Université Batna2 Le 12/03/2023
Faculté Technologie
Département LMD ST Informatique2 1iere année

TP 2 :INSTRUCTIONS REPETITIVES (LES BOUCLES CONDITIONNELLES)

But du TP : Manipulationtion des boucles **WHILE et REPEAT.**

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 debut;
  Uses crt;
 VAR x:REAL ;
BEGIN
 x:=0;
 WHILE x <= 5 DO
 BEGIN
   x:=x+1; WRITELN(x);
 END;
END.
```

1. Que fait le programme ?
2. Remplacer alors la boucle WHILE par la boucle REPEAT UNTIL.

Exercice 2 :

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 3 :

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 4:

Soit la suite définie par : S=1+1/3+1/5+1/7+1/9+……

1-Compléter le programme qui calcule la valeur de S en s'arrêtant lorsque le terme 1/x est plus petit que ε ; ε est un nombre réel <1, donné par l'utilisateur.

```
PROGRAM exo4;
  use crt ;
  var s, e: real;
BEGIN
   write ('epsilon = ');
   readln (……..);
   s := ……;
  i := ……;
  repeat
    s := ……;
    i := …… ;
 until ……..;
writeln ('S = ',……);
END.
```

2- Réécrire le programme en utilisant le WHILE.

Exercice 5:

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.

Remarque : le programme ne doit accepter que des valeurs positives pour A et B .

```
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.
```

Exercice6:

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.
```

Exercice7:

Compléter le programme SOMME qui calcule et affiche la somme suivante : S=1+1/3+1/5+……+1/n 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:=……..;
 i:=……..;
WHILE ……..…..…….. DO
  Begin
   S:=……..;
   i:=……..;
  end;
  writeln(‘la somme est: ‘ , ……);
end.
```