Algorithmique – Travaux Dirigés

Master *"Technologie et Handicap"* : Intensifs d'Automne

Exercice 1 – Affectations

1. Considérons les algorithmes ci-dessous.
   (a) Quel sera le contenu des variables a, b et éventuellement c après leur exécution ?
   (b) Dans chacun des cas ci-dessus, y a-t-il des lignes inutiles, et si oui lesquelles ?

```
// algo 1.1
var a, b, c : entier
debut
a ← 1
b ← a + 2
c ← b - 3
fin
```

```
// algo 1.2
var a, b : entier
debut
a ← 1
b ← a + 2
a ← 3
fin
```

```
// algo 1.3
var a, b : entier
debut
a ← 2
b ← a + 2
a ← a + 2
b ← a - 2
fin
```

```
// algo 1.4
var a, b, c : entier
debut
a ← 2
b ← 4
c ← a + b
a ← 1
c ← b - a
fin
```

```
// algo 1.5
var a, b, c : entier
debut
a ← 1
b ← 2
b ← a + b
c ← a + b
fin
```

```
// algo 1.6
var a, b : car
debut
a ← '1'
b ← '2'
a ← a + b
fin
```

2. Considérons l'algorithme ci-contre
   (a) Permet-il de permuter les valeurs des valeurs des variables a et b ?
   (b) Proposer des solutions pour permuter les valeurs de 2 variables numériques ? Chacune des solutions proposées marche-t-elle dans le cas de variables non numériques.
   (c) Étant données 3 variables a, b et c, proposer un algorithme pour les permuter circulairement, en transférant les valeurs initiales de a à b, de b à c, et de c à a.

```
var a, b : entier
debut
a ← 1
b ← 2
a ← b
b ← a
fin
```

Exercice 2

1. Quels sont les affichages provoqués par les algorithmes ci-contre.
2. Écrire un algorithme qui demande un entier à l'utilisateur, puis affiche son carré.

```
// algo 2.1
var a, b : reel
var c : entier
debut
a ← 4.21
b ← a * 2
ecrire a, b
ecrire b * 2
c ← b * 2
ecrire c
fin
```

```
// algo 2.2
var a, b : entier
var c : reel
debut
a ← 5
b ← 2
ecrire a, b
ecrire a / b
c ← a / b
ecrire c
fin
```

Exercice 3 – Conditionnelles

1. Écrire un algorithme qui demande un entier à l'utilisateur, teste si ce nombre est positif (≥ 0) ou non, et affiche "`positif`" ou "`négatif`".
2. Écrire un algorithme qui demande un entier à l'utilisateur, teste si ce nombre est strictement positif, nul ou strictement négatif, et affiche ce résultat.
3. Écrire un algorithme qui demande un réel à l'utilisateur et affiche sa valeur absolue (sans utiliser de fonction prédéfinie évidemment).
4. Écrire un algorithme qui demande un réel à l'utilisateur et l'arrondit à l'entier le plus proche (les x,5 seront arrondis à l'entier supérieur).
5. Écrire un algorithme qui demande le numéro d'un mois et affiche le nombre jours que comporte ce mois (sans tenir compte des années bissextiles).
6. Écrire un algorithme qui vérifie si une année est bissextile. On rappelle qu'il y a des années bissextiles tous les 4 ans, mais la première année d'un siècle ne l'est pas (1800, 1900 n'étaient pas bissextiles) sauf tous les 400 ans (2000 était une année bissextile).
7. Écrire un algorithme qui demande une date sous la forme de 2 nombres entiers (numéro du jour et numéro du mois) et affiche la saison (ex : 12/02 $\rightsquigarrow$ hiver). On supposera que le premier jour de la saison est toujours le 21.
8. Écrire un programme qui demande les coordonnées (x, y) des sommets A, B et C d'un triangle et affiche la nature du triangle (isocèle, équilatéral, rectangle ou quelconque).

Exercice 4 – Itérations

1. Écrire un algorithme qui demande un entier positif, et le rejette tant que le nombre saisi n'est pas conforme.
2. Écrire un algorithme qui demande 10 entiers, compte le nombre d'entiers positifs saisis, et affiche ce résultat.
3. Écrire un algorithme qui demande des entiers positifs à l'utilisateur, les additionne, et qui s'arrête en affichant le résultat dès qu'un entier négatif est saisi.
4. Modifier ce dernier algorithme pour afficher la moyenne de la série d'entiers positifs saisis.

