

DUT SRC – IUT de Marne-la-Vallée  
21/09/2012  
INF120 - Algorithmique

# ***Cours 1***

## ***Introduction aux algorithmes***

# Organisation pratique

- **Contact**

- Courriel : philippe.gambette@gmail.com (INF120 dans le sujet du courriel)
- Avant ou après le cours

- **Matériel**

- Ordinateur portable : pas pendant les cours, à discuter pour les TP/TD.
- Pas de téléphone portable pendant cours/TD/TP
- Salles informatiques : pas manger, pas boire, pas débrancher les câbles

- **Déroulement des enseignements**

- Page web du cours : <http://tinyurl.com/INF120-2012S1>
- Séparation cours/TP/TD :
  - nouvelles méthodes de travail
  - distinguer ce qui est important, à retenir
  - savoir où retrouver l'information
- En général, distribution de notes de cours à compléter
- En général, distribution de corrigés : → refaire les exercices !

# Organisation pratique

- **Notes et devoirs**

- Interrogations QCM en début de cours ou TD  
(signalement des absences pour rattrapage, voir intranet)
- Un devoir maison

- **Note finale**

- Préviation : environ 2/3 “compétences”, environ 1/3 “motivation”
- Compétences : 2/3 devoir final (16 janvier 2012), 1/3 QCM
- Motivation : devoir maison, note générale de TP

- **Exercices supplémentaires d'entraînement**

- Sur demande, par courriel
- Sur demande, possibilité d'organiser une séance d'exercices ou de préparation au devoir final.

# Organisation pratique

- **Notes et devoirs**

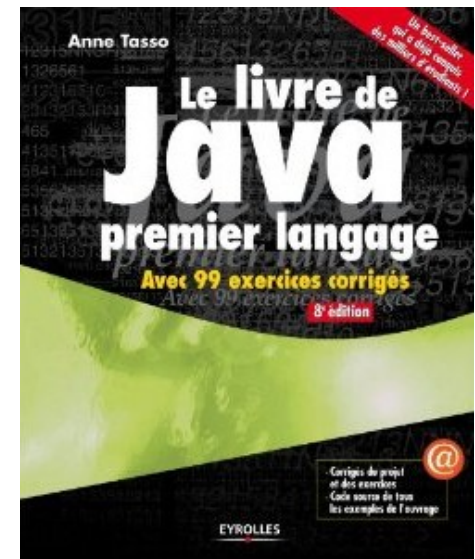
- Interrogations QCM en début de cours ou TD  
(signalement des absences pour rattrapage, voir intranet)
- Un devoir maison

- **Note finale**

- Prévion : environ 2/3 “compétences”, environ 1/3 “motivation”
- Compétences : 2/3 devoir final (16 janvier 2012), 1/3 QCM
- Motivation : devoir maison, note générale de TP

- **Exercices supplémentaires d'entraînement**

- Sur demande, par courriel
- Sur demande, possibilité d'organiser une séance d'exercices ou de préparation au devoir final.



# Sources

---

- *Le livre de Java premier langage*, d'A. Tasso
- <http://www.pise.info/algo/introduction.htm>
- Cours INF120 de J.-G. Luque
- <http://serecom.univ-tln.fr/cours/index.php/Algorithmie>
- Cours de J. Henriet : <http://julienhenriet.olymp-network.com/Algo.html>
- <http://xkcd.com>, <http://xkcd.free.fr>

# Plan des cours du semestre

---

- Introduction aux algorithmes
- Variables et affectation, type et codage des données
- Les tests et la logique
- Les boucles et leur terminaison, les tableaux
- Les entrées-sorties et les fonctions

# Plan du cours 1 – Introduction aux algorithmes

---

- Introduction aux algorithmes
  - A quoi sert un algorithme ?
  - Algorithme et programme
  - Enjeux de l'algorithmique
  - Composants d'un algorithme
- Variables et affectation

# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2 cuillères à café de sucre, 1 pincée de sel, 250g de farine et 1/2 litre de lait



Mettre l'ensemble des ingrédients dans un récipient sauf le beurre



Mélanger avec un fouet jusqu'à obtenir de la pâte liquide et sans grumeaux



Ajouter les 50g de beurre fondu: fondre au micro-onde, ça va plus vite! Pour plus de goût, ajouter de la fleur d'oranger ou du rhum...



Si possible, laisser reposer, puis étaler une dose de pâte dans une poêle chaude préalablement graissée



Laisser cuire à feu doux...



...puis retourner pour laisser cuire l'autre côté



Bon appétit !



# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2 cuillères à café de sucre, 1 pincée de sel, 250g de farine et 1/2 litre de lait



Mettre l'ensemble des ingrédients dans un récipient sauf le beurre



Mélanger avec un fouet jusqu'à obtenir de la pâte liquide et sans grumeaux



Ajouter les 50g de beurre fondu: fondre au micro-onde, ça va plus vite! Pour plus de goût, ajouter de la fleur d'oranger ou du rhum...



**Si possible**, laisser reposer, puis étaler **une dose** de pâte dans une poêle chaude préalablement graissée



Laisser cuire à feu doux...



**...puis** retourner pour laisser cuire l'autre côté



Bon appétit !

# La recette des crêpes

## Le site le plus simple pour faire la pâte à crêpes !

Oyé oyé, braves gens ! Bienvenue sur le site le plus simple pour faire la pâte à crêpe ! Cette recette facile de pâte à crêpe se transmet de bouches à oreilles et maintenant de Facebook en Facebook pour votre plus grand plaisir ! Vous allez adorer faire des crêpes.



Pâte à crêpe pour 15 crêpes:  
50g de beurre, 4 oeufs, 2 cuillères à café de sucre, 1 pincée de sel, 250g de farine et 1/2 litre de lait



Mettre l'ensemble des ingrédients dans un récipient sauf le beurre



Mélanger avec un fouet jusqu'à obtenir de la pâte liquide et sans grumeaux



Ajouter les 50g de beurre fondu: fondre au micro-onde, ça va plus vite! Pour plus de goût, ajouter de la fleur d'oranger ou du rhum...



**Si possible**, laisser reposer, puis étaler **une dose** de pâte dans une poêle chaude préalablement graissée



Laisser cuire à feu doux...



**...puis** retourner pour laisser cuire l'autre côté



Bon appétit !

# L'“algorithme des crêpes”

**Ingrédients** : beurre, oeufs, sachets de sucre vanillé, farine, lait, sel

**Récipients** : saladier, verre mesureur, poêle, assiette

**Opérations de base** : *mettre dans un récipient*, *mélanger*, *attendre pendant ... minutes*, *retourner*, *laisser cuire pendant ... minutes*

