Correction des exercices 4.4

Simplification d'algorithme

```
si ok alors
    afficher nombre
fin si

ok ← condition
```

```
si NON ok alors
    afficher nombre
fin si

ok ← a ≥ b
```

```
si ok1 ET ok2 alors
    afficher x
fin si
```

Exercice 3 – Maximum de 2 nombres

```
module max2Nb()
    nb1, nb2 : entiers
    max : entier
    lire nb1, nb2
    si nb2 ≥ nb1 alors
        max ← nb2
    sinon
        max ← nb1
    fin si
    afficher max
fin module
```

Exercice 4 – Maximum de 3 nombres

```
module max3Nb()
    nb1, nb2, nb3 : entiers
    max : entier
    lire nb1, nb2, nb3
    si nb2 ≥ nb1 alors
        max ← nb2
    sinon
        max ← nb1
    fin si
    si nb3 ≥ max alors
        max ← nb3
    fin si
    afficher max
fin module
```

Exercice 5 – Signe

```
module signe()
    nb : entier
    lire nb
    selon que
        nb > 0 : afficher "positif"
        nb < 0 : afficher "négatif"
        autre : afficher "nul"
    fin selon que
fin module
```

Exercice 6 – La fourchette

```
module fourchette()
    nb1, nb2, nb3, petit, grand : entiers
    ok : booléen
    lire nb1, nb2, nb3
    si nb2 > nb3 alors
        petit ← nb3
        grand ← nb2
    sinon
        petit ← nb2
        grand ← nb3
    fin si
    afficher ← nb1 > petit ET nb1 < grand
fin module
```

Exercice 7 – Équation du second degré

```
module racinesÉquation()
    coeffCarré, coeff, termeIndé : entiers
    delta : entier
    lire coeffCarré, coeff, termeIndé
    delta ← (coeff)^2 - 4 * coeffCarré * termeIndé
    selon que
        delta > 0 : afficher (-coeff ± √delta)/(2 * coeffCarré)
        delta = 0 : afficher -coeff/(2 * coeffCarré)
        autres : afficher "pas de racine"
    fin selon que
fin module
```

Exercice 8 – Une petite minute

```
module plusUneMin()
    heure, minute : entiers
    lire heure, minute
    si minute = 59 alors
        minute ← 0
        heure ← heure + 1
    sinon
        minute ← minute + 1
    fin si
    afficher heure, minute
fin module
```

Exercice 9 – Calcul de salaire

```
module salaireNet()
    salaireBrut : entier
    constante RETENUE : 15
    salaireNet : entier
    lire salaire
    si salaire > 1200 alors
        salaireNet ← salaire - (salaire * RETENUE) / 100
        afficher salaireNet
    sinon
        afficher salaireBrut
    fin si
fin module
```

Exercice 10 – Nombres de jours dans un mois

```
module nbJours()
    mois : chaine
    jours : entier
    lire mois
    selon que mois vaut
     "JANVIER", "MARS", "MAI", "JUILLET", "AOÛT", "OCTOBRE", "DÉCEMBRE":
        afficher 31
     "AVRIL", "JUIN", "SEPTEMBRE", "NOVEMBRE":
        afficher 30
     "FÉVRIER":
        afficher 28
    fin selon que
fin module
```

Exercice 11 – Année bissextile

```
module estBissextile()
    annee : entier
    afficher annee
    afficher annee MOD 4 = 0 ET NON(annee MOD 100 = 0) OU annee MOD 400 = 0
fin module
```

Exercice 12 – Valider une date

```
module dateValide()
    annee, mois, jour, jourMois : entiers
    bissextile : booléen
    lire jour, mois, annee
    bissextile ← annee MOD 4 = 0 ET annee MOD 100 <> 0 OU annee MOD 400 = 0
    selon que mois vaut
     1, 3, 5, 7, 8, 10, 12:
        jourMois ← 31
     4, 6, 9, 11:
        jourMois ← 30
     2:
        si bissextile alors
            jourMois ← 29
        sinon
            jourMois ← 28
        fin si
     autres :
        afficher "mois inconnu"
    fin selon que
    afficher 1 ≤ jour ≤ jourMois
fin module
```

Exercice 13 – Le jour de la semaine

```
module jourSemaine()
    dateMois : entier
    lire dateMois
    selon que dateMois MOD 7 vaut
        0 : afficher "vendredi"
        1 : afficher "samedi"
        2 : afficher "dimanche"
        3 : afficher "lundi"
        4 : afficher "mardi"
        5 : afficher "mercredi"
        6 : afficher "jeudi"
    fin selon que
fin module
```

Exercice 14 – Quel jour serons-nous ?

```
module jourFutur()
    jour : chaine
    n, jourFutur : entier
    lire jour, n
    selon que jour vaut
        "lundi" : jourFutur ← 1
        "mardi" : jourFutur ← 2
        "mercredi" : jourFutur ← 3
        "jeudi" : jourFutur ← 4
        "vendredi" : jourFutur ← 5
        "samedi" : jourFutur ← 6
        "dimanche" : jourFutur ← 7
    fin selon que
    selon que (jourFutur + n) MOD 7 vaut
        0 : afficher "lundi"
        1 : afficher "mardi"
        2 : afficher "mercredi"
        3 : afficher "jeudi"
        4 : afficher "vendredi"
        5 : afficher "samedi"
        6 : afficher "dimanche"
    fin selon que
fin module
```

Exercice 15 – Un peu de trigono

```
module cosinus()
    n : entier
    cosinus : entier
    lire n
    si NON(n MOD 2 = 0) alors
        cosinus ← 0
    sinon
        si (n/2) MOD 2 = 0 alors
            cosinus ← 1
        sinon
            cosinus ← -1
        fin si
    fin si
    afficher cosinus
fin module
```

Exercice 16 – Le stationnement alternatif

```
module bienStationné()
    dateJour, numMaison : entiers
    si 1 ≥ dateJour ≥ 15 alors
        afficher NON(numMaison MOD 2 = 0)
    sinon
        afficher numMaison MOD 2 = 0
    fin si
fin module
```