INF1256 Informatique pour les sciences de la gestion

–Introduction à l'algorithme–

Johnny TSHEKE, Ing. Jr.

Université du Québec à Montréal
Département d'Informatique
tsheke_shele.johnny@uqam.ca

Séance 01

Qu'est ce qu'un algorithme ?

Déf. : Une séquence finie d'opérations ou d'instructions non ambigues permettant d'obtenir un résultat voulu avec des données éventuelles en entrée, dans un temps fini.

Exemple : Algorithme de calcul du nombre factoriel n ($n!$)

- SI n=0 $\rightarrow$ $n! = 1$
- SINON SI n est entier > 0 $\rightarrow$ $n! = n * (n-1)!$

Comment se présente un algorithme ?

Graphique : Organigramme ou Ordinogramme (Flowchart) , grafcet, etc

Texte : Langage naturel, Pseudo-code, formalisme mathématique, etc.
⇒ le Pseudo-code est un formalisme permettant de décrire (de manière structurée) un algorithme en langage naturel.

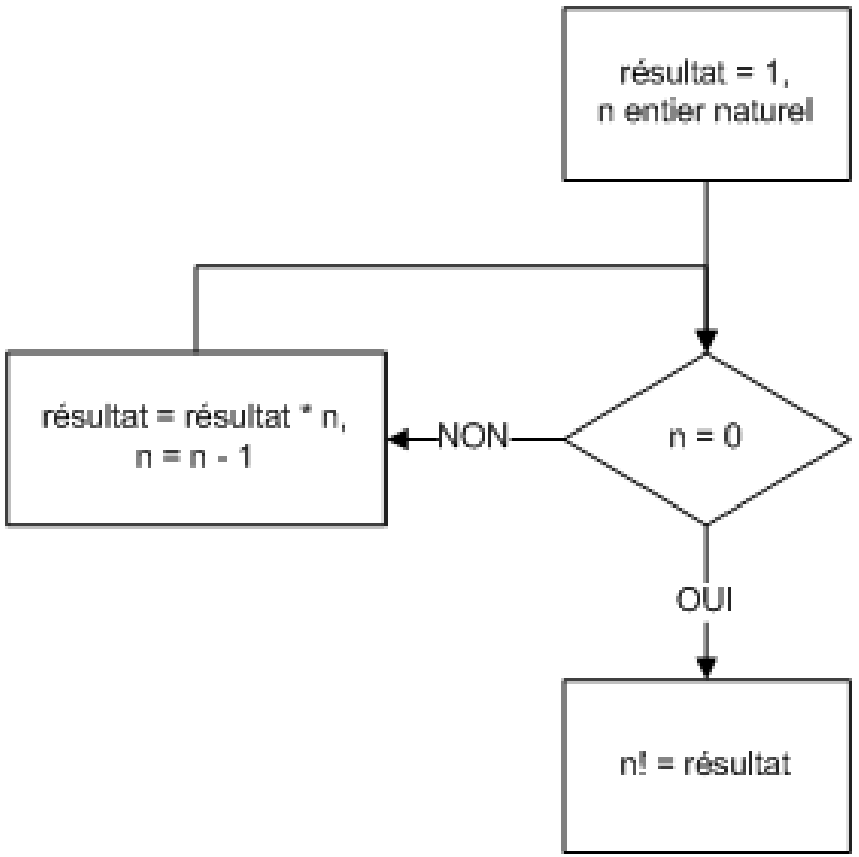


FIGURE: Algorithme de calcul du nombre factoriel

Exemple d'un Pseudo-code

Pseudo-code d'une fonction calculant $n!$

```
Fonction factorielle( n )
  Variable locale:
   r(entier)
  Instructions:
   r ← 1
   SI n > 0 ALORS
       r ← n * factorielle( n − 1 )
   FIN SI
   retourner r
Fin Fonction
```

Principales étapes d'un algorithme

Entrée, Initialisation : on reçois les données nécessaires et on procède aux initialisations éventuelles

Traitement : Exécution de la séquence d'instructions

Sortie : Le résultat du traitement

Quelques concepts relatifs aux données

Variable : Symbole renvoyant à une adresse mémoire dont le contenu peut-être modifié au cours de l'exécution du programme

Constante : La valeur ne peut-être modifié au cours de l'exécution du programme

Type :(variable, constante, donnée) ensemble de valeurs possibles et les opérations permises sur ces données

Tableau : Structure de données constitué d'un ensemble d'éléments auxquels on accède par un numéro d'index

Liste, Arbre, Objet, . . .

⇒ Plus d'info :
`http://fr.wikipedia.org/wiki/Algorithmique#Vocabulaire`

Structures de contrôle

Séquence : Structure implicite donnant l'ordre d'exécution des instructions

Conditionnelle : Structure permettant d'exécuter un ensemble d'instructions si une condition est satisfaite

Boucle : Un ensemble d'instructions à exécuter de manière répétée

⇒ Plus d'info :
`http://fr.wikipedia.org/wiki/Algorithmique#Vocabulaire`

Quelques notions importantes sur les algorithmes

Terminaison : Se terminer en un temps fini

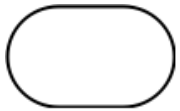
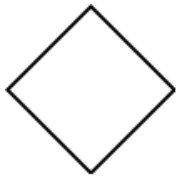
Correction : Donner des résultats corrects

Complétude : Donner toutes les solutions

Complexité : Donne l'ordre de grandeur de nombre d'opérations nécessaires (O(n), . . .)