**Algorithme des crêpes** :

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 minute

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre pendant** 60 minutes

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 minute

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** la poêle

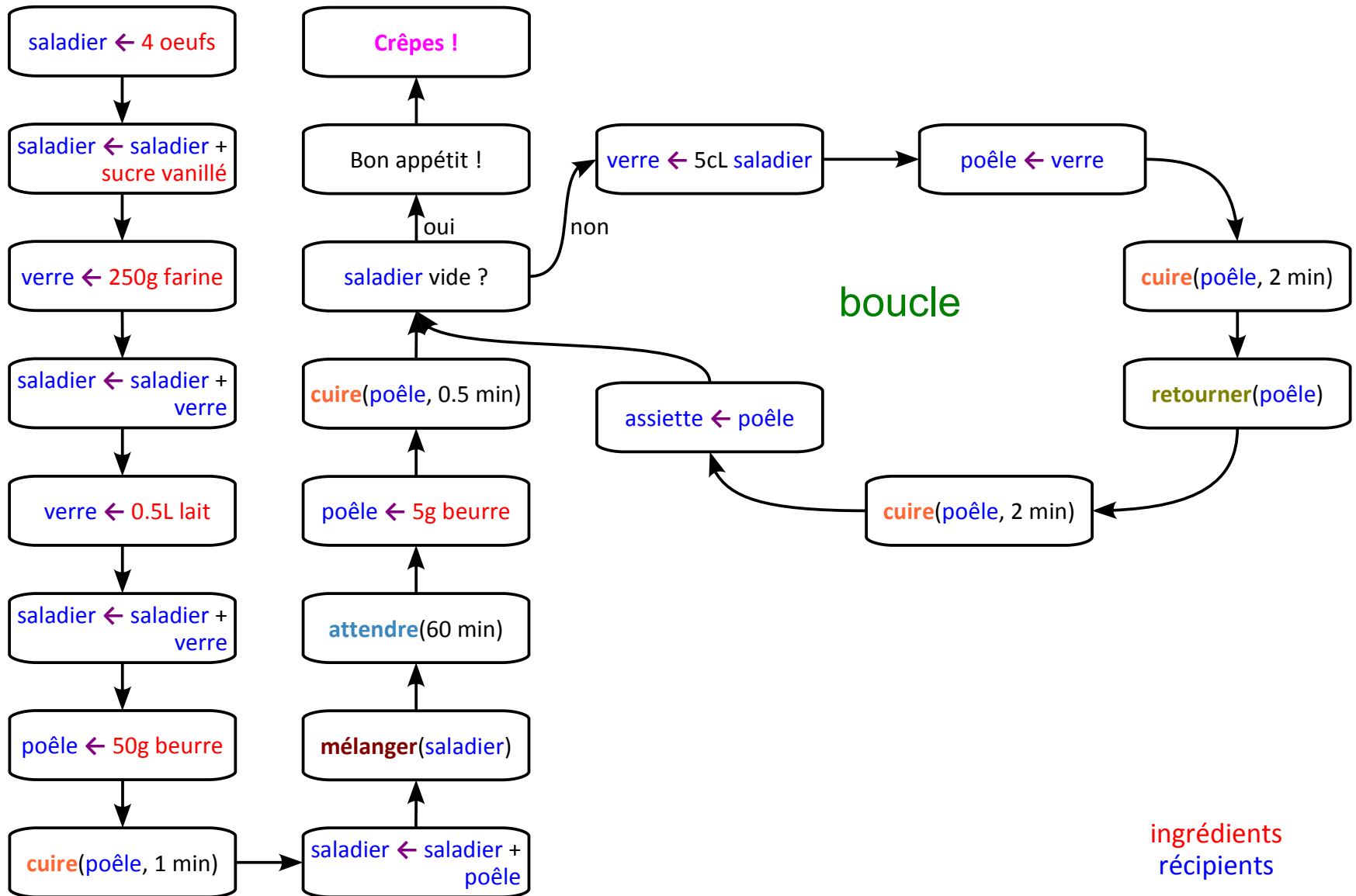
**Laisser cuire** la poêle **pendant** 2 minutes

**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 minutes

**Mettre** le contenu de la poêle **dans** l'assiette

# Organigramme de la recette des crêpes



# A quoi sert un algorithme ?

---

- **À décrire les étapes de résolution d'un problème :**
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

# A quoi sert un algorithme ?

---

- À décrire les **étapes** de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

“**étapes**” aussi appelées “**pas de l'algorithme**”



# A quoi sert un algorithme ?

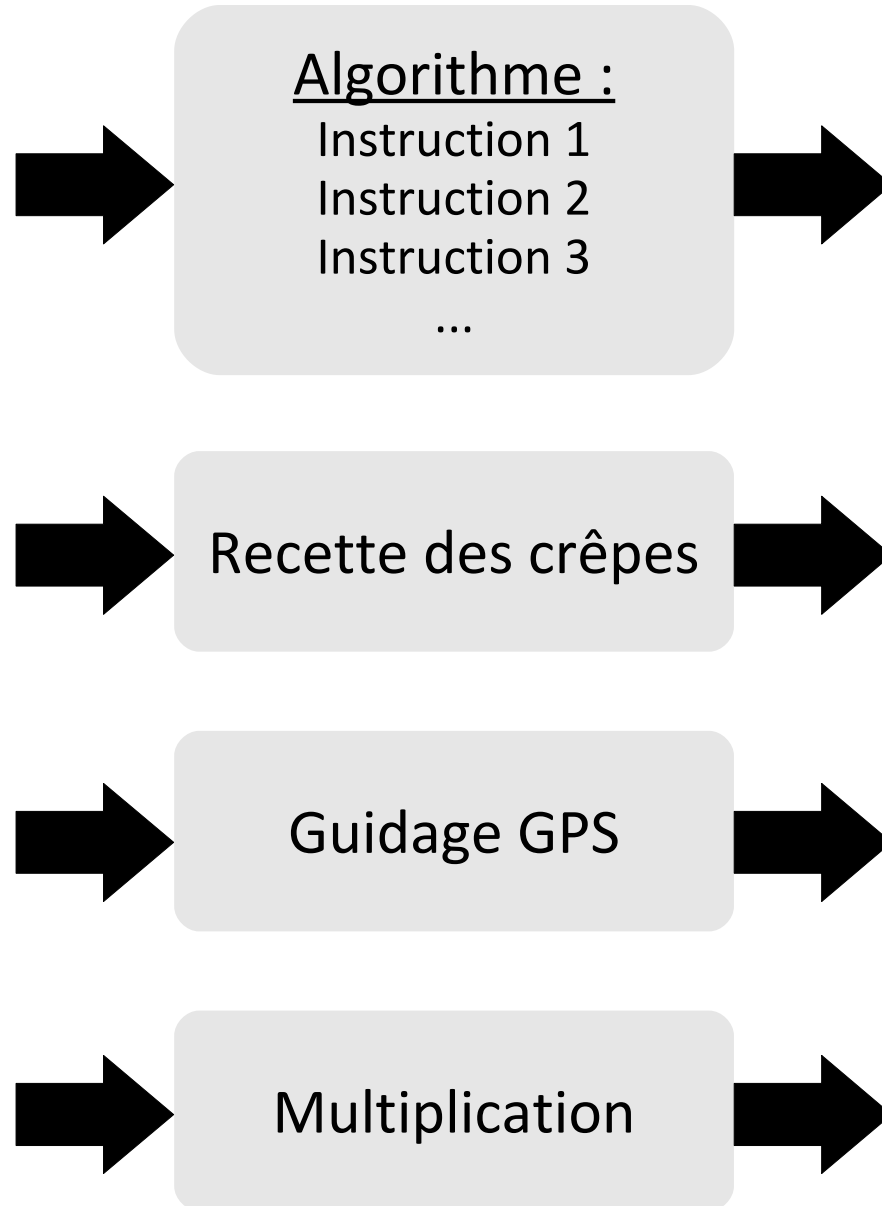
---

- À décrire les étapes de **résolution d'un problème** :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

