POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2011/2012**

http://laure.gonnord.org/pro/teaching/

Quick - 1h - Sujet commun aux trois groupes

Calculatrices, téléphones et documents interdits.

Traductrices autorisées pour les étudiants non francophones.

Instructions

Les algorithmes seront **tous** écrits en pseudo-langage algorithmique. On veillera à décrire calmement ce que l'on fait, et à bien indenter les pseudo-codes.

Exercice 1 Écrire un programme complet qui demande un entier $n > 2$ à l'utilisateur et imprime tous les entiers **pairs** positifs (> 0) et strictement inférieurs à n ($< n$).

Exercice 2 Écrire un programme complet qui demande des entiers positifs ou nuls (≥ 0) à l'utilisateur, s'arrête quand l'utilisateur entre un entier négatif, et imprime à la fin la somme des entiers positifs entrés par l'utilisateur.

Exercice 3 L'algorithme suivant est-il récursif ou itératif ?

Fonction *toto(x) :Entier*
| **D**: x : Entier>0
| **Si** *x=1* **ou** *x=2* **alors**
| | **Retourner** 1
| **Sinon**
| | **Retourner** $2 + toto(x-1) + toto(x-2)$
| **Fsi**
FFonction

Quelles opérations réalise l'appel `toto(4)`, dans quel ordre et quel est le résultat final ?

Exercice 4 Écrire une **fonction** qui prend un tableau d'entiers (> 0) en paramètre et répond vrai si ce tableau (vecteur) ne contient que des entiers pairs, et faux sinon. (On pourra s'apercevoir que si on trouve un entier impair, on peut terminer l'algorithme...)

Exercice 5 Écrire un algorithme (fonction ou action ? justifier !) qui calcule et stocke dans un tableau (vecteur) de taille N les entiers 0!, 1! ... $N-1$!, en faisant un minimum de multiplications. Quelle est la complexité de votre algorithme ?

On rappelle que $i! = 1 \times 2 \times 3 \times \ldots i$.

POLYTECH LILLE

Université Lille1 Sciences et Technologies

Département IMA / 3A (S5) **Programmation Structurée 2011/2012**

http://laure.gonnord.org/pro/teaching/

Syntaxe Algorithmique

Nom	Syntaxe	Exemple	Commentaire
Affectation	$\leftarrow$	$x \leftarrow 42$	x doit être déclaré
Type entier	Entier	x :Entier	déclaration de x entier
Type réel	Réel	x :Réel	déclaration d'un réel, en machine ce sera un flottant
Type charactère	Charactère	c :Charactère	déclaration d'un caractère ; les constantes sont 'a', 'b', ...
Type booléen	Booléen	b :Booléen	déclaration d'un booléen ; les constantes sont `Vrai` et `Faux`
Type chaîne	Chaîne	$s \leftarrow$ "*toto*"	affectation d'une chaîne, s doit être déclarée.
Tableau	Vecteur	Vecteur[10] d'Entiers	tableau de 10 entiers indexés de 0 à 9
constante	Constante	Constante N : 10	déclare une constante symbolique N qui vaut "10"

Tests

Si *condition* **alors**
| instructions si vrai
Sinon
| instructions si faux
Fsi

Si *condition* **alors**
| instructions si vrai
Fsi

Boucles

Pour *i* **de** *inf* **à** *sup* **Faire**
| instructions
Fpour

Tq *condition* **faire**
| instructions
Ftq

Programme/fonction/action : exemples

Programme *Main*
| ....
| ....
| ....
| **Retourner** *0*
FProgramme

Fonction *fonct(c) : Entier*
| **D**: c :Charactère
| **L**: s :Entier
| ....
| ....
| ....
| **Retourner** *s*
FFonction

Action *monact(a,b,c)*
| **D**: a : Entier
| **D/R**: b : Entier
| **R**: c : Charactère
| ...
| ...
| ...
FAction

- ici *fonct* est une fonction `Caractère -> Entier` :
  L'unique paramètre (la *donnée*) est un caractère, nommé c dans la suite de la fonction. La variable s est *locale*. L'appel : $resu \leftarrow fonct(d)$ où d a une valeur et $resu$ est déclaré de bon type.
- ici *monact* est une action à trois arguments. Les modifications apportées au premier paramètre ne sont pas enregistrées (c'est une *donnée*). Par contre les modifications apportées aux paramètres 2 et 3 sont enregistrées (*donnée-résultat* et *résultat*). Appel : `monact(a,b,c)` avec a, b ayant une valeur, et c étant déclaré.