Chapitre 1

Introduction à l'algorithmique

Dr Toundé Mesmin DANDJINOU

Objectifs pédagogiques du chapitre

Objectif général

A la fin du chapitre, l'étudiant est capable :

- d'expliquer ce qu'est un algorithme
- de dérouler la démarche de réalisation d'un algorithme

Objectifs pédagogiques du chapitre

Objectifs spécifiques

A la fin du chapitre, l'étudiant est capable :

- de définir les concepts d'algorithme et d'algorithmique
- de dérouler le processus d'écriture d'un algorithme
- d'expliquer ce qu'est un programme
- d'expliquer le processus de réalisation d'un programme (cycle de développement du logiciel)

Contenu

- Notions d'algorithme et d'algorithmique
- Notions de données dans l'énoncé d'un problème
- Processus de description d'un algorithme
- Notion de programme
- Processus de réalisation/développement d'un programme

Partie 1 - De l'algorithme au **programme**

Introduction à l'algorithmique

- Un **algorithme** : description sans ambiguïté du cheminement (succession de traitements) conduisant à la résolution d'un problème donné

- **Algorithmique** : domaine de l'informatique consacré à l'étude des algorithmes

Introduction à l'algorithmique

- Poser un problème à résoudre suppose l'**expression, l'énoncé, la formulation de ce problème**

- A partir de l'énoncé du problème on peut repérer :

  - les **données en entrée** : ce sont généralement des valeurs connues au départ

  - les **données en sortie/résultat** : ce sont des valeurs calculées obtenues à la fin de la résolution du problème

  - les **données intermédiaires/temporaires** : elles sont à mi-chemin entre les deux précédentes catégories de valeurs car produites par des traitements intermédiaires

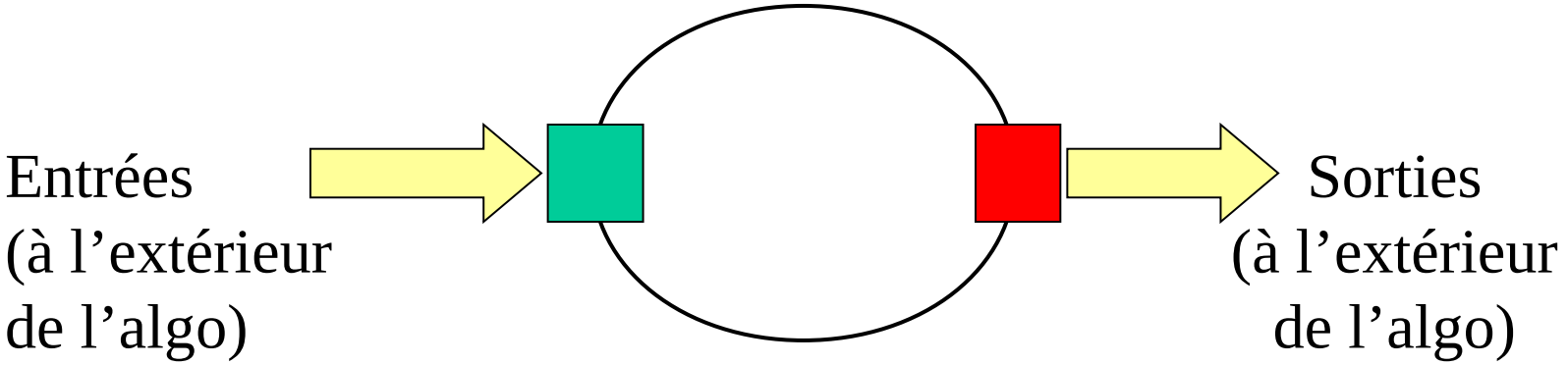
Introduction à l'algorithmique

Algorithme

NB : le nombre de données varie selon le problème à résoudre

Introduction à l'algorithmique

• **Donnée en entrée** = valeur fournie par l'utilisateur à l'algorithme qui le stocke dans une **variable en entrée** (input), ou bien une **variable supposée déjà initialisée**