Quelques symboles (shapes) graphiques

Pour représenter un algorithme sous forme de diagramme (flowchart)

Symbole	Usage
	Début ou Fin
	Action, affectation, traitement, sequence (process)
	Décision, choix, condition
	Entrée ou sortie de données
	Sous routine, routine prédéfinie
	Connecter les symboles en indiquant le sens d'exécution

⇒ Outil gratuit pour dessiner : `https://www.draw.io/`

Forme générale d'un algorithme en pseudocode

```
nomAlgorithme

 DÉBUT
          déclaration des variables

           corps de l'algorithme
 FIN
```

⇒ Plus d'info :

http://www.univ-rouen.fr/servlet/com.univ.utils.LectureFichierJoint?CODE=1222435307694&LANGUE=0

Variables en pseudo-code

Déclaration : nombre (entier)

Affectation : nombre $\leftarrow$ 0

Opérateurs logiques : ET, OU, NON

Entrée/Sortie en pseudo-code

Entrée :

```
LIRE nomVariable
ou
LIRE nomFichier
```

Sortie :

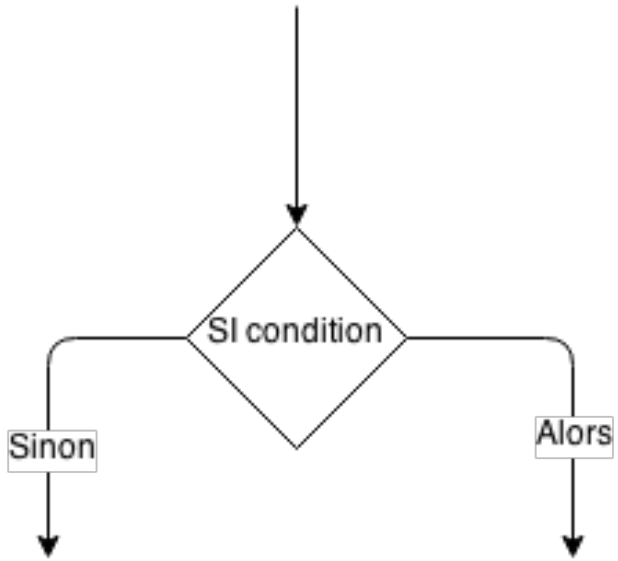
```
ECRIRE nomVariable
ou
ECRIRE nomFichier
ou
ECRIRE "message"
```

Tests des conditions (if)

```
SI (condition)
        ALORS  instructions
 FIN SI
```

ou

```
 SI (condition)
        ALORS  instructions
        SINON  autres instructions
 FIN SI
```



Important d'étiqueter les flèches sortant de la décision. Il faut les connecter aux symboles des instructions suivantes

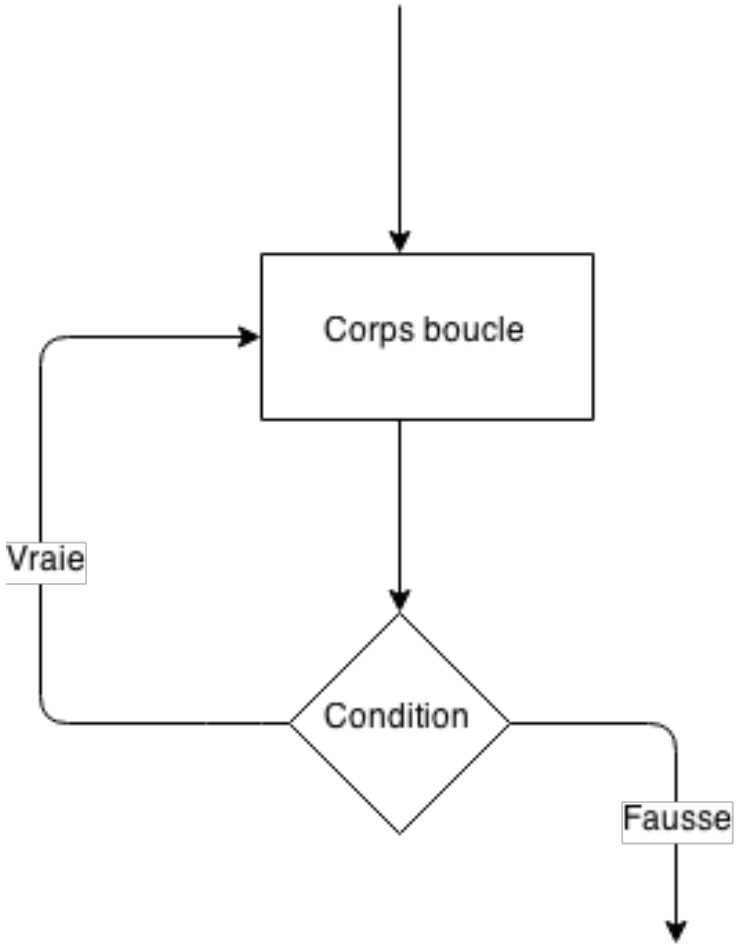
Choix des alternatifs (switch)

```
SELON choix
     CAS 1:
               Bloc instruction 1
     CAS 2:
             Bloc instruction 2
     DÉFAUT:
           Bloc instructions par défaut
FIN SELON
```

Répéter ...Tant que (do ..while)

```
REPETER
    Corps de la boucle
TANT QUE (condition)
```

⇒ Le "Corps de la boucle" sera exécuté au moins une fois !

