TD n°1

1.Rappels

A) Structure d'un programme

```
program {nom du programme};
uses {quelque_chose qui sert a pouvoir utiliser certaines instructions};
var {nom des variables utilisées dans le programme et leur type};
BEGIN
        {toutes les instructions du programme}
END.
```

- Attention aux ";" après les instructions et au "." aprés le "END".
- On peut mettre un commentaire dans un programme (quelque chose qui ne sera pas vu du compilateur) en l'écrivant entr'accolades, comme : {ceci}.
- Pour affecter 3 à l'entier "a", il faut écrire "a:=3", apres avoir déclaré a.

B) Instructions de lecture et d'écriture à l'écran

- `Write('La valeur de a est : ', a, '. Pour b, c''est : ',b);`
  Affiche les caractères entre apostrophes (Si on a besoin d'une apostrophe dans le texte, il faut en mettre deux) et la valeur des variables sans apostrophe dans l'ordre indiqué. Après l'affichage, le curseur reste en fin de ligne
- `Writeln('comme avant');`
  Fait la même chose que Write, mais le curseur retourne à la ligne.
- `Read(a,b);`
  Lit les valeurs (séparées par un espace) entrées à l'écran et les affecte aux variables entre parenthèses.
- `Readln(a);`
  Pareil, sauf que le curseur va à la ligne suivante.
- `Readln;`
  Permet de laisser l'écran jusqu'à ce que l'utilisateur appuie sur une touche.

Exercice 1 (échauffement)

Faire un programme qui affiche à l'écran la somme de deux entiers entrés par l'utilisateur.

2.Boucle for... :=... to... do...

A) Structure de la boucle

On peut avoir besoin d'exécuter un grand nombre de fois la même instruction. Pour cela, on utilise une variable entière, par exemple "i", déclarée au début du programme dans les variables, qui s'incrémente de 1 à chaque parcours du bloc d'instruction, d'un nombre entier prédéfini "m" à un second "n", grâce à l'une des boucles suivantes :

`for i:=m to n do` *{une seule instruction}*;

ou

```
for i:=m to n do
```

```
      begin
      {plusieurs instructions};
      end;
```

B) Exemple

Le programme suivant affiche les nombres entier de 1 à 20.

```
program j_apprends_a_compter;
const n=20; {n est déclarée en constante : sa valeur ne peut pas être modifiée dans le programme}
var
      i:integer;
BEGIN
      for i:=1 to n do
            writeln(i);
      readln;
END.
```

Exercice 2

Faire un programme qui demande un entier "n", et qui affiche la somme les n premiers entiers.

Exercice 3

Faire un programme qui demande un entier "n", et qui renvoie (affiche) n!.

Exercice 4

Faire un programme qui demande un entier "n", et qui renvoie $\sum_{i=1}^{n} i!$.

Exercice 5

Faire un programme qui demande un entier "n", un reel "x", qui renvoie $\sum_{i=0}^{n} \frac{x^i}{i!}$, et qui affiche exp(x).

Tester ce programme pour n=10 et x prenant différentes valeurs.

C) Boucles multiples :

On peut avoir besoin de plusieurs boucles imbriquées les unes dans les autres :

```
for i:=m to n do
      begin
      for j:=k to l do
            begin
            {intructions};
            end;
      end;
```

Exercice 6

Soient les matrices carrées $n*n$: A et B, définies par $A_{i,j}=i+j$ et $B_{i,j}=i*j$

Ecrire un programme qui affiche les coefficients de AB.

D) Suites définies par récurrence

Exercice 7

Calculer u(n) pour
u(0)=1
u(n+1)=cos(u(n))
différents n.
Remarque ?

Exercice 8

Calculer u(103) pour
u(1)=1
u(n)=cos(u(n-1))+n

Exercice 9

La vie des lapins se compose de deux périodes. Pendant leurs deux premiers mois, ils mangent beaucoup, mais ne font pas grand chose. Puis, à partir de leur troisième mois, il se reproduisent, avec une moyenne de un rejeton par mois et par lapin. On se place dans un monde idéal où les lapins ne meurent jamais, et ou chaque couple de lapins matures donne naissance chaque mois à un couple de lapins. On commence avec un couple de nouveaux nés au mois numéroté 1. On cherche le nombre total de couples lors du nième mois : u(n).

Calculer les premiers termes de la suite, trouver la relation de récurrence définissant u(n), puis l'implémenter.

Exercice 10

Calculer u(100) et v(100) pour
u(0)=1/5
v(0)=2/3
u(n+1)=4/7*(v(n)-u(n))
v(n+1)=u(n)+v(n)/5

3.Boucle for... :=... downto... do...

Si l'on veut faire une boucle sur un entier qui décroît de 1 à chaque parcours de la boucle, on utilise « `downto` » au lieu de « `to` ». Par exemple :

```
for i:=m downto 1 do
    {intructions};
```

est une boucle qui va commencer à prendre i=m, puis i=m-1, puis i=m-2... jusqu'à i=1.