Université Gaston Berger de Saint-Louis
UFR des Sciences Appliquées et de Technologie
Section Informatique

LICENCE SCIENCES & TECHNOLOGIES
UE ALGORITHMIQUE & PROGRAMMATION II - Fiche de TD/TP N° 1

Procédures et Fonctions

Si rien n'est précisé l'écriture des programmes et des sous-programmes se fera en pseudo-code (langage algorithmique)

Exercice 1 (TD)
Discuter du choix entre une fonction et une procédure puis écrire celles qui permettent :

- d'afficher le carré d'un entier donné en paramètre
- de retourner le maximum entre deux entiers donnés en paramètre
- de déterminer si un entier donné en paramètre est pair ou pas
- de savoir si un entier est divisible par un autre
- d'afficher tous les diviseurs d'un entier donné en paramètre

Exercice 2 (TD+TP)
Ecrire une procédure qui reçoit cinq paramètres et qui met dans le troisième argument le résultat de la division du premier par le deuxième et dans le quatrième argument le résultat de la division du deuxième par le premier. Le cinquième argument sera un booléen qui signifiera que toutes les divisions ont été possibles ou non. Donner une implémentation en C de la procédure précédente.

Exercice 3 (TD+TP)
Écrire l'algorithme d'un sous-programme qui calcule la somme des n premiers entiers.
Rappel : $1 + 2 + 3 + \ldots + n = n(n+1) / 2$

Exercice 4 (TD+TP)
Ecrire une fonction C qui affiche le chiffre le plus à gauche d'un nombre entier positif pris en paramètre et qui renvoie le reste de ce nombre. Par exemple, pour 456, la fonction affiche 4 et renvoie 56.
Ecrire un programme C qui utilise cette fonction pour afficher de gauche à droite les chiffres d'un entier saisi au clavier.

Exercice 5 (TD+TP)
Un nombre parfait est un nombre naturel n non nul qui est égal à la somme de ses diviseurs stricts (n exclus).
Exemple : $6 = 1 + 2 + 3$
a. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un nombre parfait, faux sinon.
b. Écrire en langage algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.
c. Donner une implementation en C des procedures et/ou fonctions.

Exercice 6 (TD+TP)

1. Écrire **en un minimum de lignes** l'algorithme d'un sous-programme permettant de dessiner le motif ci-dessous. Le nombre de motifs et la longueur du motif seront passés en paramètres du sous-programme. Dans cet exemple, le motif de base est répété 3 fois et la base d'un triangle est de longueur 4.
2. Donner la traduction du programme précédent en C.

Exercice 7 (TP)

Ecrire la fonction en en C *transfo_en_maj (minus, maj)* qui reçoit une chaîne de caractères minuscules (par le paramètre *minus*), et la transforme (dans le paramètre *maj*) en chaîne de caractères majuscules. On pourra utiliser la fonction prédéfinie de la librairie <ctype.h> toupper (*char toupper(c:char)),* qui reçoit en argument un caractère et retourne la majuscule correspondante. La fonction miroir est *char tolower(c:char)).*

Donner un exemple d'utilisation de la fonction *transfo_en_maj* à l'aide d'un petit programme en C.

Exercice 8 (TD+TP)
a) Ecrire une procédure permettant de déterminer si un nombre entier est premier. Elle comportera deux arguments : le nombre à examiner et un indicateur booléen (0 ou 1) précisant si ce nombre est premier ou non.
b) Ecrire la procédure précédente sous forme d'une fonction.
c) Donner une traduction des fonctions ainsi écrites en C.

Exercice 9 (TD+TP)
Ecrire un programme en langage C qui permet de simuler le jeu du nombre caché.
Dans ce jeu, l'ordinateur choisit un nombre (H) au hasard entre 1 et 1000 ; il faut ensuite, pour le joueur, le deviner en dix questions au plus. A chaque fois que le joueur donne une réponse (R), l'ordinateur répond par :

- « trop petit » si le nombre donné (R) est plus petit que celui à deviner (H)
- ou « trop grand » si le nombre donné (R) est plus grand que celui à deviner (H)
- ou « gagné » si c'est le nombre exact (c'est-à-dire H = R).

On utilisera la fonction ***rand()*** pour générer le nombre **H** de l'ordinateur (***rand()%1000*** donne un entier compris entre 0 et 999). Faites des recherches pour voir l'utilité de l'instruction ***srand(time(null))*** ;

Exercice 10 (TD)
On souhaite écrire un algorithme qui permet d'envoyer des SMS à partir d'un téléphone portable de marque ***SanarMobile***. Les téléphones ***SanarMobile*** disposent d'une bibliothèque de fonctions préenregistrées utilisables directement par les opérateurs téléphoniques. L'une de ces fonctions
yoneeBataaxal (message : chaine de caracteres, numeroDestinataire : tableau d'entiers) : booléen
permet d'envoyer un message à un numéro donné ; elle renvoie *true* si le message est bien envoyé et *false* sinon.
En se servant de cette fonction, écrire un algorithme qui permet de saisir un message et un numéro de téléphone, envoie le message via un téléphone ***SanarMobile*** et informe l'utilisateur sur la suite de son message *(« envoyé »* ou *« non envoyé »*).

Exercice 11 (TD+TP)
Ecrire une fonction C ***lirePharse (char maPhrase[])*** qui reçoit une phrase (contenue dans la chaîne de caractères *maPhrase*) et affiche la longueur de la phrase, le nombre de signes de ponctuation et le nombre de chiffres contenus dans la phrase.
On se servira des deux fonctions suivantes :

- *boolean* ***JeSuisUnSigneDePonctuation****(char c)* qui renvoie *true* si le paramètre reçu *c'*est un signe de ponctuation et *false* sinon.
- *boolean* ***JeSuisUnChiffre****(char c)* qui renvoie *true* si le paramètre reçu *c'*est un chiffre et *false* sinon. (On pourra définir le type booléen en langage C ou utiliser le type int)