

Devoir blanc correction

Exercice 1 :

- 1) fonction premiere (E A : réel , E B : réel) : entier
- 2) fonction seconde (E V : vecteurs de réels, E N:entier) : entier
- 3) fonction troisieme (E/S W : vecteur de 10 entier) : réel
- 4) 2 sorties donc pas souhaitable. Mais on peut faire :
fonction quatrieme (E E:entier, S C:caractère):réel
ou
fonction quatrieme (E E:entier, S R:réel):caractère
voilà ce qu'il faudrait qu'ils répondent
- 5) impossible car pas de sortie
- 6) procédure une (E VC : vecteur de 40 caractère , S A : entier)
- 7) procédure deux (E/S X : entier , E B : réel , E C : réel)
- 8)
procédure trois (S T:booléen, S U:booléen)

Exercice 2 :

Algorithme suite

```
{
    variable
        N , i: entier
        Un : réel
        Un ← 1
        Afficher (" veuillez saisir la valeur de N " )
        Saisir (N)
        Pour i de 0 à N-1
        {
            Un = (Un+2)/(Un+5)
        }
        Afficher (" La valeur de Un pour n= " , N , " est : " , Un )
}
```

Exercice 3 :

```
procédure initialisation ( E/S V : vecteur d'entier, E NB : entier )
{
    variable i : entier
    Pour i de 0 à NB-1
        V[i] ← -1
}
```

procédure SaisieVentes (E/S V vecteur d'entier, E NB : entier, E nbcommerciaux : entier)

```
{
    variable i,j,n, vente,nbvente: entier
        stop : booléen
    i ← 0
    j ← 1
    stop ← FAUX

    Tant que ((j<nbcommerciaux) ET (NON (stop)))
    {
        Afficher ( " Veuillez saisir le nombre de ventes du commercial " , j )
        Saisir ( nbvente )

        // Si il y a encore de la place dans le vecteur
        Si (i+nbvente+1<=NB) alors
        {
            V[i] ← nbvente
            i++
            Si (nbvente != 0) alors
            {
                Pour n de 1 à nbvente
                {
                    Afficher( "Veuillez saisir le montant de la vente numéro " , n ,
" du commercial numéro" , j )
                    Saisir(vente)
                    V[i] ← vente
                    i++
                }
            }
        }
        sinon // s'il n'y avait plus de place, on stoppe
        {
            stop ← VRAI
            Afficher ("Comme on ne peut pas saisir les ventes de ce commercial, on
décide d'arrêter toutes les autres saisies restantes pour les autres commerciaux – c'est une solution
parmi d'autres pour éviter de dépasser la taille maximum du vecteur.")
            Afficher("Vous n'aurez donc saisie que les ventes de ", j-1, "commerciaux")
        }

        // Commercial suivant
        j++
    }
}
```

fonction NombreDeCommerciaux (E V : vecteur d'entier, E NB : entier): entier

```
{
    variable i,n, nbcom : entier
    nbcom ← 0
    i ← 0
    Tant que ((V[i]!= -1) ET (i<NB)) // il faut les deux conditions car le vecteur peut être plein
    {
        // c'est-à-dire sans -1
        n ← V[i]
```

```

        i ← i + n + 1 // n contient le nb de vente d'un commercial +1 pour aller au prochain
commercial
        nbcom++
    }
    retourne nbcom
}

```

```

fonction ChiffreAffaires ( E V : vecteur d'entier, E NB : entier, E numcommercial: entier): entier
{

```

```

    variable i,j,n total: entier
    total ← 0
    j ← 1
    i ← 0

```

```

    // on se déplace de commercial en commercial
    Tant que ((j < numcommercial) ET (V[i]!=-1) ET (i < NB))
    {
        i ← i + V[i] + 1
        j++
    }

```

```

    // Le commercial n'est pas présent puisqu'on est à la fin du vecteur ou qu'on a atteint la fin de
la partie « utile »

```

```

    Si ((V[i]=-1) OU (i = NB)) alors
        retourne -1

```

```

    n ← V[i] // On est sur notre commercial on va chercher son nombre de vente
    i++

```

```

    // on parcourt ses ventes
    Pour k de 1 à n
    {
        total ← total + V[i]
        i++
    }

```

```

    retourne total

```

```

}

```

```

procédure AfficheVentesEtCA ( E V : vecteur d'entier, E NB : entier)
{

```

```

    variable i,n,com,vente: entier
    com ← 1
    i ← 0
    Tant que ((V[i] != -1) ET (i < NB))
    {

```

```

        n ← V[i]
        Si (n = 0) alors
        {
            Afficher(" Le commercial " , com , "n'a fait aucune vente.\n")
            i++ // pour se déplacer sur le prochain commercial
        }
    }

