

# Algorithmique & Programmation Objet

Xavier Granier <[xavier.granier@institutoptique.fr](mailto:xavier.granier@institutoptique.fr)>

Romain Pacanowski <[romain.pacanowski@institutoptique.fr](mailto:romain.pacanowski@institutoptique.fr)>

# But du cours

---

- Entrapercevoir les fondations de l'informatique
- Saisir les **capacités** de l'outil informatique
  - Capacités de calcul / de stockage
- Savoir comment concevoir un algorithme
  - Évaluer sa **complexité** / son **efficacité**
  - Évaluer son **coût** de calcul / de stockage
- Bases de la **programmation objet**

# Les non-buts du cours

---

- Être expert en informatique !
- Être expert en programmation objet

# Programmer, c'est ?

---

- Aller vers un **monde fini**
  - Formulation **discrète vs continue**
    - Nombre fini de valeurs / monde discret
    - Un nombre fini d'étapes et d'actions
    - Compter / Énumérer
  - Ressources limitées
- Notions
  - **Algorithmes**
  - **Automates**

# Machine & Automates

# Alan Turing

- 1912 – 1954
- Mathématicien britannique
- Notions de « procédure mécanique » ou algorithme
- Formalisation d'une machine (1936)
  - Mémoire (écriture / lecture)
  - Règles

