TD4

Des boucles… Des boucles… Des boucles… Des boucles… Des boucles… Des boucles…

Un programme sans boucle et sans structure de donnée
ne vaut pas la peine d'être écrit.
Alan Jay Perlis (Epigrams on Programming, 1982)

I Boucles « pour » simples

I.1. En ligne

On considère l'algorithme à trou suivant :

```
Algorithme dessinLigne
Variables n, i : entiers
Début
  Afficher("Donner le nombre de symboles sur la ligne : ")
  Saisir(n)
  {instructions permettant de dessiner la ligne}
Fin
```

a) Donner les instructions pour obtenir une ligne de n étoiles ; pour n = 8, on obtient :
****

b) Même question pour obtenir une ligne de n symboles alternant des + et des –
Si n= 7, on obtient : +–+–+–+
Si n = 10, on obtient : +–+–+–+–+–

II Fonctions simples et boucles pour

a) Ecrivez une ***fonction*** *multiplication* qui prend deux entiers naturels (≥ 0) m et n en paramètres et renvoie m * n, après avoir vérifié que m et n sont positifs ou nuls, ***en utilisant uniquement des additions***. La fonction renverra -1 si m ou n est négatif.

Pour rappel, une fonction s'écrira de la manière suivante :

```
1  Fonction toto(x) retourne réel
2  paramètre x : réel
3  début
4    retourne (x-1)*(x-2)
5  fin
```

b) Ecrivez une ***fonction*** *factorielle* qui prend un entier naturel n en paramètre et renvoie n!, ou -1 si l'entier saisi est négatif. Par définition : $n! = \prod_{i=1}^{n} i = 1 * 2 * 3 * \ldots * (n-1) * n$ et $0! = 1$.

c) Ecrivez une ***fonction*** *puissance* qui saisit deux entiers relatifs m et n et renvoie m^n. On notera que :

- si n vaut 0, m^n vaut 1 quelle que soit la valeur de m
- si n est négatif, $m^n = 1 / m^{|n|}$

III A la volée

Pour chacune des questions suivantes, on souhaite écrire une procédure qui saisisse un ensemble *n* de notes sur 20 (n passé en paramètre de la procédure) et établir au fur et à mesure un certain nombre de statistiques (moyenne, min/max, etc.), affichées à la fin de la procédure.

a) lire *n* notes et afficher le nombre de notes qui sont au-dessus de la moyenne

b) lire *n* notes et afficher la moyenne des notes lues

c) lire *n* notes et afficher la plus grande d'entre elles

IV Doubles boucles

a) Donnez la procédure qui prend un entier *n* en paramètre et dessine la figure suivante (si n = 5) :
*****
*****
*****
*****
*****
Note : un retour à la ligne s'obtient avec l'instruction Afficher("\n")

b) Même question pour la figure suivante (si n = 5) :
*
**
***
****
*****

c) Même question pour la figure suivante (si n = 5) :
*****
****
***
**
*

d) Même question pour la figure suivante (si n = 5) :
+****
++***
+++**
++++*
+++++

V Pour aller plus loin : représentation binaire

Ecrivez une procédure prenant en paramètre un entier n compris entre 0 et 255 et qui affiche la représentation binaire sur 8 bits de cet entier.

Eléments mathématiques
Un entier naturel n, codé sur 8 bits, peut s'écrire sous forme binaire de la manière suivante :
$n = a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0$, où les différents indices correspondent à 0 ou à 1,
et sa valeur décimale est obtenue par $a_7.2^7 + a_6.2^6 + a_5.2^5 + a_4.2^4 + a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0$
On peut remarquer que $2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 2^7 - 1$
donc que $a_6.2^6 + a_5.2^5 + a_4.2^4 + a_3.2^3 + a_2.2^2 + a_1.2^1 + a_0.2^0 < 2^7$
On peut en déduire que si $n >= 2^7$, a_7 vaut nécessairement 1 et qu'inversement si $n < 2^7$, a_7 vaut 0. On peut alors reproduire le raisonnement sur $n - a_7.2^7$ pour trouver a_6, et ainsi de suite pour tous les chiffres.

