Introduction à l'algorithmique et à la programmation

Algorithmique et Programmation

Unisciel/K.Zampieri

Diapositive de résumé

- L'activité de programmation
- La construction de programmes
- La programmation

L'activité de programmation

Vous savez compter ! L'ordinateur aussi…

Votre programme s'exécute, mais…

- Connaissez-vous les mécanismes utilisés ?
- Êtes-vous sûr que le résultat soit juste ?
- Combien de temps devrez-vous attendre la fin du calcul ?
- Y a-t-il un moyen pour obtenir le résultat plus vite ?
  - Indépendamment de la machine, du compilateur…

Un ordinateur ne s'utilise pas comme un boulier !

=> Connaître des algorithmes

=> Apprendre à les construire, les améliorer…

La programmation

Objectif : A*utomatiser* des tâches à l'aide d'*automates programmables* (machines particulières).

Automate : Dispositif capable d'assurer, sans intervention humaine, un enchaînement d'opérations correspondant à la réalisation d'une tâche donnée. Exemple : la montre, le réveil-matin, le « ramasse-quilles » (du bowling)…

Programmable : Lorsque la nature de la tâche peut être *modifiée* (à volonté). Dans ce cas, la *description* de la tâche se fait par le biais d'un *programme*, i.e. une séquence d'instructions et de données susceptibles d'être traitées (i.e. *comprises* et *exécutées*) par l'automate. Exemple : le métier à tisser Jacquard, l'orgue de barbarie… et l'ordinateur.

La programmation (2)

Programmer c'est : *Décomposer* la tâche à automatiser sous la forme d'une séquence de *traitements* et de *données* adaptées à l'automate utilisé.

Formalisation des traitements : *algorithmes*

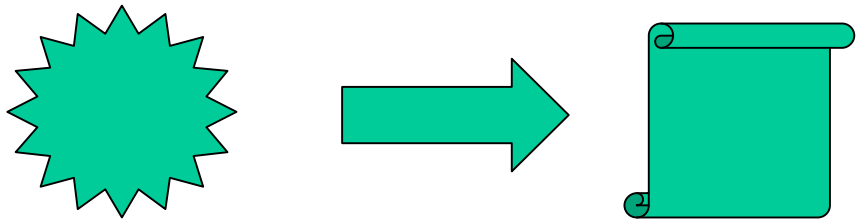
➔ distinguer formellement les bons traitements des mauvais

Formalisation des données : *structures de données* (SDD)

➔ distinguer formellement les bonnes SDD des mauvaises

Concrètement : quelles sont les instructions et les données *adaptées* à l'ordinateur.

La programmation (3)



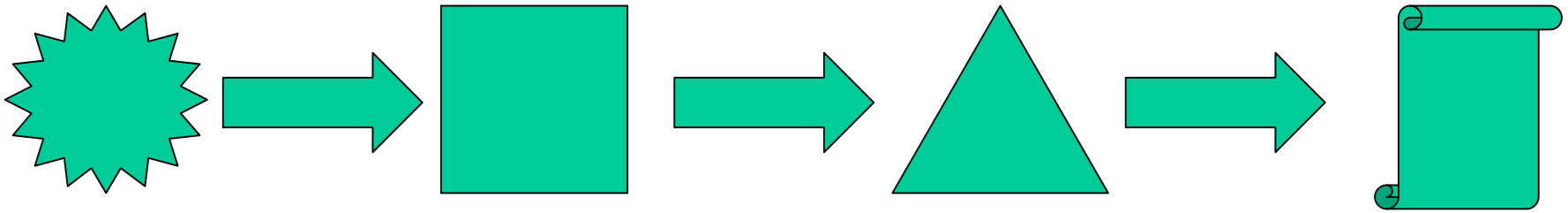
Problème Programme

- Question à résoudre par une solution informatique
- Instance d'un problème = entrée nécessaire pour calculer une solution du problème

- Ensemble de données
- Ensemble de résultats = solution informatique au problème
- Description d’un ensemble d’actions
- Exécution dans un certain ordre

Les étapes de développement

Spécification Résolution Codage



Problème Énoncé Algorithme Programme

Ne pas se laisser aveugler par l'objectif final: le codage!

