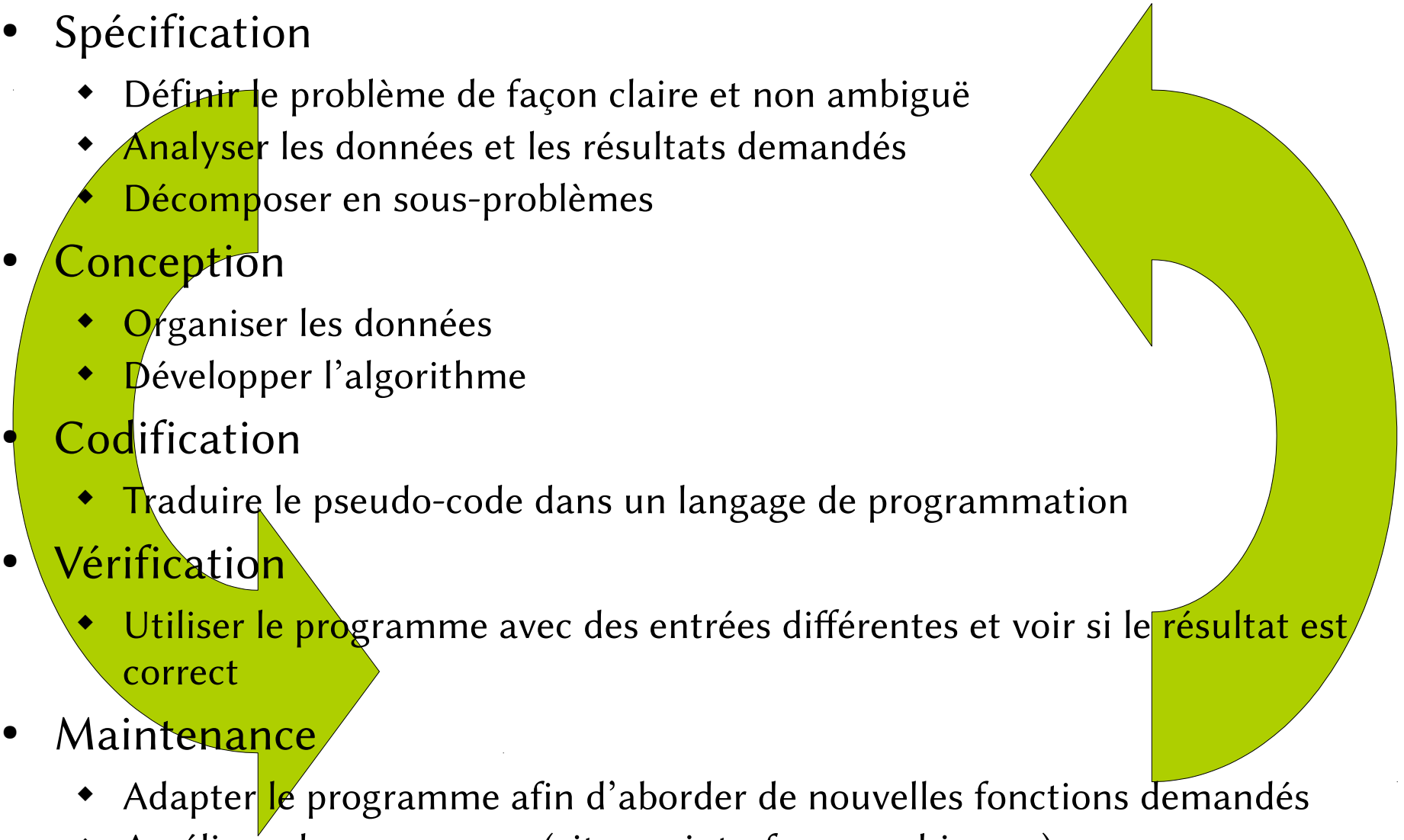
Cycle de vie d'un programme

- Spécification
  - Définir le problème de façon claire et non ambiguë
  - Analyser les données et les résultats demandés
  - Décomposer en sous-problèmes
- Conception
  - Organiser les données
  - Développer l'algorithme
- Codification
  - Traduire le pseudo-code dans un langage de programmation
- Vérification
  - Utiliser le programme avec des entrées différentes et voir si le résultat est correct
- Maintenance
  - Adapter le programme afin d'aborder de nouvelles fonctions demandés
  - Améliorer le programme (vitesse, interface graphique...)