VI Boucles « tant que » simples

VI.1. Encore des maths simples

a) Dans l'exercice II, on avait écrit le calcul de $m * n$ en utilisant une boucle ***pour***. Réécrivez la fonction en utilisant une boucle tant que

b) Dans l'exercice III.b, on calculait la moyenne de *n* notes saisies par l'utilisateur. Transformez l'algorithme pour que l'utilisateur saisisse autant de notes qu'il souhaite, ou la valeur -1 s'il souhaite arrêter la saisie et déclencher la calcul de la moyenne.

VI.2. Racine entièrement carrée

La racine carrée entière d'un entier n est le plus grand entier i tel que $i^2 \leq n$. Ecrivez une ***fonction*** qui prend un entier *n* en paramètre et calcule et renvoie sa racine carrée entière.

VI.3. C'est au menu !

a) Ecrivez une **procédure** *menu* qui affiche le menu suivant :

```
1. calcul de n!
2. calcul de m^n
3. calcul de m*n
4. calcul de la racine entière de n
5. quitter
Votre choix ?
```

b) Ecrivez un algorithme qui affiche le menu, saisit le choix de l'utilisateur puis saisit les paramètres nécessaires et appelle la bonne fonction selon le choix effectué, tant que ce choix est différent de 5.

VII Plus ou moins…

On souhaite programmer un jeu dans lequel un joueur doit deviner un nombre mystère choisi par l'autre joueur dans l'intervalle [0..1000]. Celui qui doit trouver le nombre fait des propositions, auxquelles l'autre joueur répond par « C'est plus ! » quand le nombre à trouver est plus grand que le nombre proposé, ou « C'est moins ! » quand le nombre à trouver est plus petit.

a) Ecrivez l'algorithme dans lequel c'est l'ordinateur qui choisit le nombre mystère et l'utilisateur qui fait des propositions. L'algorithme affichera « Félicitations, vous avez trouvé en xxx coups !!! » (où xxx doit être remplacé par le nombre d'essais) quand l'utilisateur trouve le nombre mystère.

```
Quel est le nombre ? 500
C'est plus !

Quel est le nombre ? 750
C'est moins !

Quel est le nombre ? 600
Félicitations, vous avez trouvé en 3 coups !!!
```

b) Ecrivez l'algorithme dans lequel c'est l'ordinateur qui doit trouver le nombre choisi par l'utilisateur en lui faisant des propositions. L'utilisateur répond '+', '-' ou '='

Pour aller plus loin (à faire chez vous pour vous entraîner)
En vous inspirant du b), écrivez un programme qui cherche la solution de *f(x)*=0 dans [a, b] où *f* est une fonction continue quelconque et ne s'annule qu'une seule fois sur [a,b]. Vous utiliserez la méthode par dichotomie (voir encadré) pour trouver la solution de *f(x)* = 0.

- on pourra prendre f(x) = (x-1)*(x-2) (cf la fonction toto du II.a) et chercher la solution dans l'intervalle [1.5, 100]

Rappel
La méthode par dichotomie consiste à réduire l'intervalle de recherche en le divisant par deux à chaque itération. Comme la fonction ne s'annule qu'une seule fois sur [a, b], cela veut dire que f(a) et f(b) n'ont pas le même signe. Si on regarde le signe de f(c) avec c = (a+b)/2, on remarque que:
- si *f(a)* et *f(c)* ont le même signe, alors la solution est dans [c, b]
- si *f(a)* et *f(c)* n'ont pas le même signe, alors la solution est dans [a, c]
- tant que |*f(c)*| > epsilon, on divise à nouveau l'intervalle en deux

VIII Les premiers des premiers…

Un **nombre premier** est un entier naturel qui admet exactement deux diviseurs distincts entiers et positifs (qui sont alors 1 et lui-même). [Wikipedia]

a) Ecrivez une ***fonction*** qui prend un entier *n* en argument et détermine s'il est premier.
   - Note 1 : Un entier *a* est divisible par un entier *b* si et seulement si $a\ mod\ b = 0$
   - Note 2 : Il est inutile de tester la divisibilité par 1…
   - Note 2 : Si on n'a trouvé aucun diviseur entre 2 et la racine carrée entière de *n* (voir exercice VI.2), inutile de chercher plus loin

b) Ecrivez une ***procédure*** qui affiche la liste des n premiers nombres premiers