Notion d’énoncé (du problème)

Souvent le problème est "mal posé"…

- Rechercher l’indice du plus petit élément d’une suite (p.ex. [7,1,3,1,5] ; indices : 2,4 ; lequel ?)

$\Rightarrow$ Spécifier = produire un énoncé

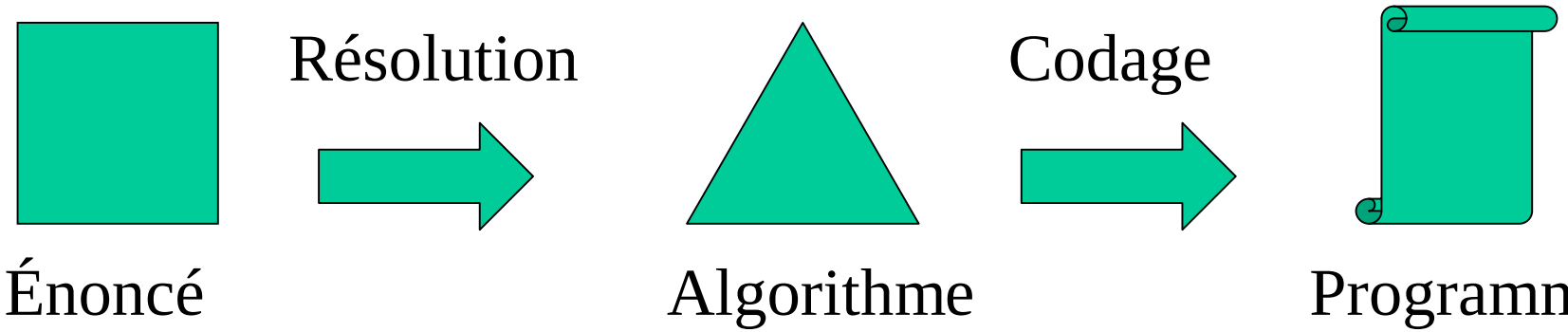
Énoncé = texte où sont définies sans ambiguïté :

- L’entrée (données du problème)
- La sortie (résultats recherchés)
- Les relations (éventuelles) entre les données et les résultats

Que dois-je obtenir ?

- Soit I l’ensemble des indices des éléments égaux au minimum d’une suite. Déterminer le plus petit élément de I.

