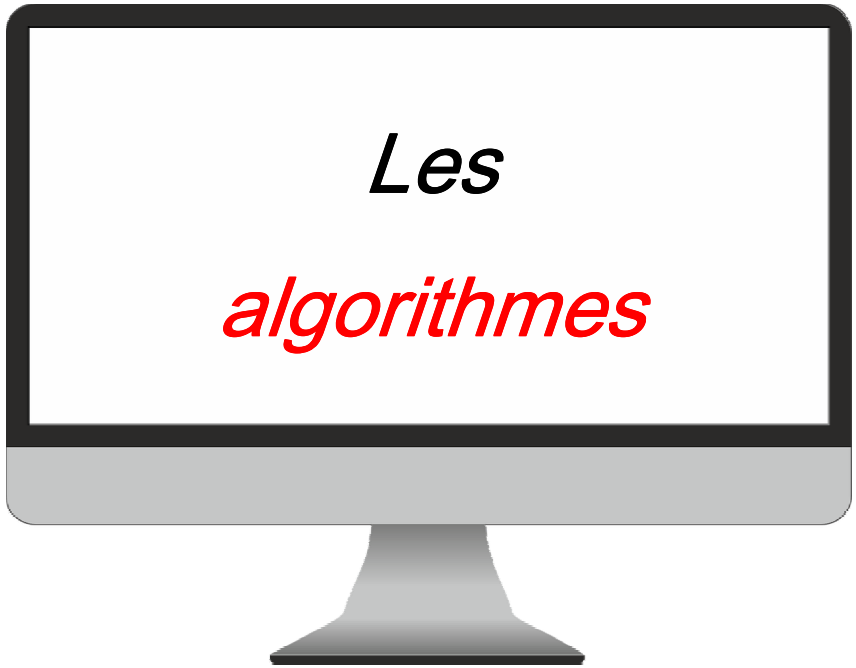
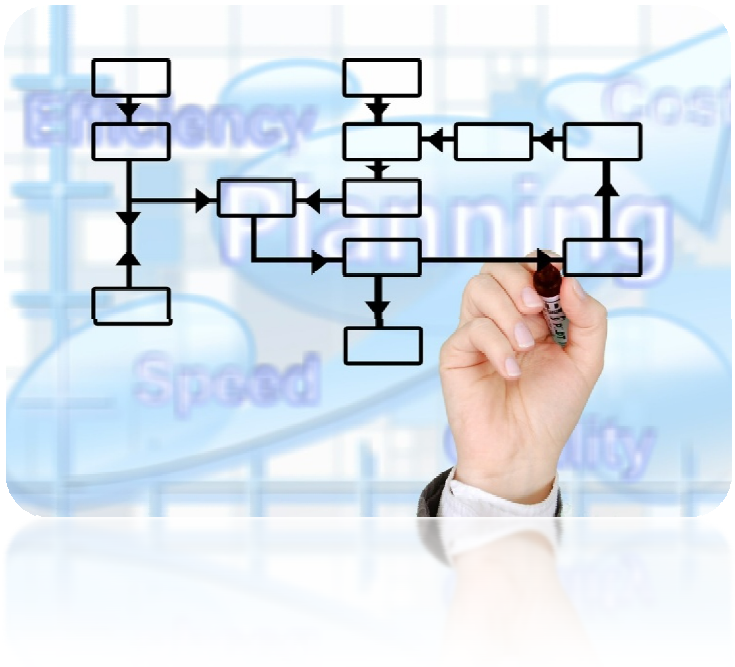


Les algorithmes

Objectifs : comprendre ce qu'est un algorithme et comment il peut être lié à la programmation.



1- Qu'est-ce qu'un algorithme?