Documentation

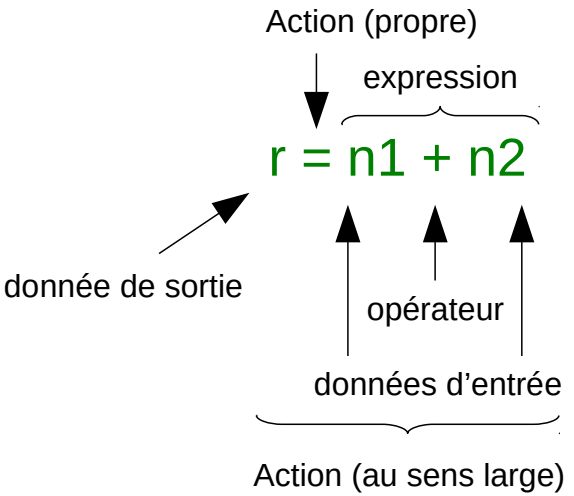
- Programmer, c'est communiquer
  - avec l'interpréteur, compilateur…
- mais aussi
  - avec les autres
  - avec soi-même (dans le futur)
- Important : bien documenter son travail
  - désignations évocatrices qui décrient le contenu ou le rôle d'une variable, fonction… : *hauteur* et *calculer_longueur* au lieu de *a* et *fonction1*
  - algorithmes en pseudo-code ou en diagrammes
  - indentation du code
    - si a = 2 alors
      afficher « a est égal 2 »
      sinon
      afficher « a n'est pas égal 2 »
  - commentaires dans le code
    - si b=0, l'equation n'a pas de solution
    - formule a consulter avec Mme la comptable

2. Éléments d'un langage de programmation ; introduction au langage Matlab et à l'environnement Octave

Syntaxe et instructions
Données et types simples
Types composés : tableaux, chaînes de caractères
Affectation
Opérateurs arithmétiques
Obtention et affichage de données

Éléments d'un langage de programmation

- Données ; classifiées selon leur rôle comme :
  - données d'entrée (entrées, arguments, paramètres [d'entrée], données)
  - données internes (intermédiaires, auxiliaires)
  - données de sortie (sorties, résultats, valeurs de retour, paramètres de sortie)
- Expressions
  - décrient les relations entre les données
  - principalement à l'aide d'opérateurs
- Actions
  - commandes, instructions, procédures, fonctions, méthodes...
  - affectation – action spécifique, souvent notée avec un opérateur et même considérée opérateur
- Structures de contrôle
  - modifient le cours d'exécution du programme : conditions, répétitions...



Nous allons nous baser sur Matlab/Octave mais nous essaierons d'aborder (dans la mesure du possible) tous les langages en général

Syntaxe

- Tout comme un langage humain, chaque langage de programmation possède sa propre syntaxe
  - plus systématique, formalisée et stricte que celles des langages humains
  - pour que les codes soient plus faciles à analyser par l'ordinateur
  - des synonymes, homonymes, alternatives, exceptions etc. sont très rares
- La **syntaxe** précise :
  - comment on saisit des valeurs et désigne des données, des actions…
  - comment on compose des expressions à l'aide d'opérateurs, fonctions…
  - comment on compose des instructions
  - comment on compose un programme et ses parties plus petites
- Le « comment » veut dire :
  - quels caractères on utilise pour un élément propre du langage
  - quels caractères servent de séparateurs, discriminants, marques de début, fin ou continuation…
  - dans quelle ordre ces éléments peuvent ou doivent apparaître
  - pour que l'ordinateur comprenne quels actions il doit exécuter

Spécificités de Matlab concernant les instructions

- **Instruction** est l'étape élémentaire d'un programme ou le fragment élémentaire d'un code
- La fin de ligne est équivalente à la fin de l'instruction
  - Pour continuer une instruction dans une ligne suivante, utiliser ... ou \
  - `a = ...` ou `a = \`
`b + 2` `b + 2`
- Affichage du résultat
  - Si une instruction finit avec le point-virgule ; , son résultat n'est pas affiché
  - Dans tous les autres cas, le résultat est affiché
- Autre application du point-virgule
  - Il sert a placer plusieurs instructions dans une ligne
`a = 0; b = pi; c = sin(a) + cos(b);`
  - Si on veut afficher les résultats d'instructions particulières, alors on les sépare avec la virgule ,
`a = 0, b = pi, c = sin(a) + cos(b)`
- Si aucune variable n'est affectée explicitement, le résultat est enregistré dans la variable spéciale *ans*
  - Après chaque instruction, son contenu est remplacé

