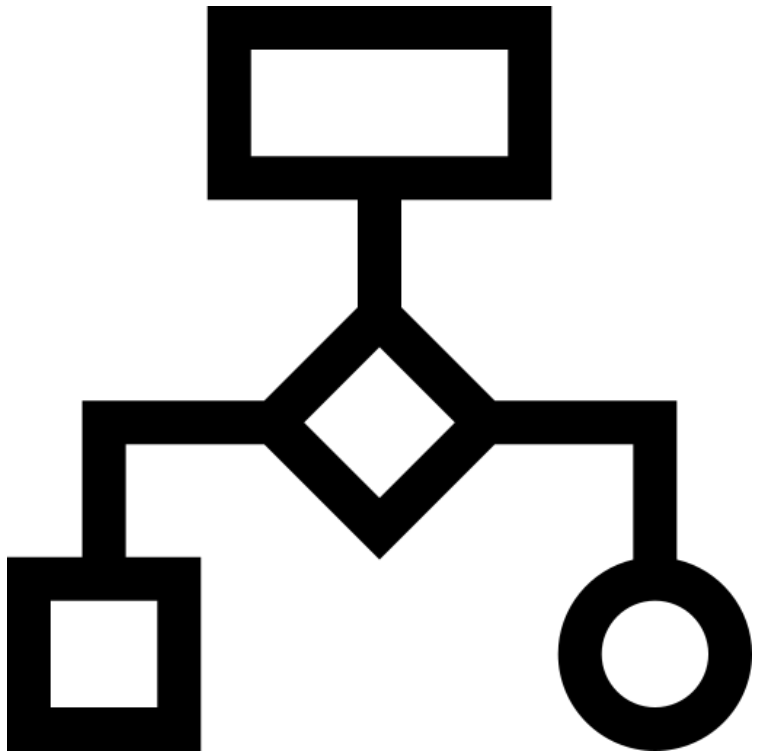
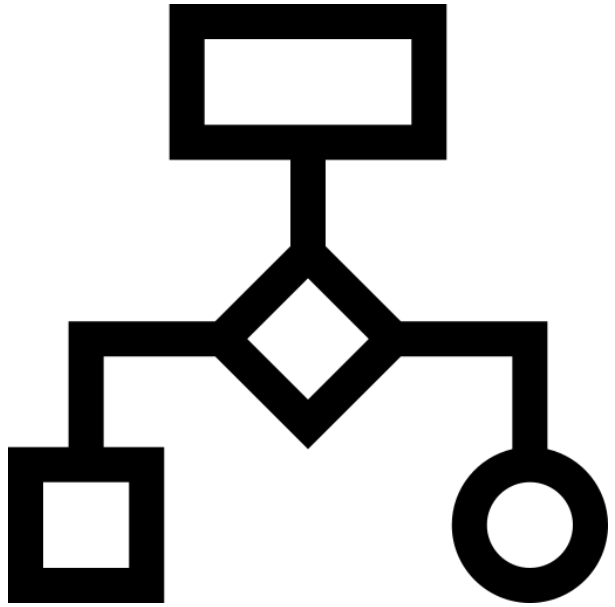




L’algorithmique

Qu'est-ce que la programmation ?

