CM2 - Introduction à la programmation

RB14

Florence Zara - Université Lyon 1 - LIRIS

E-mail: florence.zara@liris.cnrs.fr

Objectif du cours

Comprendre le terme programmation

Savoir comment des applications sont créées

Savoir ce que c’est un langage de programmation

Savoir comment un programme informatique est écrit

Il y a de la programmation partout…

Quelques exemples au quotidien

- Réveil
  - programmer l'heure de la sonnerie
  - puce contenue dans le réveil permet de stocker des instructions très simples
- Box
  - programmer un enregistrement
- Porte d'entrée
  - composer le code et ouvrir la porte

Programmation et informatique

- On utilise les ordinateurs au travers de programmes
  - le système d'exploitation
  - les applications utilisateur

  } programmés par d'autres

  - des programmes que l'on fait soi-même
    - macros suite bureautique (macro VBA dans Excel)
      - automatisation des tâches
    - des programmes plus « importants »

Programmeur et utilisateur

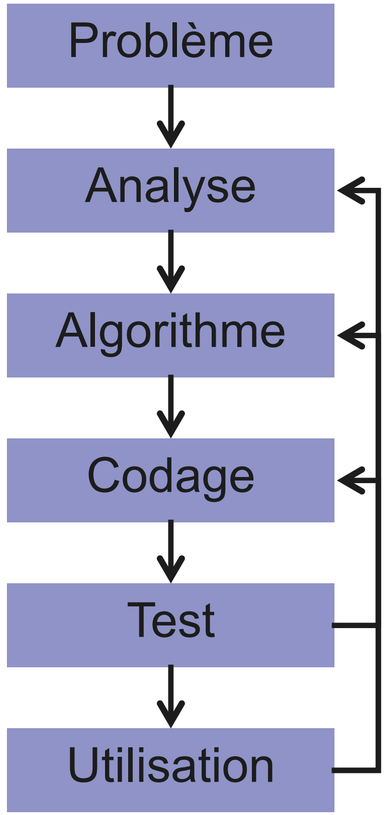
- **Programmeur**
  - conçoit et fabrique un programme
    - qui rendra des services à un utilisateur
    - en fonction d'une commande (les besoins exprimés)

- **Utilisateur**
  - utilise un programme informatique au cours de son activité
    - ne l'utilise jamais exactement comme le concepteur l'a prévu

- Remarque
  - un programmeur est un utilisateur d'un programme informatique destiné à aider à la conception de programmes informatiques
    - Visual C++ (Windows), KDevelop (linux), Xcode (Mac)

Du problème au programme

- **Problème**
- **Analyse**
  - données d'entrée
  - résultats
  - traitements
  - cas critiques
- **Algorithme**
  - indépendant du langage de programmation
- **Codage**
- **Tests**
  - simulations : vérification des cas critiques
- En cas d'erreur, on retourne en arrière



Algorithme

- **Première définition**
  - **décrit comment un humain ou une machine peuvent réaliser un objectif**
    - suivre une recette de cuisine (objectif : fabriquer une recette)
    - décomposer un numéro de Sécurité Sociale (objectif : extraire des informations sur le possesseur d'un numéro de SS)
    - utiliser les transports en commun (objectif : venir à l'Université)

- **Deuxième définition**
  - **suite d'actions**
    - chaque action est décrite par une ou plusieurs instructions
  - **à appliquer à des données**
    - indépendamment de leurs valeurs
  - **pour obtenir un résultat**
    - en un nombre fini d'étapes
      (doit s'arrêter après un certain temps)

- **Remarque**
  - devrait prévoir tous les cas possibles

Algorithme

- Description « en français » des instructions à faire pour résoudre un problème
- Ensuite traduction de cet algorithme dans un langage de programmation
- Programme écrit dans un langage : écriture du « code source »
- Ce programme sera ensuite compris par la machine

Codage de l'algorithme par une suite d'instructions

- En programmation, une **instruction**
  - décrit une **action élémentaire**
  - est spécifiée par un mot-clé
    - soit fourni par le langage
    - soit défini par le programmeur
  - peut avoir des **paramètres**
    - Sur quelles données est appliquée une instruction ?

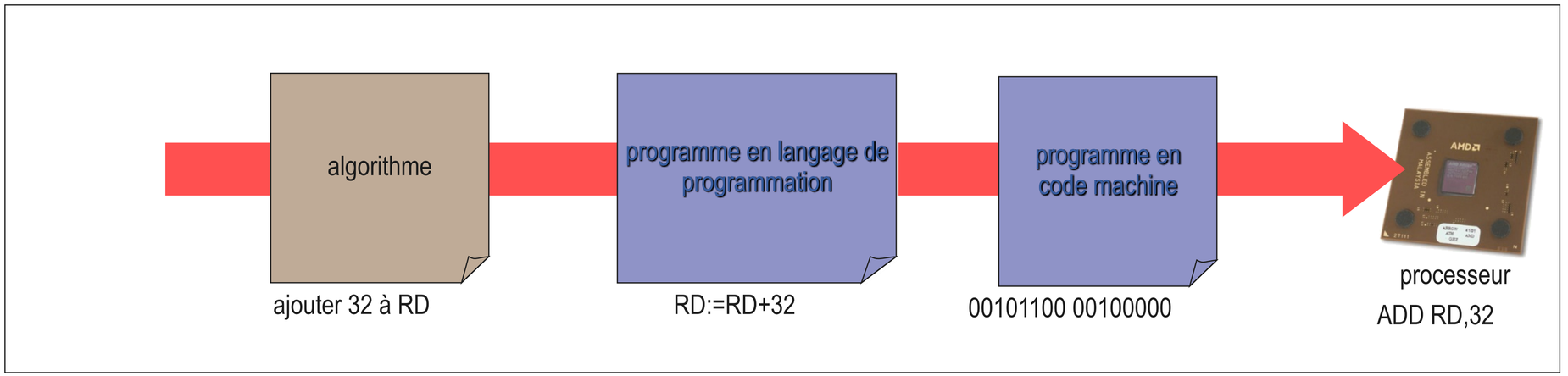
  - **Exemples**
    - trouver la troisième lettre d'une chaîne de caractères
    - prendre un nombre au hasard
    - calculer l'arrondi d'une valeur
    - compter de 1 à 100
    - …

