Fiche du cours

55 h

Titre :

C200 - Programmation C 1

Description :

Apprenez le langage C en partant de zéro jusqu'aux bases "intermédiaires" indispensables : syntaxe, contrôle de flux, fonctions, tableaux & chaînes, pointeurs, structures, mémoire dynamique, fichiers, modularisation et outillage (gcc/clang, Makefile, gdb, Valgrind/Sanitizers). On vise un C propre et sûr (discipline mémoire), avec de petits programmes console et l'implémentation de structures de données simples.

Objectifs :

- Comprendre la chaîne de compilation (préprocesseur → compile → link) et utiliser gcc/clang avec Make.*
- Écrire du C lisible : types, expressions, contrôle de flux, fonctions, en-têtes (.h) et modules (.c).*
- Manipuler tableaux & chaînes (string.h) sans débordements ; pratiquer pointeurs (y compris pointeurs de fonctions).*
- Gérer la mémoire dynamique (malloc/calloc/realloc/free) en respectant l'ownership et en évitant les fuites.*
- Lire/écrire des fichiers (texte & binaire), gérer les codes d'erreur (errno).*
- Implémenter liste chaînée, pile, file (bases) et déboguer avec gdb, Valgrind ou Sanitizers.

Chapitres :

1. Outils & workflow : gcc/clang, flags -Wall -Wextra, Makefile, structure d'un projet*
2. Bases du langage : types, opérateurs, I/O simples (printf/scanf – précautions)*
3. Contrôle de flux : if/else, switch, boucles for/while, break/continue*
4. Fonctions & portée : paramètres, retour, const, en-têtes (.h), include guards, préprocesseur (macros)*
5. Tableaux & chaînes : char[], terminator '\0', fonctions de string.h, prévention des overflow*
6. Pointeurs : adresse/déréférencement, pointeurs vs tableaux, arithmétique, pointeurs de fonctions (aperçu)*
7. Structures, enum, typedef : modèles de données, passage par valeur/référence*
8. Mémoire dynamique : malloc/calloc/realloc/free, règles d'ownership, aliasing, patterns d'initialisation*

9. Fichiers (I/O) : fopen/fclose, fscanf/fprintf, fread/fwrite, binaire vs texte, gestion d’erreurs (errno)*
10. Structures de données & qualité : liste chaînée, pile, file ; gdb, Valgrind/AddressSanitizer, conventions de code

À la fin :

Vous saurez écrire et structurer des programmes C fiables, manipuler pointeurs et mémoire dynamique sans fuites, travailler avec les fichiers, modulariser votre code et déboguer efficacement — une base solide pour la programmation système ou un C2 plus avancé (réseau, threads, performances). Vous maîtriserez aussi la chaîne de compilation et les outils (gcc/clang, Makefile, gdb, Valgrind/Sanitizers) pour garantir la qualité et la stabilité.