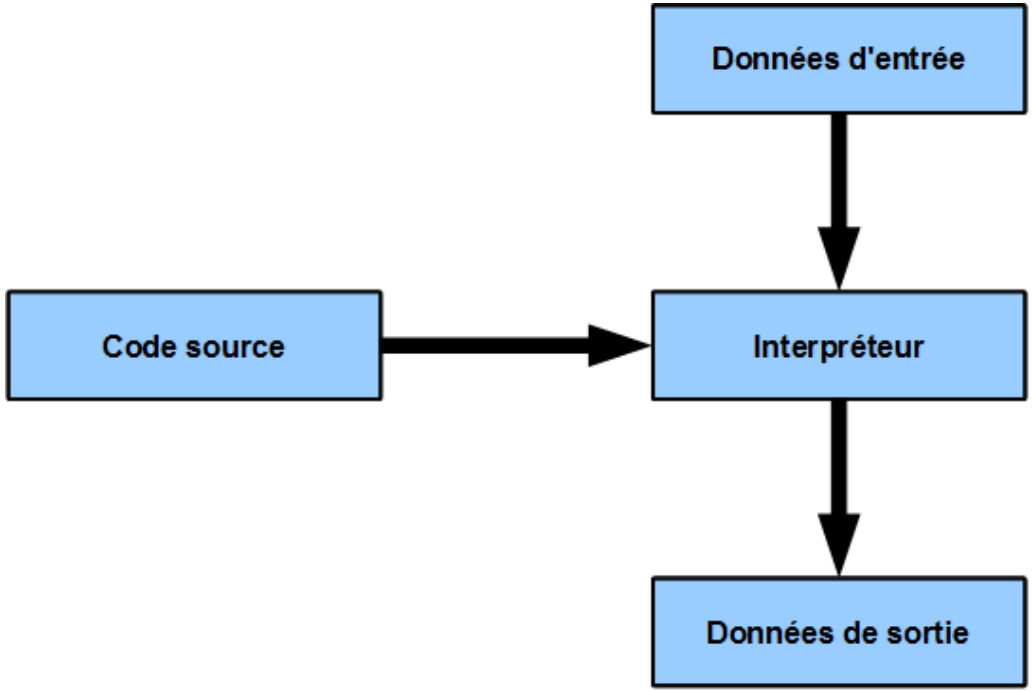
- La programmation est avant tout une **méthode** d'analyse et **non** l'apprentissage d'un langage.

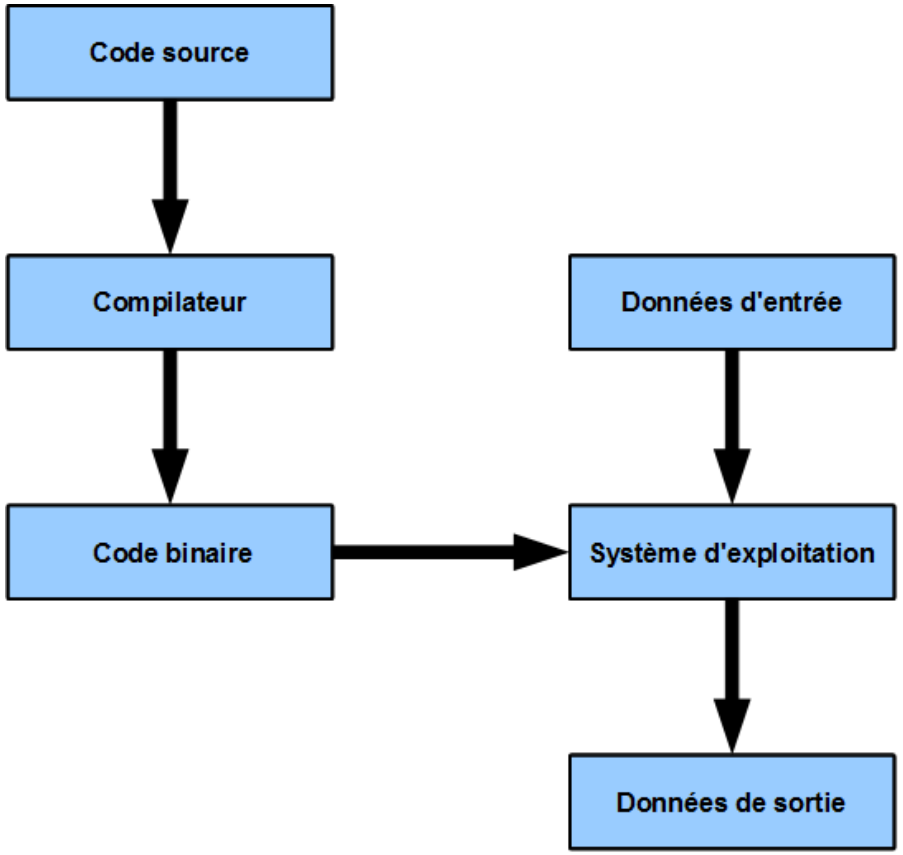
- Pour bien comprendre le fonctionnement d'un programme, il convient de le décomposer en **modules indépendants** et **élémentaires** qui se contentent d'effectuer une **action spécifique**.

