

TP 4 - Modularité et fonction à pointeurs

Exercice 1 (Module Parfait)

1. Implanter le module `parfait` vu en TD.
2. Écrire alors un programme `testeParfait` qui en boucle demande à l'utilisateur d'entrer un entier parmi et selon sa valeur :
 - 0 : quitte le programme
 - 1 : saisit un entier et dit s'il est parfait
 - 2 : saisit deux entiers a et b et affiche le nombre d'entiers parfait de l'intervalle $[a, b]$
 - 3 : saisit un entier n et affiche le nième nombre parfait.
 - 4 : saisit deux entiers n et d , et affiche le plus proche parfait de n s'il en existe un dans l'intervalle $[n - d, n + d]$ ou un message d'absence sinon.
 - autre : ne fait rien

Exercice 2 Implanter le module `temps` vu en TD et réaliser un programme testant les différents fonctionnalités.

Exercice 3 (Module premier) On souhaite créer un module `premier` contenant les fonctions :

- `estPremier` qui teste si son paramètre (entier naturel) est premier.
- `prochainPremier` qui étant donné un entier naturel n , renvoie le plus petit nombre premier supérieur ou égal à n .

On pourra à l'occasion de cet exercice, écrire un fichier **Makefile** facilitant la fabrication des différents fichiers `:premier.o`, `test.o`, `test`, `trouvePremier.o`, `trouvePremier` (Cf. l'histoire du make Alban sur site du Moodle).

1. Écrire complètement le fichier d'en-tête du module.
2. Écrire une première version du fichier source du module ne contenant que la définition de `estPremier`. Écrire alors un programme `test` qui teste votre fonction pour les entiers 0, 1, 2, 3, 4, 7, 9.
3. Compléter alors le module avec la définition de `prochainPremier`.
4. Finalement écrire un second programme `trouvePremier` qui demande un entier à l'utilisateur et affiche le nombre premier immédiatement supérieur ou égal à l'entier saisi.
5. Testez votre programme pour les entiers 11 et 21.