

Objectifs du Cours

1 Compétences attendues en Algorithmique

- Connaître la syntaxe du pseudo langage algorithmique
- Connaître les notions suivantes : les variables, les constantes, les fonctions, les actions, les boucles *tant que*, les boucles *pour*.
- Savoir écrire précisément la déclaration d'une action et l'appel de cette action (types, syntaxe, ...). Pareil pour une fonction. Savoir décrire une exécution à l'aide de schémas d'exécution. Savoir ce qu'est la signature d'une fonction.
- Connaître les principes des tris courants (insertion, sélection, bulle, fusion et rapide). Savoir retrouver les algorithmes en pseudocode. Savoir très rapidement écrire le tri sélection.
- Les tableaux : déclaration, initialisation, savoir parcourir un tableau avec une boucle *pour*, savoir parcourir avec une boucle *tant que* lorsque l'on veut s'arrêter avant.
- Les matrices ou tableaux 2d : idem
- Les chaînes de caractères codées avec marqueur de fin : parcours, et les algorithmes courants (longueur, concaténation, ...)
- Les variables modifiables : utilisation à bon escient, et déclaration à l'aide des variables D et/ou R.
- Les structures : déclaration statique.
- Conception/Analyse : savoir analyser un problème et le découper en petits algorithmes. Savoir choisir entre fonction et action, et déterminer les variables d'entrée nécessaires.
- Savoir analyser son algorithme en terme de coût. Connaître le coût en moyenne des principaux tris.

2 Compétences attendues en Programmation C

- La même chose que la partie algorithmique : variables, types, constantes, fonctions, procédures, appels de fonctions, tableaux et matrices, structures. . .
- Booléens : savoir qu'ils n'existent pas en C en tant que tels, mais que l'on peut (et l'on doit) utiliser `stdbool`.
- Pointeurs : utiliser `*` et `&` à bon escient. Utilisation des pointeurs dans le cas où l'on utilise des paramètres modifiables en algorithmique.
- Syntaxe des tableaux et matrices : déclaration, accès à une case, parcours dans tous les sens.
- Les chaînes de caractères en C. La différence entre `"a"` et `'a'`.
- Entrées/sorties : `printf`, `scanf`, et utilisation pour demander des informations à l'utilisateur du programme. Utilisation de `getc` et `getline`.
- Principes généraux de compilation pratique : ce que sont les `.o`, `.h` (savoir faire un `.h`), les bibliothèques, et comment compiler à la ligne de commande ou avec un `Makefile`. Différence entre compilation et exécution.
- Utiliser les fonctions données dans les bibliothèques, par exemple dans `string.h`.

3 Compétences TP

et plus spécifiquement en ce qui concerne le C :

- Utiliser un éditeur efficace pour éditer du C et éventuellement compiler
- Compiler, exécuter à la ligne de commande.
- Savoir lire l'énoncé, répondre aux questions posées, papier, crayon, expliquer, faire des tests pertinents.
- Commenter, documenter.
- Savoir un peu utiliser gdb.