- En utilisant cette technique, la **conception** devient **aisée**, vous faites **moins d'erreurs** et la **maintenance** du code est **simplifiée**.

Du code à l'exécution

Si le langage est interprété, le code est directement exécuté dans une application dédiée (ex: html ou batch)





Sinon le code doit être converti en instructions exécutables. Il est passé à un programme appelé "compilateur". Il en ressort un code directement exécutable, sous la forme d'un fichier exécutable (.exe, .jar …)

Ne pas tout réécrire

Lorsque vous écrivez un code informatique :

- vous pouvez bien souvent vous appuyer sur des "**bibliothèques**" pour éviter de tout réécrire.

Tous les langages proposent des **bibliothèques complémentaires** pour **simplifier** vos développements.

- N'hésitez pas à les utiliser.

- Votre temps de développement sera **réduit** d'autant ...

Qu’est-ce qu’un algorithme ?

Une solution pour résoudre un problème :

- ✓ Une recette de cuisine ;
- ✓ Protocole expérimental ;
- ✓ Des instructions pour aller quelque part ;
- ✓ Des consignes de sécurité

Dans **notre contexte**, c'est :

- une **méthode** de **résolution** d'un problème,
- **écrite** de manière non **ambigüe** ;
- et **susceptible** d'être codée sur ordinateur

Ambiguïté :

Quelques exemples à éviter:

- vends tricycle pour infirme en bon état ;
- Deux conducteurs étaient interpellés par les gendarmes en état d'ivresse.

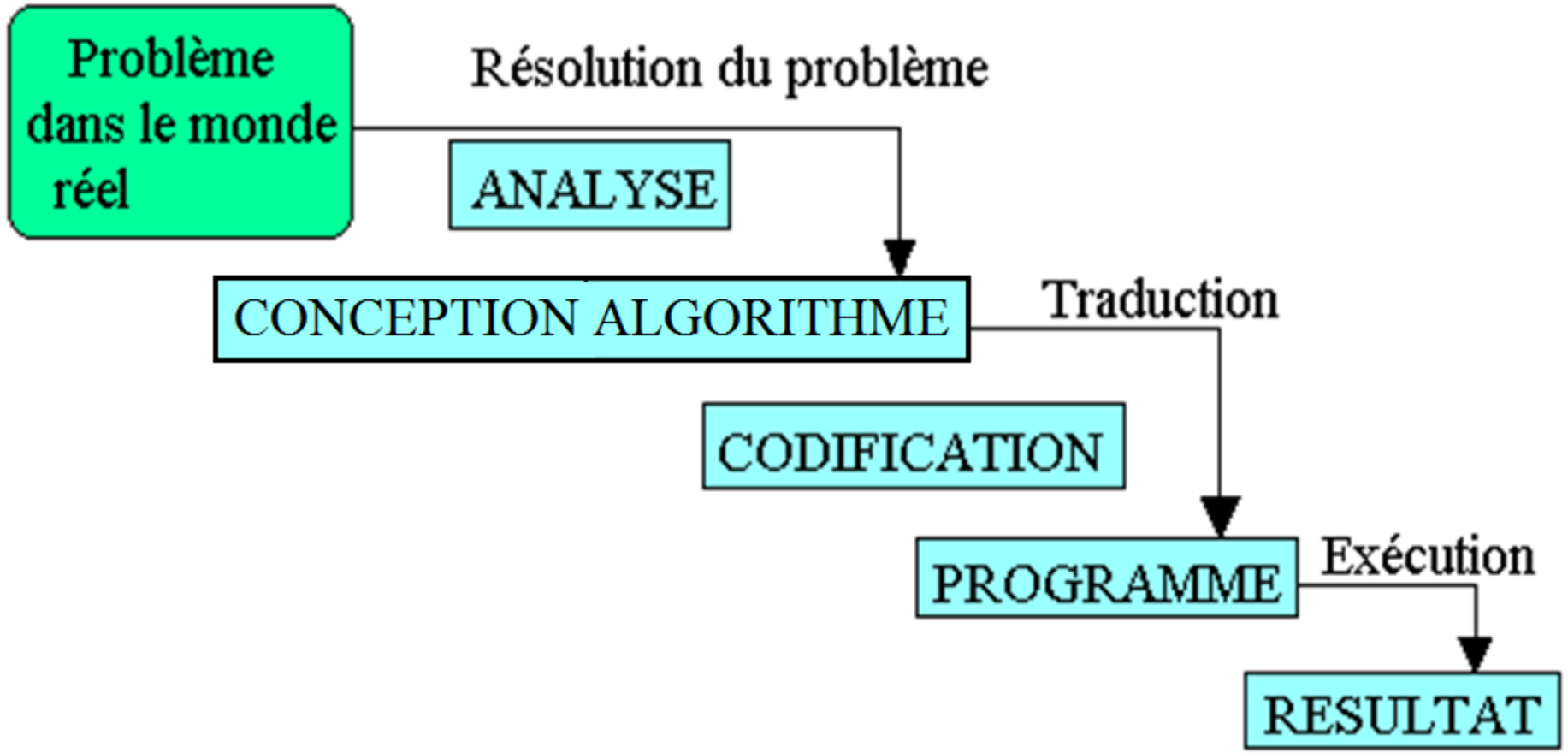
Le « **français** » **ne convient pas** :

