Université Batna2 Le 10/02/2024
Faculté Technologie
Département LMD ST Informatique2 1iere année

TP 1 :INSTRUCTIONS REPETITIVES

But du TP : Manipulationtion des boucles **.**

L'idée de la boucle FOR : on répète une suite d'instructions un certain nombre de fois, donc c'est un compteur. La syntaxe est la suivante :

```
for <indice>:=<vi> to <vf> do   (* vi: valeur initiale, vf : valeur finale*)
   BEGIN
      <instruction(s)>;
   END;
```

Attention les instructions de la boucle ne sont pas exécutées si <vi> est supérieure à <vf>.

L'idée de la boucle WHILE : on répète une suite d'instructions tant qu'une certaine condition reste vraie. La syntaxe est la suivante :

```
WHILE <condition> DO
                  BEGIN
                   <  Instruction(s) >;
                  END;
```

Le bloc d'instruction sera répété jusqu'à ce que la condition ne soit pas vérifiée.

ATTENTION : Ce type de boucle peut ne jamais s'achever si la condition reste toujours vraie. Assurez vous bien que celle-ci devient donc fausse au bout d'un certain temps!!!

L'idée de la boucle REPEAT : on répète une suite d'instructions jusqu'à ce qu'une certaine condition soit vraie. La syntaxe est la suivante :

```
REPEAT
   <  Instruction(s) >;
 UNTIL  <condition>;
```

- Ici, il n'est pas nécessaire d'encadrer les instructions qui se répètent par « **Begin** » et « **End ;** » ; elles sont délimitées par REPEAT et UNTIL.
- Le bloc d'instruction sera répété jusqu'à ce que la condition soit vérifiée.
- Au contraire de la boucle « **TantQue »**, l'utilisation de la boucle « **Répéter... Jusqu'à »** garantit que le bloc sera exécuté **au moins une fois** puisque le test a lieu après son exécution.

-La signification de la condition n'est plus la même :

• Avec « **TantQue** » la Condition est pour continuer la répétition (test d'entrée).
• Avec « **Répéter... Jusqu'à** » la Condition est pour arrêter la répétition (test de sortie).

-. On utilise généralement les instructions **While** ou **Repeat** lorsque l'on ne connaît pas, à l'avance, le nombre d'itérations

Exercice 1 :

```
PROGRAM suite;
Uses crt ;
 VAR u :REAL;
       i: INTEGER;
BEGIN
  u:=0;
 WRITELN(u);
 FOR i:= 1 TO 6 DO
 BEGIN
   u:= 2* u +3;
   WRITELN(u);  Readkey() ;
 END;
END.
```

1. Deviner l'affichage du programme précédent après exécution et reconnaître la suite mise en jeu.
2. Transformer le programme pour qu'il n'affiche que u100 (et rien d'autre).

Readkey() permet permet d'arrêter le défilement de l'affichage jusqu'à ce qu'on tape une touche du clavier.

3. Remplacer alors la boucle FOR par la boucle WHILE.

Exercice 2 :

Soit la suite u définie par $U_1 = 1$ et pour tout $n \in N^*$, $U_{n+1} = U_n + \frac{1}{n+1}$. Compléter et exécuter le programme pour qu'il calcule et affiche le terme U_{30}

```
Program suite ;
 Uses crt ;
Var i :…… ;
      U :…… ;
Begin
 U:=……..;
 For i:=……to…… do U:=……;
 Writeln('le terme U30  est:',……) ; Readkey() ;
End.
```

Exercice 3 : programme **puissance**

Compléter le programme qui demande un réel *x* et un entier naturel *n* positif ou nul, puis qui calcule x^n.

```
PROGRAM puissance;
Uses crt;
VAR x,p : …… ; n,k : …… ;
BEGIN
WRITE('Donner x et n'); ……;……..; p:=……..;
IF (n>=0) THEN
begin
  FOR …… TO …….. DO ……..;    WRITELN(x:1:2,' ^',n,'=',p:1:2);
End
else writeln(' valeur de n doit être positive ou nul');
Readkey() ;
END.
```

Exercice 4 :

Un entier naturel de trois chiffres est dit cubique s'il est égal à la somme des cubes de ses trois chiffres.

