III. Principes de programmation - Corrigé

1. Switch – case structure : Conversion d'unité d'énergie

Exemple de réalisation :

```
% Demande la valeur d'énergie, son unité et la nouvelle unité
V_in = input('quantité d''énergie à convertir: ');
Ein_unit = input('entrer son unité(J, ft-lb, cal ou eV): ','s');
Eout_unit = input('unité désirée(J, ft-lb, cal ou eV): ','s');

% Utilise un switch pour choisir entre les unités d'entrée.
% Pour les 4 unités d'entrée, convertir premièrement le V_in en J
switch Ein_unit
    case 'J'
        E_Joule = V_in;
    case 'ft-lb'
        E_Joule = V_in/0.738;
    case 'cal'
        E_Joule = V_in/0.239;
    case 'eV'
        E_Joule = V_in/(6.24*10^18);
    otherwise
        disp('L''unité d''entrée n''est pas correcte!')
        return
end

% Utilise un switch pour choisir entre les unités de sortie.
% Pour chaque unité de sortie, convertir le résultat obtenu
% précédemment (en Joules) dans l'unité finale désirée.
switch Eout_unit
    case 'J'
        E_new = E_Joule;
    case 'ft-lb'
        E_new = E_Joule*0.738;
    case 'cal'
        E_new = E_Joule*0.239;
    case 'eV'
        E_new = E_Joule*6.24e18;
    otherwise
        disp('L''unité de sortie n''est pas correcte!')
        return
end

fprintf('%g %s = %g %s\n', V_in, Ein_unit, E_new, Eout_unit)
```

Réponses :

325 J = 239.85 ft-lb // 432 cal = 1807.53 J // 6.8 eV = 2.60449e-019 cal

2. Structure en boucle : la somme d'une série

- Calcul de la somme des n termes de la série: $\sum_{k=1}^{n} \frac{(-1)^k k}{2^k}$.

Exemple de réalisation :

```
n = input('nombre de termes n ? ');

% initialiser la somme
S = 0;

% calcul de la somme dans la boucle
for k=1:n
    S = S + (-1)^k*k/2^k;
end

fprintf('la somme des %g termes de la série sont: %f \n',n, S)
```

Réponses :

la somme des 4 termes de la série sont: -0.125000

la somme des 20 termes de la série sont: -0.222216

la somme des 100 termes de la série sont: -0.222222

la somme des 1e+006 termes de la série sont: -0.222222

- La fonction cos(x) peut être écrite en une série de Taylor : cos (x) = $\sum_{k=0}^{\infty} \frac{(-1)^k x^{2k}}{(2k)!}$.

```
x = input('l''angle (en degrés)');
n = input('nombre de termes de la série de Taylor ');

% transforme l'angle x en radian
xr = x*pi/180;

% initialiser la somme à 0
S = 0;

for k=0:n
    S = S + (-1)^k * xr^(2*k) / factorial(2*k);
end

fprintf('notre calcul donne (pour %g°, n=%g)= %f\n', x, n, S)

% comparaison avec la fonction cos(x)
cosX = cos(xr);

fprintf('la fonction cos de %g° = %g\n', x, cosX)
```

Réponses :

notre calcul donne (pour 210°, n=3)= -1.564583

la fonction cos de 210° = -0.866025

notre calcul donne (pour 210°, n=8)= -0.866023

la fonction cos de 210° = -0.866025

- La fonction f(x) = e^x peut être écrite dans une série de Taylor : $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

```
% Exercice 3.2.3 fonction approximée par une série de Taylor

x = input('entrer la valeur x de la fonction exp(x) ');

% définir les termes de la série de Taylor:
% exp(x) = 1 + x + x^2/2! + x^3/3!

% l'increment n, c-à-d le nombre de termes, lequel commence à 1
n = 1;

% la somme expX commence à 1 pour n=0
expX = 1;

while n==1 || n<=20 && abs(n_term)>=0.0001
    n_term = x^n/factorial(n);
    expX = expX+n_term;
    n = n+1;
end

if n>=20
    disp('Nombre d''itérations limites dépassé!')
else
    fprintf('exp(%g) avec %g termes est de %f\n', x, n, expX)
end
```

Réponse :

exp(3) avec 15 termes est de 20.085523

exp(-4) avec 18 termes est de 0.018307

pour exp(18) : Nombre d'itérations limites dépassé!