Les **données** du problème en **entrée**

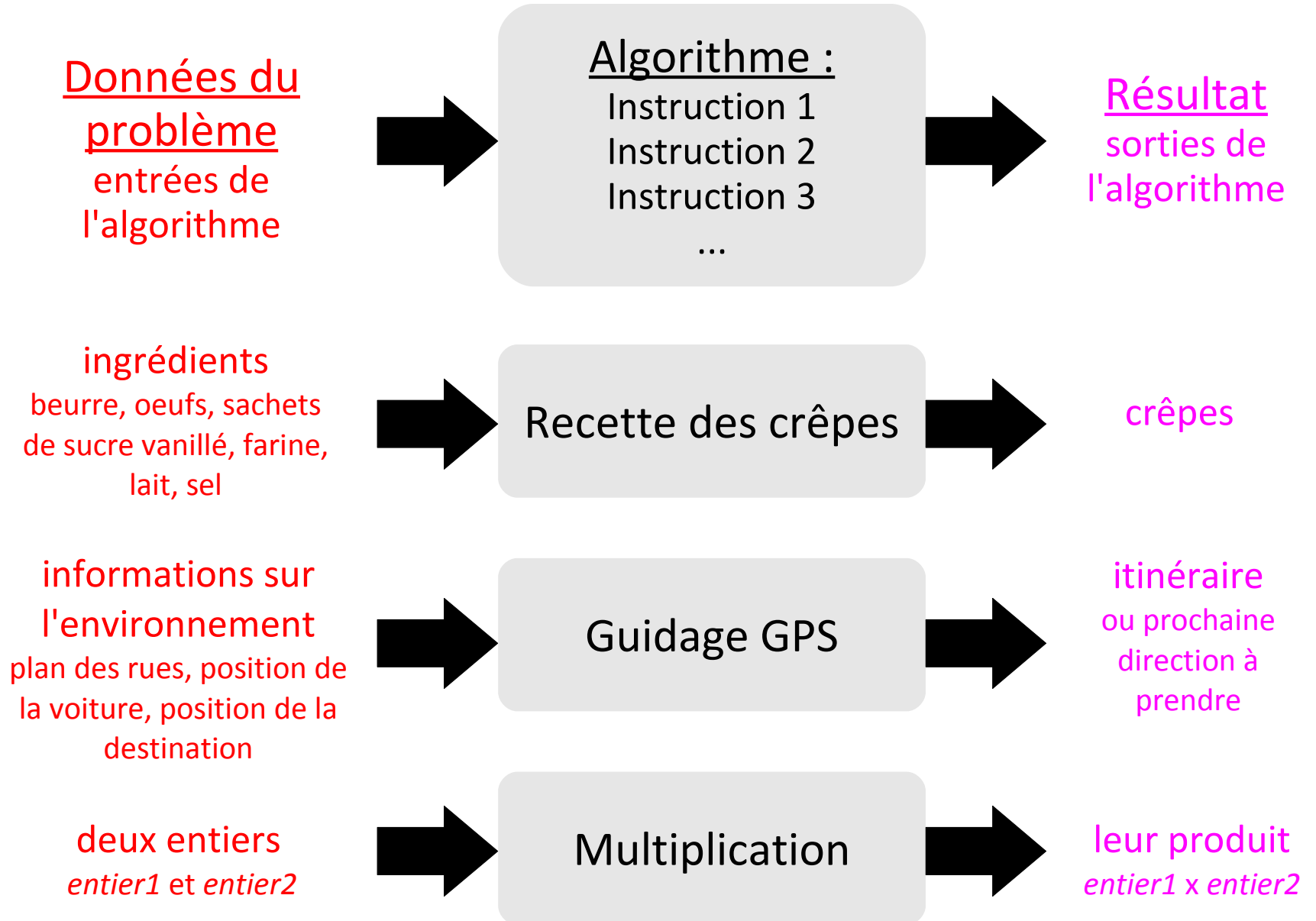
Le **résultat** de sa résolution en **sortie**

# Composants d'un algorithme





# Composants d'un algorithme



# A quoi sert un algorithme ?

---

- À décrire les étapes de résolution d'un problème :
  - **de façon structurée et compacte**
  - à partir d'opérations de base
  - indépendamment d'un langage de programmation

Méthode de résolution d'un problème :

**facile à comprendre**

**facile à transmettre**

# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - **à partir d'opérations de base**
  - indépendamment d'un langage de programmation

Méthode de résolution d'un problème :

**adaptée aux moyens à disposition**

**adaptée aux connaissances de celui qui l'utilise**

# A quoi sert un algorithme ?

---

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Méthode de résolution d'un problème :

**adaptée pour des problèmes qui se traitent sans ordinateur**  
**compréhensible sans apprendre un langage de programmation**

# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateurs** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres



# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateurs** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres
- Al-Khuwārizmī (825) : résolution d'équations



# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

## *La “minute culturelle”*

### Algorithmes **sans ordinateurs** :

- Euclide (vers -300) : calcul du PGCD de 2 nombres
- Al-Khwarizmi (825) : résolution d'équations
- Ada Lovelace (1842) : calcul des nombres de Bernoulli sur la *machine analytique* de Charles Babbage



# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Trois **langages** abordés dans ce cours :

**organigramme**

**pseudo-code**

**Java**



# A quoi sert un algorithme ?

- À décrire les étapes de résolution d'un problème :
  - de façon structurée et compacte
  - à partir d'opérations de base
  - **indépendamment d'un langage de programmation**

Trois **langages** abordés dans ce cours :