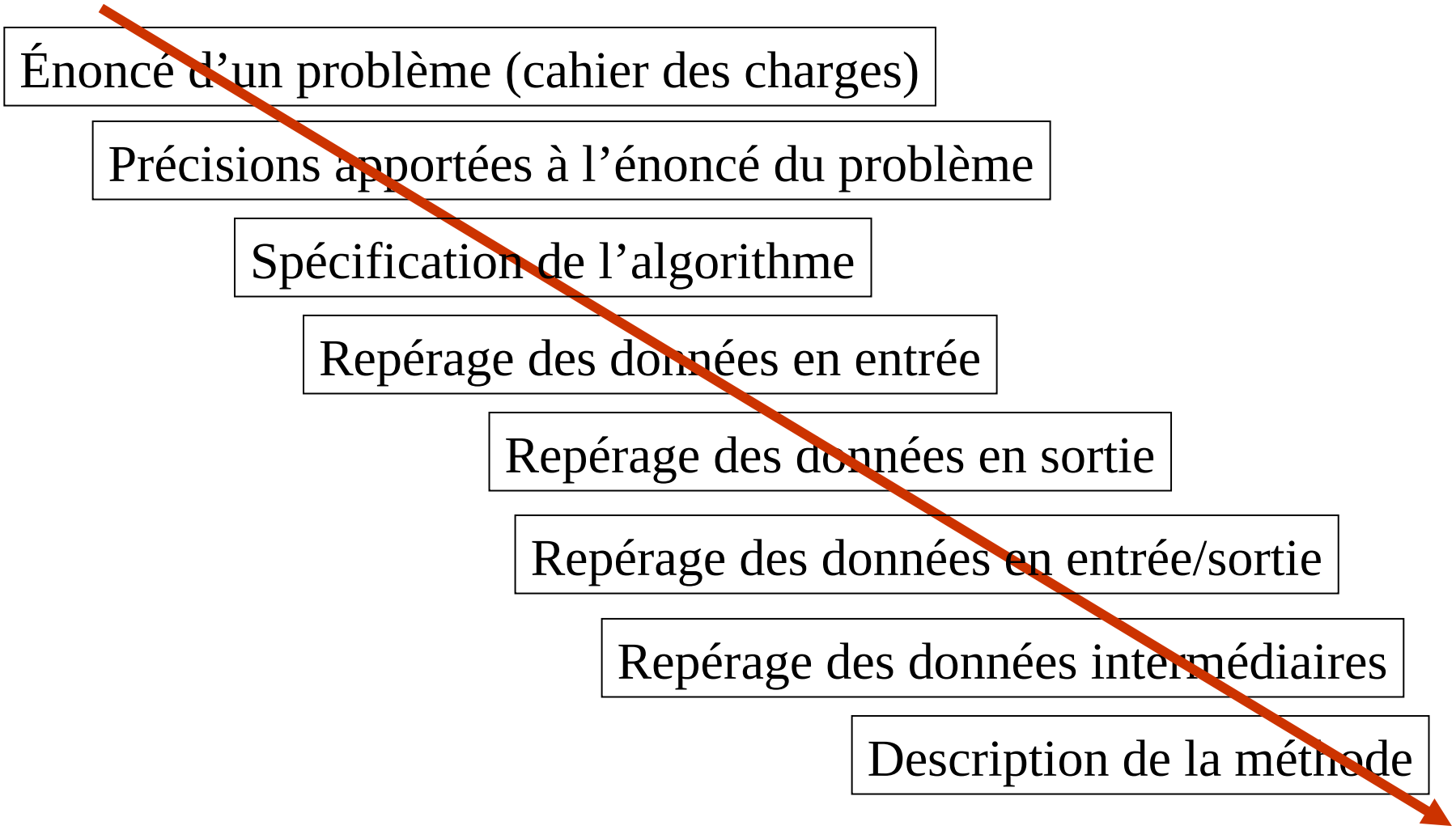
•**Donnée en sortie/résultat** = valeur calculée par l'algorithme et rangée dans une **variable en sortie/résultat** (output) attendue comme un résultat pour l'utilisateur

Introduction à l'algorithmique

- **Enoncé du problème** (**cahier des charges**) parfois incomplet ⇒ nécessité de **préciser** (compléments d'informations pour lever les ambiguïtés) le problème avant de concevoir la solution

- Le problème posé par un client dans ses termes à lui doit être traduit dans les termes des informaticiens : c'est la phase de **spécification du problème** conduisant à la **phase de spécification de l'algorithme** **où l'on dit ce que fait l'algorithme sans détailler comment il le fait**