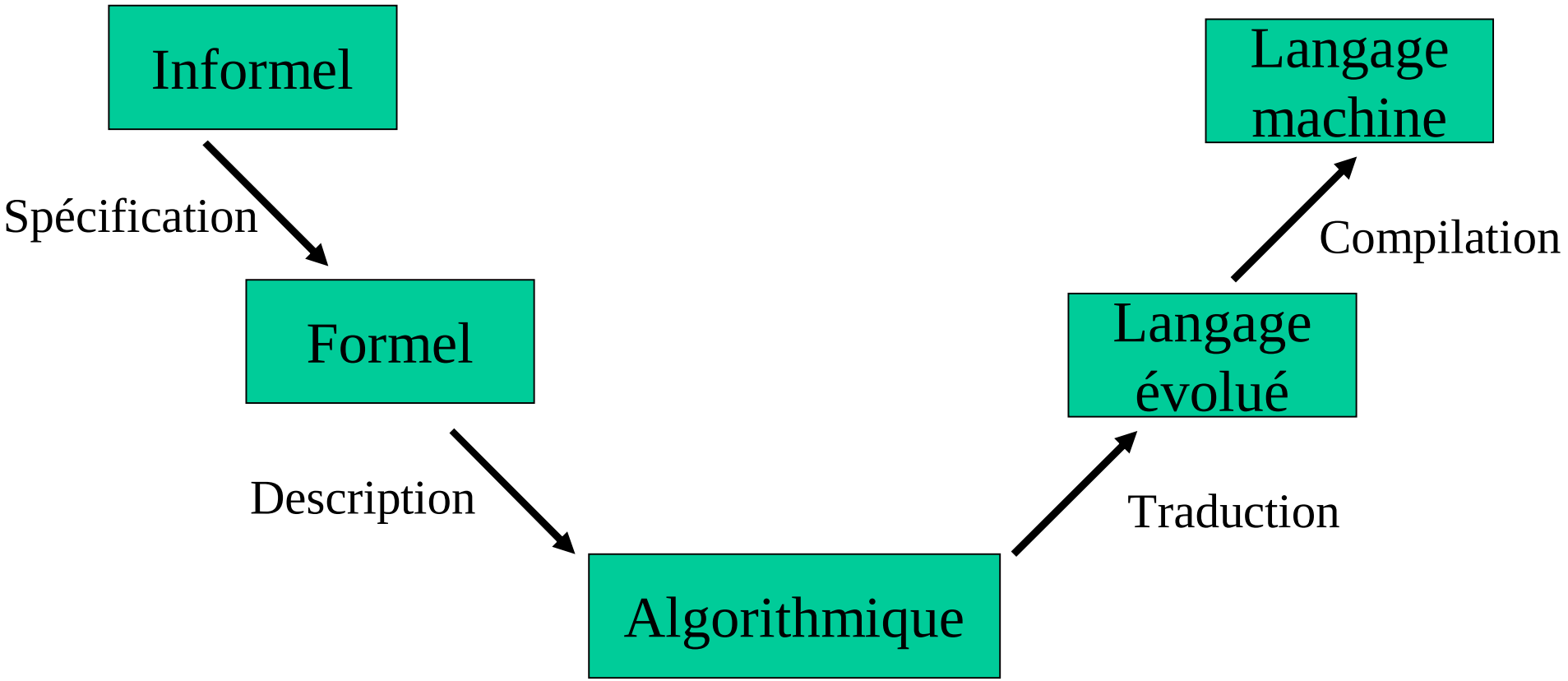
Notion d’algorithme



= Description d’un processus de résolution d’un problème bien défini

= Succession d’actions qui, agissant sur un ensemble de ressources (entrées), fourniront la solution (sortie) au problème

Schéma de la programmation



Phases

La construction de programmes

Voici quelques méthodes de constructions de programmes…

Conception structurée

Critère simple d'automatisation : un problème est *automatisable* (traitable par informatique) si l'on peut :

- Définir les données et les résultats.
- Décomposer le passage de ces données vers ces résultats en une suite d'opérations élémentaires dont chacune peut être exécutée par une machine.

Méthodologie descendante

Démarche de J. Arsac :

- Expliciter les données (nature, domaine, propriétés)
- Expliciter les résultats (structure, relations avec les données)
- Décomposer le problème en sous-problèmes.
- Pour chaque sous-problème faire :

  Si la solution est évidente alors

  écrire le morceau de programme

  sinon

  appliquer la méthode au sous-problème

Méthodologie ascendante

Démarche de la Programmation Orienté Objet :

- Débuter par les objets les plus simples/aisés à connaître
- Monter jusqu'à la connaissance des plus composés

⇔ Troisième précept de R. Descartes

Méthodologie orientée objet

Concepts de B. Meyer :

- Une analyse descendante (i.e. de l'abstraction vers le concret) avec retour sur un niveau ascendant
- Une analyse guidée par les objets

⇔ Construction d'un arbre d'analyse dirigé par les objets

Algorithme

Définition (D.E. Knuth) :

- Ensemble de règles décrivant une séquence d'opérations en vue de résoudre un problème donné bien spécifié.

Un algorithme est donc :

- une *suite finie* de règles à appliquer,
- dans un *ordre déterminé*,
- à un *nombre fini* de données.

⇔ Arriver, en un nombre fini d'étapes, à un résultat quelles que soient les données traitées.

Propriétés d'un algorithme

Un algorithme doit répondre aux caractéristiques de :

- Finitude : se termine et temps d’exécution évaluable
- Précision : étape définie et production d'un résultat correct
- Domaine des entités : toute entrée ou sortie est spécifiée
- Exécutabilité : génère un programme exécutable en un temps fini et raisonnable [et si possible est efficace]

Problèmes fondamentaux en algorithmique

Complexité :

- En combien de temps un algorithme atteint-il le résultat ?
- De quel espace a-t-il besoin ?

Calculabilité :

- Existe-t-il des tâches pour lesquelles il n'existe aucun algorithme ?
- Étant donnée une tâche, peut-on dire s'il existe un algorithme qui la résolve ?

Correction :

- Peut-on être sûr qu'un algorithme réponde au problème pour lequel il a été conçu ?