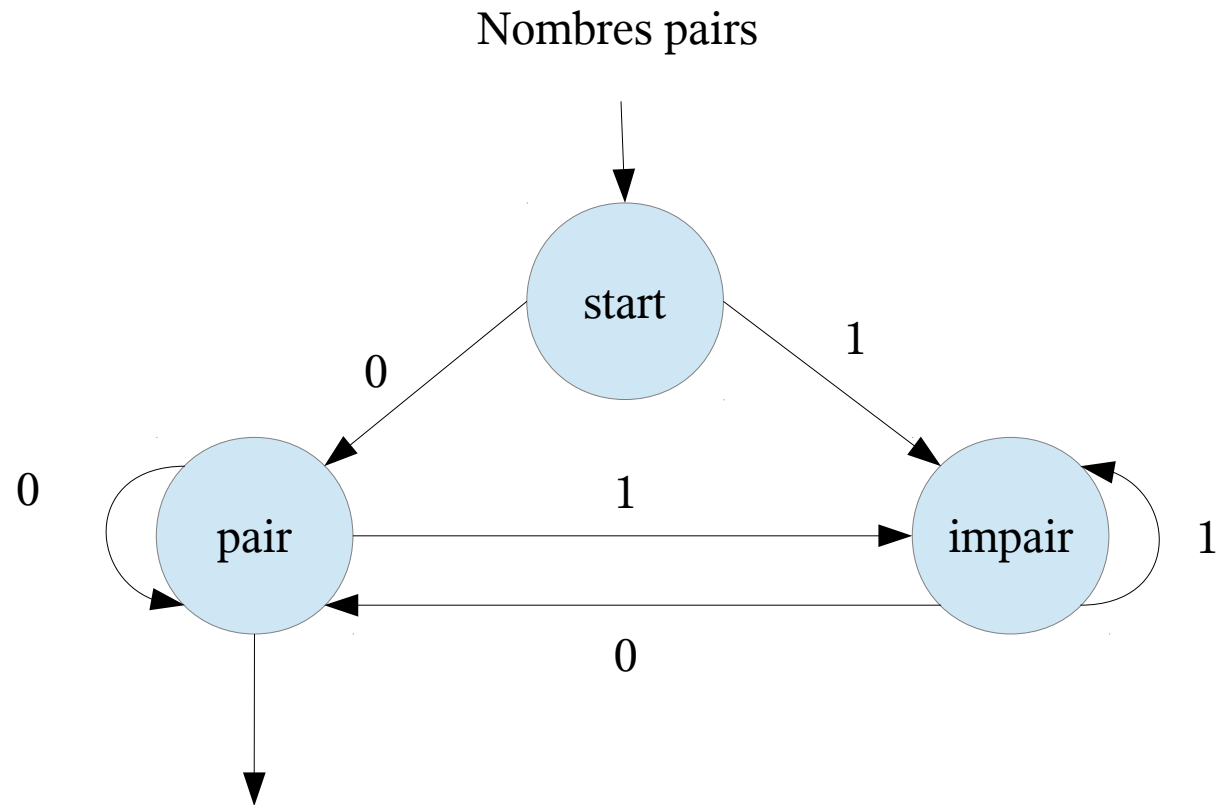


# Un ordinateur

---

- Le modèle : **machine de Turing**
- La mise en place : électronique
  - Le transistor : ouvert ou fermé
  - 2 **états** : 0 ou 1
  - Même état tant qu'aucune action ne le change

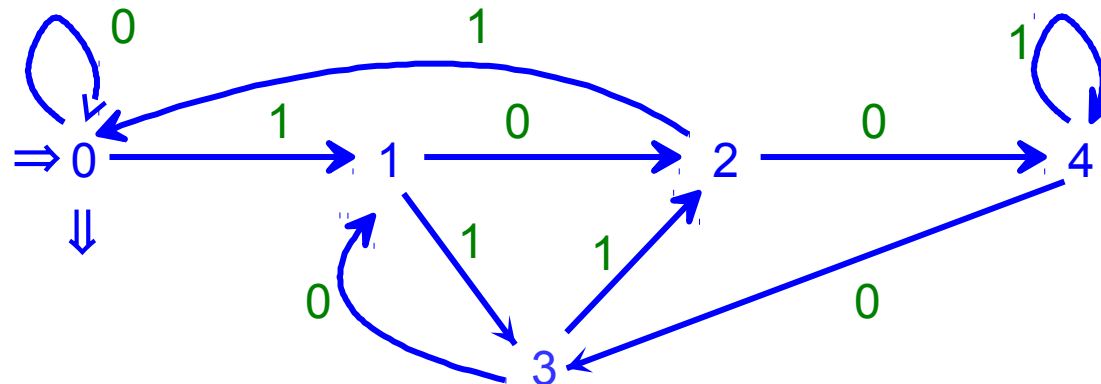
