Algorithmique - Correction du TD2

IUT 1ère Année

5 octobre 2012

1 Les tests

Exercice 1. Construire un arbre de décision et l'algorithme correspondant permettant de déterminer la catégorie sportive d'un enfant selon son âge :

- poussin de 6 à 7 ans
- pupille de 8 à 9 ans
- minime de 10 à 11 ans
- cadet de 12 à 14 ans

Algorithme 1: categorieEnfant

```
variables
    entier age

début
    lire age
    si (age < 6) ou (age > 14) alors
        afficher "hors intervalle"
    sinon
        si age < 8 alors
            afficher "poussin"
        sinon
            si age < 10 alors
                afficher "pupille"
            sinon
                si age < 12 alors
                    afficher "minime"
                sinon
                    afficher "cadet"
                fin
            fin
        fin
    fin
fin
```

Exercice 2. Construire un arbre de décision et l'algorithme correspondant permettant de lire une note, de vérifier si cette note est bien entre 0 et 20, et de déterminer la mention associée à cette note :

- insuffisant en dessous de 10
- passable de 10 à 11
- assez bien de 12 à 13
- bien de 14 à 15
- très bien de 16 à 20

Algorithme 2: mentionNote

```
variables
    entier note

début
    lire note
    si (note < 0) ou (note > 20) alors
        afficher "hors intervalle"
    sinon
        si note < 10 alors
            afficher "insuffisant"
        sinon
            si note < 12 alors
                afficher "passable"
            sinon
                si note < 14 alors
                    afficher "assez bien"
                sinon
                    si note < 16 alors
                        afficher "bien"
                    sinon
                        afficher "très bien"
                    fin
                fin
            fin
        fin
    fin
fin
```

Exercice 3. Construire un algorithme permettant de résoudre le problème suivant :
- Données : les coefficients réels a, b et c d'une équation du second degré $ax^2 + bx + c = 0$,
- Résultat : le nombre de solutions de l'équation.

Algorithme 3: nbSolutionsEquationSecondDegré

```
variables
    réel a, b, c, Δ

début
    lire a
    lire b
    lire c
    Δ ← (b × b) − (4 × a × c)
    si Δ > 0 alors
        afficher "deux solutions"
    sinon
        si Δ = 0 alors
            afficher "une solution"
        sinon
            afficher "zero solution"
        fin
    fin
fin
```

Exercice 4. Construire un algorithme permettant de résoudre le problème suivant :
- Données : une série de trois entiers a, b et c donnés par l'utilisateur
- Résultat : "vrai" si $a \leq b \leq c$ et "faux" sinon

Algorithme 4: sontRangésParOrdreCroissant

```
variables
    entier a, b, c
    booléen rangés

début
    lire a
    lire b
    lire c
    rangés ← (a ≤ b) et (b ≤ c)
    afficher rangés
fin
```

Exercice 5. Construire un algorithme permettant de résoudre le problème suivant :

- Données : une série de trois entiers a, b et c donnés par l'utilisateur
- Résultat : une permutation $\langle a', b', c' \rangle$ de $\langle a, b, c \rangle$ telle que $a' \leq b' \leq c'$

Par exemple, si l'algorithme lit la série $\langle 50, 100, 10 \rangle$ il affichera $\langle 10, 50, 100 \rangle$

Algorithme 5: rangeParOrdreCroissant

```
variables
    entier a, b, c, t

début
    lire a
    lire b
    lire c
    si a > b alors
        t ← a
        a ← b
        b ← t
    fin
    si a > c alors
        t ← a
        a ← c
        c ← t
    fin
    si b > c alors
        t ← b
        b ← c
        c ← t
    fin
    afficher a, b, c
fin
```

Exercice 6. Construire un algorithme permettant de simuler une calculette : l'algorithme lit en entrée deux nombres réels et un opérateur arithmétique, et affiche en sortie le calcul de l'opération. Les opérateurs sont +, −, ∗ et /.

