



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 L^AT_EX.
Version du 12 février 2019.

Table des matières

1	Les tableaux à 2 dimensions	3
2	L'orienté objet	9
3	La liste	11
4	Les traitements de rupture	13
5	Représentation des données	17



Les tableaux à 2 dimensions

Solution de l'exercice 1.

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

Solution de l'exercice 2.

```
algorithm assigner(tab ↑: array of  $n \times m$  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  $n \times m$  entiers, lg, col: entiers) → booléen
|   return lg = 0
end

algorithm estBordBas(tab: array of  $n \times m$  entiers, lg, col: entiers) → booléen
|   return lg = n - 1
end

algorithm estBordGauche(tab: array of  $n \times m$  entiers, lg, col: entiers) → booléen
|   return col = 0
end

algorithm estBordDroit(tab: array of  $n \times m$  entiers, lg, col: entiers) → booléen
|   return col = m - 1
end

algorithm estBord(tab: array of  $n \times m$  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  $n \times m$  entiers, lg, col: entiers)  $\rightarrow$  booléen
| return estBordGauche(tab, lg, col) ET estBordHaut(tab, lg, col)
end

algorithm estCoinHD(tab: array of  $n \times m$  entiers, lg, col: entiers)  $\rightarrow$  booléen
| return estBordDroit(tab, lg, col) ET estBordHaut(tab, lg, col)
end

algorithm estCoinBG(tab: array of  $n \times m$  entiers, lg, col: entiers)  $\rightarrow$  booléen
| return estBordGauche(tab, lg, col) ET estBordBas(tab, lg, col)
end

algorithm estCoinBD(tab: array of  $n \times m$  entiers, lg, col: entiers)  $\rightarrow$  booléen
| return estBordDroit(tab, lg, col) ET estBordBas(tab, lg, col)
end

algorithm estCoin(tab: array of  $n \times m$  entiers, lg, col: entiers)  $\rightarrow$  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  $n \times m$  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  $n \times m$  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  $n \times m$  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  $n \times m$  entiers)  $\rightarrow$  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  $\frac{nbNuls}{n \times m}$ 
end
```

Solution de l'exercice 8.

```
algorithm sommeLigne(tab: array of  $n \times m$  entiers,lg: entier)  $\rightarrow$  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  $n \times m$  entiers,lg: entier)  $\rightarrow$  réel
| return  $\frac{sommeLigne(tab)}{m}$ 
end

algorithm pourcentageRéussites(notes: array of  $n \times m$  entiers)  $\rightarrow$  réel
| nbRéussites: entier
| nbRéussites = 0
| for i from 0 to n-1
|   if moyenneLigne(notes,i)>=10
|     nbRéussites++;
|   end
| end
| return  $\frac{nbRéussites}{n} \times 100$ 
end
```

Solution de l'exercice 9.

```
algorithm trianglePascal(n: entier)  $\rightarrow$  array of  $n \times n$  entiers
| pascal: array of  $n \times n$  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
            avancer(lg, col, dirLg, dirCol)
            if caseNoireADroite(tab,lg,col,dirLg, dirCol)
                fini = vrai
            else
            end
        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( $dirLg \uparrow : entier, dirCol \uparrow : entier$ )
|   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( $lg \uparrow : entier, col \uparrow : entier, dirLg, dirCol: entiers$ )
|   lg = lg + dirLg
|   col = col + dirCol
end
algorithm caseNoireADroite( $tab, lg, col, dirLg, dirCol: entiers$ )  $\rightarrow booléen$ 
|   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

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
```

Solution de l'exercice 2.

```
algorithm RuptureNiveau2(etudiants: liste d' Etudiant)
    // on suppose les données classées en majeur sur l'option
    // et en mineur sur la date de naissance (ordre chronologique)
    etd: Etudiant
    saveOption: chaîne
    saveAnnéeNaissance: entier
    cpt, cptOpt: entier // ICI
    i: entier
    i= 0
    while i < étudiants.taille()
        saveOption= etd.option
        cptOpt= 0 // ICI
        while i < étudiants.taille() ET étudiants.get(i).option = saveOption
            saveAnnéeNaissance= étudiants.get(i).dateNaissance.année
            cpt= 0
            while i < étudiants.taille() ET étudiants.get(i).option = saveOption ET
                étudiants.get(i).dateNaissance.année = saveAnnéeNaissance
                cpt= cpt + 1
                cptOpt= cptOpt + 1 // ICI
                i= i + 1
            end
            print cpt, " étudiant(s) dans l'option ", saveOption, " est(sont) né(s) en ",
                saveAnneeNaissance
        end
        print cptOpt, " étudiant(s) dans l'option ", saveOption // ICI
    end
end
```

Solution de l'exercice 3.

```
algorithm afficherComptageEtudiants(etudiants: liste d' Etudiants)
    saveOption, saveBloc: chaine
    cptEtudiantOption, cptEtudiantBloc, i: entier
    i= 0
    while i < etudiants.taille()
        saveOption= etudiants.get(i).option
        // initialisation pré-rupture 1
        cptEtudiantOption= 0
        print saveOption
        while i < etudiants.taille() ET (etudiants.get(i)).option = saveOption
            saveBloc= etudiants.get(i).bloc
            // initialisation pré-rupture 2
            cptEtudiantBloc= 0
            while i < etudiants.taille() ET etudiants.get(i).option = saveOption ET
                etudiants.get(i).bloc = saveBloc
                // traitement des éléments de la liste
                cptEtudiantBloc= cptEtudiantBloc + 1
                i= i + 1
            end
            // traitement post-rupture 2
            cptEtudiantOption= cptEtudiantOption + cptEtudiantBloc
            print " bloc " + saveBloc + " : " + cptEtudiantBloc + " étudiants"
        end
        // traitement post-rupture 1
        print " TOTAL : " + saveOption + " étudiants"
    end
end
```

Solution de l'exercice 5.

```
algorithm club(membres: Liste de Participant)
  n: entier
  mineur: booléen
  saveNom, saveRef: chaîne
  accumRésultat, nbResultat: entier
  nbMineursTotal: entier
  n= membres.taille()
  i= 0
  while i < n
    saveRef= membres.get(i).reference
    nbMineursTotal= 0
    print "Participations de mineurs du club " + saveRef
    while i < n ET saveRef = membres.get(i).reference
      saveNom= membres.get(i).nom
      accumRésultat= 0
      nbResultat= 0
      mineur= membres.get(i).age < 18
      while i < n ET saveNom = membres.get(i).nom
        accumRésultat= accumRésultat + membres.get(i).resultat
        nbResultat= nbResultat + 1
        i= i + 1
      end
      if mineur
        nbMineursTotal= nbMineursTotal + 1
        print saveNom + accumRésultat/nbResultat
      end
    end
    if nbMineursTotal > 0
      print "Nombre total de membres mineurs de ce club : " + nbMineursTotal
    else
      print "Pas de participations de mineurs pour ce club"
    end
  end
end
```


Solution de l'exercice 13.

Solution incomplète

```
// tab est un tableau de caractères
// chaque caractère est soit :
// - une lettre (la lettre)
// - un point (case noircie)
// - une espace (case libre)
algorithm placerLettre(i, j: entiers, lettre: caractère, tab: tableau de m x n caractères)
|   if NON estCaseOccupée()
|   |   tab = lettre
|   else
|   |   error "Case déjà occupée"
|   end
end
algorithm estCaseOccupée(i, j: entiers, tab: tableau de m x n caractères) → booléen
|   return tab[i,j] != ' '
end
```