```

```

sinon
{
    Afficher(" Le commercial " , com , " a fait ")
    Selon (n)
    {
        1: Afficher("une ")
        2: Afficher("deux ")
        3: Afficher("trois ")
        4: Afficher("quatre ")
        ...
    }

    Si (n=1) alors
        Afficher (" seule vente ( " )
    sinon
        Afficher (" ventes ( " )

    i++ // pour se déplacer sur la liste des ventes

    Pour j de 1 à n
    {
        Si ( j = n ) alors
            Afficher ( V[i] , " euros ) ")
        sinon
            Afficher ( V[i], " euros et " )
        i++
    }
}
}

```

```

fonction ChiffreSociete( E V : vecteur d'entier, E NB : entier): entier
{
    variable i , nbcom, totcom, totsociete: entier
    totsociete ← 0
    nbcom ← NombreDeCommerciaux(V, NB)
    Si (nbcom > 0) alors
    {
        Pour i de 1 à nbcom
        {
            totcom ← ChiffreAffaires (V , NB, i)
            totsciete ← totsociete + totcom
        }
    }
    retourne totsociete
}

```

```

procédure MeilleurVendeur( E V : vecteur d'entier, E NB : entier, S numcom : entier, S
meilleurCA:entier)
{
    variable i , nbcom, totcom: entier
    nbcom ← NombreDeCommerciaux(V, NB)

```

```

Si (nbcom = 0) alors
{
    numcom ← -1
    meilleurCA ← -1
}
sinon
{
    numcom ← 1
    meilleurCA ← ChiffreAffaires (V , NB, 1)
    Si (nbcom > 1) alors
    {
        Pour i de 2 à nbcom
        {
            totcom ← ChiffreAffaires ( V , NB, i)
            Si (totcom > meilleurCA) alors
            {
                numcom ← i
                meilleurCA ← totcom
            }
            sinon
            {
                Si (totcom = meilleurCA) alors
                {
                    Si (numcom > 0) alors
                    {
                        numcom ← -2
                    }
                    sinon
                    {
                        numcom ← numcom-1
                    }
                }
            }
        }
    }
}
}
}
}
}
}

```

Algorithme Société

```

{
    constante entier NB ← 10
    variable
        V : vecteur de NB entier
        nbcom, numcom, meilleurCA, total : entier
        Initialisation (V, NB)
        Afficher ("Veuillez saisir le nombre de commerciaux :")
        Saisir (nbcom)
        SaisieVentes (V, NB, nbcom)
        AfficheVentesEtCA (V, NB )
        MeilleurVendeur(V, NB, numcom, meilleurCA)
        total ← ChiffreSociete(V, NB)
}

```

```

Si (numcom > 0) alors
{
    Afficher( " Le meilleur vendeur est le vendeur " , numcom, " et son pourcentage sur
la societe est de " , (meilleurCA/total) *100 , " pourcent.\n" )
}
sinon
{
    Afficher( " Il y a " , numcom, " ayant réalisé le meilleur chiffre d'affaire et leur
pourcentage individuel sur la societe est de " , (meilleurCA/total) *100 , " pourcent.\n" )
}
}

```

Exercice 4 :

```

procédure initialisation (E/S V vecteur d'entier, E MAX : entier)
{
    variable i : entier
    Pour i de 0 à MAX-1
        V[i] ← 1
}

```

```

procédure affichage (E V vecteur d'entier, E MAX : entier)
{
    variable i : entier
    Pour i de 0 à MAX-1
        Afficher ( " V[" , i , "]= " , V[i] )
}

```

```

procédure remplissage (E/S V vecteur d'entier, E MAX : entier)
{
    variable i , n : entier

    initialisation(V, MAX)

    Pour i de 0 à MAX-1
    {
        Pour n de i à MAX-1
        {
            V[n] ← V[n] * n
        }
        affichage(V , MAX)
    }
}

```

```

fonction nbchiffres(E v:entier):entier
{
    variable n : entier
    n ← 0
}

```

```

    val ← v
    Repeter
    {
        n ← n+1
        val ← val div 10
    } jusqu'à (val = 0)
    retourne n
}

```

```

procédure remplace (E/S V vecteur d'entiers, E MAX : entier )
{
    variable i: entier
    Pour i de 0 à MAX-1
    {
        V[i] ← nbchiffres(V[i])
    }
}

```

```

Algorithme vecteur
{
    constante entier MAX ← 9
    variable V: vecteur de MAX entier
    remplissage(V,MAX)
    remplace(V,Max)
    affichage(V, MAX)
}

```