2- Mise en œuvre d'un algorithme

3- Algorithmique et programmation

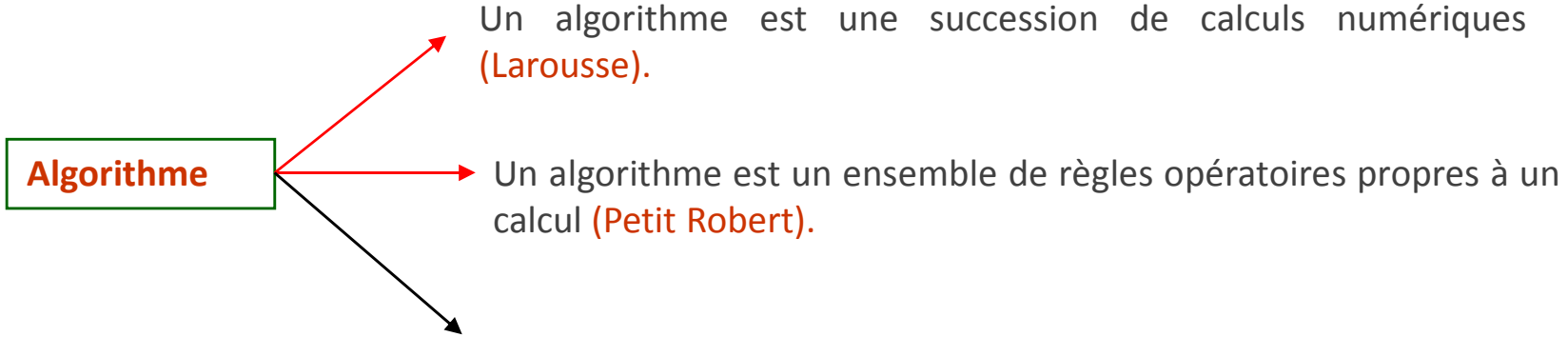
4- Programmation structurée

5- Enchaînements et données

6- Représenter un algorithme

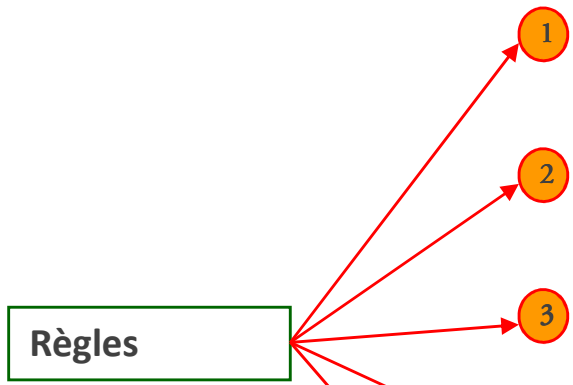
Le mot Algorithme vient du nom du mathématicien arabe du 9ème siècle, MUHAMMED IBN AL-KHAREZIN.

1 Définitions (Dictionnaires)



Un algorithme est un ensemble de règles ayant les 5 caractéristiques suivantes :

2 Règles de base à retenir



1. Il est fini et se termine après un nombre fini d'opérations.
2. Il est défini sans ambiguïté.
3. S'il y a des données en entrée, leur type doit être précisé.
4. Il doit avoir au moins un résultat.
5. Il doit être effectif : toutes les opérations doivent pouvoir être effectuées exactement et dans un temps fini par un homme utilisant des moyens manuels.

On peut suivre la démarche simple.

1 Démarche à suivre

- Identifier les données en entrée et les résultats à obtenir.
- Définir la méthode permettant d'obtenir les résultats recherchés. C'est la recherche ou la **conception de l'algorithme** proprement dit.
- Décrire l'algorithme soit en **pseudo code**, soit sous la forme d'un **organigramme**.

Pour un problème donné, il existe souvent plusieurs algorithmes de résolution. Il faut choisir le meilleur compte tenu du problème à résoudre, en fonction par exemple :

1. Du temps d'exécution
2. De l'espace mémoire nécessaire
3. De la précision des calculs

On appelle langage informatique un langage destiné à décrire l'ensemble des actions consécutives qu'un ordinateur doit exécuter.

