Structures de contrôle Itératives

Sommaire

III. Les instructions itératives 2

A. Structure itérative complète Pour .... faire ... fin pour 2

1) En analyse : Ecrire une analyse qui permet le remplissage et l'affichage d'un tableau composé par N entiers manuellement 2

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse remplissageTableau 3

3) En pascal : Ecrire le programme pascal qui permet la traduction de l'algorithme remplissageTableau 3

B. Structure itérative complète récurrente 3

1) En analyse : Ecrire une analyse qui permet le calcul et l'affichage de la somme des chiffres d'un entier aléatoire de composé par 5 chiffres 3

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse sommeChiffre 4

3) En pascal : Ecrire le programme pascal qui permet la traduction de l'algorithme sommeChiffre 4

C. Structure itérative à condition d'arrêt REPETER .... JUSQU'A 4

En analyse et algorithmique 4

En pascal 4

1) En analyse : Ecrire une analyse qui permet l'affichage du carré d'un entier saisi strictement positif 5

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse carrePositif 5

3) En pascal : Ecrire un programme pascal qui permet la traduction de l'algorithme carrePositif 5

D. Structure itérative à condition d'arrêt TANT QUE .... FAIRE 5

1) En analyse : Ecrire une analyse qui permet l'affichage de nombre des années au bout desquelles un client disposera d'une somme de 600 dinars. Sachant que le client place lors de la création du compte épargne 450 dinars au taux annuel 3% 6

1) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse nombreAnnee 6

1) En pascal : Ecrire un programme pascal qui permet la traduction de l'algorithme nombreAnnee 6

III. Les instructions itératives

A. Structure itérative complète *Pour .... faire ... fin pour*

La structure POUR permet l'exécution répétitive d'un nombre de fois fini et connu d'avance d'une suite d'instructions

En analyse et algorithmique	En pascal
R = [Initialisation] **Pour** compteur **de** valInitiale **à** valFinale **faire** {Suite de traitements à faire} **fin pour**	{suite des Initialisations} **For** compteur **:=** valInitiale **to** ValFinale **do** {Suite de traitements à faire} **;**

Le compteur est de valeur scalaire (entier , caractère , booléen , etc)

Le compteur progresse d'une façon automatique au suivant de la valeur en cours ou au précédent dans le cas d'une répétition dans l'ordre inverse :

En analyse et algorithmique	En pascal
R = [Initialisation] **Pour** compteur **de** valInitiale **à** valFinale pas = - 1 **faire** {Suite de traitements à faire} **fin pour**	{suite des Initialisations} **For** compteur **:=** valInitiale **downto** ValFinale **do** {Suite de traitements à faire} **;**

1) En analyse : Ecrire une analyse qui permet le remplissage et l'affichage d'un tableau composé par N entiers manuellement

- Analyse

```
Nom = remplissageAffichageTableau
Résultat =  Affichage
Affichage = [ ] Pour i de 1 à n faire ecrire(t[i]," | " )    fin pour
t = [n = donnée("Taille Tableau N = ")]
    pour i de 1 à n faire     t[i] = donnée("T[ ", i , " ] = ")
    fin pour
Fin remplissageAffichageTableau
```

- T.D.N.T

Types
tab = tableau de Max entiers

- T.D.O

Objet	Nature / Type	Rôle
n	Var / entier	Taille du tableau
t	Var / tab	Tableau d'entiers à remplir et afficher
i	Var / entier	compteur

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse remplissageTableau

```
0)debut remplissageAffichageTableau
1) t = [ Ecrire("Taille du tableau N = "),lire(N) ]
   pour i de 1 à n faire  ecrire( "T[ ", i , " ] = ") , lire ( t[i])
   fin pour
2) Affichage = [ ] pour i de 1 à n faire
                        ecrire ( t[i], " | " )
                   fin pour
3) Fin remplissageAffichageTableau
```

3) En pascal : Ecrire le programme pascal qui permet la traduction de l'algorithme remplissageTableau

```
Program remplissageTableau;
Uses wincrt;
Type   tab = array[1..100]of integer;
Var    n,i : integer ; t : tab;
Begin
Write('Taille du tableau N = ');readln(N);
Writeln('Remplissage du tableau : ');
for i := 1 to n do
   begin
     write(' T[ ', i ,' ] = ');readln(t[i]);
   end ;
for i := 1 to n do write(t[i], ' | ');
end.
```

B. Structure itérative complète récurrente

Il s'agit de la répétition d'une suite d'instructions afin de réaliser un calcul récurrent c-a-d que le calcul de fait au fur et à mesure en dépendance d'un calcul à une étape donnée. Si la relation lie deux étapes successifs alors , elle est d'ordre 1 , trois éléments successifs alors d'ordre 3 , etc.

