04/01/2021

# Algorithme

## Exercices Corrections

Lesur Rudy

# Exercice N°1

## Solution avec TantQue

**Programme** PremiersNombresEntiers

// Ce programme calcule la somme des N premiers nombres entiers avec la structure itérative **TANTQUE**

**Variables**

N : **entier**

resultat : **entier**

cpt : **entier**

**Début**

resultat := 0

cpt := 1

N := lireEntier()

**tantque** (cpt <= N) **faire**

resultat := resultat + cpt

cpt := cpt + 1

**fintantque**

**écrire** (« Le résultat avec tantque est : », resultat)

**Fin**

## Solution avec Répéter

**Programme** PremiersNombresEntiers

// Ce programme calcule la somme des N premiers nombres entiers avec la structure itérative **REPETER**

**Variables**

N : **entier**

resultat : **entier**

cpt : **entier**

**Début**

resultat := 0

cpt := 1

N := lireEntier()

**répéter**

resultat := resultat + cpt

cpt := cpt + 1

**jusquà** (cpt > N)

**écrire** (« Le résultat avec répéter est : », resultat)

**Fin**

## Solution avec Pour

**Programme** PremiersNombresEntiers

// Ce programme calcule la somme des N premiers nombres entiers avec la structure itérative **POUR**

**Variables**

N : **entier**

resultat : **entier**

cpt : **entier**

**Début**

N := lireEntier()

resultat :=0

**pour** (cpt := 1; cpt <= N ; cpt := cpt + 1) **faire**

resultat := resultat + cpt

**finpour**

**écrire** (« Le résultat avec pour est : », resultat)

**Fin**

# Exercice N°2

## Solution avec TantQue

**Programme** Factorielle

// Ce programme calcule la factorielle de l'entier N avec la structure itérative **TANTQUE**

**Variables**

N : **entier**

resultat : **entier**

cpt : **entier**

**Début**

resultat := 1

N := lireEntier()

cpt := N

**Si** ( N = 0 **OU** N = 1 ) **Alors**

**écrire** (« La factorielle de », N, « avec tantque est : », resultat)

**Sinon**

**tantque** (cpt > 1) **faire**

resultat := resultat * cpt

cpt := cpt - 1

**fintantque**

**écrire** (« La factorielle de », N, « avec tantque est : », resultat)

**Finsi**

**Fin**

## Solution avec Répéter

**Programme** Factorielle

// Ce programme calcule la factorielle de l'entier N avec la structure itérative **REPETER**

```
Variables
        N : entier
        resultat : entier
        cpt : entier
Début
        resultat := 1
        N := lireEntier()
        cpt := N
        Si (N = 0 ou N = 1) Alors
                écrire (« Le résultat avec répéter est : », resultat)
        Sinon
                répéter
                        resultat := resultat * cpt
                        cpt := cpt - 1
                jusquà (cpt = 1)
                écrire (« Le résultat avec répéter est : », resultat)
        Finsi
Fin
```

## Solution avec Pour

**Programme** Factorielle

// Ce programme calcule la factorielle de l'entier X avec la structure itérative POUR

**Variables**

N : **entier**

resultat : **entier**

cpt : **entier**

**Début**

resultat := 1

N := lireEntier()

**Si** ( N = 0 **ou** N = 1 ) **Alors**

**écrire** (« Le résultat avec pour est : », resultat)

**Sinon**

**pour** (cpt := N ; cpt > 1 ; cpt := cpt - 1) **faire**

resultat := resultat * cpt

**finpour**

**écrire** (« Le résultat avec pour est : », resultat)

**Finsi**

**Fin**

## Exercice N°3

**Programme** EquationSecondDegre

// Ce programme calcule et affiche les solutions d'une équation du second degré.

**Variables**

a : **entier**
b : **entier**
c : **entier**
D: **réel**
resultat : **réel**

**Début**

**tantque** (a = 0) **faire**
**écrire** (« Saisir la valeur de a : »)
a := lireEntier()
**fintantque**
**écrire** (« Saisir la valeur de b : »)
b := lireEntier()
**écrire** (« Saisir la valeur de c : »)
c := lireEntier()

D := (b*b) – (4*a*c)
**Si** (D < 0) **Alors**
**écrire** (« Il n'y a pas de solution pour cette équation »)
**Sinon**
**Si** (D = 0) **Alors**
resultat := -b/(2*a)
**écrire** (« Il y a une solution double pour cette équation du type -b/2a : »)
**écrire** (resultat)
**Sinon**
**écrire** (« Il y a 2 solutions pour cette équation : »)
**écrire** (-b + (√D))/(2*a)
**écrire** (-b - (√D))/(2*a)
**Finsi**
**Finsi**

**Fin**

## Exercice N°4

**Programme** CalculPuissance

// Ce programme calcule et affiche **X puissance Y**.

