

Chapitre 3 : Boucles simples – TD

Exercice 1 - Test de compréhension*

Question 1 : syntaxe

Quelle est la syntaxe du `while`?

Question 2 : indentation

Qu'est-ce qu'une *indentation* ? À quoi sert-elle ?

Question 3 : itération

Qu'est-ce qu'une *itération* ?

Question 4 : Nombre minimal d'itérations

Y a-t-il toujours au moins une itération dans une boucle ? Justifier votre réponse.

Question 5 : Initialisation

Qu'est-ce qu'une *initialisation* ? Pourquoi est-ce une notion importante dans le cas des boucles ?

Question 6 : Affectation compacte

À quoi correspondent les instructions compactes `i+=3`, `j-=7`, `k*=2`, `h/=4`? Comment écrire de manière compacte `a=3+a`, `a=8*a`, `a=2-a`, `a=3/a`?

Exercice 2 : Affichage d'entiers et décrémentation*

Afficher les entiers de 1 à 20 dans l'ordre décroissant.

Exercice 3 : Table de multiplication*

Afficher la table de multiplication par 7 des entiers de 1 jusqu'à 20 comme suit :

$$1 * 7 = 7$$

$$2 * 7 = 14$$

...

$$19 * 7 = 133$$

$$20 * 7 = 140$$

Exercice 4 : Saisie contrôlée d'un entier positif**

Ecrire un programme demandant à l'utilisateur de saisir un nombre entier positif. La saisie sera répétée jusqu'à ce que le nombre soit positif. On écrira un test unitaire permettant de vérifier la validité du résultat fourni.

Même question pour un nombre entier positif et multiple de 3.

Exercice 5 : Sommes séparées des nombres pairs et impairs**

Ecrire un programme permettant de calculer et d'afficher séparément les sommes d'une part des nombres pairs et d'autre part des nombres impairs compris entre 1 et 100 bornes incluses. On écrira les tests unitaires permettant de vérifier la validité des résultats fournis.

Exercice 6 - Boucle sans arrêt*

Ecrire un programme permettant d'afficher indéfiniment les entiers successifs à partir de 0.

Pour aller plus loin

Exercice 7 : Test de primalité**

Ecrire un programme permettant de vérifier si un nombre saisi par l'utilisateur est premier.

Rappel : Un nombre *premier* est un entier positif supérieur ou égal à 2 qui admet exactement deux diviseurs entiers et positifs qui sont 1 et lui-même. Le nombre 13 est donc un nombre premier, alors que 6 ne l'est pas puisque $6 = 2 \cdot 3$.

Exercice 8 : Chiffres romains***

Ecrire un programme demandant à l'utilisateur un nombre compris entre 1 et 50 et affichant ce nombre en chiffres romains.