# Exercices chapitre 4 – Les fonctions

* Pour ce chapitre, à moins d’indications contraires, vous n’avez pas à faire la compréhension du problème et le jeu de tests.

## Exercice 1.0

Écrire une fonction qui n’a aucun paramètre et qui ne retourne rien. La fonction ne fera qu’afficher la phrase “Allo, je suis dans la fonction”.

## Exercice 1.1

## Écrire une fonction qui accepte un nombre représentant le total d’une épicerie et qui retourne la taxe sur ce total (taxe de 15%).

## Exercice 1.2

Reprendre l’exercice précédent mais cette fois-ci, la fonction accepte 2 paramètres (Le total et le taux de taxe). La fonction calcule la taxe et retourne le montant de taxe.

## Exercice 1.3 - Le compte à rebours

Programmez une **procédure (rappel, une procédure est un sous-programme qui ne retourne rien à la fonction appelante)** qui effectue un compte à rebours à partir d’un nombre passé en **paramètre** jusqu’à 0. Exceptionnellement, la procédure affiche elle-même les nombres à la console. Par exemple, un compte à rebours à partir de 10 affichera: 10 9 8 7 6 5 4 3 2 1 0.

## Exercice 1.4 - La somme d’une série

Programmez une **fonction** qui calcule la somme d’une série de nombres d’un minimum à un maximum inclusivement. Le minimum et le maximum doivent être passés en paramètres. Par exemple, si on appelle la fonction avec 1 et 5, la fonction doit retourner 15 (1+2+3+4+5). Testez votre fonction avec 1 et 100. Quelle valeur la fonction retournera-t-elle ?

## Exercice 1.5 - Les nombres premiers

Programmez une fonction qui détermine si un nombre inférieur à 100 est premier ou non. Sachez que les nombres premiers inférieurs à 100 sont 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89 et 97, mais qu'il serait une très mauvaise idée de faire une immense condition pour vérifier si le nombre est un de ceux-là! (Pensez à ce qui arriverait si on décidait de passer soudainement aux nombres premiers inférieurs à 1000!)

Ensuite, écrivez un programme principal qui demande à l’utilisateur un nombre entier entre 1 et 100 inclusivement et qui utilise la fonction pour déterminer si ce nombre est premier ou non. (Clarification: votre programme doit demander un nombre, tant et aussi longtemps que l'usager n'entre pas un nombre valide (entre 1 et 100).)

## 

## Exercice 1.6 - Afficher des lignes

Écrivez la procédure AfficherLigne qui affiche une ligne construite du nombre de “o” passé en paramètre, en utilisant une boucle. Par exemple, si on appelle AfficherLigne en lui passant 10, on obtiendra à l’écran:

oooooooooo

Si on l’appelle en lui passant 5, on obtiendra alors:

ooooo

**N.B**.: **Assurez-vous de ne pas faire de changement de ligne (WriteLine) dans la procédure AfficherLigne.**

Écrivez la procédure AfficherCarré qui affiche un carré du nombre de lignes passé en paramètre, en utilisant une boucle et un appel à la procédure AfficherLigne*.*

Par exemple,

AfficherCarré en lui passant 10: AfficherCarré en lui passant 5:

oooooooooo ooooo

oooooooooo ooooo

oooooooooo ooooo

oooooooooo ooooo

oooooooooo ooooo

oooooooooo

oooooooooo

oooooooooo

oooooooooo

oooooooooo

Testez vos fonctions avec un programme principal qui appelle simplement la procédure AfficherCarré.

## Exercice 1.7 - Afficher des o et des x

La première chose que vous devez faire est modifier votre procédure AfficherLigne du numéro 4.5. Vous devez y ajouter un paramètre de type char qui permet de spécifier le caractère à utiliser pour dessiner la ligne.

C'est quoi le type char? Un char est un type de donnée qui pourra contenir un seul caractère. Pour déclarer une variable de type char et y mettre le caractère x on pourra faire:

char unCaractère = 'x';

Notez que le symbole est entouré du symbole d'apostrophe, et non du symbole de guillemets comme nous avons vu pour les strings.

