# Notion d’exécution répétée (ou boucles)

Dans une procédure, il arrive parfois que des opérations doivent être répétées plusieurs fois.

Dans la vie courante, on est très souvent confronté à cette situation :

Par exemple, si on doit remplir une baignoire avec des seaux (oui, ça peut arriver).

* **tant que** la baignoire n’est pas pleine **répéter** verser un seau.
	+ la condition, exprimée sous forme d’une expression logique, est : «la baignoire n’est pas pleine»
	+ si cette condition est vraie, et tant qu’elle est vraie, on va répéter «verser un seau»
	+ dans le cas contraire, on ne fait rien

Ou bien, si on a pris soin de calculer le nombre de seaux dont on avait besoin : nombre de seaux = volume de la baignoire / volume du seau, soit ici : 200 / 15 = 13

* **pour** mon compteur de seaux allant de 1, et tant qu’il est inférieur ou égal à 13 **répéter** verser un seau puis ajouter 1 au compteur
	+ la condition, exprimée sous forme d’une expression logique, est : «le nombre de seaux versés est inférieur ou égal à13»
	+ si cette condition est vraie, et tant qu’elle est vraie, on va répéter «verser un seau», puis ajouter 1 au compteur pour tester à nouveau si on verse de nième seau.

# Deux formes de répétitions

## 1ère forme, on ne sait pas à l’avance combien de fois on devra répéter :

TANTQUE (condition) REPETER

 liste des opérations à répéter

finTANTQUE

2ème forme, on sait à l’avance combien de fois on devra répéter

On peut utiliser un compteur pour compter ce nombre de fois

POUR compteur ALLANT DE valeurDebut A valeurFin PAR PAS DE 1

 liste des opérations à répéter

finPOUR

# Nombre de répétitions indéterminé

Dans une procédure on écrira ces tests sous la forme suivante :

**TANTQUE** (condition) **REPETER**

 liste des opérations à répéter

**finTANTQUE**

# Exemple : multiplier les nombres de 1 à 6 par 12

* calculer 1x12, puis 2x12, puis 3x12, puis 4x12, puis 5x12, et enfin 6x12
* soit, si on donne le nom *compteur* au nombre qui varie de 1 à 6 :
	+ donner la valeur 1 à compteur
	+ tant que compteur est inférieur ou égal à 6,
		- calculer *compteur* x 12
		- ajouter 1 à *compteur*

# Exemple : multiplier les nombres de 1 à 4 par 12

L’algorithme suivant affiche la table de multiplication des nombres de 1 à 6 par 12

/\* DECLARATIONS \*/

VARIABLES

 entier compteur 🡨 0 // compteur de répétition

 entier resultat 🡨 0 // resultat de la multiplication

DEBUT (des opérations)

/\* afficher la table de multiplication \*/

01 compteur 🡨 1

02 **TANTQUE** compteur <= 4 REPETER

 /\* calcul \*/

03 resultat 🡨 compteur \* 12

03bis AFFICHER resultat

04 compteur 🡨 compteur + 1

 **finTANTQUE**

05 AFFICHER « terminé ! »

FIN (des opérations)

## Trace de l’exécution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pas** | **Entrée** | **Condition****Vrai / Faux** | **CONSTANTES VARIABLES** | **Sortie** |
|  |  |  |  | **compteur** | **resultat** |  |  |
| **DEB** |  |  |  |  |  |  | 0 | 0 |  |  |  |
| 01 |  |  |  |  |  |  | 1 |  |  |  |  |
| 02 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 03 |  |  |  |  |  |  |  | 1\*12 =12 |  |  | 12 |
| 04 |  |  |  |  |  |  | 1+1 = 2 |  |  |  |  |
| 02 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 03 |  |  |  |  |  |  |  | 2\*12 = 24 |  |  | 24 |
| 04 |  |  |  |  |  |  | 2+1 = 3 |  |  |  |  |
| 02 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 03 |  |  |  |  |  |  |  | 3\*12=36 |  |  | 36 |
| 04 |  |  |  |  |  |  | 3+1 = 4 |  |  |  |  |
| 02 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 03 |  |  |  |  |  |  |  | 4\*12=48 |  |  | 48 |
| 04 |  |  |  |  |  |  | 4+1 = 5 |  |  |  |  |
| 02 |  | compteur<= 4 ? FAUX |  |  |  |  |  |  |  |  |  |
| 05 |  |  |  |  |  |  |  |  |  |  | Terminé ! |
| **FIN** |  |  |  |  |  |  |  |  |  |  |  |

# Exemple : Calculer la quantité total d’un ensemble de produits

On souhaite avoir une procédure qui permette la saisie de quantité de plusieurs quantités d’un produit et d’en afficher la somme. On arrêtera la saisie quand la quantité saisie sera 0.