1 Les langages

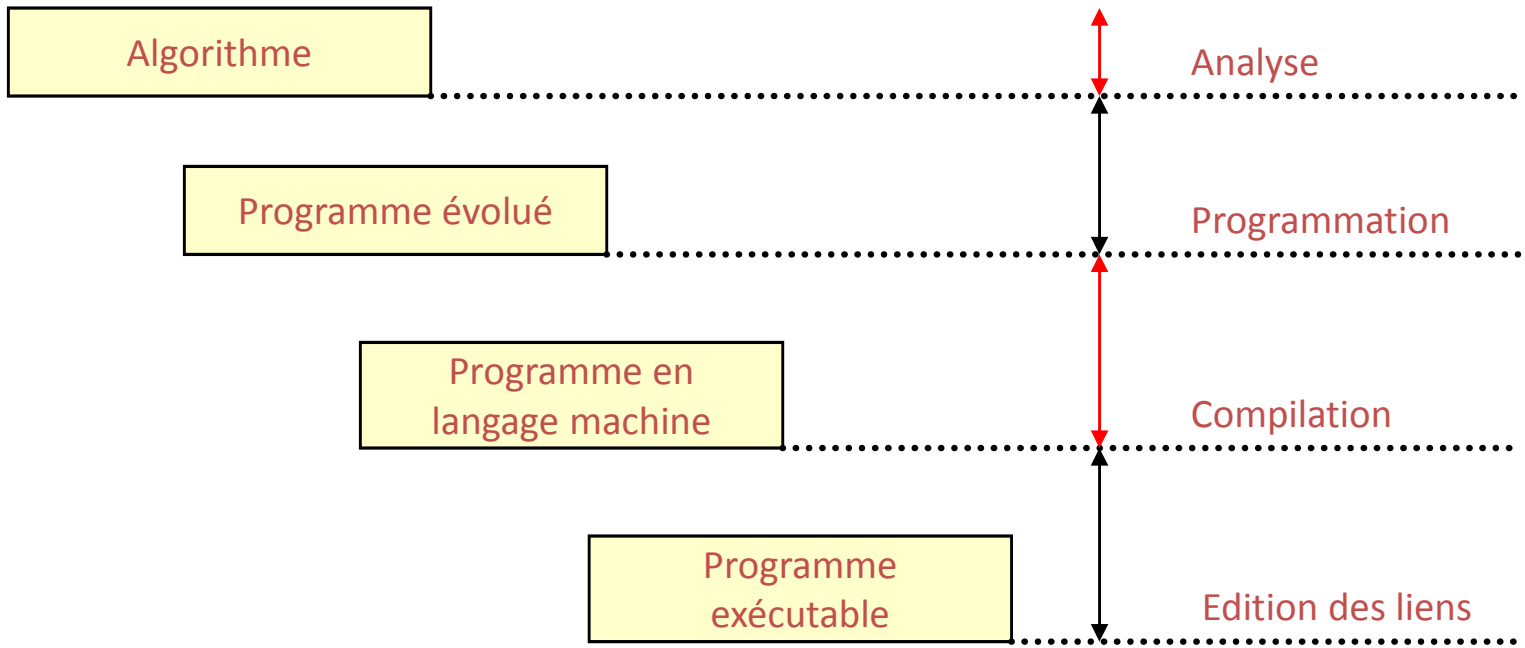
- ***Les langages naturels*** (l'anglais, le français) représentent l'ensemble des façons qu'a un groupe d'individus de communiquer.
- ***Un langage informatique*** est une façon pratique pour nous (humains) de donner des instructions à un ordinateur.

Attention :

- Les langages servant aux ordinateurs à communiquer n'ont rien à voir avec les langages informatiques, on parle dans ce cas de ***protocoles***, ce sont deux notions totalement différentes.
- Afin d'écrire des programmes structurés, modulaires, lisibles et facilement modifiables, des méthodes de programmation ont été introduites. Exemple : la méthode de programmation structurée descendante.

Pour produire un logiciel, plusieurs étapes doivent être suivies. En voici ci-dessous les principales :

2 La mise au point d'un programme



3 Etapes de la mise au point d'un programme

- **Analyse** :

  Elle consiste à mettre à jour la façon dont le problème sera résolu. C'est la recherche de l'algorithme de résolution. On peut écrire cet algorithme sur papier ou sur ordinateur en utilisant une description proche du langage humain (Pseudo code).

- **La programmation évoluée** :

  Le Pseudo code étant peu rigoureux, on convertit l'algorithme en langage de programmation évolué. On obtient alors un fichier texte ou **code source**.

