LIFAPI – TD 9 bis : Sujets TP notés

Objectifs : Corriger les 6 sujets de TP notés #1

Séquence 1 sujet A

1. Écrire en C/C++ une **procédure** `saisie_intervalle` qui demande à l'utilisateur de choisir un entier compris entre -100 et 100 inclus **et** "retourne" la valeur saisie. On recommencera la saisie tant que la valeur n'est pas dans l'intervalle [-100;100].
2. Écrire en C/C++ le programme principal qui teste le sous-programme précédent.
3. Écrire en C/C++ un sous-programme `echange` qui **échange** le contenu de deux entiers s'ils sont de même signe, sinon il place la somme dans le premier entier et le produit dans le second.
   - Exemple 1 : si x = 6 et y = 7 avant l'exécution de `echange`, ils vaudront x = 7 et y = 6 après.
   - Exemple 2 : si x = -6 et y = 7 avant l'exécution de `echange`, ils vaudront x = 1 (-6+7) et y = -42 (-6*7) après.
4. Modifier le programme principal pour qu'il demande deux entiers dans [-100;100] à l'utilisateur (on utilisera obligatoirement le sous-programme écrit en 1-) et affiche les nouvelles valeurs après exécution de `echange`.

Séquence 1 sujet B

1. Écrire en C/C++ une **fonction** `saisie_non_nulle` qui demande à l'utilisateur de choisir un **réel** non nul **et** retourne la valeur saisie. On recommencera la saisie tant que la valeur est nulle.
2. Écrire en C/C++ le programme principal qui teste le sous-programme précédent.
3. Écrire en C/C++ un sous-programme `permute` qui **échange** le contenu de deux réels s'ils sont de signe différent, sinon il place la valeur absolue de la différence dans le premier réel et le résultat de la division dans le second.
   Exemple 1 => si x = -6.5 et y = 7.2 avant l'exécution de `permute`, ils vaudront x = 7.2 et y = -6.5 après.
   Exemple 2 => si x = 6.5 et y = 7.2 avant l'exécution de `permute`, ils vaudront x = 0.7 (abs(6.5-7.2)) et y = 0.90 (6.5/7.2) après.
4. Modifier le programme principal pour qu'il demande deux réels non nuls à l'utilisateur (on utilisera obligatoirement le sous-programme écrit en 1-) et affiche les nouvelles valeurs après exécution de `permute`.

Séquence 3 sujet A

1. Écrire en C/C++ une **fonction** `alea_bornee` qui retourne une valeur entière tirée aléatoirement entre 10 et 30 inclus.
2. Écrire en C/C++ le programme principal qui teste le sous-programme précédent.
3. Écrire en C/C++ un sous-programme `petit_grand` qui effectue 20 tirages aléatoires de valeurs comprises entre 10 et 30, compte et "retourne" le nombre de valeurs supérieures ou égales à 20 **et** le nombre de valeurs strictement inférieures à 20.
4. Modifier le programme principal pour qu'il affiche le nombre de valeurs supérieures ou égales à 20 et le nombre de valeurs strictement inférieures à 20 après 20 tirages aléatoires.

Séquence 3 sujet B

1. Écrire en C/C++ une **procédure** `alea_bornee` qui "retourne" une valeur entière tirée aléatoirement entre 20 et 40 inclus.
2. Écrire en C/C++ le programme principal qui teste le sous-programme précédent.
3. Écrire en C/C++ un sous-programme `pairs_impairs` qui effectue 20 tirages aléatoires de valeurs comprises entre 20 et 40, compte et "retourne" le nombre de valeurs paires **et** le nombre de valeurs impaires.
4. Modifier le programme principal pour qu'il affiche le nombre de valeurs paires et le nombre de valeurs impaires parmi les 20 tirages aléatoires effectués.

Séquence 5 sujet A

1. Écrire en C/C++ une **fonction** `saisie_positive` qui demande à l'utilisateur de choisir un entier strictement positif **et** retourne la valeur saisie. On recommencera la saisie tant que la valeur n'est pas strictement positive.
2. Écrire en C/C++ le programme principal qui teste le sous-programme précédent.
3. Écrire en C/C++ un sous-programme `chiffres_pairs_impairs` qui compte et "retourne" le nombre de chiffres pairs et le nombres de chiffres impairs qui composent le nombre **n** passé en paramètre.
    1. Exemple 1 : si n = 12345678, `nb_chiffres_pairs` = 4 (2,4,6,8) et `nb_chiffres_impairs` = 4 (1,3,5,7)
    2. Exemple 2 : si n = 135791359, `nb_chiffres_pairs` = 0 et `nb_chiffres_impairs` = 9 (1,3,5,7,9,1,3,5,9)
4. Modifier le programme principal pour qu'il demande aussi un entier strictement positif à l'utilisateur (on utilisera la question 1 obligatoirement) et affiche le nombre de chiffres pairs et de chiffres impairs qui composent ce nombre.

Séquence 5 sujet B

1. Écrire en C/C++ une **procédure** `factorielle` qui calcule **et** "retourne" la valeur de la factorielle d'un entier n passé en paramètre. On supposera que la valeur n est strictement positive. **Attention, on ne veut pas une fonction factorielle !**
2. Écrire en C/C++ le programme principal qui affiche la factorielle d'un entier compris entre 1 et 12 saisi par l'utilisateur (on ne vérifiera pas qu'il est bien dans l'intervalle).
3. Écrire en C/C++ une **fonction** `somme_chiffres_pairs` qui calcule et retourne la somme des chiffres pairs qui composent le nombre n passé en paramètre.
    1. Exemple 1 : si n = 12345678, `somme_chiffres_pairs` = 20 (2+4+6+8)
    2. Exemple 2 : si n = 50400, `somme_chiffres_pairs` = 4
4. Compléter le programme principal pour qu'il affiche la somme des chiffres pairs de la factorielle affichée à la question 2.