Haute École Bruxelles-Brabant
École Supérieure d'Informatique
Rue Royale, 67. 1000 Bruxelles
02/219.15.46 – esi@he2b.be

Algorithmique (Solutions des exercices)

2019

Bachelor en Informatique
DEV2

M. Codutti (MCD), H. Delannoy (HDE),
S. Drobisz (SDR), A. Paquot (APA) & N. Richard (NRI)

Document produit avec LaTeX.
Version du 12 février 2019.

Table des matières

Chapitre

1

Les tableaux à 2 dimensions

Solution de l'exercice 1.

```
algorithm estNul(tab: array of n × m integers, lg, col: integers) → boolean
|  return tab[lg][col]=0
end
```

Solution de l'exercice 2.

```
algorithm assigner(tab ↕ : array of n × m integers, lg ↓ , col ↓ , val ↓ : integers)
|  if estNul(tab, lg, col)
|  |  tab[lg,col] = val
|  end
end
```

Solution de l'exercice 3.

```
algorithm estBordHaut(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return lg = 0
end
algorithm estBordBas(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return lg = n - 1
end
algorithm estBordGauche(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return col = 0
end
algorithm estBordDroit(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return col = m - 1
end
algorithm estBord(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estBordGauche(tab,lg,col) OU estBordDroit(tab,lg,col)
|     OU estBordHaut(tab,lg,col) OU estBordBas(tab,lg,col)
end
```

Solution de l'exercice 4.

```
algorithm estCoinHG(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estBordGauche(tab, lg, col) ET estBordHaut(tab, lg, col)
end
algorithm estCoinHD(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estBordDroit(tab, lg, col) ET estBordHaut(tab, lg, col)
end
algorithm estCoinBG(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estBordGauche(tab, lg, col) ET estBordBas(tab, lg, col)
end
algorithm estCoinBD(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estBordDroit(tab, lg, col) ET estBordBas(tab, lg, col)
end
algorithm estCoin(tab: array of n×m entiers, lg, col: entiers) → booléen
|  return estCoinHG(tab,lg,col) OU estCoinHD(tab,lg,col)
|     OU estCoinBG(tab,lg,col) OU estCoinBD(tab,lg,col)
end
```

Solution de l'exercice 5.

```
algorithm afficherLigneParLigne(tab: array of n × m T)
|  for i from 0 to n-1
|  |  for j from 0 to m-1
|  |  |  print tab[i,j]
|  |  end
|  end
end
algorithm afficherColonneParColonne(tab: array of n × m T)
|  for j from 0 to m-1
|  |  for i from 0 to n-1
|  |  |  print tab[i,j]
|  |  end
|  end
end
```

Solution de l'exercice 6.

```
algorithm afficherCasesAdjacentes(tab: array of n×m entiers, lg, col: entiers)
|  if NON estBordGauche(tab, lg, col)  print lg, col-1
|  if NON estBordDroit(tab, lg, col)  print lg, col+1
|  if NON estBordHaut(tab, lg, col)  print lg - 1, col
|  if NON estBordBas(tab, lg, col)  print lg + 1, col
end
```

Solution de l'exercice 7.

```
algorithm proportionNuls(tab: array of n × m entiers) → réel
    nbNuls: entier
    nbNuls = 0
    for i from 0 to n-1
        for j from 0 to m-1
            if tab[i,j]=0
                nbNuls++
            end
        end
    end
    return nbNuls/(n×m)
end
```

Solution de l'exercice 8.

```
algorithm sommeLigne(tab: array of n × m entiers,lg: entier) → réel
    somme: entier
    somme = 0
    for j from 0 to m-1
        somme = somme + tab[lg,j]
    end
    return somme ;
end
algorithm moyenneLigne(tab: array of n × m entiers,lg: entier) → réel
    return sommeLigne(tab)/m
end
algorithm pourcentageRéussites(notes: array of n × m entiers) → réel
    nbRéussites: entier
    nbRéussites = 0
    for i from 0 to n-1
        if moyenneLigne(notes,i)>=10
            nbRéussites++ ;
        end
    end
    return nbRéussites/n × 100
end
```