- **La compilation** :

  L'ordinateur ne comprend pas un langage de programmation évolué comme le C. Il convient de convertir le **code source** en langage machine : c'est la compilation. On utilise un logiciel appelé **compilateur.**

- **L'édition des liens** :

  Pour s'assurer que le programme compilé fonctionne d'un ordinateur à l'autre, on réalise l'édition des liens qui va lier le programme avec tous les éléments externes (généralement des librairies auxquelles il fait référence). On obtient un **programme exécutable (fichier d'extension .exe par exemple)**.

3 Etapes de la mise au point d'un programme (suite)

De l'algorithme au code

- **Quelques langages de programmation**

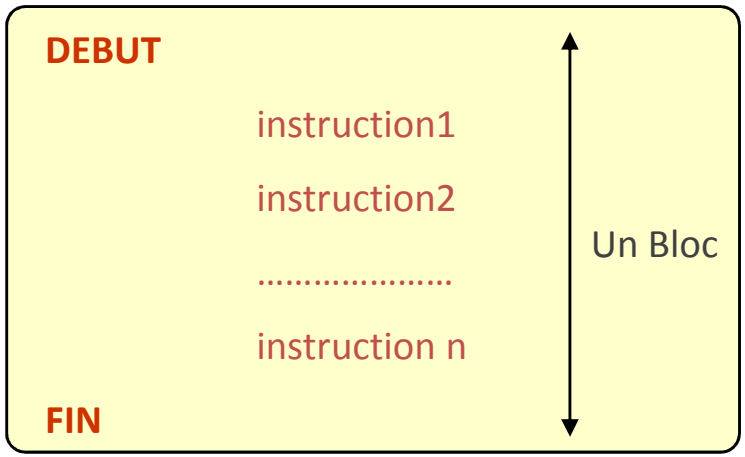
Cobol	VB (Visual Basic)	Basic
Java	VB.Net	Pascal
C	Python	PHP
C# (lire C sharp)	Smalltalk	Etc.
C++	Eiffel	
Javascript	Ruby	

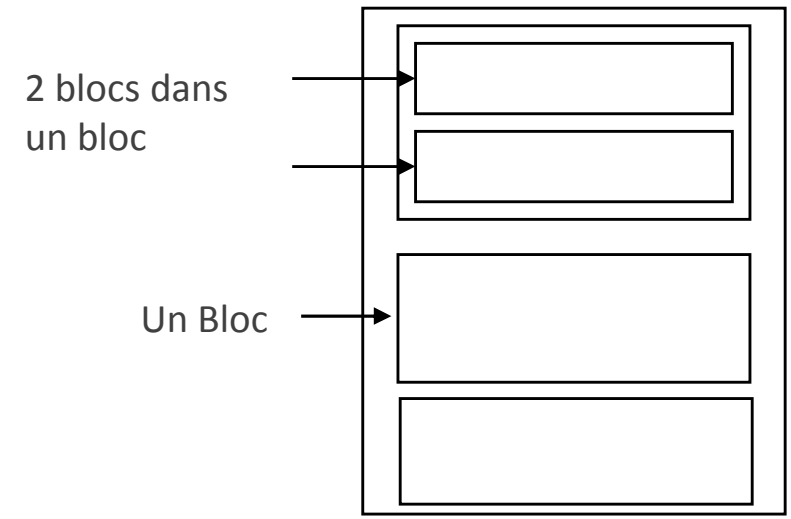
Elle est fondée sur le bloc et l'analyse descendante.

1 Notion d'instruction et de bloc

L'analyse descendante suggère qu'un algorithme complexe peut être décomposé en plusieurs sous-algorithmes plus simples.

Une **instruction** est la description non ambiguë d'une ou plusieurs actions élémentaires à effectuer dans un ordre déterminé. Selon les besoins, une suite d'instructions formera un bloc délimité par les mots DEBUT et FIN. Un bloc peut en contenir d'autres.





Elle est fondée sur le bloc et l'analyse descendante.