Exemple:153 est cubique car $153 = 1^3 + 5^3 + 3^3$. Compléter le programme Pascal **NBR_CUBE** qui cherche et affiche tous les entiers cubiques de trois chiffres (c'est-à-dire tous les entiers cubiques compris entre 100 et 999).

```
Program  Nbr_Cube;
Uses crt;
Var       k, c, d, u : ……..;
Begin
   FOR k:=…… To …… Do
   Begin
    c:=……..;   d:=……..;  u:=……..;
    IF …… = K
    Then Writeln (……., ' est un nombre cubique');
   End;
End.
```

Exercice 5 :

On s'intéresse au jeu suivant : la machine choisit au hasard un entier entre 0 et 7 et l'utilisateur doit le deviner.

Compléter le programme suivant afin de pouvoir y jouer :

```
PROGRAM jeu;
 VAR rlt, k : INTEGER;
BEGIN
  RANDOMIZE;
  rlt:=RANDOM(8); { On peut générer un nombre entier pseudo-aléatoire compris entre 0 et (8-1)=7 grâce à la fonction
Random(8)  }
  REPEAT
     WRITE(' Entrez votre proposition') ;  …… ;
 UNTIL …… ;
 WRITELN('vous avez gagné');
END.
```

2. Ajouter une variable qui compte le nombre d'essais nécessaire à l'utilisateur pour gagner, puis l'afficher.

3. Comment transformer le programme pour utiliser la boucle WHILE ?

Exercice 6 :

Compléter le programme SOMME qui calcule et affiche la somme suivante : S=1+1/2+1/4+……+1/(n+1) ou n est un nombre impair >=1 donné par l'utilisateur. On suppose qu'il n y a pas d'erreur dans la saisie de n.

```
Program SOMME ;
Uses CRT ;
Var s :…………………………… ;
    i,n :…………………………….;
begin
  writeln('donner un nombre impair');
 ………………………………………………………….;
 S:=……………………………………………….;
FOR  i:= ……………………  to ……………......…. DO
    S:=…………………………………………….;
 writeln('la somme est: ' , ………………………..);
end.
```

2- Réécrire le programme en utilisant le WHILE.

3- Réécrire le programme en utilisant la boucle repeat.

Exercice 7:

On se propose de déterminer le **PGCD** (Plus Grand Commun Diviseur اكبر قاسم مشترك) de deux entiers positifs non nuls A et B en utilisant l'algorithme d'**Euclide** : Sachant que PGCD (A, B) = PGCD(B, R), avec R = A mod B. Donc, tant que le reste R est non nul, on remplace A par B et B par R. Le dernier reste R non nul est alors le PGCD des deux nombres. Exemple : PGCD (32, 12) = PGCD (12, 8) = PGCD (8, 4) = PGCD (4, 0) = 4. A et B doivent être positifs .

```
Program PGCD_Euclide;
Uses    Wincrt;
Var     a, b, r : …………………………………………….;
Begin
  Repeat
   Writeln ('Saisir un entier a > 0');   ………………………………………………………….;
   Writeln ('Saisir un entier b > 0');   ………………………………………………………….;
  Until …………………. and …………………… ;
  While ……………………………. Do
  Begin
    r := …………………………….;  a := ……………………………….;  b := …………………………………….;
   End;
   Writeln ('PGCD = ', ……………………………….);
End.
```

Exercice8:

Compléter le programme Pascal intitulé **COMBINAISON**, qui lit deux entiers naturels n et p avec (0<p<n), puis calcule et affiche le nombre de combinaisons de p objets parmi n (C_n^p). On rappelle que $C_n^p = \frac{n!}{p!(n-p)!}$

Remarque : le programme ne doit accepter que les valeurs de p>0 et n>p.

```
Program Combinaison ;
  Uses    crt ;
  Var     cnp : ……………………………… ;
          n, p, i , nf, pf, npf: ………………………………. ;
 Begin
Repeat
  Write ('p = '); ………………………………………………..;
  Write ('n = '); ………………………………………………..;
Until ……………………………………..;
nf :=1; Pf :=1; npf :=1;  i:=2;
WHILE  ……………………………. Do
Begin
  nf := ……………………………………..;
  IF  ……………………Then pf := ……………………………..;
  IF …………………… Then npf := ……………………………..;
  i:= i+1;
 End;
 cnp := nf / (pf*npf);  Writeln ('Combinaison = ', ……………………….) ;
End.
```