- ✓ il faut avoir des **règles** de syntaxe ;
- ✓ et un vocabulaire **précis** (si possible pas trop grand pour ne pas s'encombrer l'esprit !!!).

Dans le monde réel nous utilisons des **algorithmes** pour **chacune** des actions que nous faisons.

- **Problème** :
  - ✓ j'ai soif.
- **Solution** :
  - ✓ prendre un verre dans le placard ;
  - ✓ le mettre sous le robinet ;
  - ✓ ouvrir le robinet ;
  - ✓ attendre que le verre soit plein ;
  - ✓ fermer le robinet ;
  - ✓ boire.

Du problème à la solution, trois ou quatre étapes :



Analyse

L'**analyse** est une phase de réflexion préalable qui permet d'identifier **précisément** le problème :

- ✓ données à traiter ;
- ✓ résultats attendus ;
- ✓ cas particuliers ;
- ✓ traitements à effectuer ;
- ✓ etc.

L'analyse permet également de **découper** le problème en une **succession** de **tâches simples** à enchaîner pour arriver jusqu'à la **résolution**.

L'analyse est **fondamentale**. En effet, la résolution d'un problème ne peut être dissociée de sa compréhension. Mieux vous comprendrez le problème, plus facilement vous le résoudrez.

Conception

La conception, c’est :

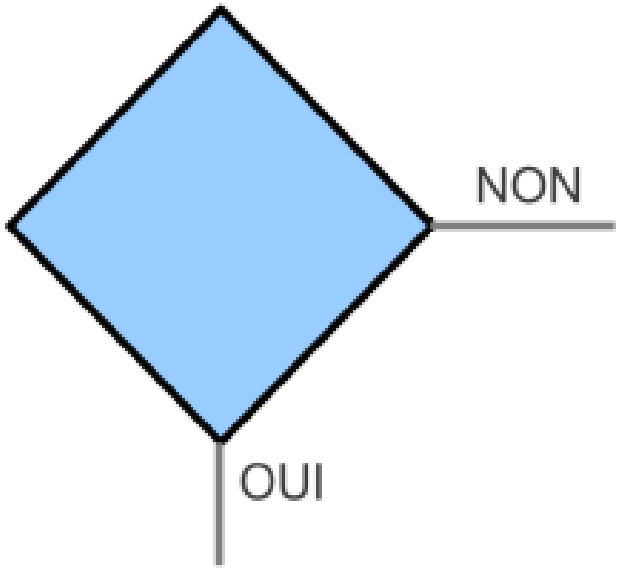
- la définition des opérations **élémentaires** à appliquer aux données pour obtenir le **résultat** attendu ;
- la mise en évidence de la **logique** d'enchaînement de ces opérations élémentaires ;
- la prise en compte des **cas particuliers**.