Introduction à l'algorithmique



Introduction à l'algorithmique

• **Écriture/rédaction de l'algorithme** : usage d'un **langage de description très proche de celui que nous avons l'habitude d'employer** (français + expressions mathématiques, graphiques)

• **Indépendance** d'un algorithme de **tout langage de programmation**

Introduction à l'algorithmique

- Pour résoudre **un problème**, généralement **plusieurs algorithmes** peuvent être proposés
- Le **meilleur algorithme** est celui qui utilise au mieux les ressources (temps de calcul, nombre de lignes décrivant l'algorithme, taille des objets requis, efforts de compréhension à faire, etc.)
- Il existe des problèmes pour lesquels on ne connaît pas d'algorithme

Introduction à l'algorithmique

Deux approches de réalisation de la solution :

- ✓ **approche ascendante** (bottom-up)
- ✓ **approche descendante** (top-down)

Introduction à l'algorithmique

- Dans **l'approche ascendante**, c'est à partir du niveau le plus fin des solutions existantes qu'on démarre, et puis par des **assemblages successifs** on réalise la solution globale

- Le gros risque de cette approche c'est de réaliser au final une solution globale qui ne réponde pas au besoin du client

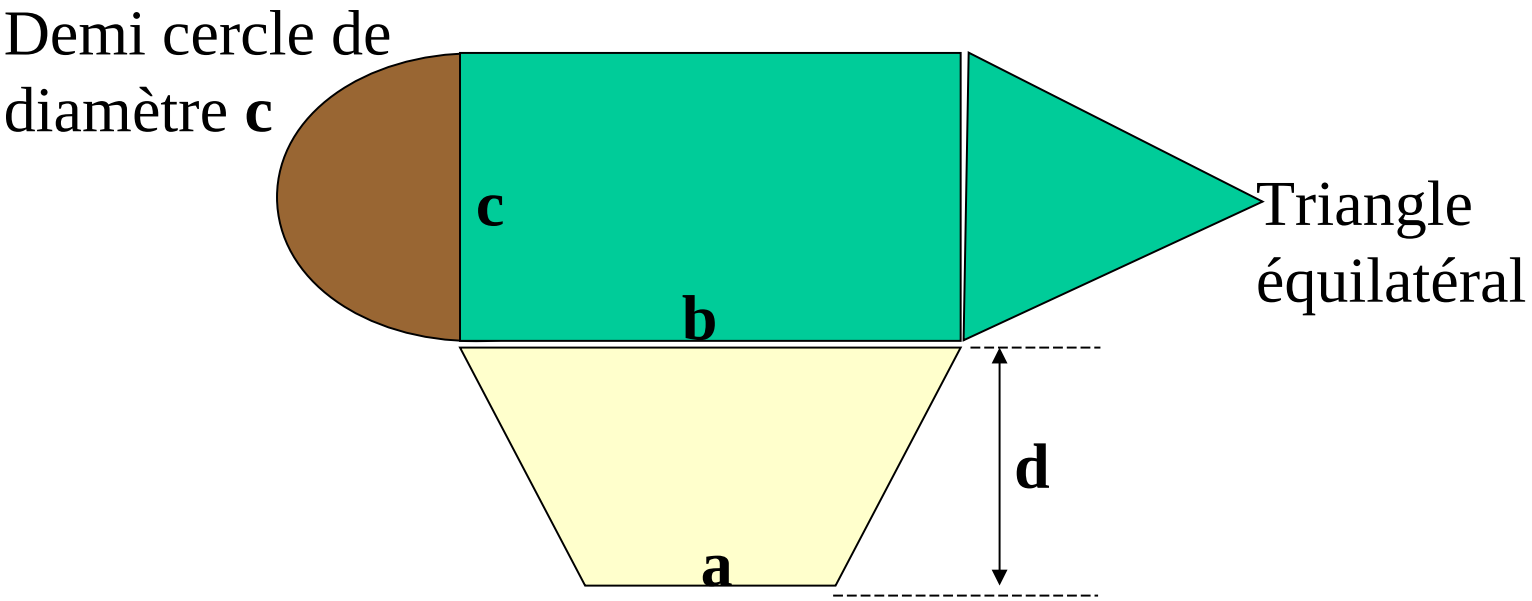
Introduction à l'algorithmique

- Dans **l'approche descendante** (la plus souvent employée) on part d'une solution globale qui va au fur et à mesure être détaillée/raffinée : on dit qu'on procède par **raffinements successifs**