Programme et langage de programmation

- **Programme**
  - c'est la traduction d'un algorithme dans un langage informatique
  - éventuellement découpé en modules (sous-programmes)

- **Langage de programmation**
  - langage intermédiaire entre l'humain et le processeur
  - permet d'exprimer les instructions algorithmiques dans un langage rigoureux

- **Programme en code machine**
  - description binaire du programme, adaptée au système et au microprocesseur



Choix du langage de programmation

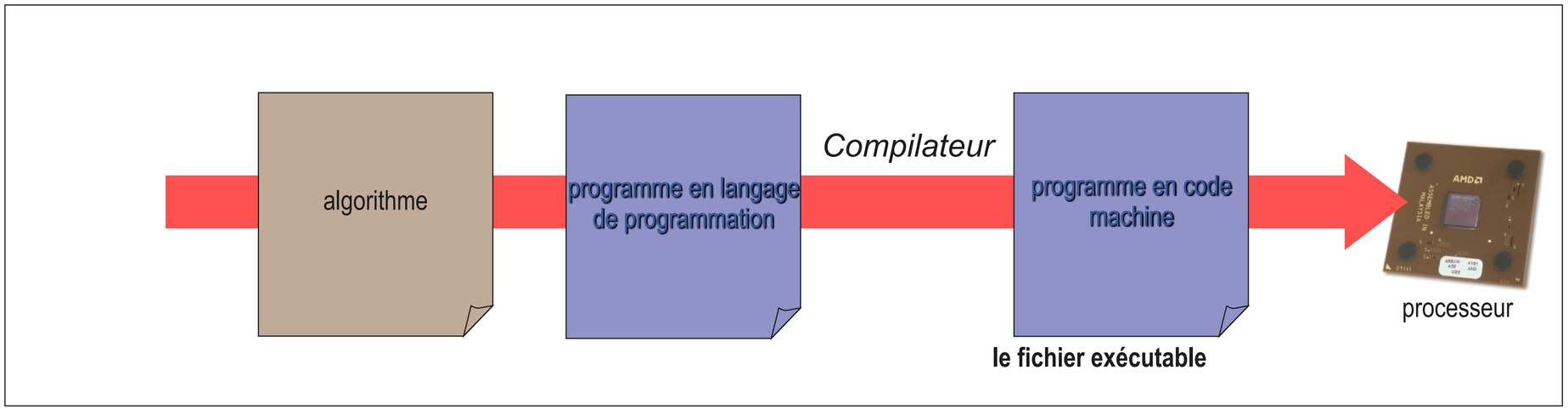
- **Critères à prendre en compte**

  - **Portabilité** : passage d'un type d'ordinateur à un autre (système d'exploitation, carte graphique)
  - **Stabilité** : langage ancien ou récent
  - **Performance** : rapidité d'exécution
  - **Sécurité** : robustesse face aux attaques
  - …

  - Classement des langages de programmation en fonction de leur utilisation (plus de 700 langages de programmation) :
    https://www.tiobe.com/tiobe-index/

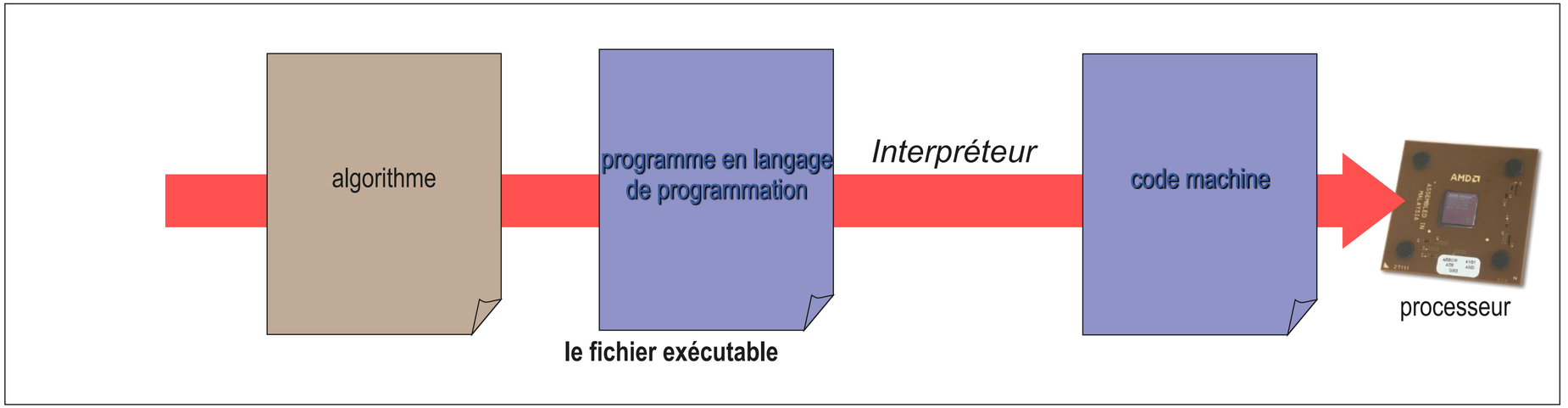
Familles de langages de programmation (1)

- **Langages compilés**
  - le programme décrit dans le langage de programmation (code source) est compilé (traduit) en code machine
  - cette traduction se fait une seule fois, avant l'exécution du programme
  - le programme est stocké sous deux formes
    - il faut le recompiler pour l'exécuter sur un système/machine différent
  - exemples de langages
    - Cobol, Fortran, Pascal, SmallTalk, C, C++, Delphi, Visual Basic…