Utilisation d'organigrammes

Un organigramme est une représentation **schématique** des **liens fonctionnels**, **organisationnels** et **hiérarchiques** d'un algorithme ou d'un programme.

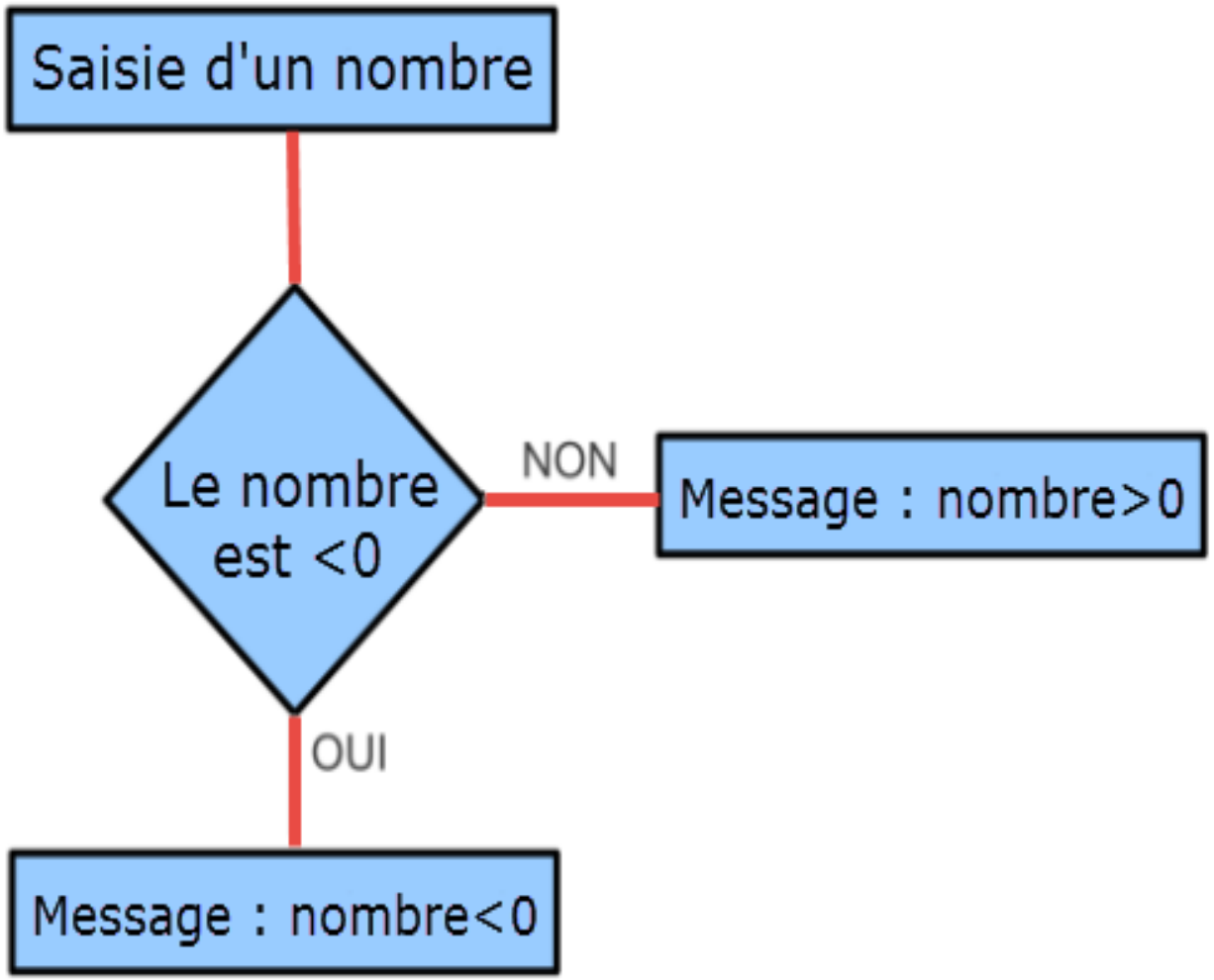
Vous utiliserez essentiellement les éléments suivants pour vos organigrammes :

Action à effectuer.



