

Chapitre 1 : Premiers programmes en C

- Types de base (`int`, `float`, `bool`, `char`, `char*`) et opérateurs sur ces types
- Structure d'un programme C simple : `#include`, définitions de fonction, `main`
- Fonctions : syntaxe, description et hypothèses en commentaire
- Affichage et saisie (`printf`, `scanf`), spécificateurs de format `%d`, `%f`, `%c`, `%s`
- Alternative (`if`, `else`, `else if`)
- Boucles `while`
- Variant de boucle et preuve de terminaison de fonction
- Invariant de boucle et preuve de correction de fonction
- Nombreux exemples : division euclidienne, PGCD, suite récurrente, évaluation de polynôme, puissance et multiplication rapide, type d'un triangle...

Chapitre 2 : Encodage des nombres

- Encodage des entiers naturels en base $b \in \mathbb{N} \setminus \{0, 1\}$
 - existence de l'écriture en base b d'un entier nature
 - quasi-unicité de l'écriture en base b
 - valeur d'un nombre écrit en base b
 - algorithme d'addition de deux nombres écrits en base b
- Encodage des entiers relatifs en base $b \in \mathbb{N} \setminus \{0, 1\}$
 - rôle particulier du premier bit, donc travail à taille fixée
 - intervalle des entiers encodés sur l bits
 - existence et unicité de l'écriture en base b (à taille fixée)
 - valeur d'un nombre écrit en base b
 - passage à l'opposé par complément à 2
 - addition par l'algorithme d'addition pour les entiers naturels
- Encodage des flottants en C : mantisse, exposant, cas particuliers non standard

Chapitre 3 : Pointeurs, tableaux et structures en C

- Organisation de la mémoire, bits, octets, adresse mémoire, différence entre pile et tas
- Pointeurs, opérateur de dé-référencement
- Passage par référence vs passage par valeur à une fonction
- Tableaux statiques : déclaration, initialisation explicite, accès aux valeurs
- Allocation dynamique d'espace mémoire : `malloc`, `sizeof`, `free`
- Déclaration de structure, création de structures

Chapitre 4 : Complexité

- Notion de complexité pire cas d'une fonction
- Comportement asymptotique et notations de Landau
- Quelques sommes à connaître
- Exemples sur des fonctions simples, comparaison de différents codes : erreurs à éviter
- Exemple moins évident : complexité de la recherche dichotomique dans un tableau trié

Chapitre 5 : Structures de données séquentielles

- Structure de données abstraite
- Pile : déf. de la structure abstraite, implémentation par liste simplement chaînée, par tableau
- File : définition de la structure abstraite, implémentation par liste simplement chaînée avec pointeurs de fin, par tableau, avec deux piles
- Liste : définition de la structure abstraite, implémentation par liste doublement chaînée
- Complexité amortie des ajouts dans une table dynamique

À travers les TP et le projet

- compilation simple avec `gcc`, en ligne de commande
- organisation du code en `xxx.h`, `xxx.c`, `test_xxx.c`
- compilation séparée : d'abord création des `xxx.o`, puis édition des liens
- utilisation de `Makefile`
- passage de paramètres à un programme C (`argc`, `argv`)
- lecture dans un fichier avec `cat`, `|` et `getchar`
- bien commenter son code, bien l'indenter