Données

- Une **donnée** peut être considérée comme une boîte contenant une information (**valeur**) d'une certaine forme (**type**), caractérisée par une étiquette (**nom**)
- Une donnée peut être :
  - **constante** – si sa valeur ne peut changer pendant l'exécution du programme
  - **variable** – si sa valeur peut être changée pendant l'exécution du programme
- Pour les noms (désignations), dans la plupart des langages il existe de très similaires :
  - règles :
    - peut être composé seulement des caractères ASCII a...z, A...Z, 0...9 et _
    - mais ne peut pas commencer avec une chiffre
  - conventions :
    - minuscules pour les variables, majuscules pour les constantes
    - si composé de plusieurs mots, on les délimite avec le tiret bas ou avec une majuscule et on omet les prépositions, p. ex. *adresseDomicile*, *AdresseDomicile* ou *adresse_domicile* (et non *adressedudomicile*)
    - en Matlab : *adresseDomicile* (variable) et *DELAI_PAIEMENT* (constante)

Spécificités de Matlab concernant les données

- Pas de nécessité de déclarer (annoncer) les variables et leurs types
  - toute variable est en fait un tableau de nombres
  - matrice, vecteur, nombre, caractère, chaîne de caractères...
- Matlab discerne les majuscules et les minuscules dans les noms des variables (et fonctions)
- Toute variable numérique est réelle
  - pas de nombreux types différents comme entier petit, entier grand, réel petite précision, réel grande précision...
  - plus facile à utiliser, parfois cause des problèmes
  - moins économique en termes de mémoire et vitesse d'opérations
- Toute variable peut être complexe (a + b*i)
- Très simple d'utiliser des matrices (et vecteurs)
- Une chaîne de caractères est un vecteur contenant les codes ASCII de ces caractères un par un

Types des données

- Les trois **types fondamentaux** (simples) des données sont :
  1) numérique
  2) alphanumérique
  3) logique
- En fait toute donnée est numérique parce ce qu'un processeur ne travaille qu'avec des nombres (binaires)
- Mais ça ce manifeste de manière différente dans de différents langages haut-niveau qui tentent de cacher ce fait
  - logique : p. ex. faux ⟺ 0, vrai ⟺ 1
  - alphanumérique : p. ex. A ⟺ 65 (selon le jeux de caractères utilisé)
    - ASCII = ISO 646 – code américain (1963), 128 caractères (0h…7Fh, 7 bits)
    - extensions de ASCII à 256 caractères (0h…FFh, 8 bits), p. ex. ISO 8859 – longtemps populaires mais problématiques : ne couvraient que de petits groupes d'alphabets et plusieurs jeux existaient pour un même groupe
    - ISO 10 646 (Unicode) – 1 114 112 codes (0h…10FFFFh, 21 bits) mais seulement une partie (env. 100 000) est utilisée ce qui permet d'utiliser 8 bits pour latin basique et 16 bits pour la plupart des alphabets/syllabaires

« Constantes » numériques de Matlab

- Nombres:
  - pi = 3,1416... le nombre π
  - i = $\sqrt{-1}$ l'unité imaginaire
  - e = 2,7183... la base du logarithme naturel
- Valeurs logiques
  - true = 1 vrai
  - false = 0 faux
- Attention :
  - En Matlab, il n'y a pas de vraies constantes ; ce sont des variables avec valeurs prédéfinies
  - Il est alors possible de modifier leurs valeurs
  - Jamais n'appelez vos variables comme les variables prédéfinies de Matlab
  - De même il est possible de changer les définitions des fonctions prédéfinies de Matlab
  - Jamais n'appelez vos fonctions comme les fonctions prédéfinies de Matlab