/\* DECLARATIONS \*/

VARIABLES

 décimal qteSaisie 🡨 0 // quantité, saisi

 décimal total 🡨 0 // somme, saisi

DEBUT (des opérations)

/\* demander la saisie de qteSaisie \*/

 AFFICHER « donnez une quantité (0 pour arrêter) : »

 SAISIR qteSaisie

/\* cumul des quantités \*/

 TANTQUE (qteSaisie <> 0) REPETER

 total 🡨 total + qteSaisie

 /\* saisir une nouvelle quantité \*/

 AFFICHER « donnez une autre quantité (0 pour arrêter) : »

 SAISIR qteSaisie

 finTANTQUE

/\* donner le résultat \*/

 AFFICHER « le total des quantités est  », total

FIN (des opérations)

# Nombre de répétitions déterminé

Dans une procédure on écrira ces tests sous la forme suivante :

**POUR** compteur **ALLANT DE** valeurDebut **A** valeurFin

 PAR **PAS DE** increment **REPETER**

 liste des opérations à répéter

**finPOUR**

* *valeurDebut* et *valeurFin* : *compteur* va prendre
* tour à tour chaque valeur de *valeurDebut* à *valeurFin*
* et à chaque tour, il va ajouter *increment* pour passer à
* la valeur suivante

# Exemple : multiplier les nombres de 1 à 6 par 12

* calculer 1x12, puis 2x12, puis 3x12, puis 4x12, puis 5x12, et enfin 6x12
* soit, si on donne le nom *compteur* au nombre qui varie de 1 à 6 :
	+ pour *compteur* de 1 à 6,
		- calculer *compteur* x 12

# Exemple : multiplier les nombres de 1 à 4 par 12

L’algorithme suivant affiche la table de multiplication des nombres de 1 à 6 par 12

/\* DECLARATIONS \*/

VARIABLES

 entier compteur 🡨 0 // compteur de répétition

 entier resultat 🡨 0 // resultat de la multiplication

DEBUT (des opérations)

/\* afficher la table de multiplication \*/

01 POUR compteur ALLANT DE 1 A 4 PAS DE 1 REPETER

 /\* calcul \*/

02 resultat 🡨 compteur \* 12

02bis AFFICHER resultat

 finPOUR

03 AFFICHER « terminé ! »

FIN (des opérations)

## Trace de l’exécution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Pas** | **Entrée** | **Condition****Vrai / Faux** | **CONSTANTES VARIABLES** | **Sortie** |
|  |  |  |  | **compteur** | **resultat** |  |  |
| **DEB** |  |  |  |  |  |  | 0 | 0 |  |  |  |
| 01 |  |  |  |  |  |  | 1 |  |  |  |  |
| 01 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 02 |  |  |  |  |  |  |  | 1\*12=12 |  |  | 12 |
| 01 |  |  |  |  |  |  | 1+1 = 2 |  |  |  |  |
| 01 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 02 |  |  |  |  |  |  |  | 2\*12=24 |  |  | 24 |
| 01 |  |  |  |  |  |  | 2+1 = 3 |  |  |  |  |
| 01 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 02 |  |  |  |  |  |  |  | 3\*12=36 |  |  | 36 |
| 01 |  |  |  |  |  |  | 3+1 = 4 |  |  |  |  |
| 01 |  | compteur<= 4 ? Vrai |  |  |  |  |  |  |  |  |  |
| 02 |  |  |  |  |  |  |  | 4\*12=48 |  |  | 48 |
| 01 |  |  |  |  |  |  | 4+1 = 5 |  |  |  |  |
| 01 |  | compteur<= 4 ? **Faux** |  |  |  |  |  |  |  |  |  |
| 03 |  |  |  |  |  |  |  |  |  |  | Terminé ! |
| **FIN** |  |  |  |  |  |  |  |  |  |  |  |

# Exemple : Calculer la quantité total d’un ensemble de produits dont le nombre est connu

On souhaite avoir une procédure qui permette la saisie de quantité de plusieurs quantités d’un produit et d’en afficher la somme. On demandera à l’utilisateur le nombre de produits.