# Exemple d'automate 1





# Représentation

- Fonction de transition d'un état vers un autre
  - Graphique : listes des flèches
  - Mathématique : matrice d'adjacence
  - Informatique : listes des successeurs
  - Informatique : table de transition



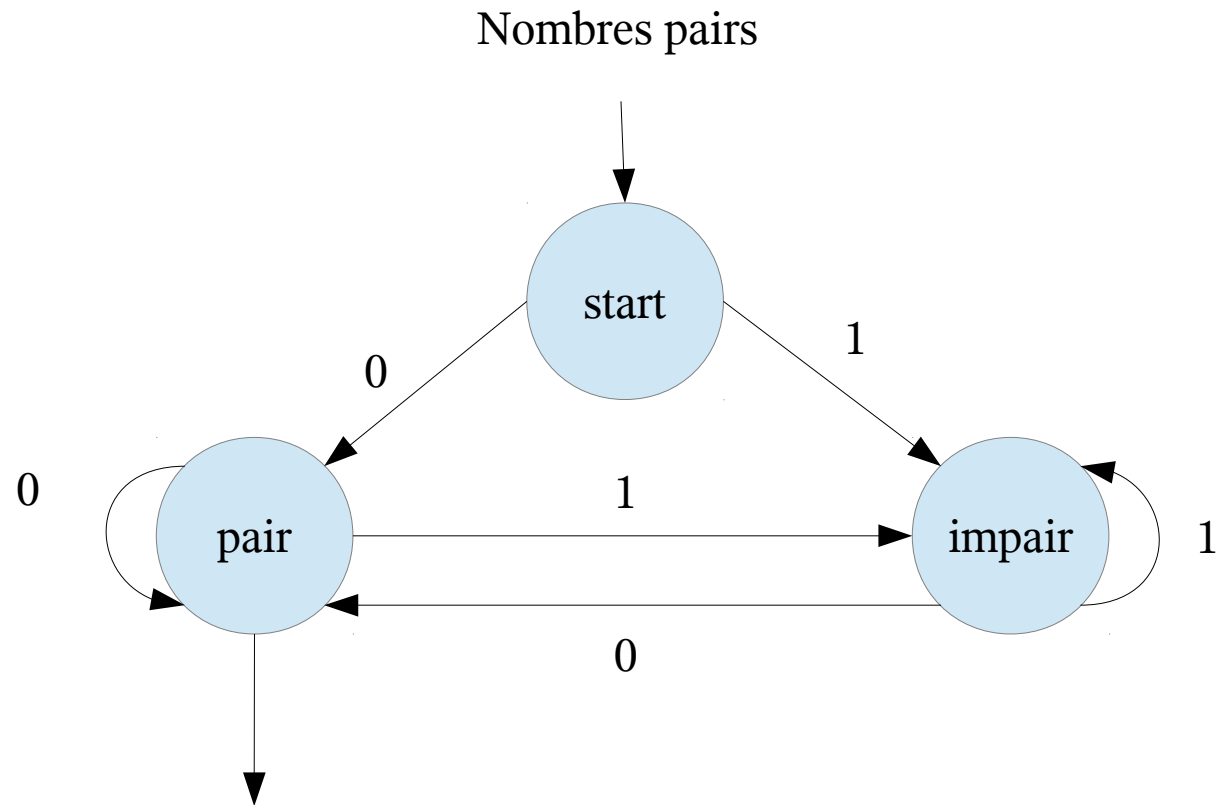
$\delta$	0	1
0	0	1
1	2	3
2	4	0
3	1	2
4	3	1

