# INF304

vendredi 7 septembre 2018 13:52

## Pourquoi tester un programme ?

-> Trouver les bugs

Avec une grande taille & complexité il y a quasi nécessairement des erreurs.
Le test est une partie importante dans le processus de développement. (environ 30-50% temps de dev)

## Méthodes de tests

Le test dans le processus de développement

- Tester à posteriori (le plus classique mais peu efficace)
- Tester au fur et à mesure (favoris des enseignants)
- Test-driven development (TDD)
  - On commence par écrire les tests avant de programmer et on cherche une solution à ceux-ci.
  - Ce qu'on a fait en Ocaml

## Différentes formes de tests pour différents objectifs

Des tests peuvent être demandés en DS, ne confondez pas

- Tests **fonctionnels** (ou "de conformité") : tester que le programme est conforme à sa spécification.
- Tests de **robustesse** : Tester le programme dans un environnement dégradé
- Tests de **sécurité**
- Tests **unitaires/ d'intégration**
- Tests de **performance**
- Tests **boite blanche** : en connaissant le contenu du programme
  - Approche 1 : **Couverture des instructions** : Créer des tests qui passent par toutes les portions du programme (entre dans toutes les boucles & conditions
  - Approche 2 : Partitionnement du domaine d'entrées
    - Cas limite 1 (ex: n=2)
    - Cas général 1 (ex: n>2)
- Tests **boite noire** : Sans regarder le contenu (source) du programme, seulement la spécification

Pour faire une spécification, préférer la simplicité à la rigueur. Il faut savoir éluder les situations inutiles et pourtant présente effectivement

## Types abstraits

- Associer un **type de données** et des **opérations** sur les valeurs de ce type
- Séparation des préoccupations
  - Séparer le pourquoi et le comment
- 3 types d'opérations :
  - De construction
  - D'accès
  - De modification / de transformation

## Phases de la compilation:

1. **Édition du fichier source :** fichier texte contenant le programme — nécessite un éditeur de texte (emacs, vi).
2. **Traitement par le préprocesseur :** le fichier source est traité par un *préprocesseur* qui fait des transformations purement textuelles (remplacement de chaînes de caractères, inclusion d'autres fichiers source, etc).
3. **La compilation :** le fichier engendré par le préprocesseur est traduit en *assembleur* i.e. en une suite d'instructions associées aux fonctionnalités du microprocesseur (faire une addition, etc).
4. **L'assemblage :** transforme le code assembleur en un fichier *objet* i.e. compréhensible par le processeur.
5. **L'édition de liens :** afin d'utiliser des librairies de fonctions déjà écrites, un programme est séparé en plusieurs fichiers source. Une fois le code source assemblé, il faut *lier* entre eux les fichiers objets. L'édition de liens produit un fichier *exécutable*.

À partir de l'adresse <http://www.lifl.fr/~sedoglav/PDC/main006.html>