**organigramme**

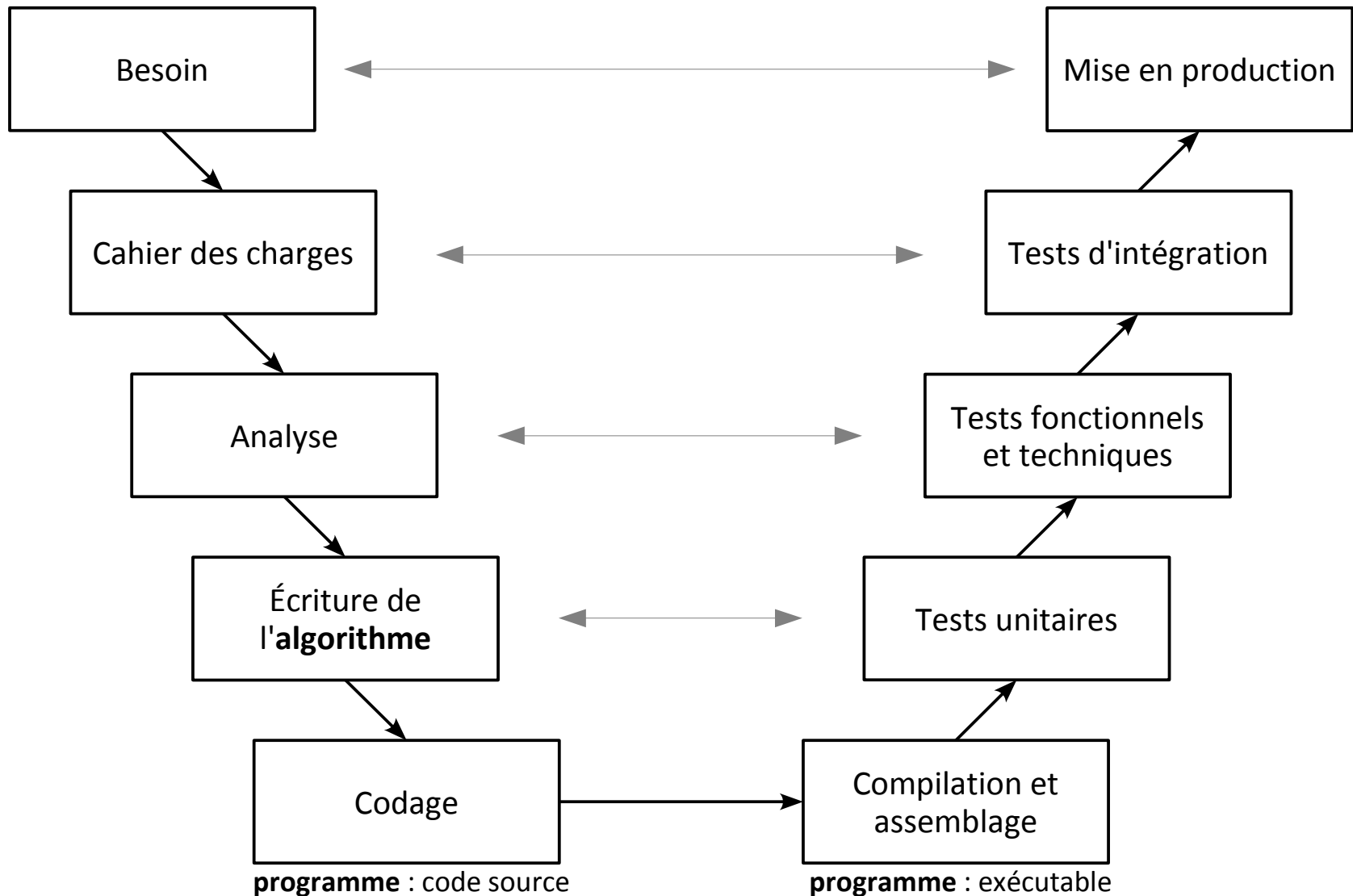
**pseudo-code**

**Java**

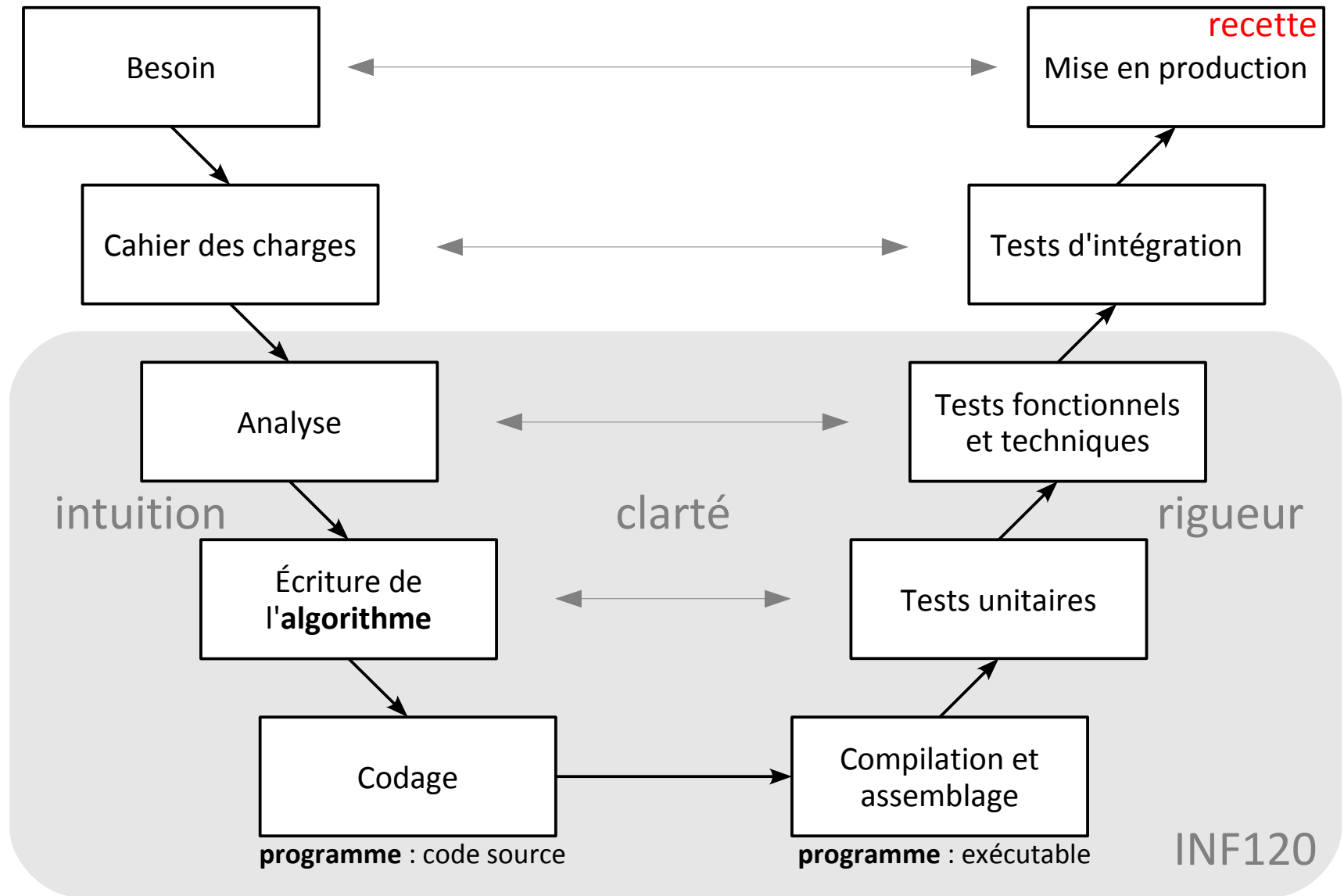


avantage aux  
littéraires !

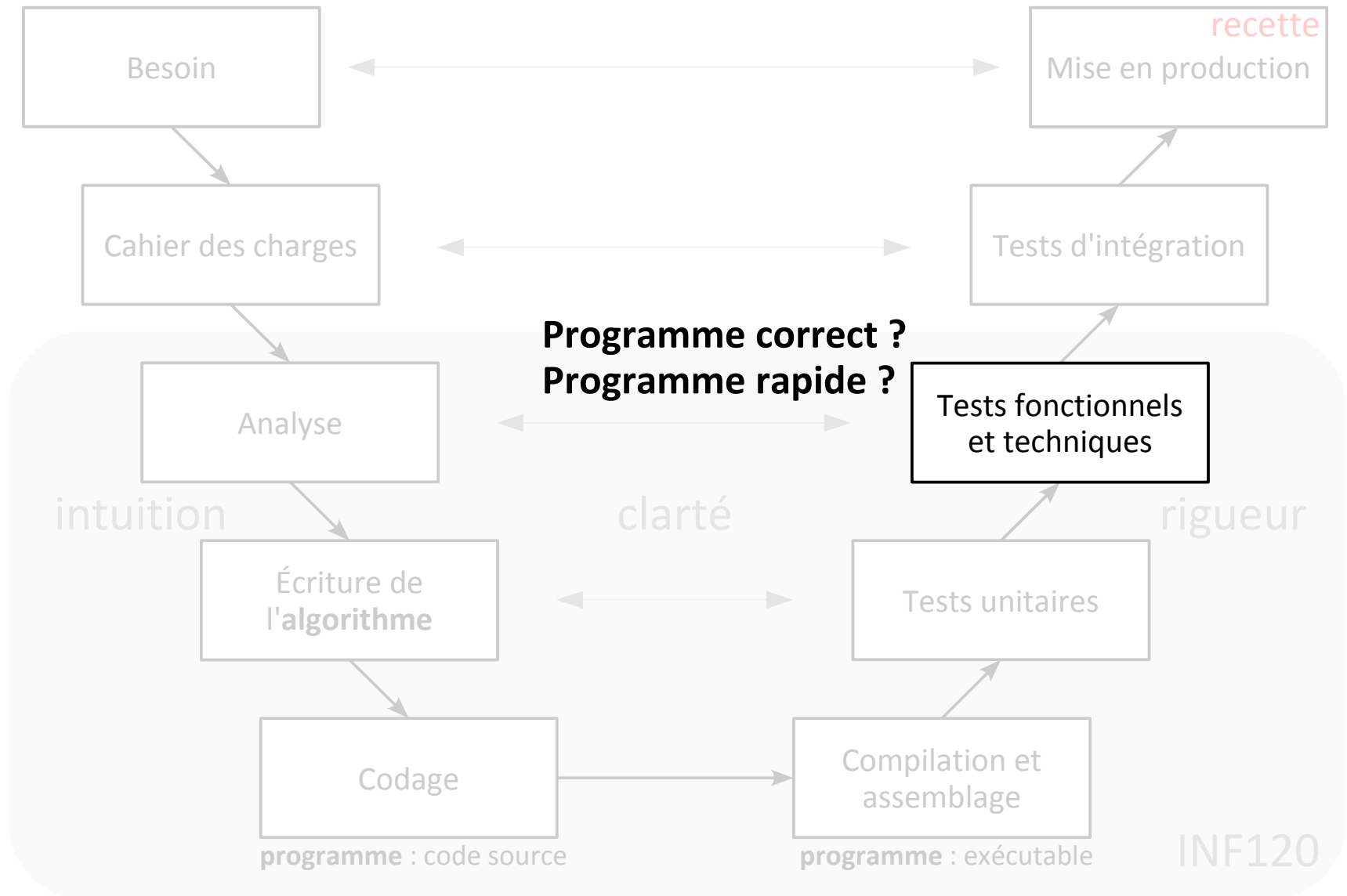
# De l'algorithme au programme, le "cycle en V"



# De l'algorithme au programme, le "cycle en V"



# De l'algorithme au programme, le "cycle en V"



# Enjeux de l'algorithmique

---

**Algorithme correct ?**

**Algorithme rapide ?**

# Enjeux de l'algorithmique

## Algorithme correct ?

- donne le résultat attendu ?
- quel que soit le type d'entrées ?

## Algorithme rapide ?

- se termine ?
- en combien de temps ?



crêpes, GPS,  
multiplication

# Enjeux de l'algorithmique

## Algorithme correct ?

- donne le résultat attendu ? → **preuve de correction**
- quel que soit le type d'entrées ? → **débuggage, tests unitaires**

## Algorithme rapide ?

- se termine ? → **preuve de terminaison**
- en combien de temps ? → **complexité**



crêpes, GPS,  
multiplication

# Enjeux de l'algorithmique - correction

**Correction** : L'algorithme donne-t-il le résultat attendu ?

**Preuve de correction** :

- « invariant » : propriété vraie tout au long de l'algorithme
    - vraie à la première étape
    - si vraie à une étape, vraie à l'étape suivante
- ⇒ vrai à la fin

**En pratique**, pour débiter :

- vérifier sur les “cas de base”
- vérifier sur des exemples aléatoires



# Enjeux de l'algorithmique - terminaison

L'algorithme se termine-t-il en un temps fini ?

