TD n°2

1.If... then... & If... then... else...

if=si, then=alors et then=sinon...

Ces boucles servent à effectuer ou non des instructions selon la valeur (vrai ou faux) d'une condition.

A) Structure

`if` *{condition}* `then`

{instructions effectuées si la condition est remplie}

`else`

{instructions effectuées si la condition n'est pas remplie};

Les instructions peuvent être une instruction unique ou un bloc de plusieurs instructions. Dans le dernier cas, le bloc doit être encadré de « `begin` » et « `end;` ». Attention : il n'y a pas de « `;` » avant le « `else` ».

Opérateurs logiques :
AND : et , OR : ou, XOR : ou exclusif , NOT : non

Opérateurs relationnels :
<, >, =, <= :inférieur ou égal, >=, <> : différent de

B) Exemple

Le programme suivant retourne le plus petit de deux entiers entrés par l'utilisateur.

```
program exemple_if;
var a,b:integer;
BEGIN
write('a = ');
readln(a);
write('b = ');
readln(b);
if a<b then writeln('Le plus petit des deux est a = ',a)
       else
           begin
                if a=b then writeln('a = b = ',a)
                       else writeln('Le plus petit des deux est b = ',b);
           end;
readln;
END.
```

Exercice 1

Faire un programme qui résoud une équation du type $a*x+b=0$. Considérer tous les cas possibles (par exemple, $a=0$) !

2.Repeat... until & While... do

A) Utilité

Ces deux boucles sont le contraire l'une de l'autre. En connaître une seule suffit pour s'en sortir, mais les connaître toutes les deux simplifie les choses.

La première répète des instructions jusqu'à ce qu'une condition soit vérifiée, et l'autre les répète tant qu'une condition est vraie :

```
repeat {instructions}
       until {condition};
```

```
while {condition} do
      {instructions};
```

Comme toujours, des instructions peuvent être une instruction unique, ou un bloc de plusieurs instructions encadrées par « begin » et « end ». Cependant, dans le cas de « repeat », l'encadrement n'est pas obligatoire, car les mots « repeat » et « until » jouent ce rôle.

B) Exemple

Les deux programmes suivants font exactement la même chose : ils affichent la première valeur entière n telle que le nième terme de la suite

$u(0)=x$

$u(n+1)=u(n)^2$

soit strictement supérieur à 100. Mais le deuxième est plus simple : pas besoin de boucle if. Selon les cas, l'une ou l'autre des deux boucles est à privilégier.

```
program exemple_repeat;
var n:integer;
    x:real;
BEGIN
    write('x=');
    readln(x);
    n:=0;
    if x>100 then writeln(n)
       else
         begin
         repeat
             begin
             n:=n+1;
             x:=sqr(x);
             end
         until x>100;
         writeln(n);
         end;
    readln;
END.
```

```
program exemple_while;
var n:integer;
    x:real;
BEGIN
    write('x=');
    readln(x);
    n:=0;
    while x<=100 do
        begin
        x:=sqr(x);
        n:=n+1;
        end;
    writeln(n);
    readln;
END.
```

Lors de l'écriture d'un programme, on peut ajouter des commentaires, entr'accolades. Ils sont invisibles lors de la compilation, mais peuvent vous servir (ou à votre correcteur) lors de la relecture d'un programme. En ajouter lors d'une épreuve écrite, pour expliquer le fonctionnement d'un programme que vous écrivez, est très apprécié.

Exercice 2

Faire un programme qui demande un entier a tel que $a<10$ Le programme doit réitérer sa demande jusqu'à ce que a soit dans le bon intervalle.

Exercice 3

Soit le polynôme x^2-x-1 . Etudier rapidement ses variations et écrire un programme permettant de calculer par dichotomie la racine positive à 0,01 puis 0,0001, ... près.

Exercice 4

Soient les suites u_n, v_n et w_n définies par : $$u_0=0$$ $$u_{n+1}=\cos(u_n)$$ $$v_n=u_{2n}$$ $$w_n=u_{2n+1}$$

a) Montrer que u_n converge et que les suites v_n et w_n sont adjacentes.

b) Faire un programme calculant sa limite à ϵ près (entré par l'utilisateur).

Exercice 5 :

$$\forall x | 0 \leq x < 1, \ln(1+x) = -\sum_{n=1}^{\infty} \frac{(-x)^n}{n}$$

Soit $S_n = -\sum_{m=1}^{n} \frac{(-x)^m}{m}$

$$R_n = -\sum_{m=n+1}^{\infty} \frac{(-x)^m}{m}$$

$$u_n = \frac{(-x)^n}{n}$$

On a donc : $\forall x | 0 \leq x < 1, \forall n \in \mathbb{N}*, \ln(1+x) = S_n + R_n$

On cherche à obtenir ln(1+x) grâce au calcul de S_n à une précision ε près :

$|\ln(1+x) - S_n| < \varepsilon$

$\Leftrightarrow |R_n| < \varepsilon$

Or, u_n est une suite décroissante et altérnée (une fois positive, une fois négative), ce qui permet de dire que :

$|R_n| < |u_{n+1}|$

Donc, si $|u_{n+1}| < \varepsilon$, on a forcément atteint la précision voulue.

Faire un programme qui calcule S_n et u_{n+1} jusqu'à avoir $\ln(1+x)$ à la précision voulue. On pourra afficher sa valeur exacte pour la comparer avec le résultat obtenu, et mettre x et ε en constantes.

Indice : utiliser une variable supplémentaire pour stocker les puissances successives de (-x).