Algorithme 6: calculette

```
variables
    réel x, y, z
    caractère op ;

début
    lire x
    lire y
    lire op
    suivant op faire
        cas où '+' :
            z ← x + y
        fin
        cas où '-' :
            z ← x − y
        fin
        cas où '*' :
            z ← x × y
        fin
        cas où '/' :
            z ← x / y
        fin
    fin
    afficher z
fin
```

Exercice 7. Construire un algorithme permettant de convertir des températures : l'algorithme lit au départ un réel (la température), une unité d'entrée et une unité de sortie. Il doit produire la conversion correspondante. Les unités possibles sont C pour degré Celcius, F pour degré Fahrenheit, et K pour Kelvin. La correspondance entre ces unités est donnée par le système d'équations suivant.

$$T_c = (T_f - 32) * \frac{5}{9} = T_k - 273.15$$

où T_c (resp. T_f, T_k) est la température en degrés Celcius (resp. degrés Fahrenheit, Kelvins).

Algorithme 7: convertitTempératures

```
variables
    // Températures d'entrée et de sortie
    réel T_e, T_s
    // Unités d'entrée et de sortie
    caractère U_e, U_s ;

début
    lire T_e
    lire U_e
    lire U_s
    si U_e = U_s alors
        T_s ← T_e
    sinon
        suivant U_e faire
            cas où 'C' :
                si U_s = 'F' alors
                    T_s ← (9 × T_e/5) + 32
                sinon
                    T_s ← T_e + 273.15
                fin
            fin
            cas où 'F' :
                si U_s = 'C' alors
                    T_s ← (T_e − 32) × 5/9
                sinon
                    T_s ← ((T_e − 32) × 5/9) + 273.15
                fin
            fin
            cas où 'K' :
                si U_s = 'C' alors
                    T_s ← T_e − 273.15
                sinon
                    T_s ← ((T_e − 273.15) × 9/5) + 32
                fin
            fin
        fin
    fin
    afficher T_s
fin
```

2 Les boucles

Exercice 8. Construire un algorithme permettant de résoudre le problème suivant :

- Données : un entier k (la taille de la séquence), une séquence de k entiers $\langle x_1, x_2, \ldots, x_k \rangle$
- Résultat : la moyenne $\frac{1}{k}\sum_{i=1}^{k} x_i$ de la séquence

Algorithme 8: moyenneSéquence

```
variables
    entier i, k, x
    réel somme, moyenne

début
    lire k
    somme ← 0
    pour i ← 1 à k faire
        lire x
        somme ← somme + x
    fin
    moyenne ← somme / k
    afficher moyenne
fin
```

Exercice 9. Construire un algorithme permettant de résoudre le problème suivant :
- Données : un entier k (la taille de la séquence), une séquence de k entiers $\langle x_1, x_2, \ldots, x_k \rangle$
- Résultat : le maximum $\max_{i=1}^{k}(x_i)$ de la séquence

Algorithme 9: maximumSéquenceBornée

```
variables
    entier i, k, x, max

début
    lire k
    max ← 0
    pour i ← 1 à k faire
        lire x
        si x > max alors
            max ← x
        fin
    fin
    afficher max
fin
```

Exercice 10. Construire un algorithme permettant de résoudre le problème suivant :
- Données : une séquence contenant un nombre arbitraire d'entiers strictement positifs, et terminée par 0 : $\langle x_1, x_2, \cdots, 0 \rangle$.
- Résultat : le maximum $\max_i(x_i)$ de la séquence

Algorithme 10: maximumSéquenceNonBornée

```
variables
    entier x, max

début
    max ← 0
    répéter
        lire x
        si x > max alors
            max ← x
        fin
    jusqu'à x = 0
    afficher max
fin
```

Exercice 11. Construire un algorithme permettant de résoudre le problème suivant :
- Données : un entier n
- Résultat : sa factorielle $n! = n(n-1)(n-2)\cdots 1$

Algorithme 11: factorielle

```
variables
    entier i, n, fact

début
    lire n
    // En démarrant par 1 on traite le cas où 0! = 1
    fact ← 1
    pour i ← 1 à n faire
        fact ← fact × i
    fin
    afficher fact
fin
```

Exercice 12. Construire un algorithme permettant de simuler une caisse automatique distribuant la monnaie :
- Données : une quantité n euros que demande l'utilisateur
- Résultat : la monnaie de n en billets de 100, de 50, de 10, de 5 euros, ainsi qu'en pièces de 2 et 1 euros.

La correspondance est donnée naturellement par :

$$n = 100b_{100} + 50b_{50} + 10b_{10} + 5b_5 + 2p_2 + 1p_1$$

où b_i est la quantité de billets de i euros, et p_j est la quantité de pièces de j euros.