Une fois que ceci est fait, vous devez écrire le programme qui affiche le dessin suivant:

oxxxxx  
ooxxxx  
oooxxx  
ooooxx  
ooooox

Indice : Pour chaque ligne, vous aurez donc à utiliser deux fois la fonction AfficherLigne (une fois pour afficher un certain nombre de 'o' et une fois pour afficher un certain nombre de 'x')

## Exercice 1.8 - La facture

Écrire un programme qui permet de calculer le total d'une facture d'épicerie. Le programme doit commencer par afficher le menu suivant:

1. Ajouter un article non-taxable
2. Ajouter un article taxable
3. Afficher le total
4. Quitter

Ensuite le programme demande à l'usager de faire un choix.

Si le choix 1 est sélectionné, le programme demande le montant de l'article à l'usager (qui doit être entre 0 et 1000$ inclusivement, sinon on le redemande) et ajoute le montant au total.

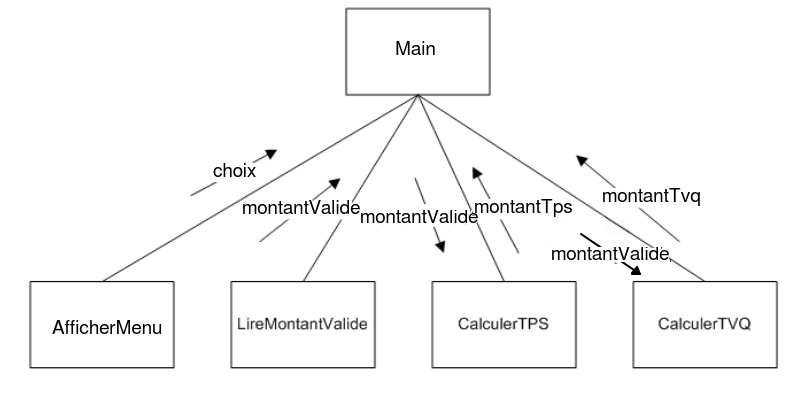
Si le choix 2 est sélectionné, le programme demande le montant de l'article à l'usager (qui doit être entre 0 et 1000$ inclusivement, sinon on le redemande), calcule les taxes (TPS=5% et TVQ=9,975%) et ajoute le montant et les taxes au total.

Si le choix 3 est sélectionné, le programme affiche le total actuel de la facture d'épicerie.

Si le choix 4 est sélectionné, le programme affiche "au revoir!" et se termine.

Après les choix 1, 2 et 3, le menu doit être réaffiché, et un autre choix doit être fait par l’usager. Le traitement sera recommencé tant que l’usager ne choisit pas de quitter le programme.

Votre programme doit obligatoirement suivre le diagramme hiérarchique suivant:



IMPORTANT : Réalisez cet exercice seulement avec une ou des boucles **do … while**.

## Exercice 1.9

Écrire un programme qui calcule la somme des nombres pairs de 2 à 32 inclusivement et la somme des nombres impairs de 33 à 65. Votre programme doit calculer la somme totale et l’afficher à la console. Saurez-vous y arriver avec une seule boucle?

## Exercice 1.10

Écrire une fonction qui calcule la somme des nombres d'un minium à un maximum inclusivement en incrémentant de 2 nombres à la fois. Par exemple, si on appelle la procédure avec 1 et 5, la fonction doit retourner (1+3+5=9). Si on appelle la procédure avec 2 et 5, la fonction doit retourner (2+4=6).

Finalement, votre programme principal doit faire usage de la fonction que l'on vient de créer pour calculer la somme des nombres pairs de 2 à 32 inclusivement et la somme des nombres impairs de 33 à 64. Votre programme doit calculer la somme totale et l’afficher à la console.

## Exercice 1.11

Écrire un programme qui demande un nombre entre 1 et 40 et qui affiche un motif. Par exemple, voici les motifs avec N=3, N=5 et N=10.

|  |  |  |
| --- | --- | --- |
| N=3 | N=5 | N=10 |
| o oo ooo | o oo ooo oooo ooooo | o  oo  ooo  oooo  ooooo  oooooo  ooooooo  oooooooo  ooooooooo  oooooooooo |

## Exercice 1.12

Un menu doit afficher 3 choix à l’usager:

1. Afficher “Bonjour”
2. Afficher “Hi”
3. Quitter le programme

Le programme doit attendre que l’usager entre un choix. Il doit recommencer la lecture du choix tant que le choix n’est pas valide. Le programme effectue le choix de l’usager et recommence tant que l’usager ne veut pas quitter le programme.

Réalisez cet exercice de deux façons, une fois avec des boucles while; et une fois avec des boucles do ... while.