TP - consolidation

Objectif(s)

✯ Consolider les acquis de ces premières semaines

Exercices obligatoires

Exercice 1 – Triplets pythagoriciens

Derrière ce nom effrayant se cache un triangle [1]. Trois nombres entiers a, b et c forment un *triplet pythagoriciens* s'ils satisfont la propriété $a^2 + b^2 = c^2$.

Question 1

Écrivez une fonction `est_triplet` qui prend en paramètre trois entier et vérifie s'ils forment un triplet pythagoricen. La fonction retournera 1 si les trois nombres forment un triplet pythagoricien, 0 sinon.

Pour tester la fonction précédente, plutôt que de faire des affichages simples dans le main, nous allons utiliser une fonction de test qui sera appelée dans le main.

Question 2

Écrivez une fonction `test_triplet` qui ne prend pas d'argument, et teste la fonction ainsi : elle appelle la fonction précedente sur chacun des trois triplets $(3, 4, 5)$, $(63, 16, 65)$ et $(36, 77, 84)$. Si la fonction constate que l'une des réponses fournies est incorrecte (les deux premiers triplets sont pythagoriciens, le troisième ne l'est pas), elle affiche un message du type : "KO : réponse erronée de est_triplet sur l'entrée 3,4,5" (avec les valeurs qui ont fait échoué le test). Si la fonction précédente donne le bon résultat sur chacun des trois triplets, elle affiche "OK : test de la fonction est_triplet réussi".

Question 3

Écrivez une fonction qui affiche tous les triplets pythagoriciens ne comportant que des nombres inférieur à un paramètre n. On s'appliquera à afficher les triplets une seule fois, et dans l'ordre croissant ($3, 4, 5$ mais pas $4, 3, 5$).

Pour $n = 19$, vous devriez trouver $(3, 4, 5)$, $(6, 8, 10)$, $(5, 12, 13)$, $(9, 12, 15)$, $(8, 15, 17)$.

Question 4

Écrivez une fonction `nb_triplets` qui prend en paramètre un nombre n et retourne le nombre de triplets pythagoriciens dont les trois éléments sont compris entre 1 et n.

Vous devriez trouver 878 triplets ayant des valeurs strictement inférieures à 1000.

Exercice 2 – Dominos

Un domino est un rectangle séparé en deux parties dont chacune possède une valeur comprise entre 0 et 6 inclus. Dans la suite de cet exercice, on envisage des dominos généralisés dont la valeur peut aller de 0 à une constante `MAX_DOMINO`, initialement définie à 6 mais susceptible d'être modifiée.

Question 1

Déclarez la constante `MAX_DOMINO` à 6 en ajoutant en début de fichier

```
#define MAX_DOMINO 6
```

Puis écrivez une fonction `affiche_dominos` qui affiche la liste des dominos sous la forme suivante :

$[0,0]$ $[0,1]$ $[0,2]$ $[0,3]$ $[0,4]$ $[0,5]$ $[0,6]$
$[1,0]$ $[1,1]$ $[1,2]$ $[1,3]$ $[1,4]$ $[1,5]$ $[1,6]$

1. Vous reconnaitrez dans la formule suivante la relation satisfaite par les longueurs des trois côtés d'un triangle rectangle.

$[2,0]$ $[2,1]$ ...
...
$[6,0]$ $[6,1]$ $[6,2]$ $[6,3]$ $[6,4]$ $[6,5]$ $[6,6]$

Question 2

Dans la version précédente vous avez affiché des dominos en doublon ($[0,1]$ et $[1,0]$) qui n'existent qu'une fois dans le jeu de base.

Écrivez une fonction `affiche_dominos_sans_doublons` affichant sous une forme similaire à la question 1 les 28 dominos existant sans doublons. La première ligne comportera les mêmes dominos, la dernière ligne ne comportera que le domino $[6,6]$.

Pour la suite de l'exercice, on utilisera toujours les dominos sans doublons.

Question 3

On souhaite maintenant afficher les dominos ayant la somme des deux valeurs fixée (à un paramètre `total`), mais uniquement si un des côtés du domino est strictement plus grand que 3. Par exemple, pour un paramètre `total` égal à 6, on souhaite afficher :
$[0,6]$ $[1,5]$ $[2,4]$

Écrivez une fonction `affiche_dominos_egal_a(int total)` affichant les dominos correspondant.

Question 4

On souhaite calculer la somme des valeurs sur certains ensembles de dominos existant (toujours sans doublons). On veut pouvoir calculer soit :

- La somme des dominos sauf ceux possédant deux parties paires ;
- La somme des dominos sauf ceux possédant deux parties impaires ;
- la somme de tous les dominos.

Pour gérer cela on utilisera un nombre passé en paramètre :

- Si le nombre est strictement négatif, la fonction renverra la somme de tous les dominos **sauf** ceux ayant deux parties paires ;
- S'il vaut 0 la somme de tous les dominos ;
- S'il est strictement positif, on renverra la somme de tous les dominos **sauf** ceux ayant deux parties impaires.

Écrivez une fonction de prototype `int total_dominos(int somme)` calculant et retournant ces valeurs. Voici par exemple les résultats attendus pour `MAX_DOMINO` égal à 6 :

- `total_dominos(-1)` -> 108
- `total_dominos(0)` -> 168
- `total_dominos(1)` -> 132

Renforcements

Exercice 3 – Chiffres Romains

Question 1

Écrivez une fonction qui prend en paramètre un nombre entier n et affiche l'écriture de n en chiffres romains.

Question 2

Faites de même pour l'écriture Maya des nombres, utilisant la base 20. (cf `https://fr.wikipedia.org/wiki/Num%C3%A9ration_maya`). On utilisera quatre '-' pour faire une ligne horizontale, et le motif "`<(()>`" pour représenter le zéro.