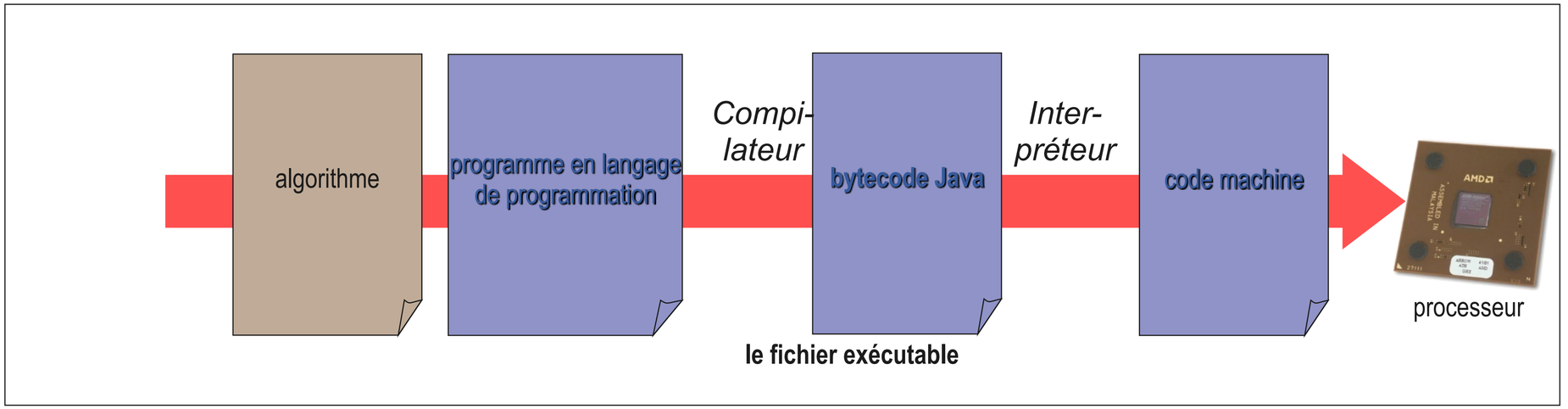
Familles de langages de programmation (2)

- **Langages interprétés**
  - la traduction en code machine se fait à chaque exécution du programme
  - le programme n'est stocké que sous une seule forme, qui est le fichier exécutable
  - il pourra être utilisé tel quel sur plusieurs systèmes/machines différents, si chacun dispose d'un interpréteur
  - exemples de langages
  - commandes DOS, shell Unix, Javascript, Perl, PHP, Python, Visual Basic for Applications (VBA)



Familles de langages de programmation (3)

- **Le langage Java**
  - le programme en langage de programmation est traduit (compilé) en en bytecode Java (code machine indépendant du processeur)
  - le bytecode Java est exécuté (interprété) par une machine virtuelle Java
    - la machine virtuelle est dépendante du système/machine sur lequel elle s'exécute
  - la compilation se fait avant l'exécution du programme, et le programme est stocké sous deux formes



Autres classifications des langages

- **Programmation impérative**
  - Très proche des instructions du processeur (affectation de variables, lecture, condition, boucle, etc.)
  - C, Pascal, Delphi, VBA, Visual Basic

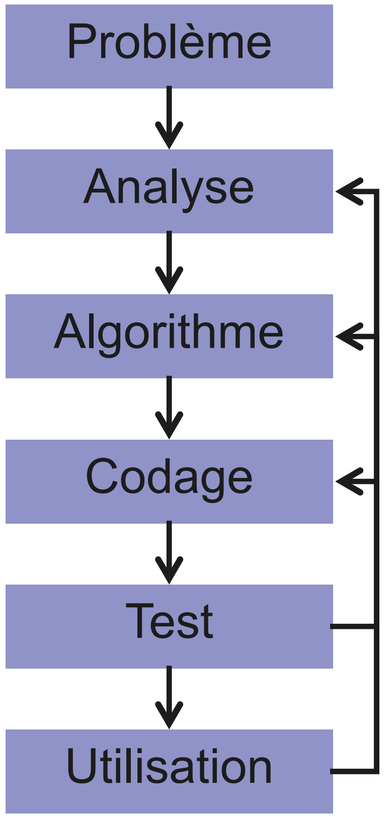
- **Programmation fonctionnelle**
  - On ne peut décrire que des fonctions
  - Un programme = 1 expression avec des fonctions
  - On ne peut pas changer la valeur de variable en cours d'exécution
  - Scheme, Lisp

- **Programmation objet**
  - Smalltalk, C++, Java, Delphi

- **Programmation événementielle**
  - Delphi, Visual Basic, Javascript

- **Programmation logique**
  - Prolog

Du problème au programme : un exemple



Phase 1 : problème

- **Calculer le montant d'un placement sur un compte rémunéré après un certain nombre d'années**

- Exemple d'écran d'interaction
  - communication avec l'utilisateur du programme

```
Ce programme calcule le montant d'un placement sur un compte rémunéré

Donnez le montant du placement : 100
Donnez le taux d'intérêt (ex : 3 pour 3%) : 4
Donnez la durée en années : 4

Après 4 ans, le montant sera de : 112,55 euros
```

Phase 2 : analyse

- **Données d'entrée fournies**
  - un nombre représentant la valeur placée
  - un nombre représentant le taux d'intérêt (pour 10% : 10)
  - un nombre représentant une durée

- **Résultat souhaité**
  - un nombre représentant le montant après versement des intérêts, après une certain durée

- **Démarche à adopter**
  - prendre connaissance de la somme initiale, du taux d'intérêt et de la durée
  - calculer le résultat :
    - calculer 1+taux/100
    - mettre le résultat à la puissance durée
    - multiplier le résultat par la somme initiale
  - afficher le nouveau montant ainsi obtenu

Phase 3 : algorithme

Algorithme CalculDeRémunération
Variables *MontantInitial*, *NouveauMontant, Durée*, *Taux* : réels
début
    /* Saisie des données */
    Afficher "Ce programme calcule le montant d'un placement après un an sur un compte rémunéré "
    Afficher "Donnez le montant du placement"
    Saisir *MontantInitial*