## *Algorithme des crêpes :*

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 **minute**

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre pendant** 60 **minutes**

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 **minute**

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans**  
le verre mesureur

**Mettre** le contenu du verre mesureur  
**dans** la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Mettre** le contenu de la poêle **dans**  
l'assiette

# Enjeux de l'algorithmique - terminaison

L'algorithme se termine-t-il en un temps fini ?

## *Algorithme des crêpes :*

**Mettre** 4 oeufs **dans** le saladier

**Mettre** 1 sachet de sucre vanillé **dans** le saladier

**Mettre** 250 g de farine **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 0,5 litre de lait **dans** le verre mesureur

**Mettre** le contenu du verre mesureur **dans** le saladier

**Mettre** 50 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 1 **minute**

**Mettre** le contenu de la poêle **dans** le saladier

**Mélanger** le contenu du saladier

**Attendre pendant** 60 **minutes**

**Mettre** 5 grammes de beurre **dans** la poêle

**Laisser cuire** la poêle **pendant** 0.5 **minute**

**Tant que** le saladier n'est pas vide :

**Mettre** 5 cL du contenu du saladier **dans**  
le verre mesureur

**Mettre** le contenu du verre mesureur  
**dans** la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Retourner** le contenu de la poêle

**Laisser cuire** la poêle **pendant** 2 **minutes**

**Mettre** le contenu de la poêle **dans**  
l'assiette

→ Le saladier sera forcément vide à un moment donné !

→ preuve mathématique...

# Enjeux de l'algorithmique - complexité

**Complexité** : Combien de temps l'algorithme prend-il pour se terminer ?

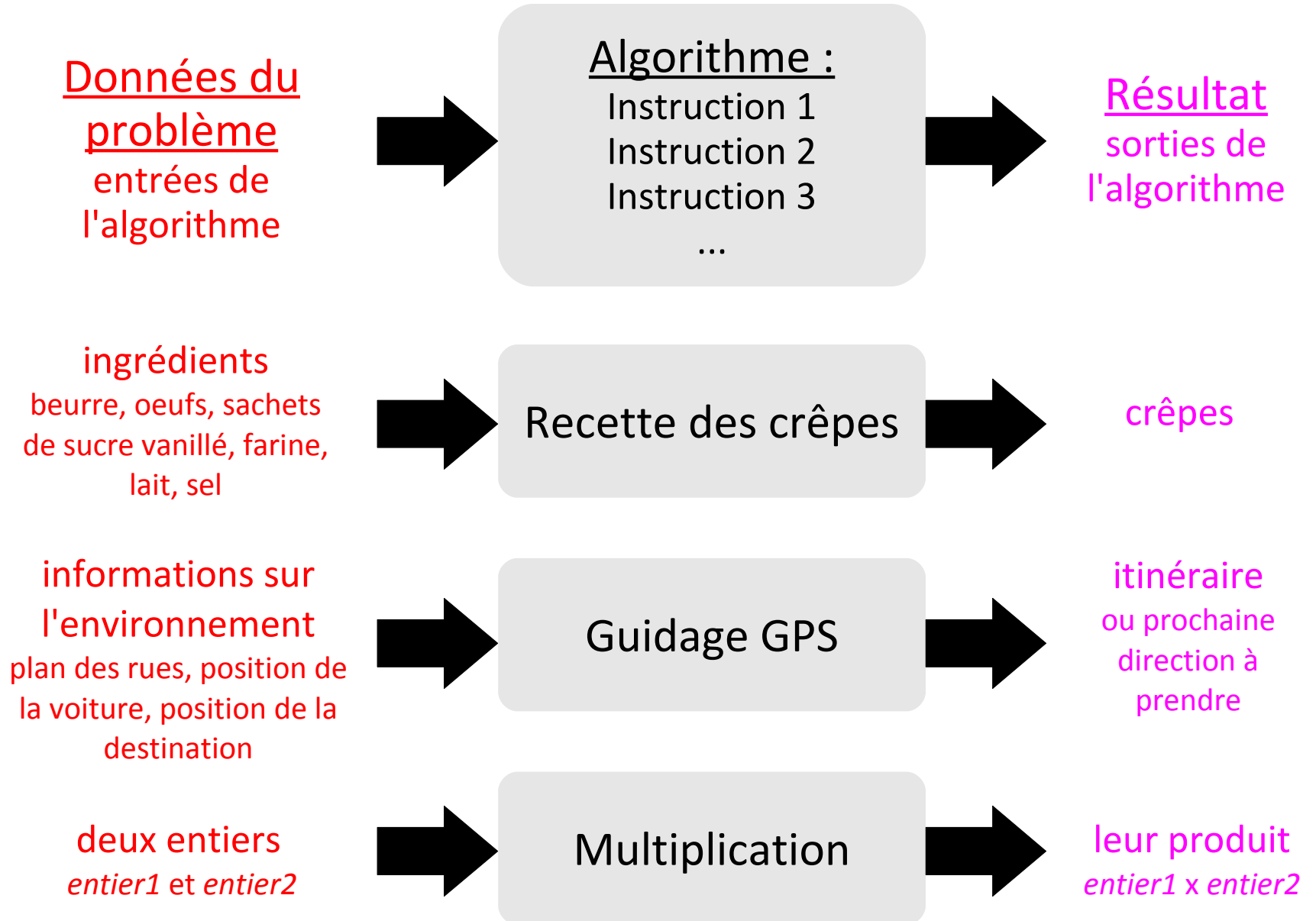
**Théorie de la complexité** :

- nombre d'opérations en fonction de la taille du problème, dans le pire cas
- prouver qu'on ne peut pas utiliser moins d'opérations pour résoudre le problème, dans le pire cas

**En pratique**, pour débiter :

- vérifier sur des exemples aléatoires
- connaître les cas difficiles

# Composants d'un algorithme



# Quels types d'instructions ?

Divers types d'instructions :

