UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence Informatique
Info0101 – 2018/2019

Travaux dirigés n° 3

Les fonctions

Exercice 1 (Traces d'exécution)

Donnez la trace d'exécution de ces deux algorithmes :

```
Algorithme Exercice1.1
Déclarations
      Variables
            x, y : entier
Début
{m1}  x ← 1
{m2}  y ← 2
{m3}  x ← f(x+y, y)
{m4}  écrire(x + " " + y)
{m5}  y ← f(x+y, y)
{m6}  écrire(x + " " + y)
Fin


Fonction f(u : entier, v : entier) : entier
Déclarations
      Variables locales
            x, resultat : entier
Début
{f1}  x ← 2*u + 1
{f2}  Si x > 4 Alors
{f2a.1}     resultat ← v - u
      Sinon
{f2b.1}     resultat ← v + u
      FinSi
{f3}  retourner(resultat)
Fin
```

```
Algorithme Exercice1.2
Déclarations
      Variables
            i : entier
Début
{m1}  i ← 2
{m2}  écrire("Principal: " + i + " " + g(i))
Fin


Fonction f(i : entier) : entier
Début
{f1}  i ← i + 1
{f2}  écrire("f: " + i)
{f3}  retourner(i)
Fin


Fonction g(j : entier) : entier
Début
{g1}  j ← j + 2
{g2}  écrire("g: " + j)
{g3}  retourner(f(j) + j)
Fin
```

Exercice 2 (Date du lendemain)

On désire écrire un algorithme permettant, après saisie de la date du jour, d'afficher la date du lendemain :

- un message d'erreur doit être affiché si la valeur saisie pour le mois n'est pas dans [1 ; 12]
- un message d'erreur doit être affiché si la valeur saisie pour le jour n'est pas conforme (en tenant compte du mois et de l'année)

Rappel : une année bissextile est une année comportant 366 jours (au lieu de 365). Une année est bissextile lorsqu'elle est soit divisible par 4, mais non divisible par 100, soit divisible par 400.

1°) Écrivez une fonction permettant de déterminer si une année est bissextile.

2°) Écrivez une fonction donnant le nombre de jours d'un mois d'une année déterminée.

3°) Écrivez une fonction permettant de déterminer si une date est valide.

4°) Écrivez, finalement, un algorithme qui, après saisie d'une date et vérification de sa conformité, permet d'obtenir la date du lendemain.

5°) Écrivez une fonction permettant de comparer deux dates. Elle retournera vrai si la première date est inférieure ou égale à la seconde.

Exercice 3 (Nombres premiers)
Nous disposons d'une fonction `estPremier(n : entier) : booléen` qui teste si l'entier n est un nombre premier.

1°) Écrivez une fonction qui compte le nombre de nombres premiers compris dans un intervalle d'entiers positifs fixé à $[a, b]$. Par exemple : `comptePremiers(5,15) == 4`.

2°) Écrivez une fonction qui, pour un entier n donné, calcule le nombre premier de rang n. Par exemple, `premier(1)` retourne 2 car 2 est le premier nombre premier et `premier(5)` retourne 11 car 11 est le cinquième.

3°) Écrivez une fonction qui détermine le rang d'un nombre premier.

4°) Écrivez une fonction qui détermine le plus petit diviseur ($\neq 1$) d'un entier au moins égal à 2.

5°) Déduisez-en une fonction qui teste si un entier est premier.

Exercice 4 (Les classiques)
1°) Soit une suite numérique $(u_i)_{i\in\mathbb{N}}$.

Pour $n \in \mathbb{N}$ nous posons :

$$S_n = \sum_{i=0}^{n} u_i \qquad P_n = \prod_{i=0}^{n} u_i \qquad M_n = \max\{u_i \mid 0 \leq i \leq n\} \qquad m_n = \min\{u_i \mid 0 \leq i \leq n\}$$

Donnez les fonctions permettant de calculer les termes des suites $(S_n)_{n\in\mathbb{N}}$, $(P_n)_{n\in\mathbb{N}}$, $(M_n)_{n\in\mathbb{N}}$ et $(m_n)_{n\in\mathbb{N}}$. Nous supposons disposer de la fonction `u(i : entier) : entier` qui calcule le terme u_i.

2°) Soit $(n, p) \in \mathbb{N} \times \mathbb{N}$ avec $0 \leq p \leq n$, donnez la fonction pour calculer le coefficient binomial.
Rappel : $C_n^p = \begin{pmatrix} n \\ p \end{pmatrix} = \frac{n!}{p!(n-p)!} = \frac{n(n-1)\ldots(n-p+1)}{p!}$

3°) Le triangle de Pascal :

a) Écrivez la procédure `afficheLigne(n : entier)` qui affiche la ligne "n" du Triangle de Pascal. Si l'on passe 3 comme paramètre, l'appel à cette fonction produira l'affichage 1, 3, 3 et 1 (la première ligne est la ligne 0).

b) Écrivez la procédure `affichePascal(n : entier)` qui affiche les "n" premières lignes du Triangle de Pascal.