2 Constituants d'un algorithme

Un algorithme est la décomposition d'une action complexe qui, partant de données toutes définies, permet d'obtenir un (des) résultat(s) déterminé(s). Tout algorithme est fait :

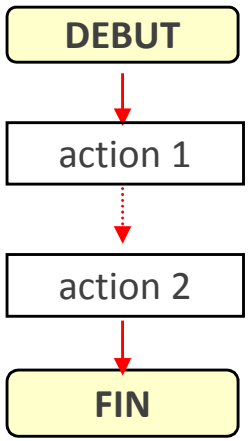
- d'enchaînements d'actions ou traitements : séquence, sélection ou répétition.
- de données sur lesquelles ces actions agissent. Ces données sont des constantes ou des variables, et sont d'un certain type (entier, chaine, etc.)
- L'algorithme produit des résultats.

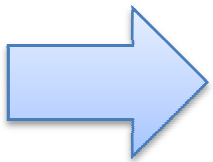
Cheminement

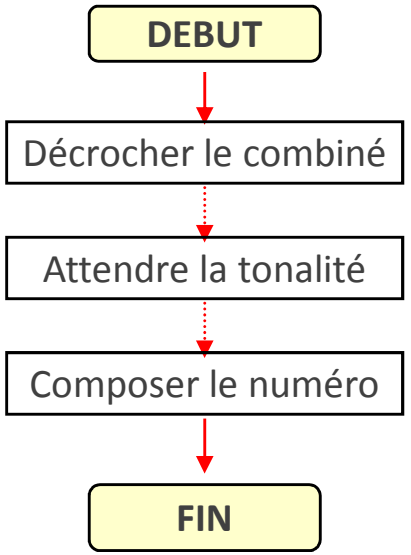
Séquences, décisions, répétitions

1 Les enchaînements : séquences

L'enchaînement **séquentiel** consiste à exécuter des actions l'une après l'autre, du début à la fin du bloc logique qu'elles forment (l'action plus complexe) :







Exemple : appeler quelqu'un au téléphone

Séquences, décisions, répétitions

2 Les enchaînements : sélection

L'enchaînement **sélectif** consiste à exécuter certaines actions si une condition préalable est vérifiée, et à exécuter d'autres actions si ce n'est pas le cas. On peut le représenter ci-dessous.

SI *[Condition préalable vraie]*

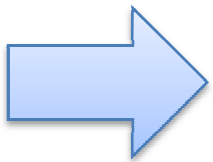
action 1

action 2

SINON

action 3

action 4



SI le nombre B est égal à zéro

Ecrire division impossible

SINON

Faire la division A/B

Exemple : diviser A par B

Nota : on peut imbriquer des sélections (ne pas en abuser).

Séquences, décisions, répétitions

2 Les enchaînements : sélection (suite)

Un bloc peut tout à fait ne comporter aucune action. Dans ce cas, la branche **« SINON »** peut ne pas être représentée. Par exemple, en faisant des courses :

SI *le produit convient*

Le prendre du rayon

SINON

RIEN

Séquences, décisions, répétitions

3 Les enchaînements : répétition

L'enchaînement **répétitif** consiste à exécuter certaines actions aussi longtemps qu'une condition préalable est vérifiée, ou à exécuter certaines actions jusqu'à ce qu'une condition pré-établie soit vérifiée.

REPETER

Les actions suivantes

JUSQU'A *[ce que la condition soit vraie]*

Cette boucle s'effectue donc de **1 à plusieurs fois**.

TANT QUE *[cette condition est vraie]*

Exécute les actions suivantes

Cette boucle s'effectue donc de **0 à plusieurs fois**.

Nota : on peut imbriquer des répétitions (ne pas en abuser).

Séquences, décisions, répétitions

3 Les enchaînements : répétition (suite)

L'enchaînement **répétitif** consiste à exécuter certaines actions aussi longtemps qu'une condition préalable est vérifiée, ou à exécuter certaines actions jusqu'à ce qu'une condition pré-établie soit vérifiée.

REPETER

Manger un peu

JUSQU'A *ne plus avoir faim*

Cette boucle s'effectue donc de **1 à plusieurs fois**.

