TP 5

Procédures avec paramètres (en entrée, en sortie, en entrée/sortie)

Exercice 1

Proposez et testez une procédure qui admet deux nombres entiers **`a`** et **`b`** en paramètres d'entrée et qui rend en paramètres de sortie le **quotient** et le **reste** de la division de **`a`** par **`b`**.

Exercice 2

Proposez et testez une procédure qui demande à l'utilisateur une suite de nombres entiers positifs, terminée par un zéro (qui constitue le marqueur), et qui rend en paramètres de sortie le plus petit et le plus grand d'entre eux. La procédure rend en paramètre de sortie -1 (pour les deux résultats) si la suite est vide, à charge pour le programme appelant d'afficher un message ad hoc.

Exercice 3

Le salaire net d'un employé se calcule en retirant taux% de charges de son salaire brut (cf TP3 exercice 5) ; par exemple taux =20.

1) écrivez et testez une procédure `traiterUnEmploye` ayant en paramètres d'entrée le nom et le salaire brut d'un employé, et en paramètre d'entrée/sortie le total des salaires nets des employés déjà traités.
La procédure calcule et affiche le salaire net de l'employé, précédé de son nom, et ajoute ce salaire net au total.

2) écrivez et testez une procédure `afficherMoyenne` ayant en paramètres d'entrée le total des salaires nets et le nombre de salariés traités.
La procédure calcule et affiche le salaire moyen ; attention à la division par zéro.

Écrivez et testez un programme, utilisant les procédures précédentes, qui calcule et affiche le salaire net en fonction du salaire brut pour autant de personnes qu'on le souhaite, puis qui affiche à la fin le salaire net moyen.
L'utilisateur saisit pour chaque personne le nom et le salaire brut (nom puis entrée, salaire brut puis entrée), et tape * à la place du nom pour indiquer la fin des données.

N.B. On considère qu'un nom saisi au clavier ne comporte pas d'espaces et est limité à 19 caractères pour l'utilisateur. On envisage le cas de la liste vide (cas où l'utilisateur tape '*' dès le départ).

Exercice 4 : jeu de dé

Dans cet exercice, le programme ne « tire » pas le dé lui-même ; il se contente de cumuler les points en fonction des données (valeur du dé) que le joueur communique au programme.

règle n°1 :

Deux joueurs A et B lancent le dé alternativement ; le premier qui atteint 50 points (50 est un exemple) a gagné (même si l'autre joueur a joué un coup de moins).

1) écrivez une procédure :
`faireJouer(entF j:caractère, entF/sortF score:entier)`
qui fait jouer le joueur j ; pour cela, elle demande au joueur j la valeur du dé qu'il vient de lancer, et ajoute cette valeur à son score score .

2) écrivez une procédure :
`afficherResultat(entF score1:entier, entF score2:entier)` qui affiche les scores respectifs score1 et score2 des 2 joueurs et le vainqueur (A ou B).

3) écrivez un programme, utilisant les procédures spécifiées ci-dessus, qui demande alternativement à chaque joueur de jouer, puis la valeur du dé que ce joueur a tiré, et affiche en fin de partie le joueur gagnant, son score, et le score du perdant.

Exercice 5 : jeu de dé n° 2

Règle n°2 : à la règle du jeu n°1 s'ajoute le fait que tirer un 6 donne le droit de rejouer (attention, en rejouant on peut encore tirer un 6).

1) Modifiez la procédure :
`faireJouer(ent j : caractère, ent/sort score:entier)`
et testez-la.

2) Constatez que le programme principal ne change pas (ni les autres procédures).
Testez le tout.

Exercice 6

Reprenons les exercices du jeu de dé :

1) Afin d'améliorer l'interface de ces programmes, écrivez et testez une procédure `unDeValide(sort valeur : entier)` qui fournit en paramètre de sortie une valeur comprise entre 1 et 6, donc qui redemande à l'utilisateur une valeur autant de fois que nécessaire pour satisfaire cette exigence.

2) Introduire l'utilisation de cette procédure dans les programmes des exercices 4 et 5.

Exercice complémentaire

Exercice 7

Proposez et testez une procédure qui demande à l'utilisateur un nom de figure (rectangle, carré ou cercle) et demande ensuite les informations nécessaires pour calculer et rend en paramètre de sortie l'aire d'une telle figure (ex. : la longueur et la largeur pour un rectangle, la longueur d'un côté pour un carré, le rayon pour un cercle).

NB : la procédure fournira la valeur -1 en paramètre de sortie si la figure saisie n'est pas l'une des trois prévues.

Aide : Vous utiliserez la fonction `strcmp` pour identifier la figure.

Pour cela : `#include <string.h>`

`strcmp(chaine1, chaine2)` vaut soit 0, soit un entier négatif, soit un entier positif, selon que chaine1 est, respectivement, égale, inférieure, supérieure, à chaine2.