    Afficher "Donnez le taux d'intérêt (ex : 3 pour 3%)"
    Saisir *Taux*

    Afficher "Donnez la durée en années"
    Saisir *Durée*

    /* Calcul à effectuer */
    NouveauMontant ← *MontantInitial* × (1+*Taux* /100)^*Durée*

    /* Affichage du résultat */
    Afficher "Après" & Durée "ans, le montant sera de : " & *NouveauMontant* & "Euros"
fin

Phase 4 : codage

Par exemple en Javascript

```
…
<script>
function placement() {
montant = parseFloat ( window.prompt("Entrez le montant initial :" ) );
taux = parseFloat ( window.prompt("Entrez le taux en pourcents :" ) );
duree = parseInt ( window.prompt("Entrez la durée en années :" ) );
mntfinal = montant * Math.pow ( 1+taux/100,duree);
alert ("Après " + duree + " an(s), le montant sera de : " +mntfinal);
}
</script>
<h1>Placement</h1>
<p>Ce programme calcule...<p>
<form name="placementform">
<input type="button" value="Calculer" onClick="placement()">
</form>
```

Démonstration - fichier Placement.html

Phase 5 : simulation de fonctionnement

Simulation 1

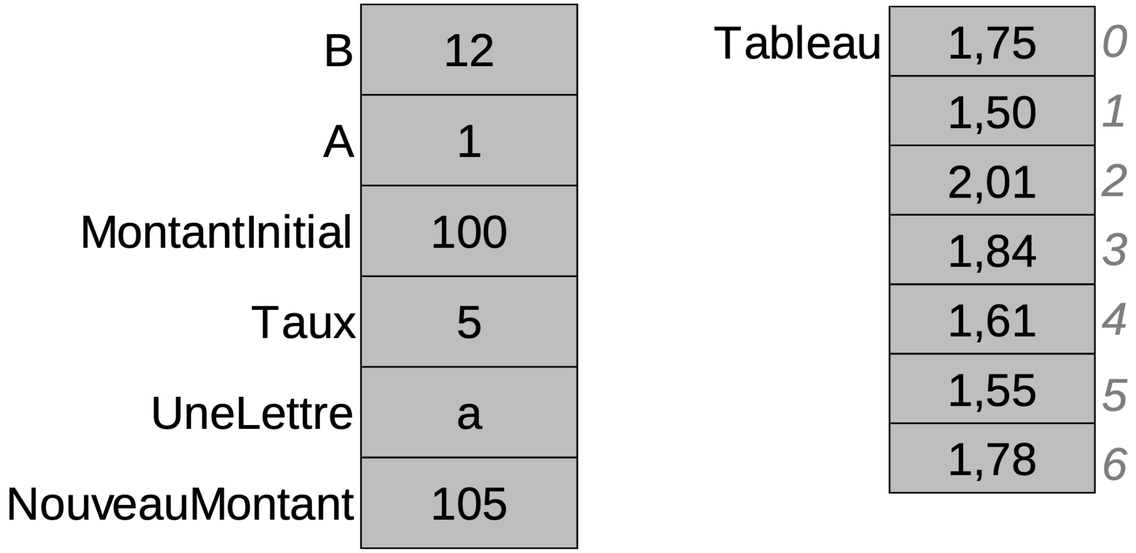
MontantInitial	-	100	100	100	100
Taux	-	-	5	5	5
Durée	-	-	-	1	1
NouveauMontant	-	-	-	-	105
Affichage					… 105 €

Simulation 2

MontantInitial	-	200	200	200	200
Taux	-	-	10	10	10
Durée	-	-	-	5	5
NouveauMontant	-	-	-	-	322,102
Affichage					… 322,102 €

Variables

- D’un point de vue matériel
  - zone de stockage en mémoire centrale
  - définie par son nom et son type (entier, réel, caractère, …)



Affectation et initialisation des variables

- Le programmeur peut
  - remplir la zone mémoire en lui attribuant une valeur
  - modifier à tout moment le contenu de la zone mémoire en changeant de valeur
  - consulter la valeur contenue dans la zone mémoire (uniquement si elle est remplie)

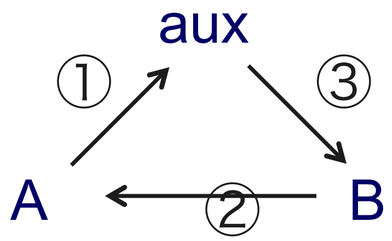
- **Affectation**
  - c'est le processus par lequel on attribue une valeur à une variable

- **Initialisation**
  - c'est le processus par lequel on attribue une première valeur à une variable

Affectation

- On donne une valeur à une variable
  - le contenu de la variable est modifié
  - la valeur précédente est définitivement perdue

- **Exemples**
  - variable ← valeur
  - variable ← variable
  - variable ← résultat du calcul
  - incrémentation
    - compteur ← compteur + 1
  - permutation du contenu de 2 variables A et B
    - besoin d'une variable auxiliaire (aux)



Catégories de variables

- **Trois catégories de variables**
  - les données (entrée)
  - les résultats (sortie)
  - les utilitaires (données intermédiaires)

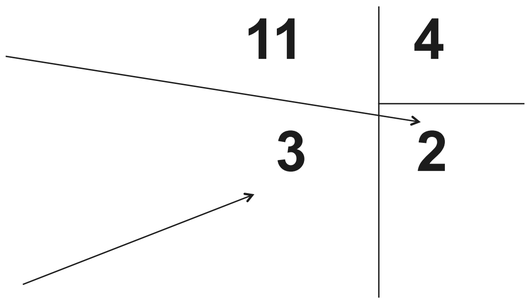
- **Constantes**
  - variables dont la valeur est fixe (pi, taux de TVA...)
  - définies dès le début du programme
  - ne peuvent être modifiées pendant l'exécution du programme

Types de variables

- **entier**
  - 23 ; 0 ; 3
- **réel**
  - -104,324 ; 0,25
- **caractère**
  - ‘a’ ; ‘A’ ; ‘1’ ; ‘?’
- **chaîne de caractères**
  - "caractère" ; "c" ; « "
- **booléen**
  - vrai ; faux
- la taille de la zone de stockage dépend du type de la variable