TANT QUE *j'ai faim*

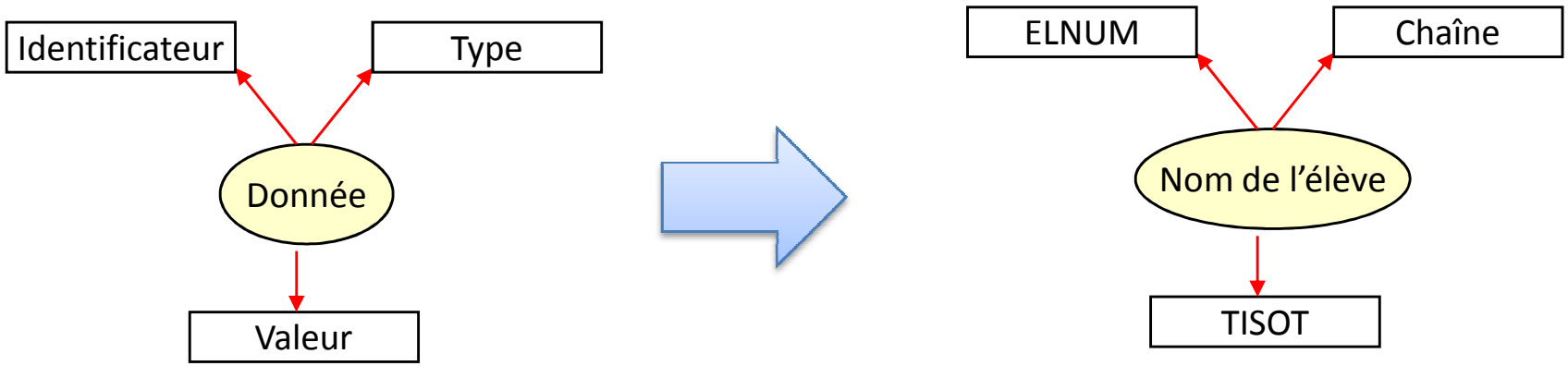
Manger un peu

Cette boucle s'effectue donc de **0 à plusieurs fois**.

Donnée = case mémoire

4 Données : identificateur, type, variable

Les instructions d'un algorithme opèrent sur des données dont la description peut être schématisée ci-dessous. Une donnée possèdera un **identificateur**, un **type** et une **valeur** :



Exemple décrivant un élève

Donnée = case mémoire

5 Données : identificateur, type, variable (suite)

- ***Un identificateur :*** c'est le nom de la donnée. Il devra être unique dans l'algorithme. Dans l'exemple précédent, ELNUM fera référence au nom de l'élève. ELNUM est un identificateur de Nom de l'élève.

  ELNUM="TISOT" ou **ELNUM** a la valeur **TISOT** (ou contient TISOT).

- ***Un Type :*** détermine l'ensemble dans lequel la donnée prend ses valeurs. Ici, **ELNUM** est de type **chaîne** de caractères, car le nom d'une personne est une chaîne de caractères.

- ***Une Constante :*** lorsque la donnée ne prend qu'une seule valeur dans tout l'algorithme, on l'appelle constante.

- ***Une Variable :*** si, au cours du déroulement de l'algorithme la donnée peut changer de valeur, on l'appelle variable.

Donnée = case mémoire

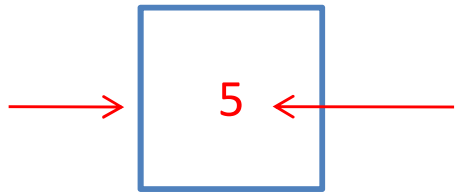
6 Constante et variable

La donnée est une sorte de case mémoire. L'***identificateur*** est le nom qu'on donne à la case mémoire. On stocke dans la case une valeur ***constante*** ou ***variable***. Enfin la donnée peut être d'un certain type.

type :

1- ensemble de valeurs possibles.
Exemple : entier, chaine, date, etc.
5 est de type entier.

2- opérations possibles :
Exemple : +, - , & , / , etc.



5 est la ***valeur*** stockée dans la case.