4°) Soit $(x, n) \in \mathbb{R} \times \mathbb{N}$, donnez une fonction pour calculer $S = \sum_{k=0}^{n} \frac{x^k}{k!}$

Exercice 5 (Syracuse, le retour)
On rappelle que la suite de Syracuse est définie par la donnée de $S_0 \in \mathbb{N}^*$ et par la relation de récurrence :

$$S_{n+1} = \begin{cases} \frac{S_n}{2} \text{ si } S_n \text{ est pair} \\ 3 \times S_n + 1 \text{ si } S_n \text{ est impair} \end{cases}$$

1°) Écrivez une fonction qui calcule la longueur de la suite de Syracuse de premier terme S_0 (passé en paramètre), c'est-à-dire le premier indice n tel que $S_n = 1$.

2°) Écrivez une fonction...

a) ...qui détermine son maximum.

b) ...qui retourne le nombre de termes de S_n supérieurs ou égaux à m.

c) Est-il possible de fusionner les deux précédentes questions (*i.e.* retourner le maximum et le nombre de termes supérieurs ou égaux à m) ?

3°) Écrivez une fonction permettant de déterminer la somme des "m" premiers termes. En utilisant cette fonction, écrivez une fonction permettant de calculer la moyenne des "m" premiers termes.

4°) Calculez l'écart type des "m" premiers termes. Pour rappel, il s'agit de calculer la dispersion des termes autour de leur moyenne. En notant σ l'écart type, X la variable aléatoire et $E[X]$ la moyenne (appelée aussi espérance) : $\sigma = \sqrt{E[X^2] - E[X]^2}$

Annexes - Trace d'exécution sans appel de fonction/procédure

Nous supposons l'algorithme suivant :

```
Algorithme SansFonction
Déclarations
        Variables
                a, b : entier
Début
{1}     a ← 2
{2}     b ← 5
{3}     TantQue a ≠ b Faire
{3.1}           Si a > b Alors
{3.1a.1}                a ← a - b
                Sinon
{3.1b.1}                b ← b - a
                FinSi
        FinTantQue
{4}     écrire("PGCD : " + a)
Fin
```

La trace d'exécution est la suivante :

SansFonction				
Instructions	Contrôle	Variables locales		Écran
		a	b	
Avant		?	?	
1	×	2		
2	×		5	
3	(a!=b) ? oui			
3.1	(a>b) ? non			
3.1b.1	×		3	
3	(a!=b) ? oui			
3.1	(a>b) ? non			
3.1b.1	×		1	
3	(a!=b) ? oui			
3.1	(a>b) ? oui			
3.1a.1	×	1		
3	(a!=b) ? non			
4	→			"PGCD : 1"

Annexes - Trace d'exécution avec appel de fonction/procédure

Nous supposons l'algorithme et la fonction suivants :

```
Algorithme AvecFonction
Déclarations
      Variables
            x, y : entier
Début
{m1}    x ← 5
{m2}    y ← x × x - 1
{m3}    TantQue x < y Faire
{m3.1}        Si y % x ≠ 0 Alors
{m3.1a.1}           x ← x + 3
              Sinon
{m3.1b.1}           y ← f(x)
              FinSi
{m3.2}        écrire(x + "-" + y)
      FinTantQue
Fin
```

```
Fonction f(y : entier) : entier
Déclarations
      Variables locales
            x : entier
Début
{f1}    x ← y / 4
{f2}    TantQue y % 2 ≠ 0 Faire
{f2.1}        y ← y + 7
      FinTantQue
{f3}    x ← 3 × x - y + 3
{f4}    retourner(y + 5 × x)
Fin
```

La trace d'exécution est la suivante :

AvecFonction			
Instructions	Contrôle	Variables locales	
		x	y
Avant		?	?
m1	×	5	
m2	×		24
m3	(x<y) ? oui		
m3.1	(y%x != 0) ? oui		
m3.1a.1	×	8	
m3.2			
m3	(x<y) ? oui		
m3.1	(y%x != 0) ? non		
m3.1b.1			
			13
m3.2			
m3	(x<y) ? oui		
m3.1	(y%x != 0) ? oui		
m3.1a.1	×	11	
m3.2			
m3	(x<y) ? oui		
m3.1	(y%x != 0) ? oui		
m3.1a.1	×	14	
m3.2			
m3	(x<y) ? non		

f(8)

f			
Instructions	Contrôle	Param. formel	Variable locale
		y	x
Avant		8	?
f1	×		2
f2	(y%2 != 0) ? non		
f3	×		1
f4			

13

Écran

"8–24"

"8–13"

"11–13"

"14–13"