Opérateurs

- Addition
- Soustraction
- Multiplication
- Division réelle
  - 11/4 → 2,75
- Division entière (euclidienne)
  - sur des entiers
    - 11 DIV 4 → 2
- Reste de la division entière
  - sur des entiers
    - 11 RESTE 4 → 3



Relations d’ordre

- **Relations d’ordre**
  - égal =
  - différent ≠ (<>, !=)
  - supérieur >
  - supérieur ou égal ≥ (>=)
  - inférieur <
  - inférieur ou égal ≤

- Attention
  - on ne compare que des éléments de types compatibles

Opérateurs logiques

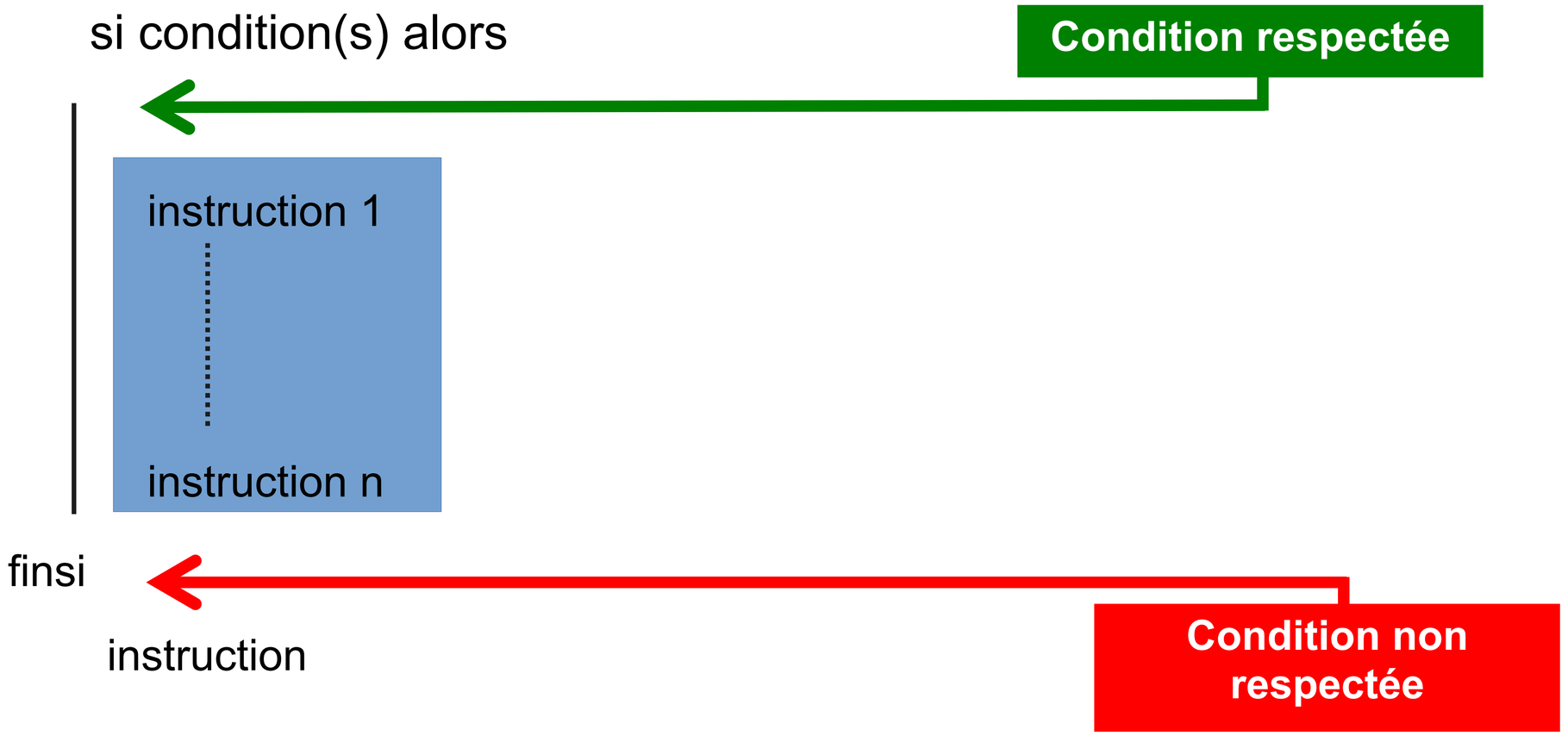
- Opérateurs **logiques**
  - ET
  - OU
  - NON
  - exemple : a=b ou a=c