écritures en base 2 des multiples de 5

# Automate fini

- Définition : quintuplet  $(Q, V, \delta, I, F)$ 
  - $Q$  : ensemble fini d'**états**
  - $V$  : ensemble fini de symboles (**vocabulaire**)
  - $\delta$  : ensemble fini de **transitions**  
triplets  $(q, v, q)$  avec  $q \in Q$  et  $v \in V \cup \{\epsilon\}$
  - $I \subset Q$  : ensemble des **états initiaux**
  - $F \subset Q$  : ensemble des **états finals**

# Exemple d'automate 1



Exercice : définir les états (initiaux, finals), le vocabulaire et les transitions

## Exercice automate 2

---

- Faire un automate reconnaissant les triangles
  - Vocabulaire  $\{s\}$  où  $s$  est un sommet

# Retenir

---

Un machine c'est :

- Un nombre fini d'états
- Une capacité de stockage
- Un ensemble de transitions

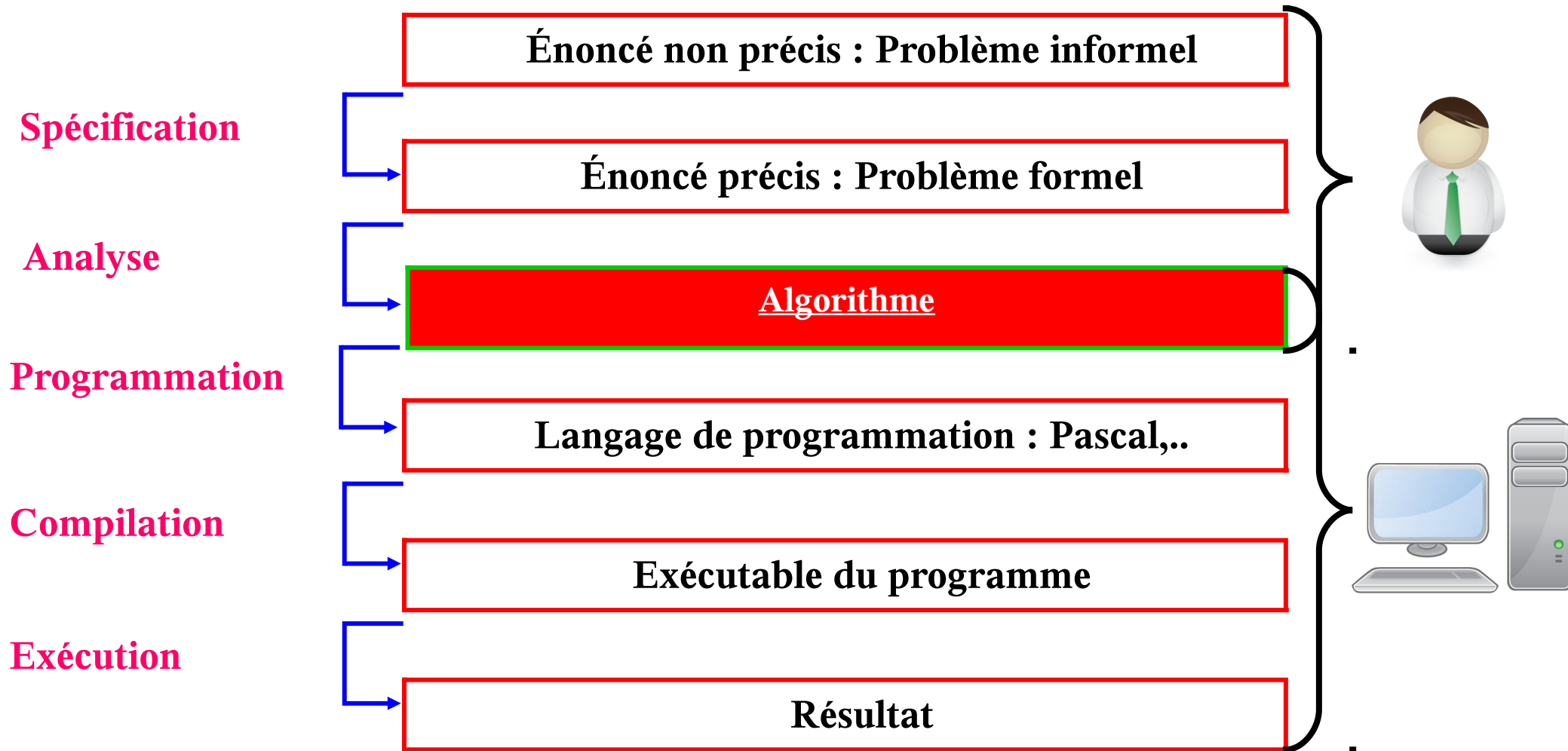
# Algorithmes

# Algorithme

---

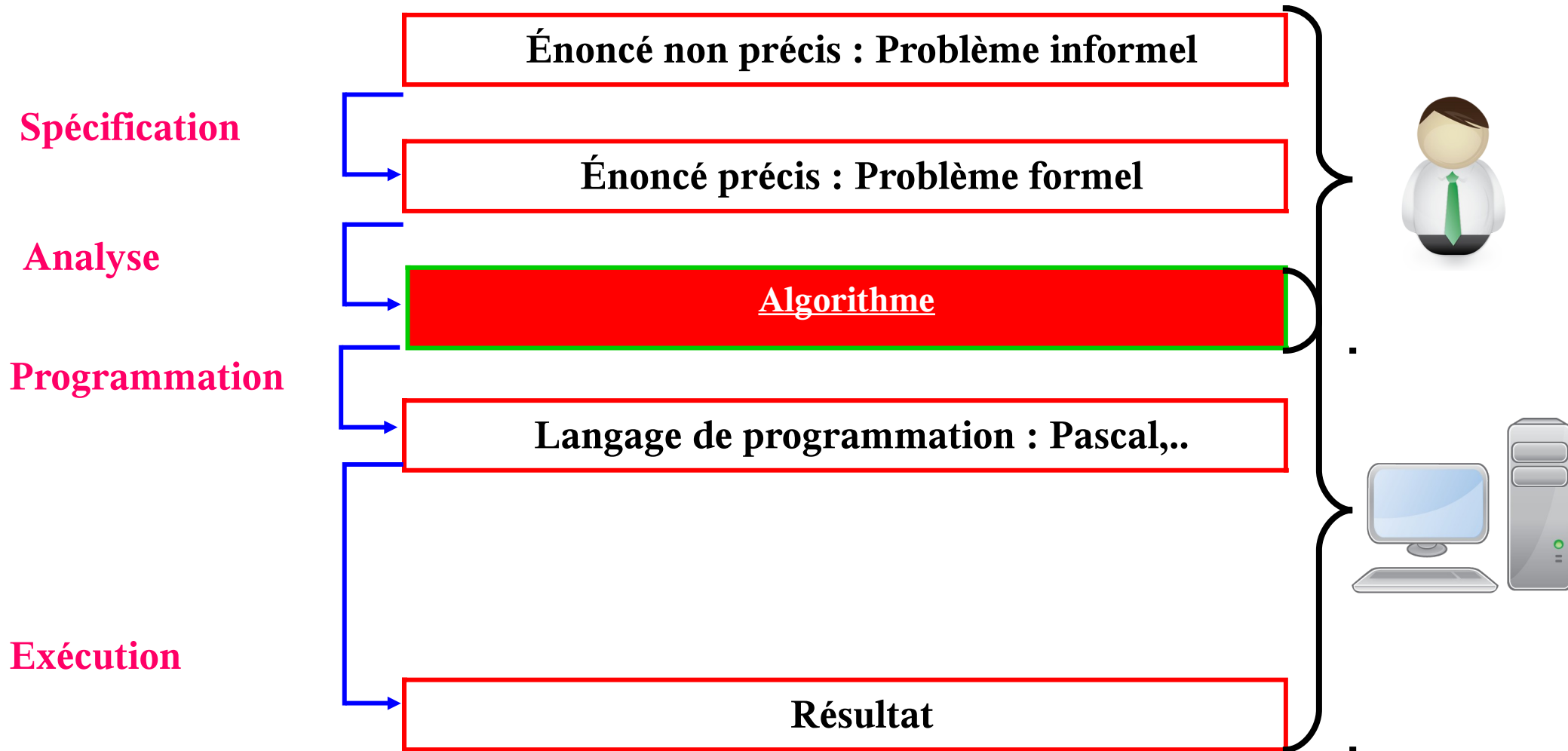
- Nom masculin
  - d'Al-Khârezmi, médecin arabe
- **Suite** de raisonnements ou d'opérations
  - qui fournit la solution de certains problèmes.
- Objectif
  - décrire les **étapes** à suivre pour réaliser un travail.
  - expliciter clairement les idées de solution d'un problème **indépendamment** d'un langage de programmation.

# Du problème au programme





# Du problème au programme



# Formalisation d'un problème

---

- Quelles sont les données à traiter ?
  - Les paramètres d'une onde ? Des énergies ?