- déclaration d'un algorithme

- appel d'un algorithme

- déclaration d'une **variable**

- **affectation** d'une **variable**

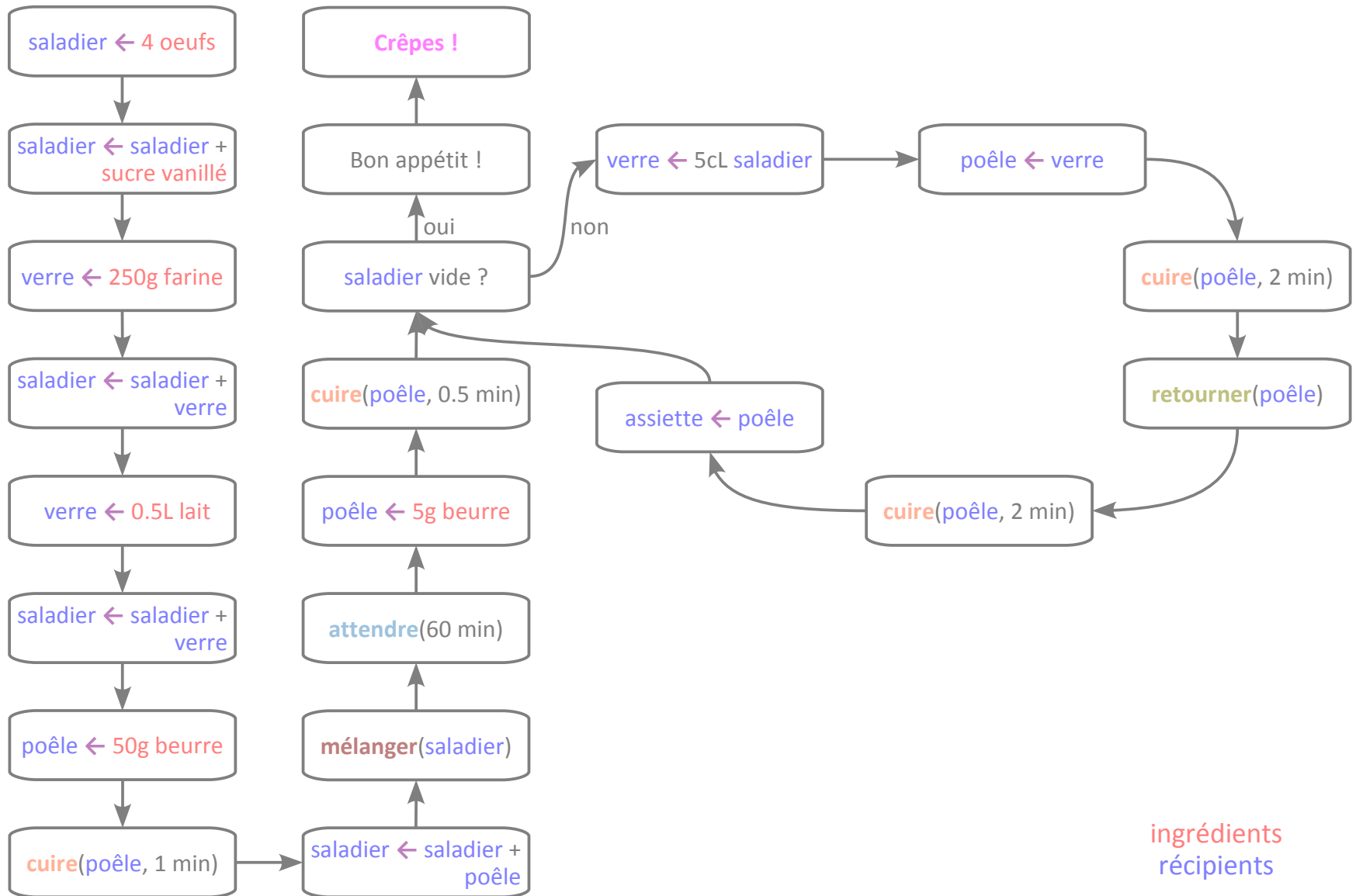
pour stocker des valeurs, des résultats intermédiaires