- **Tables de vérité**
  - X et Y, 2 variables booléennes

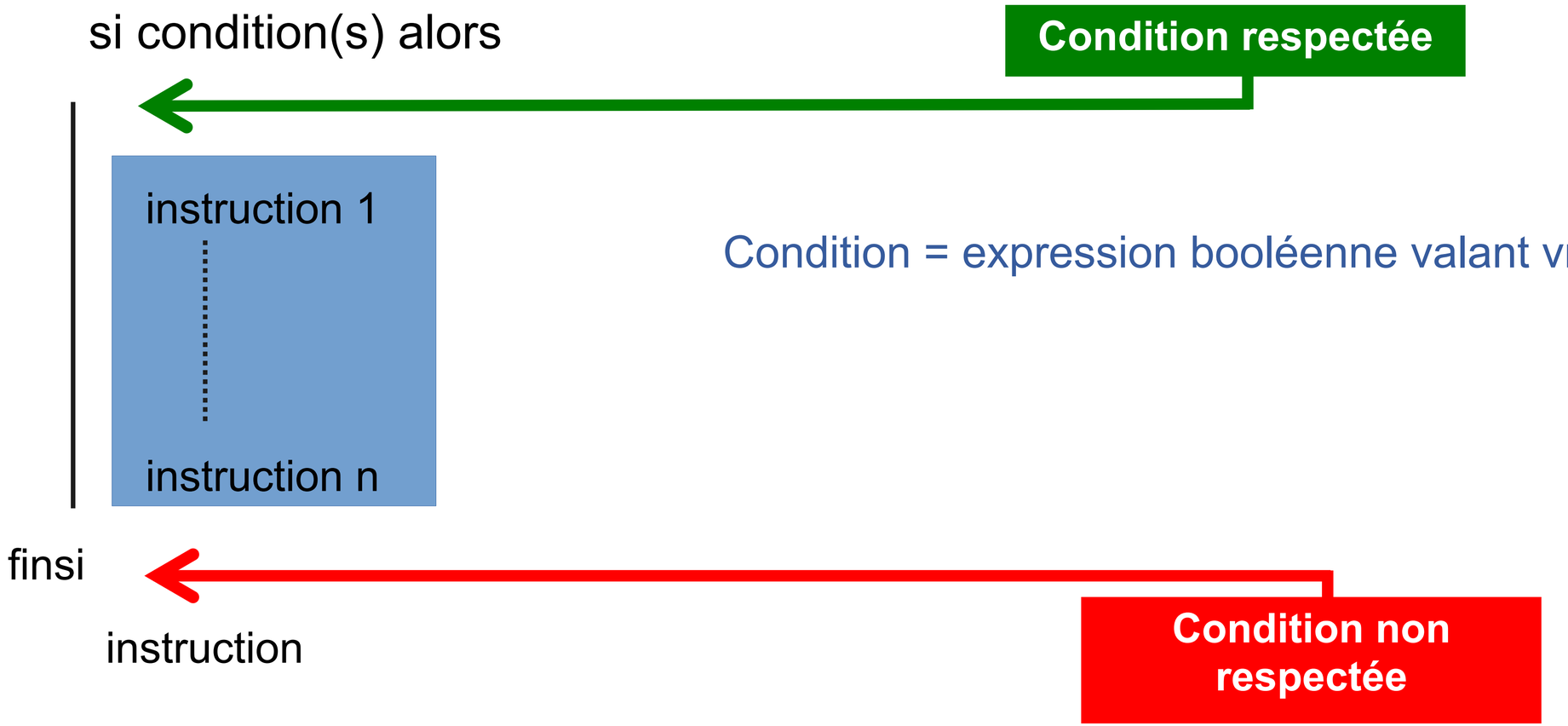
X	Y	X ET Y	X OU Y	NON X
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

- Lois de De Morgan
  - NON (A ET B) ⇔ (NON A) OU (NON B)
  - NON (A OU B) ⇔ (NON A) ET (NON B)