- **Cette approche descendante** correspond au découpage/décomposition du « gros problème » en de « plus petits problèmes » à résoudre : c'est la politique du « **diviser pour régner** » qui est employée

Introduction à l'algorithmique

- Exemple : On demande de calculer la superficie de la figure géométrique dont le schéma est donné ci-après :



Introduction à l'algorithmique

a) Spécification : Soit S la superficie recherchée de la figure polyforme donnée qui se compose :

- d'un demi-cercle de diamètre **c**,
- d'un rectangle de longueur **b** et de largeur **c**,
- d'un trapèze de grande base **b**, de petite base **a** et de hauteur **d**,
- d'un triangle équilatéral de côté **c**.

Introduction à l'algorithmique

b) Données/variables en entrée

- **a** est de type réel strictement positif, correspond à la petite base du trapèze constituant la figure polyforme ;
- **b** est de type réel, correspond à la longueur du rectangle et à la grande base du trapèze constituant la figure polyforme ;

Introduction à l'algorithmique

b) Données/variables en entrée (suite)

- **c** est de type réel, correspond au diamètre du demi-cercle, au côté du triangle équilatéral et à la largeur du rectangle constituant la figure polyforme ;
- **d** est de type réel, correspond à la hauteur du trapèze constituant la figure polyforme ;

Introduction à l'algorithmique

c) Données/variables en sortie

Néant

d) Données/variables en entrée-sortie

- **S** est de type réel, correspond à la superficie de la figure polyforme ;

Introduction à l'algorithmique

e) Données/variables intermédiaires

- **Sr** est de type réel, correspond à la superficie du rectangle ;
- **Sc** est de type réel, correspond à la superficie du demi cercle ;
- **St** est de type réel, correspond à la superficie du trapèze ;
- **Ste** est de type réel, correspond à la superficie du triangle équilatéral ;

Introduction à l'algorithmique

f) Description de l'enchaînement logique

/* APPROCHE ASCENDANTE */

Début

Sr ← c*b /* calcul de la superficie du rectangle */

Sc← $(\pi * c^2/4)/2$ /* calcul de la superficie du demi cercle */

St← (b+a)*d/2 /* calcul de la superficie du trapèze */

Ste← $((c^2)*(3)^{1/2})/4$ /* calcul de la superficie du triangle équilatéral */

S← Sr+Sc+St+Ste /* superficie de la figure polyforme */

Fin

Introduction à l'algorithmique

f) Description de l'enchaînement logique

/* APPROCHE DESCENDANTE */

Début

S← Sr+Sc+St+Ste /* superficie de la figure polyforme */

Fin

/* calcul superficie rectangle */

Sr ← c*b

/* calcul superficie demi cercle */

Sc← $(\pi * c^2/4)/2$

/* calcul superficie trapèze */

St← (b+a)*d/2

/* calcul superficie triangle équilatéral */

Ste← $((c^2)*(3)^{1/2})/4$

Introduction à l'algorithmique

Partie 2 : Du programme aux **résultats de l'exécution**

Introduction à l'algorithmique

- **PROGRAMME = Résultat de la traduction d'un algorithme dans un langage de programmation** que comprend un ordinateur

- De **nombreux langages de programmation existent** (langages de bas niveau et langages haut niveau ou évolués, langages de programmation impérative/procédurale, langages de programmation fonctionnelle, langages de programmation objet, langages de programmation événementielle, etc.)

Compilation & interprétation

- **Les programmes écrits par les programmeurs** sont appelés **PROGRAMMES SOURCES** car rédigés dans des langages de programmation

- **L'apprentissage** de l'**écriture** de ces **programmes** à travers le cours de **« programmation »**

Interprétation & Compilation

Les programmes sont :

- **soit directement** fournis à des **INTERPRETEURS :** on est dans le processus d'**INTERPRETATION**

- **soit traduits** en **PROGRAMMES EXECUTABLES** qui seront exécutés par les ordinateurs, après une succession de traitements ; on est dans le processus de **COMPILATION**