Solution de l'exercice 9.

```
algorithm trianglePascal(n: entier) → array of n × n entiers
    pascal: array of n × n entiers
    for i from 0 to n-1
        pascal[i,0] = 1
        pascal[i,i] = 1
        for j from 1 to i-1
            pascal[i,j] = pascal[i-1,j-1] + pascal[i-1,j]
        end
    end
    return pascal ;
end
```

Solution de l'exercice 15.

```
algorithm pinceauZebre(tab↕ : tableau de n x n entiers)
    colDepart= 0
    for lg from 0 to n-1
        for col from colDepart to n-1 by 3
            tab= NOIR
        end
        if colDepart > 0
            colDepart= colDepart - 1
        else
            colDepart= 2
        end
    end
end
algorithm pinceauSpirale(tab↕ : tableau de n x n entiers)
    lg= 0
    col= 0
    dirLg= 0
    dirCol= 1
    fini= faux
    while NON fini
        tab= NOIR
        if bord(lg,col,dirLg, dirCol) OU noircieDansDeuxCase(tab, lg,col,dirLg,dirCol)
            tournerADroite(dirLg, dirCol)
        else
        end
        avancer(lg, col, dirLg, dirCol)
        if caseNoireADroite(tab,lg,col,dirLg, dirCol)
            fini= vrai
        else
        end
    end
end
algorithm bord(lg, col, dirLg, dirCol: entiers) → booléen
    tmpLg= lg
    tmpCol= col
    avancer(tmpLg, tmpCol, dirLg, dirCol)
    return 0 <= tmpCol ET tmpCol < n ET 0 <= tmpLg ET tmpLg < n
end
algorithm noircieDansDeuxCase(tab, lg, col, dirLg, dirCol: entiers) → booléen
    tmpLg= lg
    tmpCol= col
    avancer(tmpLg, tmpCol, dirLg, dirCol)
    return tab[tmpLg, tmpCol] = NOIR
end
```

```
algorithm tournerADroite(dirLg↕ : entier, dirCol↕ : entier)
    dirLg= dirCol
    dirCol= -dirLg
end
// Tests : tournerADroite(0, 1) = (1, 0)
// tournerADroite(1, 0) = (0, -1)
// tournerADroite(0, -1)= (-1, 0)
// tournerADroite(-1, 0)= (0, 1)
algorithm avancer(lg↕ : entier, col↕ : entier, dirLg, dirCol: entiers)
    lg= lg + dirLg
    col= col + dirCol
end
algorithm caseNoireADroite(tab, lg, col, dirLg, dirCol: entiers) → booléen
    tmpLg= lg
    tmpCol= col
    tmpDirLg= dirLg
    tmpDirCol= dirCol
    tournerADroite(tmpDirLg, tmpDirCol)
    if NON bord(lg,col, tmpDirLg, tmpDirCol)
        avancer(tmpLg, tmpCol, dirLg, dirCol)
        return tab[tmpLg, tmpCol] = NOIR
    else
        return faux
    end
end
```

Chapitre

2

L'orienté objet

Chapitre

3

La liste

Chapitre

4

Les traitements de rupture

Solution de l'exercice 1.

```
structure Date
    année, mois, jour: entiers
end
structure Job
    login: chaîne
    date: Date
    nombre: entier
end
algorithm stopGaspi(jobs: Liste de Job, limitePrn: entier)
    // jobs est triée en majeur sur le login
    i: entier
    cptPrn: entier
    saveLogin: chaîne
    while i < jobs.taille()
        cptPrn= 0
        saveLogin= jobs.get(i).login
        while i < jobs.taille() ET jobs.get(i).login = saveLogin
            cptPrn= cptPrn + jobs.get(i).nombre
            i= i + 1
        end
        if cptPrn > limitePrn
            print "Alerte : " + saveLogin + " " + cptPrn
        end
    end
end
```

Chapitre

5

Représentation des données