Algorithme 12: caisseAutomatique

```
variables
    entier b100, b50, b10, b5, p2, p1, n, reste

début
    lire n
    b100 ← n/100
    reste ← n mod 100
    b50 ← reste /50
    reste ← reste mod 50
    b10 ← reste /10
    reste ← reste mod 10
    b5 ← reste /5
    reste ← reste mod 5
    p2 ← reste /2
    p1 ← reste mod 2
    afficher "Billets de 100 : ", b100
    afficher "Billets de 50 : ", b50
    afficher "Billets de 10 : ", b10
    afficher "Billets de 5 : ", b5
    afficher "Pièces de 2 : ", p2
    afficher "Pièces de 1 : ", p1
fin
```

Note : nous n'avons pas toujours besoin de boucles pour résoudre un problème !

Exercice 13 (*) Construire un algorithme permettant d'associer à un nombre entre 0 et 365, le mois et le jour qui lui correspondent dans l'année. Nous supposerons que l'année n'est pas bissextile. Rappelons que :
- Le mois de février fait 28 jours,
- Les mois d'avril, juin, septembre et novembre font 30 jours,
- Les autres mois font 31 jours

Par exemple, le nombre 60 correspond au premier jour du troisième mois (mars).

Algorithme 13: jourEtMoisDeLAnnée

```
variables
    entier jours, jourDuMois, mois, somme

début
    lire jours
    somme ← 0
    mois ← 0
    répéter
        jourDuMois ← jours - somme
        mois ← mois + 1
        si mois = 2 alors
            somme ← somme + 28
        sinon
            si (mois = 4) ou (mois = 6) ou (mois = 9) ou (mois = 11) alors
                somme ← somme + 30
            sinon
                somme ← somme + 31
            fin
        fin
    jusqu'à jours ≤ somme
    Afficher "Mois de l'année : ", mois
    Afficher "Jour du mois : ", jourDuMois
fin
```

Note : si nous voulons absolument afficher la chaîne de caractères correspondant au mois, alors il faut tester douze cas possibles (ou plus simplement utiliser un tableau de chaînes comme nous le verrons dans la suite).

Exercice 14 (*) Construire un algorithme permettant de calculer le plus grand commun diviseur (PGCD) de deux entiers naturels x et y. Rappelons que :

(1) $\text{PGCD}(x, x) = x$
(2) $\text{PGCD}(x, y) = \text{PGCD}(y, x)$
(3) $\text{PGCD}(x, y) = \text{PGCD}(x - y, x)$ si $x > y$

Par exemple, le PGCD de 60 et 40 est 20.

Algorithme 14: PGCD

```
variables
    entier x, y, t

début
    lire x
    lire y
    répéter
        si x > y alors
            // On applique la règle 3
            x ← x − y
        sinon
            // On applique la règle 2 en permutant les variables
            t ← x
            x ← y
            y ← t
        fin
    jusqu'à x = y
    // On applique la règle 1
    Afficher x
fin
```

Note : il s'agit de l'algorithme d'Euclide.

Exercice 15 (*) Construire un algorithme permettant de convertir un entier naturel n en base 2. Rappelons que :

$$n = \sum_{i=0}^{\lfloor \log_2 x \rfloor} a_i 2^i$$

où a_i est le $i$ème chiffre booléen dans la conversion binaire de n.

Algorithme 15: conversionBinaire

```
variables
    entier n, max, val

début
    lire n
    // Le nombre de chiffres de la conversion sera égal à max + 1
    max ← log2(n)
    pour j ← 0 à max faire
        // On calcule le ième chiffre
        i ← max - j
        // On stocke la puissance de 2 correspondant au ième chiffre
        val ← 2^i
        si n ≥ val alors
            // Le ième chiffre est à 1 ; on continue alors avec le reste
            afficher "1"
            n ← n − val
        sinon
            // Le ième chiffre est à 0 ; on garde le nombre courant
            afficher "0"
        fin
    fin
fin
```

Note : cet algorithme peut se généraliser facilement à n'importe quelle base. Concernant la conversion binaire, il existe d'autres algorithmes (ex : lire à l'envers le résultat des divisions par 2, ou utiliser les opérateurs de rotation de bit en C)