09/11/2022

ALGORITHMIQUE

Exercices : généralités - corrections

Rudy Lesur
SOFIP

Table des matières

I Somme des n premiers nombre entiers

1/ Correction avec Répéter Jusqu'à

Programme PremiersNombresEntiers
// Calculer 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
    Jusqu'à (cpt > N)

    Ecrire (« Le résultat avec répéter est : », resultat)
Fin

2/ Correction avec Tant que Faire

Programme PremiersNombresEntiers
// Calculer la somme des N premiers nombres entiers avec la structure itérative TANT QUE

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

    Ecrire (« Le résultat avec tantque est : », resultat)
Fin

3/ Correction avec Pour

```
Programme PremiersNombresEntiers
// Calculer 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

        Ecrire (« Le résultat avec pour est : », resultat)
Fin
```

II Factorielle des n premiers nombre entiers

1/ Correction avec Répéter Jusqu'à

```
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
                Ecrire (« Le résultat avec répéter est : », resultat)
        Sinon
                Répéter
                        resultat := resultat * cpt
                        cpt := cpt - 1
                Jusqu'à (cpt = 1)
                Ecrire (« Le résultat avec répéter est : », resultat)
        Finsi
Fin
```

2/ Correction avec Tant que Faire

```
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
                Ecrire (« La factorielle de », N, « avec tantque est : », resultat)
        Sinon
                Tantque (cpt > 1) Faire
                        resultat := resultat * cpt
                        cpt := cpt - 1
                Fintantque
                Ecrire (« La factorielle de », N, « avec tantque est : », resultat)
        Finsi
Fin
```

3/ Correction 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
                Ecrire (« Le résultat avec pour est : », resultat)
        Sinon
                Pour (cpt := N ; cpt > 1 ; cpt := cpt - 1) Faire
                        resultat := resultat * cpt
                Finpour
                Ecrire (« Le résultat avec pour est : », resultat)
        Finsi
Fin
```

III Résolution d'une équation du second degré

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
resultat2 : réel
Début

Tantque (a = 0) **Faire**
Ecrire (« Saisir la valeur de a : »)
a := lireEntier()
Fintantque

Ecrire (« Saisir la valeur de b : »)
b := lireEntier()

Ecrire (« Saisir la valeur de c : »)
c := lireEntier()

D := (b*b) – (4*a*c)

Si (D < 0) **Alors**
Ecrire (« Il n'y a pas de solution pour cette équation »)
Sinon
Si (D = 0) **Alors**
resultat := -b / (2*a)
Ecrire (« Il y a une solution double : x= », resultat)
Sinon
resultat := (-b + (√D)) / (2*a)
resultat2 := (-b - (√D)) / (2*a)
Ecrire (« Il y a 2 solutions : x1= », resultat, « et x2= », resultat2)
Finsi
Finsi
Fin

IV Calcul de X à la puissance Y

```
Programme CalculPuissance
// Ce programme calcule et affiche X puissance Y.

Variables
        x : entier
        y : entier
        resultat : entier

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

Début

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

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

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

V Recherche dichotomique d'une valeur dans un tableau

Pour des besoins de lisibilités j'ai séparé le programme principal et la définition de la fonction

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

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

Ecrire (« Veuillez saisir la valeur de X à rechercher dans le tableau »)
x := lireEntier()

position := rechercherEntier (tabEntiers, x);
Si (position == -1) **Alors**
Ecrire(« x n'existe pas dans le tableau »)
Sinon
Ecrire(« 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

        trouve : booléen                                    // booléen pour indiquer si trouvé

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

        trouve := faux

        Tantque (trouve = faux ET indiceBas <= indiceHaut) Faire
                Si (x < tab[indiceMilieu]) Alors
                        indiceHaut := indiceMilieu - 1
                Sinon
                        Si (x > tab[indiceMilieu]) Alors
                                indiceBas := indiceMilieu +1
                        Sinon
                                trouve := true
                        Finsi
                Finsi

                indiceMilieu := (indiceBas + indiceHaut) div 2
        Fintantque

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