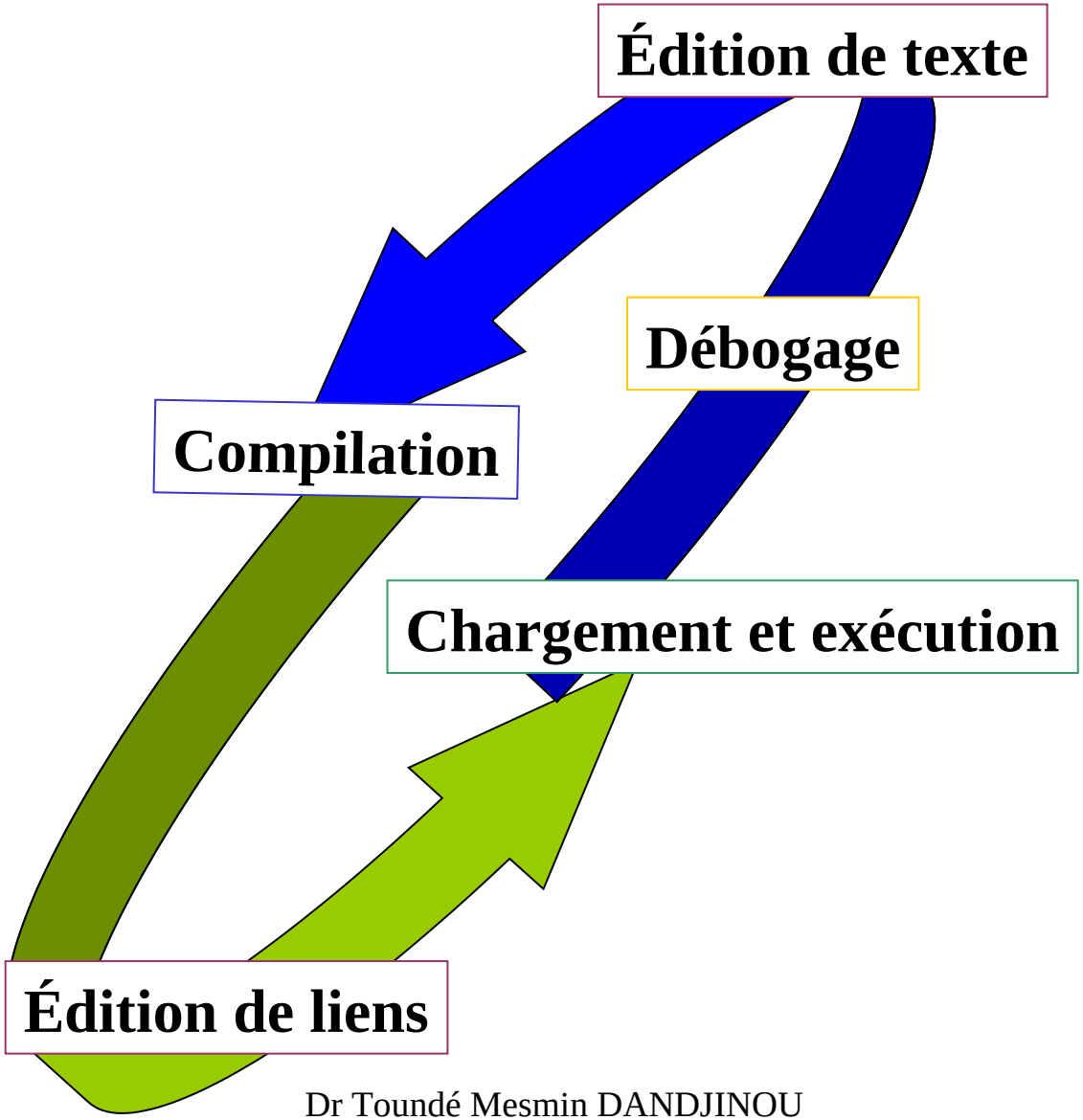
Interprétation

- **Le processus d'interprétation est employée lorsque :**
  - l'utilisateur souhaite **écrire rapidement un petit programme jetable (maquette)**
  - le **source** peut sans problème être **donné au client**
- **A chaque exécution du programme, chacune des instructions est analysée et traduite avant exécution, et le résultat de cette traduction n'est pas conservée dans un fichier exécutable**