- **entrées** / **sorties**

- **boucle**

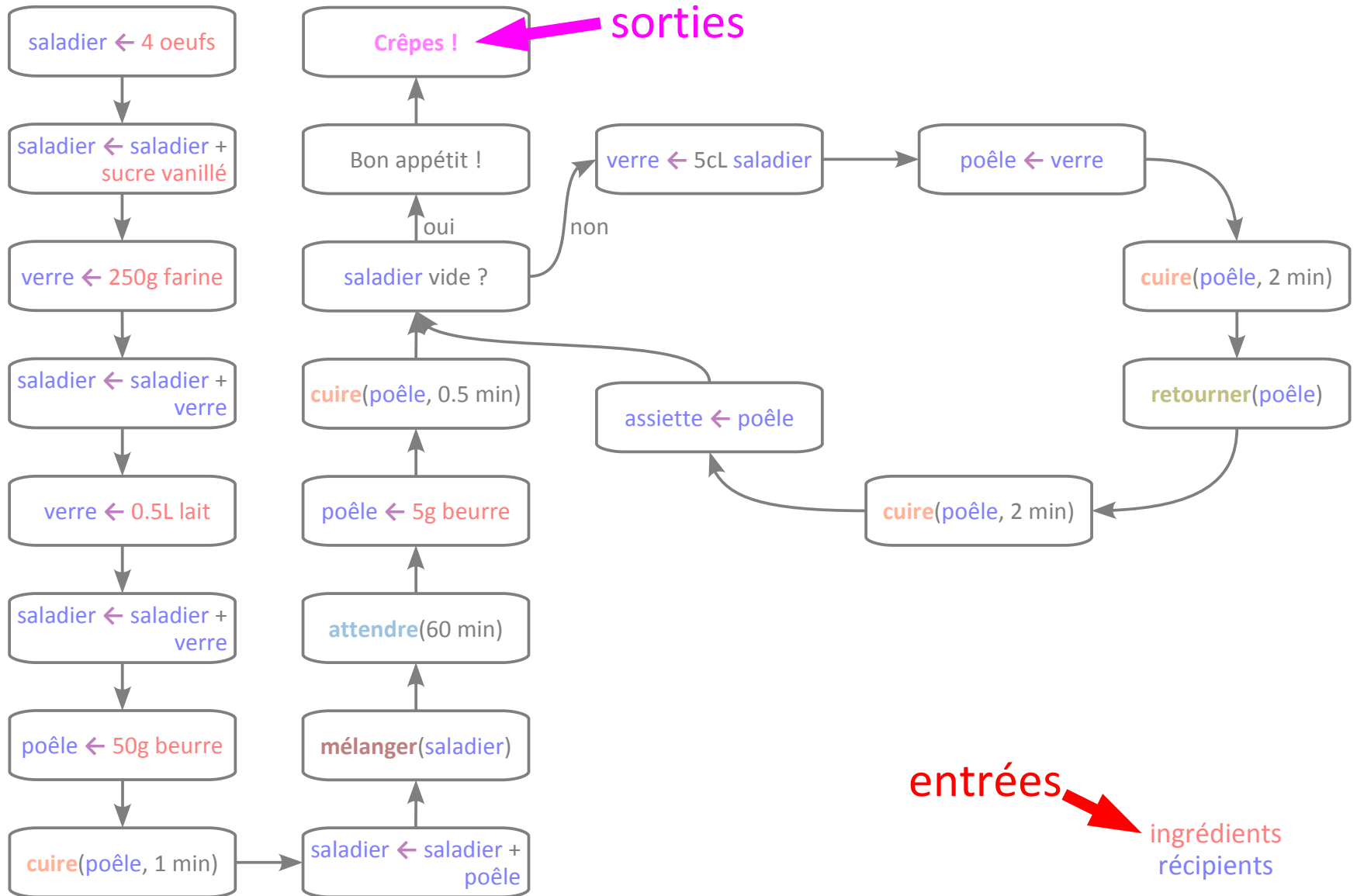
- test

# Organigramme de la recette des crêpes

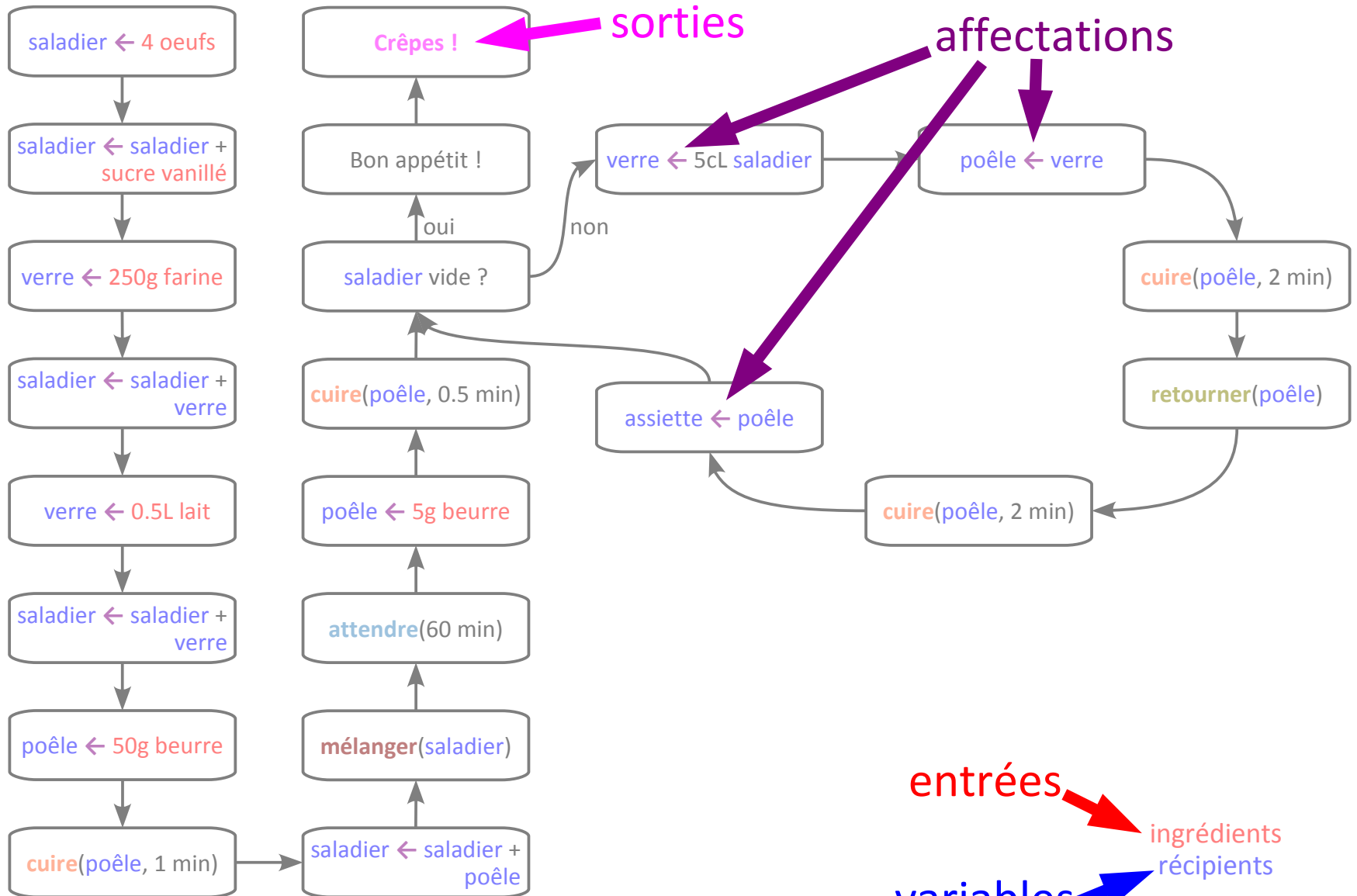


ingrédients  
réceptifs

# Organigramme de la recette des crêpes



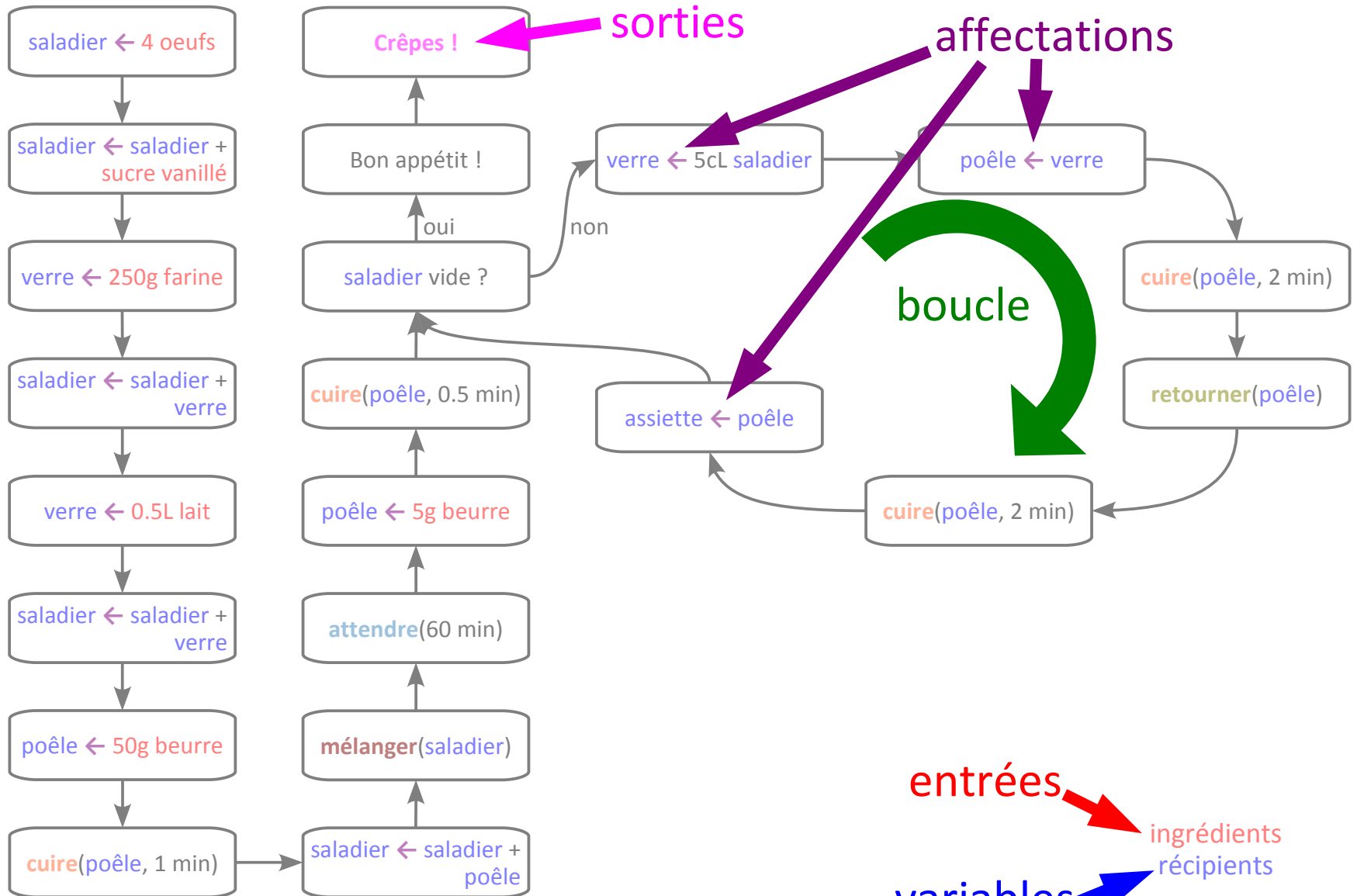
# Organigramme de la recette des crêpes



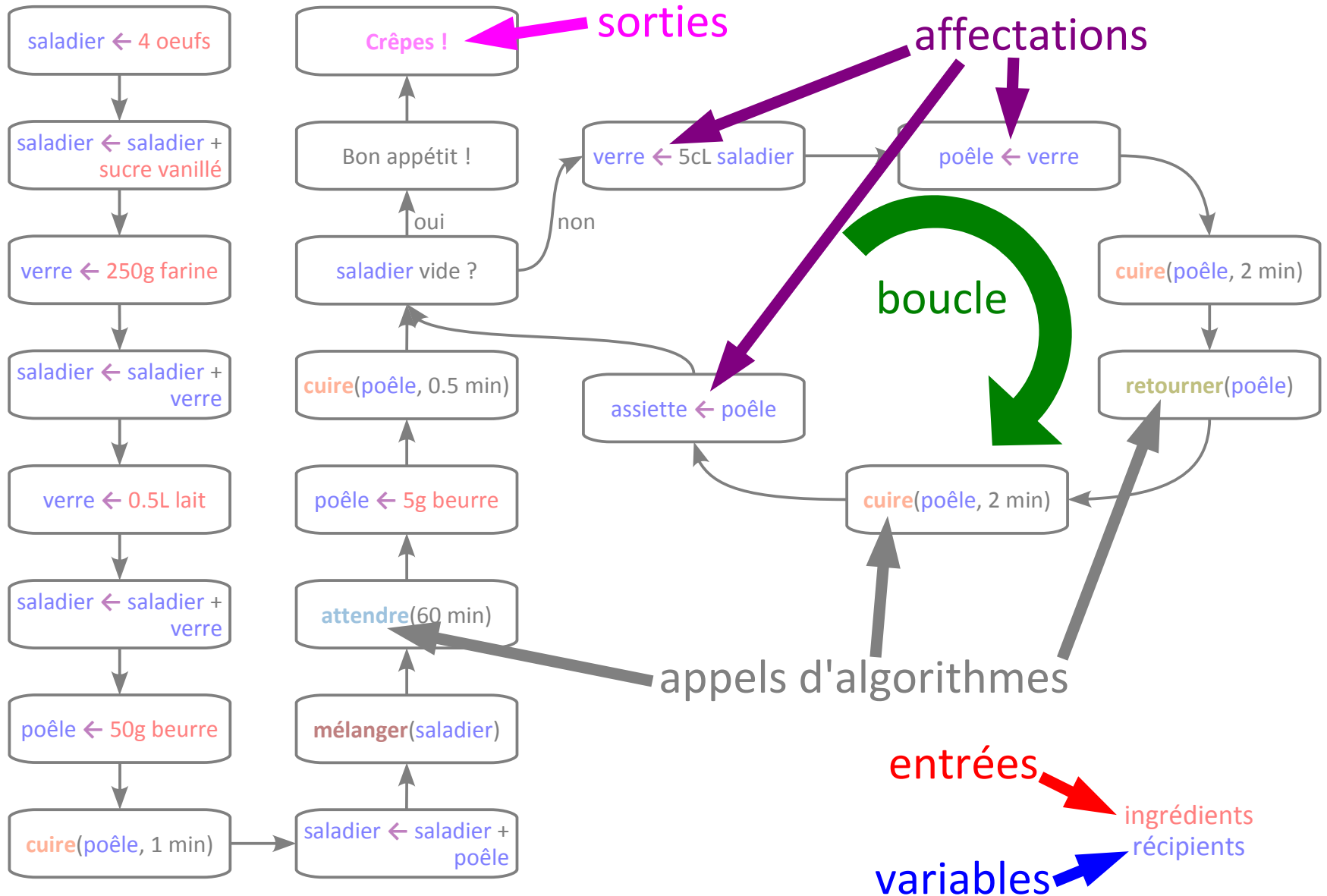
entrées → ingrédients  
variables → récipients