**Variables**

x : **entier**
y : **entier**
resultat : **entier**

**Début**

**écrire** (« Saisir la valeur de x : »)
x := lireEntier()

**écrire** (« Saisir la valeur de y : »)
y := lireEntier()

**Si** (y=0) **Alors**
**écrire** (« Le résultat est 1 »)
**Sinon**
**Si** (x=0) **Alors**
**écrire** (« Le résultat est 0 »)
**Sinon**
resultat := puissance (x,y)
**écrire** (« Le résultat est », resultat)
**Finsi**
**Finsi**

**Fin**

**fonction** puissance (**entrée** : x : entier, **entrée** y : entier) : entier

// Cette fonction calcule et retourne x élevé à la puissance y.

**Variables**

compteur : **entier**

resultat : **entier**

**Début**

compteur :=1

resultat :=1

**tantque** (compteur <= y) **faire**

resultat :=resultat * x

compteur := compteur + 1

**fintantque**

**retourner** (resultat)

**Fin**

## Exercice N°5

**Programme** RechercheDichotomique

// Ce programme effectue une **recherche dichotomique**, dans un tableau d'entiers **déjà trié**.

// Si l'entier recherché est trouvé, on affiche le rang où il se trouve dans le tableau, sinon on indique que l'entier n'existe pas dans le tableau.

**types** tabent = tableau[10] de **entier** // Création du type *tabent*

**Variables**

x : **entier**
position: **entier**
resultat : **entier**
tabEntiers : tabent

**Début**

tabEntiers = { -2, -1, 0, 13, 24, 37, 44, 56, 99, 117}; // Tableau trié
**écrire** (« Veuillez saisir la valeur de X à rechercher dans le tableau »)
x := lireEntier()

position =rechercherEntier (tabEntiers, x);
**Si** (position == -1) **Alors**
**écrire**(« x n'existe pas dans le tableau »)
**Sinon**
**écrire**(« x se trouve à la position », position, « dans le tableau »)
**Fin**

**Fin**

```
fonction rechercherEntier (entrée tab : tabent, entrée x : entier) : entier
// Cette fonction recherche la valeur de x dans le tableau tab et retourne sa position.
// Si y n'existe pas, la fonction retourne -1

Variables
    indiceBas : entier
    indiceHaut : entier
    indiceMilieu : entier
    trouvé : booléen

Début
    indiceBas := 1
    indiceHaut := tab.taille
    indiceMilieu := (indiceBas + indiceHaut) div 2

    trouvé := faux

    tantque (trouvé = faux ET indiceBas <= indiceHaut) faire
        Si (x < tab[indiceMilieu]) Alors
            indiceHaut := indiceMilieu - 1
        Sinon Si (x > tab[indiceMilieu]) Alors
            indiceBas := indiceMilieu +1
            Sinon faire
                trouvé := true
            Finsi
        Finsi
        indiceMilieu := (indiceBas + indiceHaut) div 2
    fintantque

    Si (trouvé = vrai) Alors
        retourner (indiceMilieu)
    Sinon
        retourner (-1)
    Finsi
```

## Exercice N°6

**Programme** TriPermutation

// Ce programme trie un tableau d'entiers.

**types** tabent = tableau[10] de **entier** // Création du type *tabent*

**Variables**

tabEntiers : tabent

cpt : **entier**

N : **entier**

**Début**

tabEntiers = {-5, 12, 50, 23, -2, 10, 5, 7, 81, -50}; // Tableau non trié

N = tabEntiers.taille

tabEntiers =trierTableau (tabEntiers);

// affichage du contenu du tableau

**pour** (cpt := 0; cpt < N ; cpt := cpt + 1) **faire**

**écrire**(tabEntiers[cpt] + " ")

**finpour**

**Fin**

```
fonction trierTableau (entrée tab : tabent) : tabent
// Cette fonction trie un tableau reçu en paramètre et retourne un tableau trié

Variables
    cptEl1 : entier
    cptEl2 : entier
    N : entier

Début
    N = tab.taille

    pour (cptEl1 := 0; cptEl1 < N-1 ; cptEl1 := cptEl1 + 1) faire
        pour (cptEl2 := cptEl1 +1; cptEl2 < N ; cptEl2 := cptEl2 + 1) faire
            Si (tab[cptEl1] > tab[cptEl2]) Alors
            // permutation des 2 éléments
                mem = tab[cptEl1]
                tab[cptEl1] = tab[cptEl2]
                tab[cptEl2] = mem
            Finsi
        finpour
        retourner (tab)
    finpour
```