La programmation

Programmer, c'est communiquer

- Avec la machine
- Avec soi-même
- Avec les autres

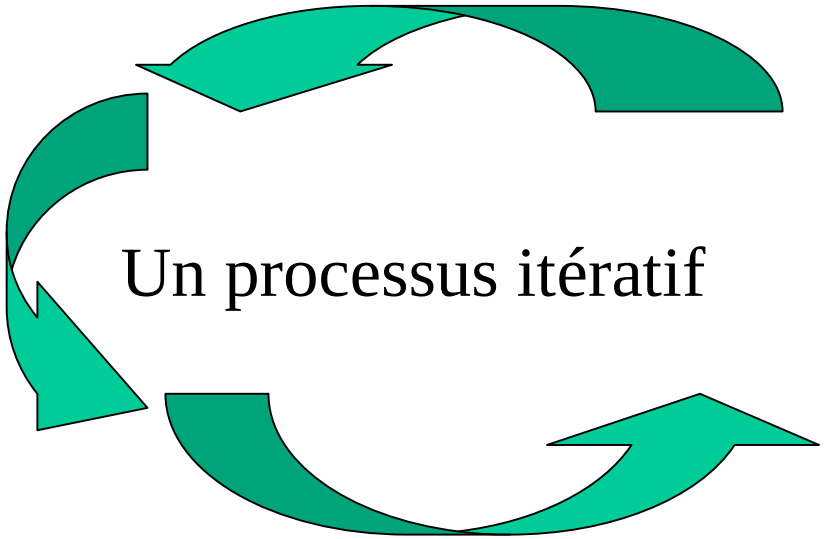
- Désignations évocatrices
- Algorithmes en pseudo-code concis et clairs
- Programmes indentés et commentés

Cycle de vie d'un programme

[1] Analyse + spécification

[4] Vérification

[2] Conception



[5] Maintenance

[3] Codification

Une documentation doit être associée à chaque étape.

Cycle de vie d'un programme (2)

- Analyse + spécification
  - Définir clairement le problème
  - Recenser les données
  - Dégager les grandes fonctionnalités
- Conception
  - Organiser les données
  - Concevoir l'algorithme en pseudo-code
- Codification
  - Traduire l'algorithme dans un langage de programmation

Cycle de vie d'un programme (3)

- Vérification (test et mise au point)
  - Utiliser le programme avec des entrées spécifiques
  - Utiliser un outil de mise au point
- Maintenance
  - Adapter le programme existant pour de nouvelles fonctionnalités et/ou pour corriger les erreurs

Critères de qualité d'un logiciel

Aptitude du système :

- Valide : assure ses fonctions (respect du cahier des charges)
- Fiable : fonctionne même dans des conditions anormales
- Flexible, réutilisable : adaptable en tout ou partie à de nouvelles applications
- Extensible : peut être étendu à de nouvelles fonctionnalités
- Portable : se transfère dans différents environnements logiciels et matériels
- Compatible : se combine avec d'autres
- Facile d'utilisation : utilisable par un client

Critères de qualité d'un logiciel (2)

(Suite) Aptitude du système :

- Maintenable : se modifie, se corrige ou s'adapte (clarté du code, commentaires, choix des structures de données)
- Efficace (complexité) : utilise de façon optimale les ressources disponibles :
  - Exécution la plus courte possible
  - Espace mémoire nécessaire le plus petit possible…
- Intègre : protége son code et ses données contre les accès non autorisés
- Correct (preuve)

Raisons d’être des méthodes de programmation

- Augmentation de la taille et de la complexité des logiciels
- Nécessité de construire des programmes corrects, efficaces, vérifiables et modifiables

➔ Méthodologie de conception des programmes

Principes méthodologiques

- Abstraire : Retarder le plus longtemps l'instant du codage
- Décomposer : « …diviser chacune des difficultés en autant de parties qu'il se pourrait et qu'il serait requis pour les mieux résoudre. » Descartes
- Combiner : Résoudre le problème par combinaison d'abstractions
- Mais aussi : Vérifier, modulariser, réutiliser…

Résumé

- Algorithme = partie conceptuelle d'un programme indépendante du langage de programmation.
- Programme = implémentation de l'algorithme dans un langage de programmation et sur un système particulier.
- Écriture : transcription de l'algorithme dans un langage de programmation.