Test d'une condition.

Un exemple d'organigramme :



Algorithme

La définition d'un algorithme consiste en la **mise en œuvre d'actions élémentaires** à l'aide d'une **notation dédiée** :

- ✓ déclaration de **variables** ;
- ✓ demande de **données** à l'utilisateur ;
- ✓ **Boucles** ;
- ✓ Test ;
- ✓ Affichage ;
- ✓ etc.

L'écriture :

```
Algorithme Puissance
// algorithme qui calcule une puissance d'un nombre
Variables
     nombre,puissance : réels;
     total,i : entier;
Début
     nombre ← Saisie(); // L'utilisateur doit entrer un réel
     puissance ← Saisie(); // L'utilisateur doit entrer un entier
     total ← 1; // initialisation de la variable puissance
     Pour i allant de 1 à nombre faire // « répéter k fois » n'existe
pas…
          total ← puissance * nombre;
     fin pour
     Ecrire(total);
Fin
```

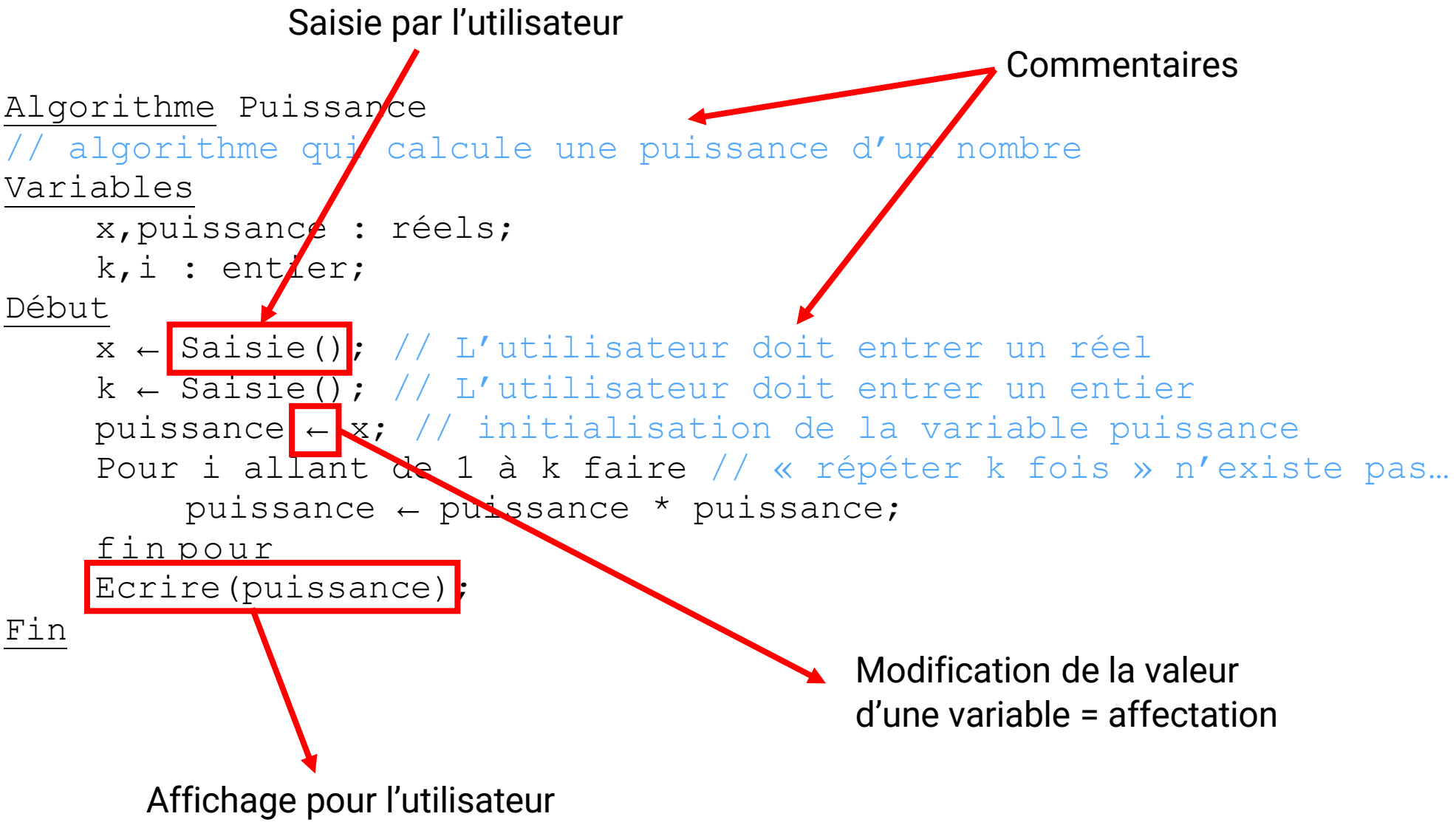
L'écriture :

```
Algorithme Puissance                                    ENTÊTE
// algorithme qui calcule une puissance d'un nombre
```

```
Variables
    x,puissance : réels;                                DECLARATIONS
    k,i : entier;
```

```
Début
    x ← Saisie(); // L'utilisateur doit entrer un réel
    k ← Saisie(); // L'utilisateur doit entrer un entier
    puissance ← x; // initialisation de la variable puissance
    Pour i allant de 1 à k faire // « répéter k fois » n'existe pas…
        puissance ← puissance * puissance;
    fin pour
    Ecrire(puissance);                                  Corps = description du calcul et des intéractions
Fin
```

