L2 Informatique Semestre S3 Automne 2021
Structures de Données et Algorithmes SDA1

Responsable SDA1 :
Dominique Bechmann
bechmann@unistra.fr

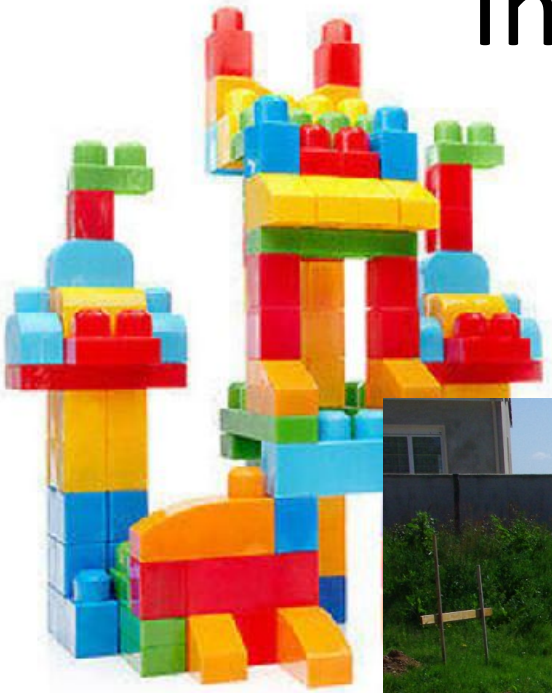
Organisation pratique

- 20 heures de cours le mardi de 8H-10H
- 22 heures de travaux dirigés
- 12 heures de travaux pratiques

Contrôles

- CC1 30 % contrôle sur table en TD en octobre
- CC2 30 % contrôle sur ordinateur en TP en décembre (dernière des 6 séances de TP)
- CC3 40 % contrôle écrit mardi 14 décembre

Introduction



CONSTRUCTION DE PROGRAMMES

- **Objectif du cours** : apprendre à résoudre un problème informatique en suivant une démarche concrète de résolution des problèmes :

1. spécification informelle puis formelle
2. décomposition fonctionnelle du problème
3. programmation s'appuyant sur la spécification
4. tests par des jeux d'essai reprenant tous les cas particuliers

Spécification informelle

C'est toujours le point de départ.

La spécification *informelle* du problème est souvent imprécise, redondante, incomplète, voire incohérente.

- **De l'informel au formel**

C'est une sorte de mise en équations.

Spécification formelle

- **Structuration horizontale** : *classifier* les données, nous disons les *objets*, et les *opérations* qui entrent en jeu.
- **Raffinement vertical** : une suite de spécifications, du cahier des charges au programme, énoncés du problème donné à plusieurs *niveaux d'abstraction*.
  - **De l'abstrait au concret** : Choix des Structures de données et des procédés algorithmiques.

Procédés de résolution

- **Décomposition fonctionnelle** : décomposer un problème complexe en un certain nombre de problèmes simples que nous savons résoudre.
- **Décomposition récursive** : un problème P que nous ne savons pas résoudre sauf dans un cas simple P0, ou si on nous fournit la solution d'un cas un peu moins complexe alors nous pouvons en déduire la solution de P.

➢ **Transformer l'énoncé formel autant que nécessaire …**

Programmation visée

A. *correction* (ou *justesse*, *fiabilité*) : il fait ce qu'il est censé faire ;
B. *robustesse* : il prévoit tous les cas de mauvaises utilisations ;
C. *rapidité* : il a des temps de réponse convenables ;
D. *économie* : il n'abuse pas des ressources, notamment de la place mémoire ;
E. *convivialité* : il est facile et agréable à utiliser ;
F. *facilité de maintenance* : il peut être aisément corrigé ou modifié ;
G. *portabilité* : il peut être adapté à une autre machine.

Prouver la qualité (A et B)

- En évaluant la qualité d'une spécification via des preuves de consistance et de complétude,
- En évaluant la validité d'une implantation par rapport à la spécification via des preuves de correction,
- En prouvant la terminaison des fonctions.

Chapitre 3 : 3H CM – 2H TD (1 séance)

Qualité d'une spécification et validité d'une implantation

Evaluer la rapidité (C et D)

- écrire des algorithmes efficaces en terme de complexité en temps et en espace :

➢ Calculer l'ordre de grandeur de la complexité,

➢ Identifier si un algorithme est optimal.

Chapitre 4 : 3H CM - 2H TD (1 séance)

Initiation à l'analyse d'algorithmes (complexité)

A ne pas négliger …

- Code propre et ordonné, fonctions indentées méthodiquement,
- Présentation facilitant la compréhension et le suivi,
- écrire des algorithmes concis,
- fonctions courtes.

Et le **traitement des jeux de données « type »** via une exécution du programme pour chacun.

= SDA1 =
Structures de données linéaires

- Spécifier un type de données linéaires avec ses opérations :
  - donner la liste des axiomes et des pré-conditions, respecter les types des prototypes,
  - identifier les différents cas pour chaque opération,
  - intégrer toutes les configurations possibles et imaginables de données, avec une réponse adaptée.

= SDA1 =
Structures de données linéaires

- Programmer les variantes d'une structure de données linéaire avec ses opérations :
  - en distinguant les types de programmation , respecter les types des prototypes,
  - en comparant les avantages et les inconvénients des structures de données et de leurs implantations

Chapitre 2 : 8H CM - 12H TD - 10H TP

Piles, Files et Listes

S'appuyer sur la spécification pour programmer

Dans le programme on doit retrouver :

- Les fonctions spécifiées avec leurs pré-conditions, leurs prototypes,
- Les algorithmes de la spécification avec tous ses cas particuliers.

➢ Traduire une définition récursive en une itération et réciproquement.

Questions ?

On démarre !

Chapitre 1 : 5H CM - 6H TD

Rappels programmation et introduction à la spécification