1) En analyse : Ecrire une analyse qui permet le calcul et l'affichage de la somme des chiffres d'un entier aléatoire de composé par 5 chiffres

- Analyse

```
Nom = sommeChiffre
Résultat =  Ecrire(" Somme des chiffres de " , n," égale à ", s)
s = [ s ← 0, convch(n , ch) ] Pour i de 1 à 5 faire
                              valeur(ch[i],chiffre,e)
                              s ←  s + chiffre
                           fin pour
n = 10 000 + aléa(90 000)
Fin sommeChiffre
```

- T.D.O

Objet	Nature / Type	Rôle
n	Var / entier Long	Un entier aléatoire composé apr 5 chiffres

ch	Var / chaine[5]	Chaine contient le résultat de la conversion de l'entier n en une chaine
s	Var / entier	Somme des 5 chiffres d'un entier
chiffre	Var / entier	Les chiffres de l'entier choisi
e	Var / entier	Position du caractère qui a levé l'erreur lors de la conversion
i	Var / entier	compteur

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse sommeChiffre

```
0) debut sommeChiffre
1) n = 10 000 + aléa ( 90 000)
2) s = [ s ← 0 ,convch(n, ch) ]  pour i de 1 à 5 faire
                                    valeur( ch[i], chiffre , e)
                                   s ← s + chiffre
                                fin pour
3) Ecrire(" Somme des chiffres de " , n," égale à ", s)
4) Fin sommeChiffre
```

3) En pascal : Ecrire le programme pascal qui permet la traduction de l'algorithme sommeChiffre

```
Program sommeChiffre ;
Uses wincrt;
Var    n,i,chiffre, e : integer ; ch : string[5];
Begin
randomize ;   n := 10000 + random( 90000) ;
str(n , ch ) ; s : = 0 ;
for i := 1 to 5 do
   begin
        val(ch[i], chiffre , e) ;
        s := s + chiffre ;
   end ;
 write(' Somme des chiffres de ' , n,' égale à ', s) ;
end.
```

C. *Structure itérative à condition d'arrêt* *REPETER .... JUSQU'A ...*

La structure REPETER permet l'exécution répétitive au moins une fois d'une suite d'instructions . L'arrêt est généré par une condition.

En analyse et algorithmique	En pascal
R = [Initialisation] **Répéter** {Suite de traitements à faire} **Jusqu'à (** *condition d'arrêt* **)**	{suite des Initialisations} **repeat** {Suite de traitements à faire} **until (** *conditionArret* **) ;**

1) En analyse : Ecrire une analyse qui permet l'affichage du carré d'un entier saisi strictement positif

- Analyse

```
Nom = carrePositif
Résultat =  Ecrire ( n, " au carré = " , carre ( n ))
n = [ ]  répéter
          n = donnée("'Saisie entier Positif  N = ")]
jusqu'à n > 0
Fin carrePositif
```

- T.D.O

Objet	Nature / Type	Rôle
n	Var / entier	Un entier positif saisi

2) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse carrePositif

```
0) debut carrePositif
1) n = [ ]  répéter
                N = donnée( "'Saisie entier Positif N = ")
          Jusqu'à n > 0
3) Ecrire(n , " au carre = " , carre( n ) )
4) Fin carrePositif
```

3) En pascal : Ecrire un programme pascal qui permet la traduction de l'algorithme carrePositif

```
Program carrePositif ;
Uses wincrt ;
Var n : integer ;
Begin
Repeat
      begin
            write( 'Saisie entier Positif N = ') ; readln(n);
      end ;
        until n > 0 ;
 write(n , ' au carre = ' , sqr( n ) );
end.
```

D. *Structure itérative à condition d'arrêt* *TANT QUE .... FAIRE ...*

La structure TANT QUE permet l'exécution répétitive d'un nombre inconnu d'itérations d'une suite d'instructions . La répétition est généré tant que la condition est vérifié.

En analyse et algorithmique	En pascal
R = [Initialisation] **TANT QUE (**condition de non arrêt**) FAIRE** {Suite de traitements à faire}**FIN TANT QUE**	{suite des Initialisations} **WHILE (**condition de non arrêt**) DO** {Suite de traitements à faire} **;**

1) En analyse : Ecrire une analyse qui permet l'affichage de nombre des années au bout desquelles un client disposera d'une somme de 600 dinars. Sachant que le client place lors de la création du compte épargne 450 dinars au taux annuel 3% .

- Analyse

```
Nom = nombreAnnee
Résultat =  Ecrire (" Le nombre des années est : " , nb)
nb = [ nb ← 0 , compte ← 450]
      tant que compte <> 600 faire
           compte ← compte + 15
          nb ← nb + 1
      fin tant que
Fin nombreAnnee
```

- T.D.O

Objet	Nature / Type	Rôle
nb	Var / entier	Nombre des années au bout desquelles le client disposera de la somme voulue

1) En algorithme : Ecrire l'algorithme qui permet la traduction de l'analyse nombreAnnee

```
0) debut  nombreAnnee
1) nb = [ nb ← 0 , compte ← 450]
      tant que compte <> 600 faire
           compte ← compte + 15
          nb ← nb + 1
      fin tant que
2)  Ecrire (" Le nombre des années est : " , nb)
3) Fin nombreAnnee
```

1) En pascal : Ecrire un programme pascal qui permet la traduction de l'algorithme nombreAnnee

```
Program nombreAnnee ;
Uses wincrt ;
Var nb,compte : integer ;
Begin
Nb := 0 ; compte := 450 ;
While compte <> 600 do
      begin
            compte := compte + 15;
             nb := nb + 1 ;
     end ;
 write('Le nombre des annees est : ' , nb); end.
```

Remarque :

❖ R est un objet dont son calcul est itératif , Initialisation contient les instructions nécessaires à la marche du processus d'itérations

❖ Dans la structure Répéter ... jusqu'à le nombre d'itérations est au moins une seule fois

❖ Dans la structure Tant que ... faire ... le traitement peut ne pas être exécuter