TP C no 2

Objectifs:

Comprendre et mettre en œuvre les structures algorithmiques de bases en langage C

Modalités :

Le TP est disponible sous la forme d'un dépôt Github distribué de manière privée et nominative à l'ensemble des étudiants de la classe. Utilisez le lien communiqué sur Team pour activer votre dépôt en prenant soin d'utiliser votre adresse universitaire en junia.com.

Le TP est fourni sous la forme d'une solution Visual Studio 2019.

Nous vous invitons à réaliser un 'commit' après chaque exercice et à pousser l'ensemble du travail sur GitHub dès la fin de séance.

Exercice 1

Rappel :

```
// La boucle FOR est utilisée lorsqu'on connaît le nombre d'itérations à effectuer.
//
// D'autres boucles comme le WHILE expriment de façon plus naturelle qu'un évènement est
// attendu et va provoquer la sortie de boucle et donc mettre fin aux répétitions.
// Le langage C permet d'écrire toutes sortes de choses originales mais pas forcément très
// structurées donc sources de bugs et difficiles à maintenir.
// Dans le cadre de ces TPs, on utilise le C pour coder des algorithmes de manière structurée.
// On s'interdit donc d'utiliser des instructions comme le GOTO qui permet des sauts difficiles à
// modéliser en programmation structurée.
// Pour simplifier le propos, les trois types de structures suivantes (pseudo-langage) :
// Pour i allant de debut à la fin , faire instructions , fait.
// Tant que condition est vraie, faire instructions, fait.
// et, Répéter faire instructions, fait, jusqu'à condition est vraie.
// se traduisent respectivement en C par :
// for ( i=debut ; i<= fin ; i++) {   instructions ; }
// while (condition) { instructions ; }
// do { instructions ; } while ( !condition );
// Dans la structure do .. while , les instructions seront exécutées au moins une fois.
// REMARQUE : Penser à donner une valeur initiale aux variables !!
```

- **Somme des n premiers entiers positifs** :
  Ecrire un programme qui calcule la somme des n premiers entiers positifs, avec n > 1, en utilisant les trois types de boucles. La valeur de *n* (prendre 100 pour commencer) est fixée dans le programme. Afficher la valeur de n et la somme correspondante.
- **Trouver la valeur maximum de n sans dépassement de capacité :**
  La somme est représentée par une variable de type **unsigned short int,** quelle est la valeur maximum de **n** pour que la variable représentant la somme ait une valeur correcte ?
  Concevoir un test qui permettra de sortir de la boucle **while** de manière anticipée en cas de risque de dépassement de capacité.
  Aide : si *A + valeur <= B* alors *B – A >= valeur*. En calculant *B-A* on ne risque pas de faire un dépassement, contrairement à celui qui peut apparaître en calculant *A + valeur*.
- **Demander la valeur de n à l'utilisateur :**
  Modifier le programme, en ce qui concerne la boucle **for**, pour que la valeur de **n** soit demandée à l'utilisateur avec la fonction **scanf_s("%hu",&n)** et tester la boucle au-delà des valeurs permises pour **n**, que vaut la somme ?
  ```
  // scanf_s utilise l'adresse de la variable qui va stocker la valeur et non le nom de la
  // variable elle-même. Si on omet de mettre le &, un message du type 'segmentation fault'
  // apparait lors de l'exécution du programme.
  ```
- **Recommencer tant que n est trop grand :**
  Modifier le programme afin que l'utilisateur soit invité à saisir une nouvelle valeur pour **n** tant qu'il n'aura pas entré une valeur permettant d'effectuer un calcul correct de la somme des **n** premiers entiers.

Exercice 2

Rappel :

```
// Le problème suivant fait essentiellement appel à une suite de tests organisés de telle manière à ce
// que l'on puisse conclure : Le paquet a ou n'a pas une taille réglementaire.
// Cela est relativement compliqué à déterminer quand on fait la différence entre hauteur, largeur et
// profondeur. Cependant, notre gabarit ne fera pas la différence et il est souhaitable d'imaginer
// une astuce qui va faciliter la tâche : commencer par ordonner les trois valeurs..
// Objectif : apprendre à permuter les valeurs de deux variables en utilisant une variable temporaire
// et écrire des conditions de test de manière structurée.
// SI .. ALORS .. SINON
// Conseil pour éviter les erreurs : utiliser les accolades et indenter le code pour écrire des
// if (   ) {  ..  } else {  .. } , surtout lors de l'imbrication de ces structures.
```

Ecrire un programme nommé **Exo2**, qui permet à l'utilisateur d'entrer les trois dimensions d'un bagage et vérifie qu'il est éligible à l'embarquement dans la cabine d'un avion puisque sa taille n'excède pas 55 cm x 35 cm x 25 cm.

- Les dimensions sont des nombres à virgule.
- Le programme pose des questions jusqu'à ce qu'il ait obtenu trois valeurs comprises entre 1 et 150 cm
- Le programme affiche un résultat sous la forme : VALIDE / NON VALIDE
- Le programme propose de traiter le cas d'un autre colis sans devoir être relancé.

Exercice 3

Ecrire un programme qui propose à un joueur de deviner un nombre entier. La valeur est fournie par le générateur de nombres aléatoires (cf. fonction rand() et srand()).
Lorsque la valeur proposée est inférieure à la valeur à deviner, le programme répond "Trop petit" ou "Trop grand" dans le cas contraire. Le jeu s'arrête quand la valeur a été trouvée et le programme fournit un score qui correspond au nombre de propositions effectuées par le joueur.

Exercice 4

Ecrire un programme qui décompose un nombre en une suite de chiffres séparés par des tirets.

L'algorithme est basé sur une méthode numérique et non une méthode basée sur le traitement d'une chaîne de caractères.

L'entrée de la valeur 9245 produit l'affichage 9-2-4-5

Exercice 5

Ecrire un programme qui calcule le plus petit multiple commun – **ppcm**- de trois nombres entiers naturels non nuls.

Le **ppcm** est la plus petite valeur entière qui peut être divisée par chacun de ces nombres.

Le programme doit afficher la valeur du **ppcm** ainsi que le nombre d'opérations arithmétiques réalisées pour calculer le résultat. Comparer cette dernière valeur avec celle obtenue par d'autres étudiants.

Exemple d'affichage : Le ppcm de 16, 240 et 64 est égal à 960

Il y a eu 76 opérations