- Quels sont les résultats attendus ?
- Quels sont les a-priori
  - Données connues en prérequis
  - Contexte d'utilisation
  - Domaine de validité, ordre de grandeur
- Quelles sont les actions nécessaires

# Variables et constantes

---

- Constante

- 1 **identificateur**
- 1 **valeur**

constante **pi** = **3.14**

- Variable

- 1 **identificateur**
- 1 conteneur pour 1 **valeur** typée

entier **N**

# Typage

---

- But
  - Éviter de mélanger des éléments incompatibles
  - Facilite la vérification
  - Ex : un entier naturel ne peut être négatif
- Un type
  - Ensemble de valeurs possibles
    - Ex : les 12 mois de l'année
  - Ensemble d'opérations possibles sur ces valeurs
    - Ex : +, -, \*, / (division euclidienne)
  - Une structure : combinaison de types
    - Ex : tableau

# Instructions

---

- Commandes / actions élémentaire
  - Passer d'un état initial vers un
  - Produit un résultat
- Exemples :
  - Affectation :
    - $A \leftarrow 10.324$
  - Opérations arithmétiques
    - $B \leftarrow A * 4.0$
  - Appels de procédures ou des fonctions
    - $C \leftarrow \text{sqrt}(B)$

# Organiser les instructions

---

- Bloc d'instructions
  - Sous-ensemble d'instructions
  - Entre des séparateurs
  - Variables / Constantes locales
- Procédures
  - Bloc d'instructions
  - Paramètre en entrée : in entier **N**
  - Paramètre en sortie : out entier **N**
  - Paramètre en entrée/sortie : inout entier **N**