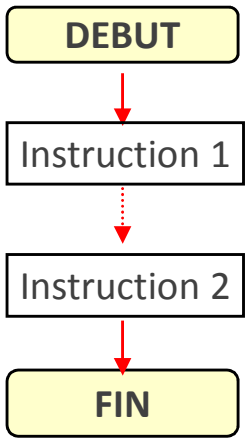
Cette case s'appelle ***nbre***. C'est son ***identificateur***.

Donnée : *identificateur + valeur + type*

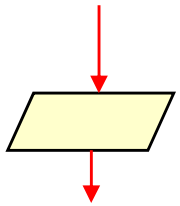
Pseudo code ou organigramme

1 Utiliser un organigramme

Le déroulement d'un algorithme peut être représenté par un organigramme (ordinogramme). Ce dernier donne une impression de déroulement successif des instructions plus grande, mais ne met pas en évidence la notion de bloc.



Le losange symbolise une alternative



Le parallélogramme indique qu'on introduit des données et qu'on fournit un résultat.

Symboles les plus courants

Pseudo code ou organigramme

1 Utiliser un organigramme (suite)

Limites de l'organigramme

- Ne met pas en évidence la notion de bloc.
- Si on doit structurer une pensée complexe, l'organigramme deviendra vite très complexe et difficile à manipuler.
- Sert surtout à l'analyse ascendante (c'est-à-dire du détail vers le général).
- Ce formalisme est considéré plutôt comme un bon outil de communication – et non comme un outil de conception.

On préferera utiliser le pseudo code

Pseudo code ou organigramme

2 Utiliser le pseudo-code

Chaque élément d'algorithme est défini par un **terme représentatif et non ambigu**. Ces différents termes, peu limités, forment le pseudo-code.

- **pseudo-code = langage qui décrit l'algorithme.**
  compréhensible par l'homme (utilisateur, programmeur, etc).

- langage de programmation (code) = transcrit l'algorithme en code source.
  accessible au programmeur et à l'ordinateur.

Nota :

L'utilisation d'un petit nombre d'opérations élémentaires pour représenter des algorithmes simplifie leur expression. Ce qui facilite ainsi leur conception, mise au point et maintenance.

D'ailleurs en 1966, Böhm et Jacopini ont démontré qu'on peut décrire tout algorithme en n'utilisant que l'enchaînement séquentiel et la boucle « Tant que ».

Pseudo code ou organigramme

3 Eléments du pseudo-code

Il n'existe pas de norme internationale syntaxique du pseudo-code. Voici un exemple qui respecte l'esprit du langage, notamment au niveau des structures d'enchaînement :

- **Pour une alternative**

SI *<condition>*
| actions 1
SINON
| actions 2
FIN-SI

- **Pour l'itération**

TANT QUE *<condition>*
| Exécute les actions suivantes
FIN-TANT- QUE

REPETER
| Les actions suivantes
JUSQU'A *<condition>*

Pseudo code ou organigramme

3 Eléments du pseudo-code (suite)

L'affectation est une opération qui permet de mettre une valeur dans une case mémoire. On la représente par une flèche vers la gauche .

- **Pour une affectation**

  A ← B

  A ← B / 250

  A ← A + 12

- **Lecture d'une valeur (du clavier, disque, etc.)**

  LIRE A

  LIRE A, B

- **Ecriture d'une valeur (à l'écran, sur le disque, etc.)**

  ECRIRE A

On peut parfois différencier l'affichage (à l'écran) de l'écriture (sur le disque) par **AFFICHER** A et **ECRIRE** A respectivement.

Pseudo code ou organigramme

4 Exemple d'algorithme en pseudo-code

L'algorithme ci-dessous demande son nom à un utilisateur, lit le nom en question et le range dans la case mémoire **Nom**. Enfin, le nom fourni est affiché en sortie à l'écran.

Nom de l'algorithme

```
ALGO MONNOM

VAR Nom : Chaine

DEBUT

        ECRIRE  'Veuillez saisir votre nom'
        LIRE  Nom
        ECRIRE  'Vous êtes', Nom
FIN
```

Déclaration des données

Le système attend une saisie avant de poursuivre

Un petit algorithme