# Organigramme de la recette des crêpes



# Organigramme de la recette des crêpes



# Premier programme Java

```
public static void main(String[]arg){
    String nom, reponse;
    int nombreATrouver, nombreUtilisateur;

    //Identification de l'utilisateur
    nom = reponseALAQuestion("Comment vous appelez-vous ?");

    Affiche(" ");
    Affiche("Bonjour "+nom+" !");

    //Choix du nombre aléatoire
    nombreATrouver = nombreAleatoire(1,10);
    Affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
    Affiche("Essayez de le deviner.");

    //Premier essai de l'utilisateur
    reponse = reponseALAQuestion("Premier essai :");
    nombreUtilisateur = Integer.parseInt(reponse);

    if (nombreUtilisateur == nombreATrouver){
        Affiche("Bravo "+nom+", vous avez trouve !");
    }
}
```

entrées

sorties

# Premier programme Java

```
public static void main(String[] arg){
    String nom, reponse;
    int nombreATrouver, nombreUtilisateur;

    //Identification de l'utilisateur
    nom = reponseALAQuestion("Comment vous appelez-vous ?");

    Affiche(" ");
    Affiche("Bonjour "+nom+" !");

    //Choix du nombre aléatoire
    nombreATrouver = nombreAleatoire(1,10);
    Affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
    Affiche("Essayez de le deviner.");

    //Premier essai de l'utilisateur
    reponse = reponseALAQuestion("Premier essai :");
    nombreUtilisateur = Integer.parseInt(reponse);

    if (nombreUtilisateur == nombreATrouver){
        Affiche("Bravo "+nom+", vous avez trouve !");
    }
}
```

The diagram illustrates the flow of data between the code and external inputs/outputs. Red arrows labeled 'entrées' (inputs) point to the string literals in the first two `reponseALAQuestion` calls. Magenta arrows labeled 'sorties' (outputs) point to the `Affiche` statements that print text to the console.

variables

# Premier programme Java

```
public static void main(String[] arg){
    String nom, reponse;
    int nombreATrouver, nombreUtilisateur;

    //Identification de l'utilisateur
    nom = reponseALAQuestion("Comment vous appelez-vous ?");

    Affiche(" ");
    Affiche("Bonjour "+nom+" !");

    //Choix du nombre aléatoire
    nombreATrouver = nombreAleatoire(1,10);
    Affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
    Affiche("Essayez de le deviner.");

    //Premier essai de l'utilisateur
    reponse = reponseALAQuestion("Premier essai :");
    nombreUtilisateur = Integer.parseInt(reponse);

    if (nombreUtilisateur == nombreATrouver){
        Affiche("Bravo "+nom+", vous avez trouve !");
    }
}
```

entrées

sorties

variables affectations

# Premier programme Java

```
public static void main(String[] arg){
    String nom, reponse;
    int nombreATrouver, nombreUtilisateur;

    //Identification de l'utilisateur
    nom = reponseALAQuestion("Comment vous appelez-vous ?");

    Affiche(" ");
    Affiche("Bonjour "+nom+" !");

    //Choix du nombre aléatoire
    nombreATrouver = nombreAleatoire(1,10);
    Affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
    Affiche("Essayez de le deviner.");

    //Premier essai de l'utilisateur
    reponse = reponseALAQuestion("Premier essai :");
    nombreUtilisateur = Integer.parseInt(reponse);

    if (nombreUtilisateur == nombreATrouver){
        Affiche("Bravo "+nom+", vous avez trouve !");
    }
}
```

variables affectations appel d'algorithme

# Premier programme Java

```
public static void main(String[] arg){
    String nom, reponse;
    int nombreATrouver, nombreUtilisateur;

    //Identification de l'utilisateur
    nom = reponseALAQuestion("Comment vous appelez-vous ?");

    Affiche(" ");
    Affiche("Bonjour "+nom+" !");

    //Choix du nombre aléatoire
    nombreATrouver = nombreAleatoire(1,10);
    Affiche("L'ordinateur a choisi un nombre entre 1 et 10.");
    Affiche("Essayez de le deviner.");

    //Premier essai de l'utilisateur
    reponse = reponseALAQuestion("Premier essai :");
    nombreUtilisateur = Integer.parseInt(reponse);

    if (nombreUtilisateur == nombreATrouver){
        Affiche("Bravo "+nom+", vous avez trouve !");
    }
}
```

entrées

sorties

variables affectations appel d'algorithme test

# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

En revanche, le nom et le type d'une variable ne changent pas.



# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

L'**affectation** change la valeur d'une variable :

- $a \leftarrow 5$  (pseudo-code) /  $a=5$  (Java) :
  - la variable  $a$  prend la valeur 5
  - la valeur précédente est perdue (“écrasée”)
- $a \leftarrow b$  (pseudo-code) /  $a=b$  (Java) :
  - la variable  $a$  prend la valeur de la variable  $b$
  - la valeur précédente de  $a$  est perdue (“écrasée”)
  - la valeur de  $b$  n'est pas modifiée
  - $a$  et  $b$  doivent être de même type (ou de type compatible)

# Variables et affectation

Dans un algorithme, une **variable** possède :

- un **nom**,
- une **valeur**,
- un **type** (ensemble des valeurs que peut prendre la variable).

La **valeur** d'une variable :

- est **fixe à un moment donné**,
- peut **changer au cours du temps**.

L'**affectation** change la valeur d'une variable :

- $a \leftarrow 5$  (pseudo-code) /  $a=5$  (Java) :
  - la variable  $a$  prend la valeur 5
  - la valeur précédente est perdue (“écrasée”)
- $a \leftarrow b$  (pseudo-code) /  $a=b$  (Java) :
  - la variable  $a$  prend la valeur de la variable  $b$
  - la valeur précédente de  $a$  est perdue (“écrasée”)
  - la valeur de  $b$  n'est pas modifiée
  - $a$  et  $b$  doivent être de même type

La recette de cuisine avec  
récipients n'est qu'une métaphore

(ou de type compatible)

# Noms des variables

Dans un **algorithme**, choisir pour les variables :

- un nom composé de **lettres** et éventuellement de **chiffres**
- un nom **expressif**, par exemple :
  - *chaine, requête1...* pour une chaîne de caractères
  - *n, a, b, compteur, nbOperations, longueur...* pour un entier
  - *x, y, température* pour un réel
  - *estEntier, testEntier, trouvé...* pour un booléen
- un nom **assez court** (il faut l'écrire !)
- éviter les **noms réservés** : *pour, tant que, si...*

Dans un **programme** :

- **éviter** les lettres accentuées et la ponctuation
- préférer l'**anglais** si votre code source est diffusé largement
- être **expressif** et **lisible** :
  - *est\_entier* ou *estEntier* plutôt que *estentier*

Votre code sera relu, par vous ou par d'autres...