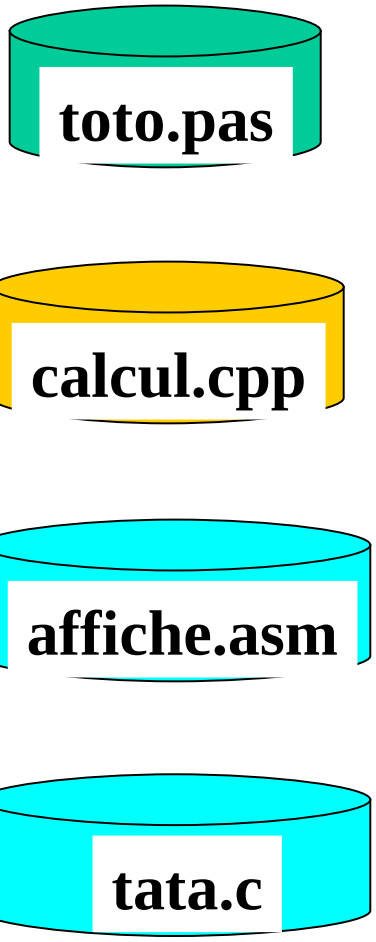
Production du code exécutable

- **Le processus de production d'un code exécutable (code binaire) est généralement employé quand :**
  - les **programmes** associés sont **de grande taille**
  - on souhaite **un temps d'exécution réduit**
  - on ne désire **pas** mettre le **code source** du programme à **la disposition du client final**
- **Les exécutions ne nécessitent plus aucune traduction car se faisant à partir du code exécutable**

Phases du développement de programmes



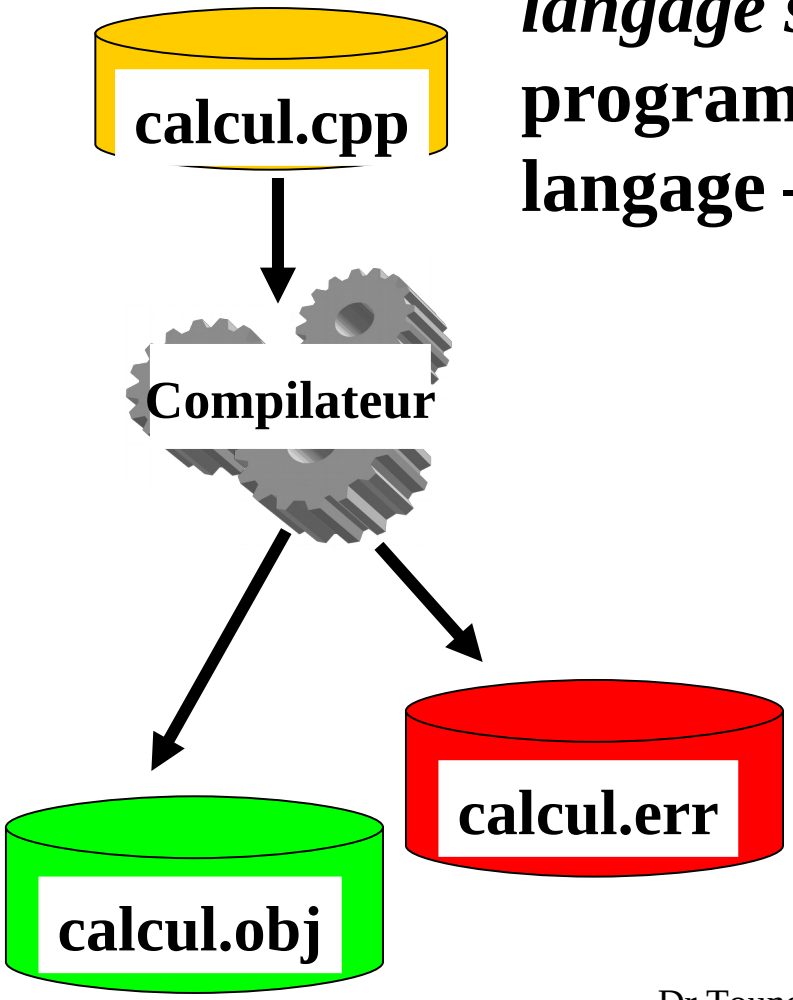
Édition de texte



- **Éditeur de texte = *programme de saisie de texte* par l'utilisateur *sans mise en forme*. Ce texte va correspondre à un programme écrit dans un langage de programmation (respect de la syntaxe d'écriture du programme et des instructions)**
- **Un *fichier source* résulte de cette opération et porte un nom dont l'extension rappelle le langage de programmation employé**