L'écriture :


Saisie par l'utilisateur
Commentaires
Algorithme Puissance
// algorithme qui calcule une puissance d'un nombre
Variables
x,puissance : réels;
k,i : entier;
Début
x ← Saisie(); // L'utilisateur doit entrer un réel
k ← Saisie(); // L'utilisateur doit entrer un entier
puissance ← x; // initialisation de la variable puissance
Pour i allant de 1 à k faire // « répéter k fois » n'existe pas…
puissance ← puissance * puissance;
fin pour
Ecrire(puissance);
Fin
Modification de la valeur
d'une variable = affectation
Affichage pour l'utilisateur

Ecrire un algorithme

99% des algorithmes se décomposent en **3** parties :

1. ce qu'on **demande** à l'utilisateur ;
2. ce qu'on **calcule** (la partie difficile en général) ;
3. ce qu'on **restitue** à l'utilisateur (qui ne peut pas démonter l'ordinateur pourvoir l'état de la mémoire !).

```
Début

1   x ← Saisie(); // L'utilisateur doit entrer un réel
    k ← Saisie(); // L'utilisateur doit entrer un entier

    puissance ← x; // initialisation de la variable puissance
2   Pour i allant de 1 à k faire // « répéter k fois » n'existe pas…
        puissance ← puissance * puissance;
    fin pour

3   Ecrire(puissance);

Fin
```

Exercice

Ecrire un algorithme qui demande à l'utilisateur de saisir une année de naissance et calcule l'âge qu'aura la personne au 31 décembre 2024 à minuit.

Solution possible :

Ecrire un algorithme qui demande à l'utilisateur de saisir une année de naissance et calcule l'âge qu'aura la personne au 31 décembre 2024 à minuit.

```
Algorithme      Calcule âge
Variables

                annee : entier
                age : entier

Début

                annee ← Saisie();
                age ← 2024 - annee;
                Ecrire(age);

Fin
```

Questions ??

Prochain chapitre: Variables, constantes et instructions.