/\* DECLARATIONS \*/

VARIABLES

 entier nbreProd 🡨 0 // nombre de produits, saisi

 décimal qteSaisie 🡨 0 // quantité, saisi

 décimal total 🡨 0 // somme, saisi

 entier ct 🡨 0 // compteur de répétition (

DEBUT (des opérations)

/\* demander la saisie de nbreProd \*/

 AFFICHER « donnez le nombre de produits : »

 SAISIR nbreProd

/\* cumul des quantités \*/

 total 🡨 0

 POUR ct ALLANT DE 1 A nbreProd PAS DE 1 REPETER

 /\* saisir une quantité \*/

 AFFICHER « donnez une quantité : »

 SAISIR qteSaisie

 total 🡨 total + qteSaisie

 finPOUR

/\* donner le résultat \*/

 AFFICHER « le total des quantités est  », total

FIN (des opérations)

La boucle TANTQUE peut être utilisée dans la plupart des cas : on peut ainsi remplacer la boucle POUR précédente par une boucle TANTQUE.

# Exemple : Calculer la quantité total d’un ensemble de produits dont le nombre est connu

On souhaite avoir une procédure qui permette la saisie de quantité de plusieurs quantités d’un produit et d’en afficher la somme. On demandera à l’utilisateur le nombre de produits.

/\* DECLARATIONS \*/

VARIABLES

 entier nbreProd 🡨 0 // nombre de produits, saisi

 décimal qteSaisie 🡨 0 // quantité, saisi

 décimal total 🡨 0 // somme, saisi

 entier ct 🡨 0 // compteur de répétition (

DEBUT (des opérations)

/\* demander la saisie de nbreProd \*/

 AFFICHER « donnez le nombre de produits : »

 SAISIR nbreProd

/\* cumul des quantités \*/

 total 🡨 0

 ct 🡨 1

 **TANTQUE** (ct <= nbreProd) REPETER

 /\* saisir une quantité \*/

 AFFICHER « donnez une quantité : »

 SAISIR qteSaisie

 /\* ajouter qyeSaisie à total\*/

 total 🡨 total + qteSaisie

 /\* incrémenter ct (=ajouter 1)\*/

 ct 🡨 ct + 1 //

 **finTANTQUE**

/\* donner le résultat \*/

 AFFICHER « le total des quantités est  », total

FIN (des opérations)

# Nombre de répétitions lié à une liste de valeurs

Dans une procédure on écrira ces tests sous la forme suivante :

**POUR CHAQUE** valeur **DE** listeDeValeurs **REPETER**

 liste des opérations à répéter

**finPOUR**

* *valeur* : une variable valeur de même type que les éléments de la liste
* *listeDeValeurs* : une liste de valeurs d’un certain type

Par exemple, si on est amené à calculer la somme d’une liste de cellules d’une plage :

|  |  |  |
| --- | --- | --- |
|  | A | B |
| 1 | 2 | 3 |
| 2 | 5 | 0 |

Un algorithme correspondant pourrait être :

/\* DECLARATIONS \*/

VARIABLES

 Plage pla // une plage de cellules

 Cellule cel // une cellule

 réel som // la somme, résultat

DEBUT (des opérations)

LIRE pla

/\* pour chaque cellule de la plage, l’additionner à somme \*/

POUR CHAQUE cel DE pla REPETER

 som 🡨 som + cel

finPOUR

AFFICHER som

FIN (des opérations)

Un algorithme alternatif pourrait être :

/\* DECLARATIONS \*/

VARIABLES

 Cellule cel // une cellule d’une plage de cellules

 décimal som // la somme, résultat

DEBUT (des opérations)

LIRE cel

/\* tant que la cellule n’est pas vide, l’additionner à somme \*/

TANTQUE cel <> Fin REPETER

 som 🡨 som + cel

 LIRE cel

finPOUR

AFFICHER som

FIN (des opérations)