les boucles infinies
- Gérer les cas particuliers (nombres négatifs, zéro, etc.)
- Utiliser des noms de variables explicites pour les compteurs