Types composés – tableaux

- Les **types composés** (évolués) des données sont :
  - tableaux
  - chaînes de caractères (chaînes) – genre de tableau
  - d'autres **structures de données** – enregistrements, listes, arbres, piles…
- Un **tableau** est une suite de données du même type
  - Il correspond à la notion de matrice ou vecteur en mathématiques
  - Tout élément possède une ou plusieurs **indices** (en fonction du nombre des dimensions du tableau qui peut être 1, 2 ou plus)
  - Numérotation des éléments la plus évidente : 1, 2, 3, 4… (Matlab)
  - mais beaucoup de langages commencent par 0 – c'est plus efficace
- Un **enregistrement** est un ensemble de données qui peuvent être de types différents
  - données d'un employé : numéro d'identification – nombre
prénom, nom – chaîne de caractères
date de naissance – nombre(s) ou chaîne(s)

Saisie des matrices en Matlab

- Une matrice type
  - [*a11, a12, a13 … ; a21, a22, a23 …*]
  - [1, 4, 6 ; 2, -1, 0.5 ; 3, 6, -2 ; 4, -7, 18] ⟺ $\begin{bmatrix} 1 & 4 & 6 \\ 2 & -1 & 0{,}5 \\ 3 & 6 & -2 \\ 4 & -7 & 18 \end{bmatrix}$
  - les crochets [ ] signalisent qu'on va composer une matrice à partir de plusieurs éléments
  - la virgule , sépare les éléments d'une ligne (les colonnes) ; remplaçable avec l'espace
  - le point-virgule ; sépare les lignes
- Un vecteur ligne (matrice 1×n)
  - [*a1, a2, a3 …*]
- Un vecteur colonne (matrice n×1)
  - [*a1; a2; a3 …*]
- Un élément simple (matrice 1×1)
  - [*a*]
- Une matrice vide (0×0)
  - []

Création raccourcie des matrices en Matlab

- Si les éléments forment une suite arithmétique
  - Raccourci : [*premier* : *raison* : *dernier*] (produit toujours un vecteur ligne)
  - [2 : 0.5 : 4.5] équivaut [2 2.5 3 3.5 4 4.5]
  - Si *raison* est omise, 1 sera supposé
  - [2 : 5] équivaut [2 3 4 5]
  - Pour transformer une ligne en une colonne, il suffit de la transposer (')
- Matrice d'uns
  - ones(*nombre_lignes, nombre_colonnes*)
  - ones(2,3) $\Longleftrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - ones(*taille*) produit une matrice carrée et non pas un vecteur !
  - ones(1,3) $\Longleftrightarrow \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$
  - ones(3) $\Longleftrightarrow \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
  - par analogie – matrice de zéros : zeros
- Matrice d'identité
  - eye(*taille*)
  - eye(3) $\Longleftrightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Manipulations des matrices en Matlab

- Combinaison
  - matrices placées l'une à droite de l'autre = ajout de colonnes
    [M1, M2, M3 ...]
  - matrices placées l'une au dessous de l'autre = ajout de lignes
    [M1; M2; M3 ...]
  - les *Mx* désignent des matrices en général ; ça comprend également – en tant que cas spécifiques – vecteurs (n×1 / 1×n) ou éléments simples (1×1)
- Extraction
  - un élément d'une matrice : *M(no_ligne, no_colonne)*
  - un élément d'un vecteur : *V(no_element)*
  - une colonne d'une matrice : *M(:, no_colonne)*
  - une ligne d'une matrice : *M(no_ligne, :)*
  - plusieurs colonnes : *M(:, premiere_colonne : derniere_colonne)*
  - plusieurs lignes : *M(premiere_ligne : derniere_ligne, :)*
  - une sous-matrice :
    M(premiere_ligne : derniere_ligne, premiere_colonne : derniere_colonne)

Taille des matrices

- Détermination du nombre d'éléments en Matlab :
  - numel(*M*) – nombre d'éléments total
    - > matrice = [3 5 ; 7 -2 ; -9 11];
    - > numel(matrice)
      ans = 6
  - size(*M*) – nombre de lignes et nombre de colonnes
    - > size(matrice)
      ans = 3 2
    - > [nl, nc] = size(matrice)
      nl = 3
      nc = 2
  - Cela fonctionne aussi avec des tableaux à plus de 2 dimensions
- Pour le cas standard de 2 dimensions on peut se servir également de :
  - columns(*M*) – nombre de colonnes
  - rows(*M*) – nombre de lignes

Dans les exemples d'instructions Matlab entrées en ligne, « > » va signaliser une entrée dans la ligne de commandes d'Octave