TP 3 : Fonctions et programmation modulaire

Exercice 1 : Fonctions pgcd et ppcm

- Ecrire une fonction `pgcd` qui calcule et retourne le pgcd de deux entier p et q
- Ecrire une fonction `ppcm` qui calcule et retourne le ppcm de deux entier p et q

Exercice 2 : Fonction factorielle

1. Ecrire une fonction `factorielle` (non récursive) qui calcule $n! = n \times (n-1) \times \ldots \times 2 \times 1$, n étant un entier naturel avec par convention $0! = 1$. Cette fonction doit afficher un message d'erreur et arrêter l'exécution du programme si le nombre fournit en argument n'est pas un entier naturel valide ($n < 0$).
2. Ecrire un programme qui affiche la valeur $n!$ pour un entier naturel n saisi par l'utilisateur.

Exercice 3 : Fonction puissance

1. Écrire une fonction `puissance` qui calcule x^n où x est un réel et n un entier naturel. Cette fonction doit afficher un message d'erreur et arrêter l'exécution du programme si $n < 0$.
2. Écrire un programme qui affiche la valeur x^n pour x et n saisis par l'utilisateur.

Exercice 4 : Utilisation de fonctions de la bib standard

Implémenter et tester le programme de l'exercice 3 du TD sur la conversion de caractères.

- Ecrire une fonction `conversion_caractere` qui converti un caractère alphabétique (parmi 'a'...'z') minuscule (respectivement majuscule (parmi 'A'...'Z')) et retourne le caractère majuscule (respectivement minuscule) associé. Cette fonction doit afficher un message d'erreur et retourner le caractère inchangé si le caractère en argument

n'est pas un caractère alphabétique. En utilisera les fonctions de la bibliothèque standard `ctype.h`.
– Écrire un programme qui teste cette fonction sur un caractère saisi au clavier et affiche le caractère majuscule (respectivement minuscule) associé.

Exercice 5 : Programme modulaire et outil make

Implémenter et tester le programme de l'exercice 5 du TD sur les nombres premiers. Pour rappel il s'agit de

1. Écrire une fonction `est_premier` qui permet de tester si un nombre (entier naturel) est premier.
2. Écrire une fonction `premiers_premier` qui appliquée à un nombre entier positif n, affiche les n premiers nombres premiers calculés en utilisant la fonction `est_premier`.
3. crire un programme modulaire comprenant les modules `bib_premier.h`, `bib_premier.c` et `main.c` qui demande à l'utilisateur de saisir le nombre des premiers nombres premiers qu'il désire voir afficher et les affiche en appelant la fonction `premiers_premiers`. On pourra tester le programme en calculant les 10 premiers nombres premiers qui sont : 2, 3, 5, 7, 11, 13, 17, 19, 23 et 29.
4. Générer l'exécutable en effectuant un copilation séparée
5. Générer l'exécutable à parti d'un fichier makefile

1. Décomposer le programme de l'exercice 3 en trois fichiers : `bib_premier.h`, `bib_premier.c` et `main.c` et donnez le contenu de chaque fichier source /en-tête
2. donnez les commandes à lancer pour avoir le code éxécutable lorsqu'on effectue une compilation séparée des fichiers
3. Construire le fichier makefile du programme