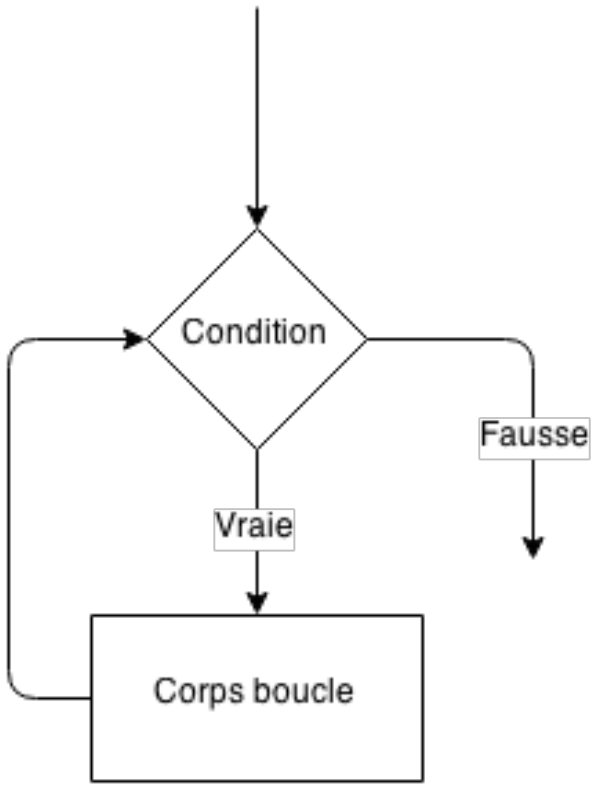


⇒ Le corps de la boucle pourrait avoir plusieurs actions !

Tant que ... (while ..do)

```
TANT QUE (Condition)
  Corps de la boucle
FIN TANT QUE
```

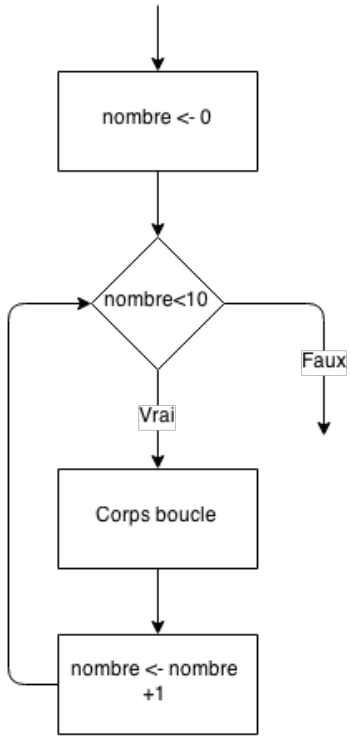
⇒ Le "Corps de la boucle" peut ne pas être exécuté si "condition" est fausse dès le départ !



⇒ Le corps de la boucle pourrait avoir plusieurs actions !

Pour ... (for)

```
POUR nombre ← 0 , nombre < 10 , PAS de 1
    Corps de la boucle
FIN POUR
```



⇒ Le corps de la boucle pourrait avoir plusieurs actions !

Les étapes importantes pour résoudre un problème

- Énoncer le problème clairement
- Décomposer le problème complexe en sous problèmes simples pour en concevoir des algorithmes
- Définir les données d'entrée et de sortie à manipuler par chaque algorithme
- Développer le logiciel

Principales étapes

Analyse : Comprendre le problème dans son contexte et les besoins. Essayer d'anticiper les besoins futurs

Conception : Chercher une solution indépendamment du langage de programmation

Codage : Programmer la solution dans un langage spécifique (Python, Java, etc.)

Tests : Vérifier le bon fonctionnement du logiciel et sa conformité au cahier de charges

Déploiement : Installation du logiciel

Maintenance : Amélioration, correction, modifications, ...

Quelques exercices

Ecrire le pseudo code et les organigrammes (ordinogramme ou flowchart) pour résoudre les problèmes suivants

- Initialiser la variable n avec la valeur 3 et la variable m avec la valeur 4 puis calculer leur produit (n*m) r
- Initialiser 3 variables avec des nombres réels et calculer la moyenne
- Calculer la somme des doubles des nombres allant de 1 à 5
- Demander à l'utilisateur d'entrer un nombre et afficher ce nombre s'il est strictement positif. Répéter l'opération jusque quand l'utilisateur entre le nombre 0.