

CM2 : Analyse et conception

ISia2A3 - Algorithmique et programmation structurée

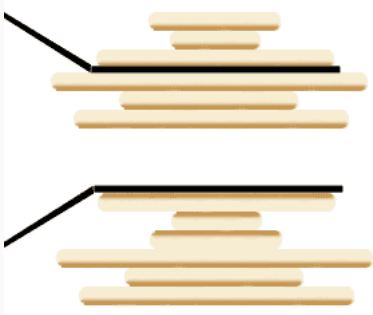
Thibault Liétard

2026-2027

Définition

Un algorithme est une suite d'instructions à suivre pour résoudre un problème donné.

Exemple : le problème du tas de crêpes



Méthode d'écriture

- Analyse
- Conception

Attention

Un algorithme est indépendant du langage de programmation. Il répond à un formalisme qu'on appelle **pseudo-code**.

Étapes

- Identifier les informations manipulées
- Étudier LES méthodes de résolution possibles
- Les formuler
- Comparer les coûts (mémoire et exécution)

Exemples

Exemple 1

Calculer la somme des x^n pour n donné.

Exemple 2

Trouver le maximum d'un vecteur ainsi que le nombre de fois où il apparaît

Exemple 3

Voir si un nombre donné x apparaît dans un vecteur ordonné V .

Principe

Découper l'algorithme (action) en sous-algorithmes (sous-actions) plus simples, jusqu'à des opérations considérées comme primitives au niveau algorithmique

Buts

- Simplification (problèmes plus simples à résoudre)
- Structuration (meilleure lisibilité, maintenabilité, testabilité)

Exemple

Action *nbMax*

- initialiser *max* et *nbMax*
- $\forall i \in [1; N - 1]$, traiter $v[i]$

Formalisme utilisé

- Pseudo-langage (structures de contrôle, primitives lecture/écriture : lire et écrire)
- Actions et fonctions paramétrées

Action <nom d'action> (paramètres)

D : <déclaration> {signification}

D/R : <déclaration> {signification}

R : <déclaration> {signification}

L : <déclaration> {signification}

<corps>

FinAction

Vocabulaire

- **Paramètres formels** : dans la définition d'une action ou fonction. Servent à définir l'algorithme indépendamment de données réelles.
- **Données** : informations fournies à l'algorithme
- **Résultats** : informations produites par l'algorithme
- **Données/Résultats** : informations modifiées
- **Variables locales** : informations temporaires nécessaires au déroulement de l'algorithme

Attention

Une déclaration se fait sous la forme <nom> :<type>

Exemple : nbMax

Action initNbMax(val, max, nb)

D : val : entier {valeur initiale du max}

R : max, nb : entiers {le maximum et son nombre d'occurrence}

nb ← 1

max ← val

FinAction

Exemple : nbMax

Action traiter(ent, max, nb)

D : ent : entier {entier courant à traiter}

R : max, nb : entiers {le maximum et son nombre d'occurrence}

Si ent=max alors

nb←nb+1

Sinon

Si ent>max alors

initNbMax(ent, max, nb)

FinSi

FinSi

FinAction

Exemple : nbMax

Action nbMax(v, N, max, nb)

D : v : vecteur d'entiers {N entiers à traiter }

N : entier {taille du vecteur d'entiers}

R : max, nb : entiers {le maximum et son nombre d'occurrence}

initNbMax(v[1], max, nb)

Pour i allant de 2 à N faire

traiter(v[i], max, nb)

FinAction

paramètres formels vs effectifs

- **paramètre formel** : sert à définir l'algorithme indépendamment de données réelles.
- **paramètre effectif** : désigne une information manipulée par l'algorithme appelant sur laquelle appliquer l'action appelée

Association

- Autant de paramètres effectifs à l'appel que de paramètres formels
- L'association se fait par position (pas par nom)
- Ils doivent être de même type

paramètres formels vs effectifs

- **Données** = valeur ou expression
- **Résultat** = Variable (pour stockage du résultat)
- **Données/Résultats** = Variable évaluée

nbMax

Dans les 3 actions précédentes, identifier les paramètres formels et effectifs.

Avantages

- Sous-algorithmes plus simples, isolables
- Structure et lisibilité
- Réutilisation possible
- Tests plus facile

Attention

Ne pas nuire à l'efficacité ou à la lisibilité par excès inverse

Action

- Sous-programme manipulant des paramètres D, D/R, R
- = instruction

Fonction

- Sous-programme manipulant des paramètres D **uniquement**
- Produit **un seul** résultat

Définition d'une fonction

Fonction <identificateur> (paramètres) : <type résultat>

D : <déclaration> {signification}

L : <déclaration> {signification}

 <corps>

Retourner (<expression>)

FinFonction

Bien sûr, <expression> est de type <type résultat>.

Différence fonction/action 1/3

Action max(a, b, m)

D : a, b : entiers {valeurs à comparer}

R : max : entier {maximum de a et b}

Si a<b alors

m ← b

Sinon

m ← a

FinSi

FinAction

Différence fonction/action 2/3

Fonction max(a, b) : entier

D : a, b : entiers {valeurs à comparer}

L : max : entier {maximum de a et b}

Si a < b alors

 m ← b

Sinon

 m ← a

FinSi

Retourner (m)

FinFonction

Fonction max(a, b) : entier

D : a, b : entiers {valeurs à comparer}

Si a < b alors

Retourner (b)

Sinon

Retourner (a)

FinSi

FinFonction

Principe

- considéré comme une expression
- peut donc apparaître dans n'importe quelle expression
- même règle de paramétrage que pour les actions

Exemple

`max(max(a,b),c)`

Programmation impérative

Style de programmation dans lequel les instructions qui modifient les données sont exécutées simplement les unes après les autres (C, Java, Python,...)

Programmation fonctionnel

Style de programmation n'utilisant ni affectation ni séquencement d'instructions. Programmation de fonctions qui appellent d'autres fonctions, qui appellent des fonctions, ... (Caml, Lisp,...)

Principe

- **Analyse** : définir les informations manipulées et attendues (D, D/R, R, L) et leur type
- **Conception** : écrire l'action ou la fonction réalisant le traitement