# Branchements conditionnels

---

Si <condition> alors faire

    <bloc instructions>

Sinon faire

    <bloc instructions>

Fin Si

# Les boucles

---

Pour <var> = <début> jusqu'à <condition> faire  
    <bloc instructions>

Fin Pour

Tant que <condition> faire  
    <bloc instructions>

Fin Pour

Faire  
    <bloc instructions>

Jusqu'à <condition>



# Organiser les instructions

---

- Procédures
  - Bloc d'instructions
  - Paramètre en entrée : in entier **N**
  - Paramètre en sortie : out entier **N**
  - Paramètre en entrée/sortie : inout entier **N**

# Organiser les instructions

---

- Fonctions
  - Bloc d'instructions
  - Paramètre en entrée : in entier **N**
  - Paramètre en sortie : out entier **N**
  - Paramètre entrée/sortie : inout entier **N**
  - Retourne un résultat

# Validation d'un algorithme

---

Il faut prouver que

- Le programme s'arrête ?
  - Mettre des conditions d'arrêt
  - Pour le contexte d'utilisation
    - S'assurer qu'elles soient atteignable
- Le programme donne le bon résultat
  - Dans le contexte d'utilisation

# Exercice

---

- Hypothèse :
  - On a les opérateurs  $*$ ,  $+$ ,  $-$ , `ispair(in entier a) : bool`
- Créer la fonction
  - `euclide(in entier a, in entier b, out entier r) : entier`
  - `r` est le résidu
  - le résultat est le quotient

# Programmation

# Programmer, c'est ?

---

- Savoir parler une **langue étrangère**
  - Impératif : C / Pascal / Ada / Assembleur / ...  
Séquences d'instructions exécutées
  - Objet : C++ / Java / C# / Ada  
Définition de blocs et de leur interactions
  - Fonctionnel : Scheme / Lisp  
Définition de fonctions
  - Logique : Prolog  
Définition de règles

# Prolog (programmation logique)

---

```
fact(1, 1).  
fact(A, B) :-  
    fact(C, D),  
    C is A-1,  
    B is A*D.
```

-----

```
?-fact(10,R).
```

# Scheme (programmation fonctionnelle)

---

```
(define (factorial n)
  (if (= n 0)
      1
      (* n (factorial (- n 1)))))
```

-----

```
(factorial 10) ;;
```



# Ada (programmation impérative)

---

```
function factorial (n: natural) return positive is
begin
    if n=0 then
        return 1 ;
    else
        return n * factorial(n-1) ;
    end if;
end factorial ;
```

-----

```
factorial(10) ;
```

# Programmer, c'est ?

---

- Savoir parler une **langue étrangère**
  - Impératif : C / Pascal / Ada / Assembleur / ...
  - Objet : C++ / Java / C# / Ada
  - Fonctionnel : Scheme / Lisp
  - Logique : Prolog
- Notions Communes:
  - Définition d'un **langages**

# Théorie des langages

---

- Un langage, c'est ?
  - Langue naturelle
  - **Système de notation** (formalisme)
    - Mathématiques, logique, chimie,
    - Informatique
- Un ensemble d'objets que l'on peut composer
  - Plusieurs niveaux
    - Alphabet → mots
    - Mots → phrases

# Langage

---

- Définition
  - Ensemble d'objets élémentaires
    - Vocabulaire ou alphabet
  - Suites d'objets élémentaires ayant un «sens»
    - Syntaxe
  - Sens des suites
    - Sémantique
- Deux niveaux
  - **Lexical** : alphabet  $\rightarrow$  mot
  - **Syntaxique** : mot  $\rightarrow$  phrase

# Classification des langages

---

- Langage compilé
  - Le code est converti une fois pour toute en instruction machine.
  - Une fois cela fait, on peut exécuter le programme généré autant de fois que l'on veut, mais on ne peut plus changer le programme
- Langage interprété
  - La conversion se fait au vol, lors de la lecture du code.
  - Le code peut être modifié au vol
- Java : compilé et interprété !
  - Machine virtuelle.