Défi Personnel : 30 Jours d'Algorithmique, Pascal/C & Git

Objectif : développer la logique algorithmique, la discipline personnelle et les bases professionnelles en programmation structurée (Pascal ou C) et en gestion de versions avec Git. Ce défi est non noté et repose sur l'engagement personnel.

Règle fondamentale

Chaque jour : analyser sur papier, coder en Pascal ou en C, puis effectuer un commit Git. Principe clé : 1 jour – 1 algorithme – 1 commit.

Exercices détaillés – Jour par jour

Jour 1 – Affichage simple

Écrire un programme qui affiche le message « Bonjour le monde ». Aucune saisie ni calcul. Objectif : comprendre la structure minimale d'un programme.

Jour 2 – Somme de deux nombres

Lire deux nombres saisis par l'utilisateur, calculer leur somme et afficher le résultat. Travailler avec des variables et des opérateurs arithmétiques.

Jour 3 – Aire et périmètre d'un rectangle

Lire la longueur et la largeur d'un rectangle, calculer l'aire et le périmètre, puis afficher les résultats.

Jour 4 – Échange de deux variables

Lire deux variables A et B, échanger leurs valeurs et afficher le résultat. Bonus : réaliser l'échange sans variable temporaire.

Jour 5 – Conversion du temps

Lire une durée en secondes et la convertir en heures, minutes et secondes restantes.

Jour 6 – Pair ou impair

Lire un nombre entier et indiquer s'il est pair ou impair en utilisant une condition.

Jour 7 – Maximum de trois nombres

Lire trois nombres et afficher le plus grand. Les nombres peuvent être égaux.

Jour 8 – Année bissextile

Lire une année et déterminer si elle est bissextile selon les règles connues.

Jour 9 – Moyenne et mention

Lire trois notes, calculer la moyenne et afficher la mention correspondante.

Jour 10 – Conditions combinées

Lire plusieurs informations (ex : âge, sexe) et afficher un résultat en combinant plusieurs conditions.

Jour 11 – Affichage de 1 à N

Lire un nombre N et afficher tous les entiers de 1 à N à l'aide d'une boucle.

Jour 12 – Table de multiplication

Lire un nombre et afficher sa table de multiplication de 1 à 10.

Jour 13 – Factorielle

Lire un entier positif N et calculer sa factorielle à l'aide d'une boucle.

Jour 14 – Suite de Fibonacci

Lire un nombre N et afficher les N premiers termes de la suite de Fibonacci.

Jour 15 – Somme des N premiers entiers

Lire un entier N et calculer la somme des entiers de 1 à N.

Jour 16 – Tableau : saisie et affichage

Lire la taille d'un tableau, saisir les valeurs et les afficher.

Jour 17 – Minimum et maximum d'un tableau

À partir d'un tableau saisi, rechercher et afficher le minimum et le maximum.

Jour 18 – Tri par sélection

Trier un tableau par ordre croissant en utilisant l'algorithme du tri par sélection.

Jour 19 – Tri à bulles

Trier un tableau par ordre croissant à l'aide du tri à bulles.

Jour 20 – Matrice 2D

Lire une matrice et calculer la somme des lignes et des colonnes.

Jour 21 – Longueur d'une chaîne

Lire une chaîne de caractères et compter sa longueur sans utiliser de fonction intégrée.

Jour 22 – Palindrome

Vérifier si un mot saisi est un palindrome.

Jour 23 – Chiffrement de César

Implémenter un chiffrement de César avec un décalage donné.

Jour 24 – Fonction puissance

Écrire une fonction qui calcule a puissance b sans utiliser de fonction prédéfinie.

Jour 25 – Menu interactif

Créer un menu permettant d'appeler plusieurs fonctions selon le choix de l'utilisateur.

Jour 26 – Enregistrement simple

Créer un enregistrement (structure) représentant un étudiant : nom, âge, note.

Jour 27 – Tableau d'enregistrements

Gérer un tableau d'étudiants avec saisie et affichage.

Jour 28 – Recherche dans les enregistrements

Rechercher un étudiant par son nom dans un tableau d'enregistrements.

Jour 29 – Mini-projet libre

Réaliser un petit projet au choix : gestion de notes, de produits ou d'emprunts.

Jour 30 – Finalisation

Nettoyer le code, vérifier les commits Git et rédiger un README simple décrivant le travail réalisé.

Conclusion

Ce défi vise à installer une discipline durable. Celui qui termine les 30 jours aura acquis une base solide en algorithmique, en programmation Pascal ou C et en utilisation de Git.