Humanbooster

Algorithmie

TP02 – Pratique du TDD en binôme

DWWM PE6 LYON

Barthélémy DELUY –

Introduction

Objectifs :

- Maîtriser la logique en s'entraînant sur des algos de base
- Mettre en application le TDD
- Maîtriser git en travail collaboratif (avec Alice et Bob)

Git

Vous devez réaliser l'ensemble de ces mises en application sur Git, en suivant le TDD.

Pour chaque exercice, créez une nouvelle branche, et à la fin de l'exercice, fusionnez sur la branche principale.

- Créez un dépôt sur github pe6lyon-algo-tp02
- Clonez le dépôt dans VS Code
- Créer une branche « tdd1 »
- Faites l'algo 1
- Fusionnez avec la branche principale
- Créer une branche « tdd2 »
- Faites l'algo2
- ...

Consignes spécifiques au TDD

Vous allez pratiquer à fond git et le TDD. Le git log DOIT faire apparaître l'historique de commit suivant, énoncé par énoncé : (exemple se basant sur l'énoncé 1)

Alice & Bob : écriture des scénarios de test et du jeu d'essai
Alice : écriture du squelette de la routine mean
Alice : écriture du test testMean_N1(1, 1)
Bob : modification de la routine mean pour valider le test testMean_N1
Bob : écriture du test testMean_N2(10, 20)
Alice : modification de la routine mean pour valider le test testMean_N2

Énoncés

1. Moyenne de deux nombres entiers
2. Mention
3. Commande en volume
4. Somme des n premiers entiers
5. Suite de Fibonacci

1. Moyenne de deux nombres entiers

Votre collègue vous donne la signature suivante :

```
FONCTION mean(var nb1, var nb2) : décimal
```

Proposez un algorithme permettant de coder cette routine, puis la coder en PHP

2. Mention

Votre collègue vous donne la signature suivante :

```
FONCTION mention(var note) : chaîne de caractères|FAUX
```

Cette routine doit renvoyer :

- « AB » si 12 ≤ note <14
- « Bien » si 14 ≤ note < 16
- « TB » si 16 ≤ note < 18
- « Excellent » si note ≥ 18
- Attention, une note n’est valide que si elle est comprise entre 0 et 20 (renvoyer false en cas d’erreur)

3. Commande en volume

Votre collègue vous donne la signature suivante :

```
FONCTION getCost(var volume) : décimal|FAUX
```

Cette routine reçoit un nombre de produits commandés en paramètre, et renvoie le prix total en fonction du volume, selon la règle :

- les 100 premiers produits sont facturés 1 € HT pièce
- les 50 suivants sont facturés 0,98€HT pièce
- les 25 suivants sont facturés 0,95€HT pièce
- les 25 suivants sont facturés 0,90€HT pièce
- pour toute commande de plus de 500 articles, ils sont facturés 0,80€HT pièce pour l'intégralité de la commande
- la routine renvoie FAUX si le nombre de produits commandés n'est pas valide

Proposez un algorithme permettant de coder cette routine, puis codez-là en PHP

4. Somme des n premiers entiers

Votre collègue vous donne la signature suivante :

```
FONCTION sumNIntegers(var n) : entier|FAUX
```

Cette routine reçoit un nombre n en paramètre, et renvoie la somme des n premiers entiers en commençant à 1.

Exemple :

- sumNIntegers(1) => 1
- sumNIntegers(2) => 1+2 => 3
- sumNIntegers(3) => 1+2+3 => 6
- renvoyer FAUX si n n'est pas valide

Proposez un algorithme permettant de coder cette routine, puis codez-là en PHP

5. Suite de Fibonacci

Votre collègue vous donne la signature suivante :

```
FONCTION fibonacci(var n) : entier|FAUX
```

Cette routine reçoit un nombre n en paramètre, et renvoie le n-ième élément de la suite de Fibonacci.

Exemple :

- fibonacci(1) => 0
- fibonacci(2) => 1
- fibonacci(3) => 1
- fibonacci(6) => 5
- renvoyer FAUX si n n'est pas valide

Proposez un algorithme permettant de coder cette routine, puis codez-là en PHP

Conclusion