Exercices sur les Fonctions en C

Challenge 1 : Addition avec Fonction

Créez une fonction qui additionne deux entiers et retourne le résultat.

Signature : `int somme(int a, int b)`

Programme principal : Saisir deux nombres, appeler la fonction et afficher le résultat.

Objectif : Maîtriser la création de fonctions simples avec paramètres et valeur de retour.

Challenge 2 : Multiplication avec Fonction

Développez une fonction qui multiplie deux entiers et retourne le produit.

Signature : `int produit(int a, int b)`

Programme principal : Tester avec différents nombres et afficher les résultats.

Objectif : Renforcer l'utilisation des fonctions pour les opérations arithmétiques.

Challenge 3 : Recherche du Maximum

Créez une fonction qui détermine le plus grand de deux entiers.

Signature : `int maximum(int a, int b)`

Méthode : Utiliser une condition `if-else` ou l'opérateur ternaire.

Objectif : Implémenter la logique conditionnelle dans les fonctions.

Challenge 4 : Recherche du Minimum

Développez une fonction qui trouve le plus petit de deux entiers.

Signature : `int minimum(int a, int b)`

Extension possible : Adapter pour fonctionner avec trois ou plusieurs nombres.

Objectif : Compléter les fonctions de comparaison de base.

Challenge 5 : Calcul de Factorielle Récursive

Créez une fonction qui calcule la factorielle d'un nombre entier positif.

Signature : `long long factorielle(int n)`

Approches possibles :

- Récursive : `n! = n × (n-1)!`
- Itérative : Boucle de multiplication

Cas particulier : `0! = 1`

Objectif : Comprendre la récursion et les calculs cumulatifs.

Challenge 6 : N-ième Terme de Fibonacci

Développez une fonction qui calcule le n-ième terme de la suite de Fibonacci.

Signature : `long long fibonacci(int n)`

Définition : F(0) = 0, F(1) = 1, F(n) = F(n-1) + F(n-2)

Approches :

- Récursive (simple mais lente)
- Itérative (efficace)

Objectif : Implémenter des suites mathématiques avec les fonctions.

Challenge 7 : Inversion de Chaîne de Caractères

Créez une fonction qui inverse une chaîne de caractères in-place.

Signature : `void inverser_chaine(char str[])`

Méthode :

1. Utiliser deux indices (début et fin)
2. Échanger les caractères symétriquement
3. Avancer vers le centre

Note : Modification directe du tableau passé en paramètre.

Objectif : Manipuler les chaînes de caractères et comprendre le passage par référence.

Challenge 8 : Test de Parité

Développez une fonction qui vérifie si un nombre est pair ou impair.

Signature : `int est_pair(int nombre)`

Valeurs de retour :

- `1` (vrai) si le nombre est pair
- `0` (faux) si le nombre est impair

Test : Utiliser l'opérateur modulo `%`

Objectif : Créer des fonctions booléennes et comprendre les conventions de retour.

Concepts Clés des Fonctions

Déclaration et Définition

```
// Déclaration (prototype)
int ma_fonction(int param1, int param2);

// Définition
int ma_fonction(int param1, int param2) {
    // Corps de la fonction
    return resultat;
}
```

Types de Paramètres

- **Passage par valeur :** Les modifications n'affectent pas l'original
- **Passage par référence :** Utiliser des pointeurs ou des tableaux

Bonnes Pratiques

- **Noms explicites :** Fonctions et paramètres avec des noms clairs
- **Une responsabilité :** Chaque fonction fait une seule chose
- **Gestion d'erreurs :** Valider les paramètres d'entrée
- **Documentation :** Commenter le but et les paramètres

Avantages des Fonctions

- **Réutilisabilité :** Code utilisable plusieurs fois
- **Modularité :** Division du programme en blocs logiques
- **Lisibilité :** Code plus facile à comprendre
- **Maintenance :** Modifications localisées

Extensions Possibles

- Créer des fonctions qui acceptent des tableaux
- Implémenter des fonctions mathématiques avancées
- Développer une bibliothèque de fonctions utilitaires
- Utiliser la récursion pour résoudre des problèmes complexes