Fonctionnement des TP de langage C / C++

Principe n°1 Acquisition et consolidation	Chaque étudiant doit faire tous les exercices d'acquisition et si possible un ou plusieurs exercices de consolidation, "à la carte".
Principe n°2 Validation expérimentale	Tout exercice doit donner lieu à la réalisation d'un programme solution et d'un programme de test permettant la **validation expérimentale** de la solution proposée.
Principe n°3 Répartition du travail	Dans le cas d'un travail en binôme, un étudiant réalise le programme solution et l'autre le programme de test. Pour l'exercice suivant, les rôles sont permutés.
Principe n°4 Validation du travail par l'enseignant	A l'issue de chaque exercice, l'enseignant doit vérifier la solution proposée et le programme de test, et peut demander des corrections ou modifications.
Principe n°5 Évaluation des compétences	Les compétences des étudiants sont évaluées individuellement par l'enseignant après chaque exercice traité. Lors de cette évaluation, l'enseignant examine le travail réalisé et peut poser à l'étudiant concerné des questions relatives au programme solution ou au programme de test proposé.
Principe n°6 Suivi individuel	A l'issue de l'évaluation de chaque exercice, l'enseignant met à jour le profil de compétence de chacun des étudiants concernés.

Niveau	Signification
R	**Rudiments.** Vous avez retenu quelques notions mais vous avez des difficulté à les utiliser, même dans certains cas simples.
P	**Maîtrise partielle.** Vous savez faire face à certaines situations simples en trouvant des solutions acceptables.
A	**Compétence acquise.** Vous pouvez faire face seul à la plupart des situations ne comportant pas de piège où de difficulté particulière.
C	**Compétence consolidée.** Vous êtes capable de trouver des solutions pertinentes y compris face à des situations plutôt complexes, présentant certaines difficulté au regard du niveau attendu dans le cadre de l'enseignement dispensé.

TP n°1 de langage C : scalaires, fonctions, pointeurs

Vous devez faire tous les exercices d'acquisition. Lorsque ces exercices sont maîtrisés (si applicable), vous pouvez tenter de faire un ou plusieurs exercices de consolidation. Choisissez les exercices qui présentent pour vous une difficulté raisonnable et qui sont donc susceptibles de vous faire progresser.

En cas de difficulté, demandez un indice ou une explication à l'enseignant.

Exercice 1 (acquisition)

Réalisez une fonction `void dnz(int* p)` qui décrémente la variable pointée par p si sa valeur est positive et n'a aucun effet dans le cas contraire.

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 2 (acquisition)

Réalisez une fonction `void tlc(int n)` qui affiche à l'écran tous les couples (x,y) tels que x et y sont compris entre 1 et n.

Par exemple, `tlc(2)` doit afficher (1,1), (1,2), (2,1), (2,2).

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 3 (acquisition)

La fonction `rand()` retourne une valeur pseudo-aléatoire comprise entre 0 et la constante prédéfinie `RAND_MAX`.

Réalisez une fonction `int random(int a, int b)` qui retourne un nombre pseudo-aléatoire compris entre **a** et **b** inclus.

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 4 (acquisition)

Réalisez une fonction `int isPrime(int x)` qui retourne **vrai** si **x** est un nombre premier, **faux** sinon.

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 5 (consolidation)

Réalisez une fonction `void div(int a, int b, int n)` qui affiche le résultat de la division de a par b avec n chiffres après la virgule (n pouvant être très grand, par exemple 1000).

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 6 (consolidation)

Réalisez *sans utiliser de tableau* une fonction *récursive* `void inv(int n)` permet la saisie de n entiers au clavier et l'affichage de ces entier dans *l'ordre inverse* de leur saisie.

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 7 (consolidation)

On souhaite évaluer la probabilité qu'une suite aléatoire de **n** valeurs comprises entre 1 et un entier **max** soit croissante.

Réalisez une fonction `double stat(int n, int max, int k)` qui retourne une estimation de cette probabilité basée sur la génération de **m** séquences de **n** nombres aléatoires.

Décomposez ce travail en plusieurs fonctions.

Validez expérimentalement votre solution en réalisant les tests appropriés.

Exercice 8 (consolidation)

On souhaite simuler le déroulement d'un jeu dans lequel un candidat cherche à deviner un nombre secret produit par un arbitre.

A chaque tentative, le candidat propose une valeur à l'arbitre et ce dernier dit au candidat si la valeur est correcte, trop grande, ou trop petite.

Décomposez ce travail en plusieurs fonctions.

Validez expérimentalement votre solution en réalisant les tests appropriés.