Sommaire

1. Introduction : pourquoi un fondement plutôt qu'un langage .
   1. Objectifs et limites du propos
   2. Comparaison entre langages, idées communes et divergentes
2. Notions de base : variables, références et mémoire
   1. Types de données primitifs
   2. Variables et affectation
   3. Pointeurs, références, aliasing
   4. Allocation statique et dynamique
3. Structures de données : tableaux, enregistrements, structures.
   1. Tableaux unidimensionnels et multidimensionnels
   2. Structures / enregistrements
   3. Combinaisons (tableaux de structures, etc.)
4. Fonctions et modularisation
   1. Déclaration, définition, prototypes
   2. Passage de paramètres (par valeur, par référence)
   3. Valeurs de retour multiples / conventions
   4. Portée, liaison (scope), visibilité
5. Pile d'exécution, appels et retour
   1. Modèle de pile (stack)
   2. Comment un appel fonctionne "en machine"
   3. Variables locales, sauvegarde de contexte
   4. Recursion et profondeur d'appel
6. Récursivité et ses variantes
   1. Récursion simple / multiple
   2. Récursion terminale
   3. Cas limites, débordement de pile

4. Comparaison itératif vs récursif

7. Traduction vers une machine virtuelle simplifiée

1. Architecture d'une machine virtuelle pédagogique

2. Instruction, cycle fetch-execute

3. Modélisation de la mémoire, du saut, des appels

8. Impact des choix de conception : performance, mémoire, fiabilité

1. Coûts des appels

2. Optimisations possibles

3. Gestion des erreurs

9. Exemples en plusieurs langages

1. Java

2. C / C++

3. Autres exemples (selon les auteurs)

4. Comparaisons interlangages

10. Cas d'étude / programmes commentés

1. Exemples de programmes simples (fichiers, calculs)

2. Projets un peu plus élaborés

3. Analyse de performance

11. Conclusion : vers une programmation éclairée

1. Résumé des concepts

2. Recommandations pour le code "bon"

3. Pistes pour aller plus loin

12. Annexes / ressources / codes sources

1. Codes source téléchargeables

2. Glossaire des termes

3. Bibliographie