Exercice 5 – Conversion en binaire

1. Écrire un algorithme de conversion d'un nombre entier en binaire.
2. *(Facultatif)* Écrire un algorithme de conversion d'un nombre entier en une base b quelconque.

Exercice 6 – Suites

1. Écrire un algorithme pour afficher les n premiers termes des suites suivantes (n demandé à l'utilisateur) :
   (a) *Suite arithmétique*
   $$\begin{cases} u_0 = 1 \\ u_{n+1} = u_n + 2 \end{cases}$$
   (b) *Suite de Newton*
   $$\begin{cases} u_0 = \frac{a}{2} \\ u_{n+1} = \frac{1}{2}(u_n + \frac{a}{u_n}) \end{cases}$$
   (a réel demandé à l'utilisateur)
   (c) *Suite de Fibonacci*
   $$\begin{cases} u_0 = 0 \\ u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \end{cases}$$
2. *(facultatif)* La suite de Newton converge vers la racine carrée de a. Modifier l'alogorithme (b) pour calculer la racine carrée d'un nombre selon une précision donnée.

Exercice 7 – Devinez un nombre

1. Écrire un algorithme permettant de joueur au jeu du *plus petit-plus grand*. On tire un nombre au hasard pour le faire deviner au joueur en lui disant à chaque tour si le nombre proposé est plus grand ou plus petit que le nombre à chercher. Lorsque le joueur a trouvé, l'algorithme se termine en affichant le nombre de tours.
   Note : On suppose qu'on a une fonction `entier nombreAléatoire(var max : entier)` qui tire un nombre au hasard et le renvoie.
2. Modifier ensuite cet algorithme pour limiter à 10 le nombre de propositions du joueur, et afficher "Perdu !" si le joueur n'a pas trouvé.

Exercice 8 – Boucles imbriquées

1. Échiquiers
   (a) Écrire un algorithme permettant d'écrire un carré de 8 fois 8 caractères 'x'.
   (b) Écrire un algorithme permettant d'écrire un échiquier. On représentera les case noires par des 'x' et les cases blanches par des espaces.
   (c) Modifier l'algorithme précédent pour afficher un cadre autour de l'échiquier, en utilisant les caractères '|' , '−' et '+'.
   (d) Modifier de nouveau cet algorithme pour afficher l'ensemble des cases avec ces mêmes caractères (voir exemple 4 ci-dessous – incomplet).

```
// a                 // b                 // c                 // d
                                           +---------+           +-+-+-+-+ ... +
xxxxxxxx             x x x x             |x x x x |           |x| |x| ... |
xxxxxxxx              x x x x            | x x x x|           +-+-+-+-+ ... +
xxxxxxxx             x x x x             |x x x x |           | |x| | ... |
xxxxxxxx              x x x x            | x x x x|           +-+-+-+-+ ... +
xxxxxxxx             x x x x             |x x x x |           |x| |x| ... |
xxxxxxxx              x x x x            | x x x x|           +-+-+-+-+ ... +
xxxxxxxx             x x x x             |x x x x |            ...             .
xxxxxxxx              x x x x            | x x x x|            ...             .
                                           +---------+           +-+-+-+-+ ... +
```

2. Écrire un algorithme permettant d'écrire une table de multiplication comme celle présentée ci-contre. Dans un premier temps on ne s'occupera pas du nombre d'espaces entre les nombres, puis on affinera en en tenant compte.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
...									
10	20	30	40	50	60	70	80	90	100

Exercice 9 – Tableaux

1. Que font les algorithmes suivants ?

```
1 var i, n[10] entier
2 debut
3 n[0] ← 1
4 pour i allantde 1 a 9
5 faire
6 n[i] ← n[i-1] + 2
7 fin
```

```
1 var i, n[10] entier
2 debut
3 pour i allantde 0 a 9
4 faire
5 n[i] ← 2 * i
6 fin
```

2. Écrire un algorithme qui déclare un tableau de 10 éléments et initialise toutes ses valeurs à 1
3. Écrire un algorithme qui calcule les n premiers nombres premiers.