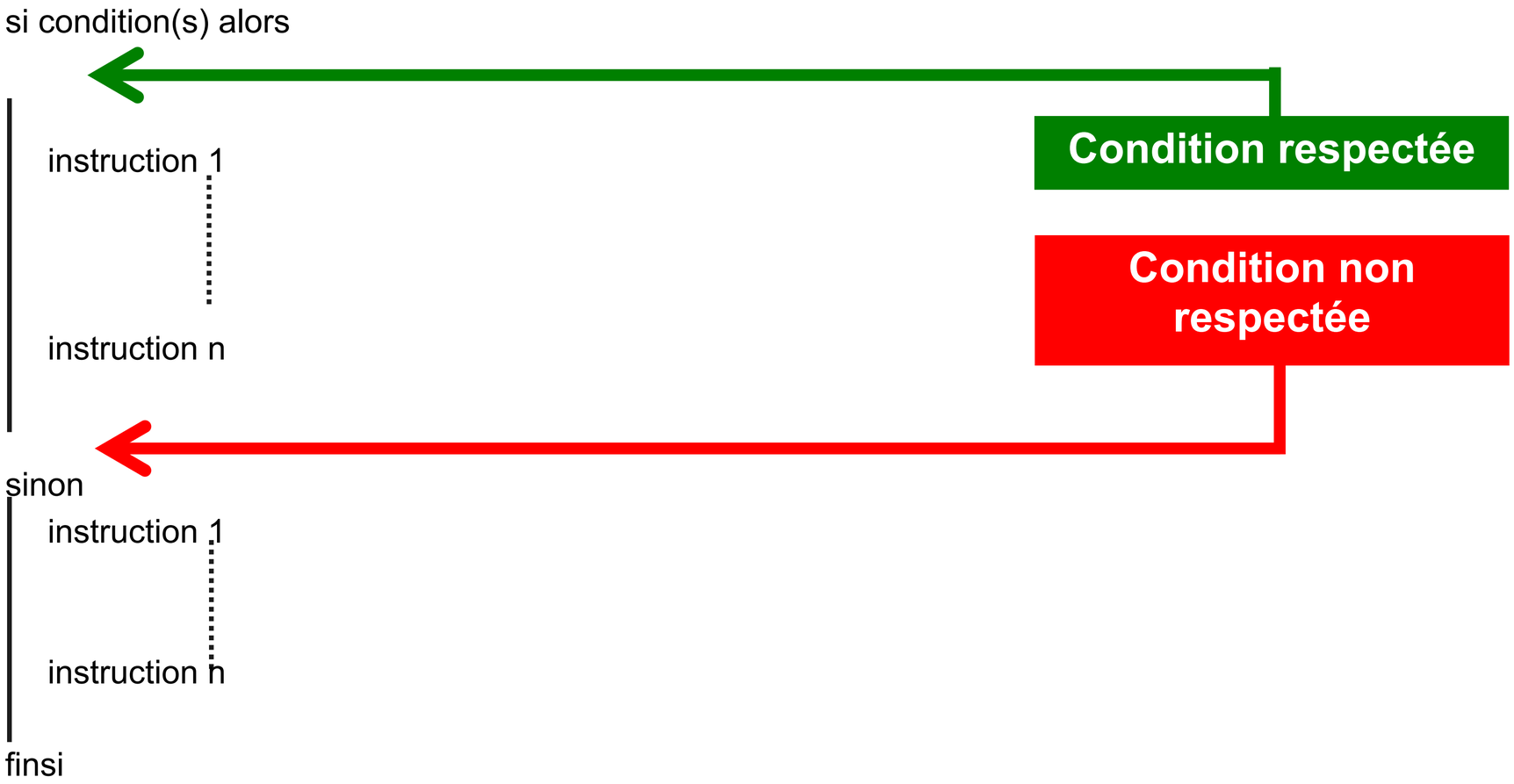
Conditionnelle : si…alors



Conditionnelle : si…alors



Conditionnelle : si…alors…sinon



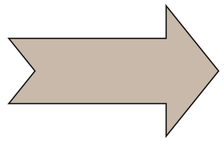
Conditionnelle : exemple

```
/* Saisie des données */
Afficher "Calcul du résultat de la division de 2 entiers"
Afficher " Donnez le premier entier "
Saisir A
Afficher " Donnez le deuxième entier "
Saisir B
/* Calcul et affichage du résultat */
si B ≠ 0 alors
   Resultat ← A / B
   Afficher Resultat
sinon
   Afficher " Impossible de diviser un nombre par 0"
finsi
```

Conditions et expressions booléennes

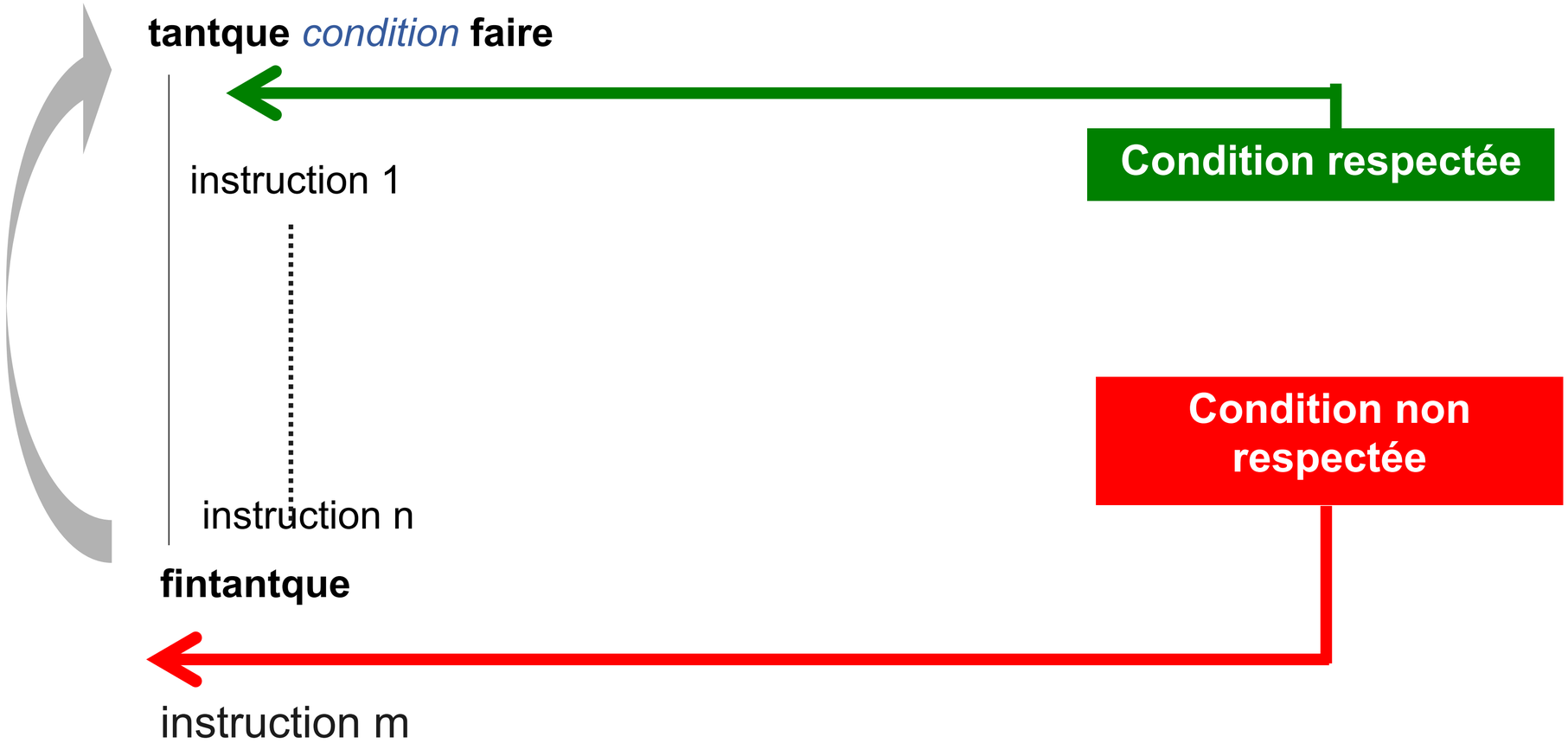
- La condition est une expression booléenne
- Elle renvoie une valeur booléenne
  - Vrai
  - Faux

```
si condition alors
    instruction(s)
sinon
    instruction(s)
finsi
```



```
si condition = Vrai alors
    instruction(s)
si condition = Faux alors
    instruction(s)
```

Instruction de répétition : tantque



Attention à prévoir la sortie de la boucle (boucle infinie)

