Programmation rapide

2023-2024

Fonctionnement

Un CTP par semaine de 2h.
Équipes de 5 étudiant-e-s (composition fixe).

Dépôt git du cours :
https://moule.informatique.univ-paris-diderot.fr/poulalho/prog-eff-2023-2024

Évaluation :
Session 1 : 50% CC + 50% examen final
CC :

- Participation aux séances de CTP
- Qualité des programmes remis

Examen final :

- un programme à réaliser en groupe de 2 ou 3 en 3h.

Session 2 : 50% CC + 50% examen

Les équipes

Les équipes doivent être fixées au plus tard vendredi soir !

-> un mail à l'enseignant-e du groupe.

En retour, nous vous donnerons un numéro d'équipe, vous pourrez alors créer un projet sur le **nouveau** serveur **gitlab : moule**.

-> Le travail par équipe commence dès le CTP2.

Langages suggérés : Java, Python, C…
Possibles : OCaml, Rust…

Utiliser git à l'UFR

Il faut dorénavant utiliser impérativement le nouveau serveur gitlab de l'UFR, `moule`, et non plus `gaufre`.

- (étape 0 : si ce n'est pas déjà fait, générer une paire de clés ssh avec `ssh-keygen`, par exemple `id-dsa` et `id-dsa.pub`)

- étape 1 : se connecter une première fois à `moule.informatique.univ-paris-diderot.fr` (login et mot de passe du compte UFR)

- étape 2 : cliquer sur l'icône ronde (avatar) dans la colonne de gauche, choisir « Edit profile », puis « SSH keys »

- étape 3 : cliquer sur « Add new key », copier-coller `id-dsa.pub` dans la fenêtre « Key » et confirmer en cliquant sur « Add key »

- étape 4 : récupérer le dépôt du cours avec `git clone git@moule.informatique.univ-paris-diderot.fr:poulalho/prog-eff-2023-2024.git`

- étape 5 : créer votre propre dépôt d'équipe, et ajouter tous les membres de l'équipe comme `Maintainer`, et les enseignants (François Laroussinie et Dominique Poulalhon) comme `Reporter`

Déroulement des CTP

3 types de séances :

Classique :

- vous programmez en 2 heures une solution à un problème; vous disposez d'un sujet et de fichiers d'exemples.

Debriefing :

- on reprend un problème donné et on donne quelques éléments d'algorithmique pour compléter le travail effectué.

Présentation :

- après une séance « classique », chaque groupe présente son approche aux autres groupes.

Objectifs

Programmer rapidement… à plusieurs.

Bien sûr, on vise à obtenir un programme efficace (mais ce n'est pas toujours possible).

On se pose des *problèmes algorithmiques* : décomposer le problème, faire le lien avec des algorithmes vus en cours, évaluer la complexité…

On se pose des *problèmes de programmation* : faire un parseur, manipuler les entrées/sorties, tester son programme, debugger… le tout en temps limité !

Les problèmes étudiés

On verra essentiellement deux types de problèmes :

- les problèmes pour lesquels on peut développer des algorithmes exacts et efficaces (problèmes « simples »)

- les problèmes pour lesquels on doit développer des heuristiques ou des approximations pour avoir des résultats (problèmes « durs »)

Dans le premier cas, on vérifiera que le programme fourni est correct (donne les bons résultats) y compris sur des instances de grande taille.

Dans le second cas, on évaluera la qualité des résultats obtenus et la capacité du programme à traiter les problèmes de grande taille.

Les entrée/sorties

En général, les tests sont décrits dans des fichiers d'entrée (`.in`).
Les résultats doivent être écrits dans des fichiers (`.out`).

—> il faut savoir manipuler des fichiers d'entrée/sortie.

Sachez le faire rapidement !

Une équipe ?

C’est un travail d’équipe !

Discuter du problème,
discuter des algos,
se répartir le code à faire,
tester des idées,
debugger le code des collègues…