Exercice 10 – Le triangle de Pascal

Écrire un algorithme permettant de calculer le triangle de Pascal au rang n, dans lequel à la ligne i et à la colonne j ($0 \leq j \leq i$) est placé le coefficient binomial C_j^i.
On le construit aisément par récurrence, en remarquant qu'à chaque ligne i, le coefficient numéro j est la somme des coefficients $j-1$ et j de la ligne précédente (pour $0 < j < i$). Les lignes 1 et 2 sont initialisées à 1 ainsi que les coefficients 1 et i de chaque ligne i.
Autrement dit :

$$\begin{cases} \forall i, j \mid 0 < j < i, C_j^i = C_{j-1}^{i-1} + C_j^{i-1} \\ C_i^0 = C_i^i = 1 \end{cases}$$

1					
1	1				
1	2	1			
1	3	3	1		
1	4	6	4	1	
1	5	10	10	5	1

Exercice 11 – Fonctions numériques

1. Écrire une fonction qui calcule la somme de 3 entiers fournis en arguments
2. Écrire une fonction qui calcule la moyenne des éléments d'un tableau. Le tableau et la taille du tableau sont fournis en argument
3. Écrire une fonction qui affiche la décomposition en facteurs premiers d'un nombre. Indication : le plus petit diviseur strictement supérieur à 1 d'un nombre est nécessairement premier.
4. En utilisant la fonction PGCD vue en cours, écrire une fonction PPCM. On rappelle que $PPCM(a,b) = \frac{|ab|}{PGCD(a,b)}$.

Exercice 12 – Fonctions de traitement de chaînes de caractères

On donne le type chaine pour représenter les chaînes de caractères. On utilisera l'opérateur $+$ pour concaténer des chaînes et on donne les 2 fonctions suivantes :
`fonction taille(var s : chaine par variable)` : renvoie la taille de la chaîne, en entier.
`fonction charAt(var s : chaine par variable, var i : entier par valeur)` : renvoie le caractère numéro i de la chaîne.

1. Écrire une fonction contient qui prend 2 chaînes en paramètres en renvoie vrai si la seconde est incluse dans la première.
2. Écrire une fonction qui purge une chaîne d'un caractère, la chaîne comme le caractère étant passés en argument. Si le caractère spécifié ne fait pas partie de la chaîne, celle-ci devra être retournée intacte. Par exemple :
   — purge("Bonjour","o") renverra "Bnjur".
   — purge("J'aime pas les espaces"," ") renverra "J'aimepaslesespaces".
   — purge("Moi, je m'en fous", "y") renverra "Moi, je m'en fous".
3. Écrire deux fonctions start et end qui prennent 1 chaîne s et un entier n en paramètres et qui renvoient une chaîne contenant respectivement les n premiers caractères et les n derniers caractères de s
4. Écrire une fonction middle qui prend 1 chaîne et 2 entiers p et q en paramètres et qui renvoie la partie de la chaîne comprise entre les caractères p et q. On utilisera les fonctions définies précédemment.

Exercice 13 – Fonctions récursives

1. Écrire un algorithme de PGCD récursif. On rappelle que si $q < p$ alors $PGCD(p,q) = PGCD(p-q,q)$.
2. On reprend la Suite de Fibonacci, qu'on a déjà utilisée dans l'exercice 6 sur les suites.

$$\begin{cases} u_0 = 0, \quad u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \end{cases}$$

   (a) Écrire une fonction récursive de calcul du terme de rang n de cette suite. Qu'en pensez vous ?
   (b) On peut montrer que :

$$\begin{cases} u_{2k} = 2u_{k-1}u_k + u_k^2 = (2u_{k-1} + u_k)u_k \\ u_{2k+1} = u_{k+1}^2 + u_k^2 \end{cases}$$

   En déduire une nouvelle fonction récursive. Est-elle plus efficace ?

Exercice 14 – Problème des 8 reines

Il s'agit de trouver les différentes façons de placer 8 reines sur un échiquier (8x8 cases) sans qu'elles ne se menacent mutuellement, conformément aux règles du jeu d'échecs (on ne tient pas compte de la couleur des pièces).