Instruction de répétition : exemple

Je monte dans le tram

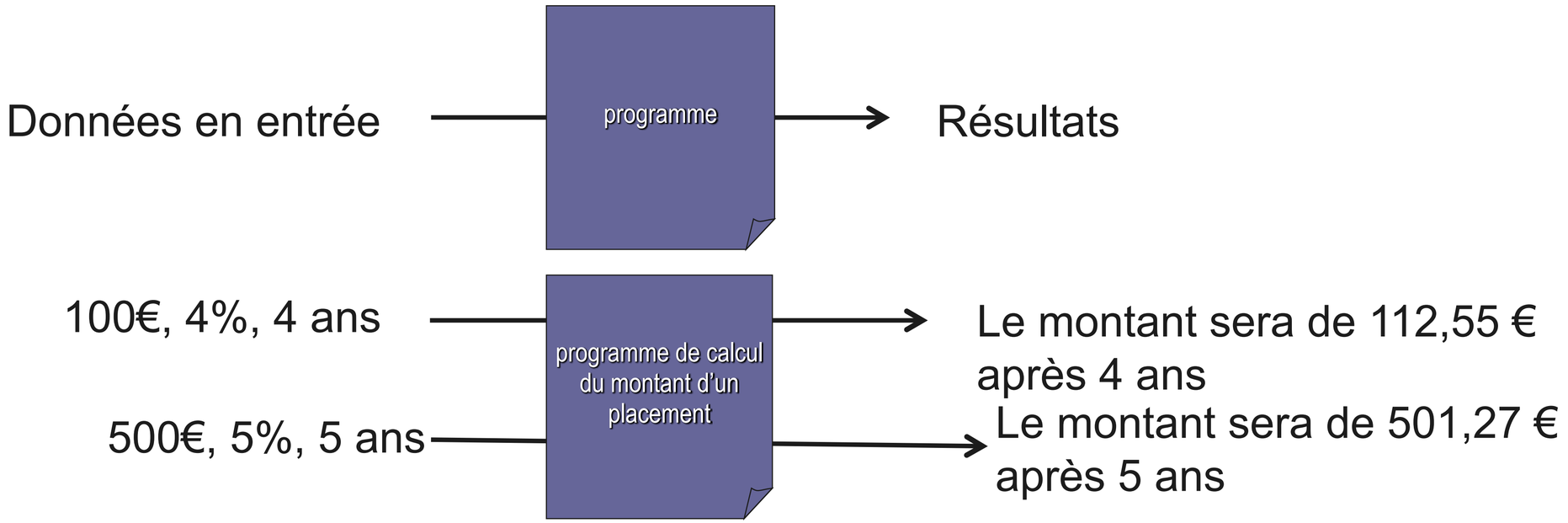
Tant que (arrêt ≠ "Université Lyon 1") faire
    Je me tiens à une barre
    Je surveille les arrêts
Fin tantque

Je descends du tram

Appel de programme et paramètres

On peut appliquer un programme à des données différentes

- les paramètres
- exemples
  - calcul du montant d'un placement sur un compte rémunéré



Exemple : numéro de sécurité sociale

- Aussi appelé NIR (Numéro d'Inscription au Registre)
- Numéro unique composé de 13 caractères suivis d'une clé de contrôle
- Exemple : 2 80 07 75 073 004 83

Clé de contrôle

N° d'ordre de naissance

Commune de naissance
(N° INSEE, ici Suresnes)

Département de naissance
(ici Paris,ex-Seine)

Mois de naissance
(ici juillet)

Année de naissance
(ici 1980)

Sexe
(1 : homme ;
2 : femme)

Exemple : décomposition du numéro de SS

- **Problème**
  - décomposer mon numéro de Sécurité Sociale
- **Données en entrée**
  - numéro de Sécurité Sociale
- **Résultat**
  - le sexe, l'année et le mois de naissance, l'âge de l'assuré
- **Méthode**
  - trouver le sexe associé au premier chiffre du numéro
  - trouver l'année associée au deux chiffres suivants
  - trouver le mois associé au deux chiffres suivants
  - calculer l'âge de l'assuré

Exemple : algorithme

```
demander numeroSS
sexe ← 1er caractère de numeroSS
annee ← 2ème et 3ème caractères de numeroSS
mois ← 4ème et 5ème caractères de numeroSS
departement ← 6ème et 7ème caractères de numeroSS

afficher   "Vous êtes un"
si sexe="1" alors
        afficher  "homme"
sinon
        afficher   "e femme"
finsi

afficher  "né(e) en »

selon mois
        01 : afficher "janvier"
        02 : afficher "février"
        03 : afficher "mars"
        04 : afficher "avril"
        05 : afficher "mai"
        06 : afficher "juin"
        07 : afficher "juillet"
        08 : afficher "août"
        09 : afficher "septembre"
        10 : afficher "octobre"
        11 : afficher "novembre"
        12 : afficher "décembre"
        autre : afficher "erreur sur le mois"
finselon

anneeNaissance ← "19" & annee
afficher anneeNaissance
age ← 0
anneeCourante ← 2025
tantque (age + anneeNaissance < anneeCourante) faire
        age ← age + 1
finTantque

afficher "Vous avez " & age &  "ans"
```

Exemple : codage en JavaScript

Démonstration - fichier Secu.html

Exemple : codage en JavaScript

```
…
<script language="javascript">
function decoupage () {
numSECU =
document.forms["saisie"].elements["secu"].value
sexe = parseInt (numSECU.substr(0,1));
annee = parseInt(numSECU.substr (1,2));
mois = parseInt(numSECU.substr(3,2));
dept = parseInt(numSECU.substr (5,2));

texte = "Vous êtes un";
if (sexe ==1) texte = texte + " homme, né en ";
else texte = texte + "e femme, née en ";

switch (mois) {
case 1: texte = texte + "janvier"; break;
case 2: texte = texte + "février"; break;
case 3: texte = texte + "mars"; break;
case 4: texte = texte + "avril"; break;
case 5: texte = texte + "mai"; break;
case 6: texte = texte + "juin"; break;
case 7: texte = texte + "juillet"; break;
case 8: texte = texte + "août"; break;
case 9: texte = texte + "septembre"; break;
case 10: texte = texte + "octobre"; break;
case 11: texte = texte + "novembre"; break;
case 12: texte = texte + "décembre"; break;
}
```

```
annee = annee + 1900;
texte = texte + " " + annee;

age = 0;
while (annee < 2025 ) { age ++; annee ++; }
texte = texte + "\nVous avez " + age + " ans";

alert(texte);
}
</script>

<h1>N° Sécu</h1>
<p>Ce programme calcule…</p>
<form name="saisie">
<input type="text" name="secu" value="">
<input type="button" value="Calculer"
onClick="decoupage()">
</form>
…
```