Compilation

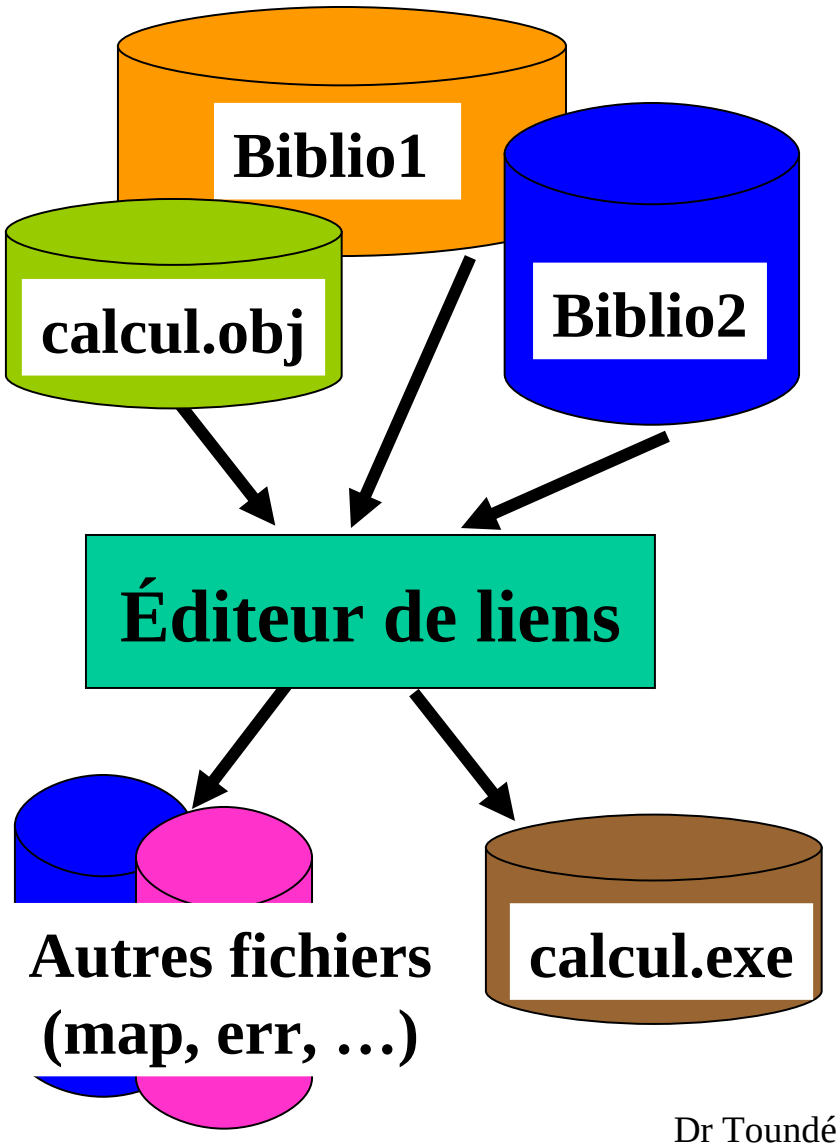
- **Compilateur = programme lisant un programme écrit dans un 1er langage – *langage source* – et le traduisant en un programme équivalent écrit dans un 2nd langage – *langage cible ou objet* -**



- Lors de ce processus **les erreurs rencontrées sont signalées** à l'utilisateur dans un ***fichier d'erreurs de syntaxe***

- *Est associé* ***à chaque langage de programmation*** *un* ***Compilateur***

Édition de liens



- Tous les *liens irrésolus* (variables, étiquettes, fonctions connues ailleurs que dans le programme compilé) sont traités, grâce à l'*utilisation des bibliothèques* et autres fichiers renfermant du code objet fourni à l'*éditeur de liens qui est un programme*
- Le fichier exécutable résultant contient le *programme traduit en langage machine (code binaire)*

Chargement et exécution

- **Chargeur = programme ayant pour rôle d'*installer en mémoire centrale le code exécutable d'un programme* dont on connaît le nom : trouver la place, copier le code en mémoire centrale avec les adaptations liées aux translations, préparer le contexte d'exécution**
- **Un programme exécutable ⇔ un nom de fichier renfermant du code exécutable**
- **Lorsque le processeur sera attribué au processus correspondant au programme chargé, celui-ci s'exécutera et produira des résultats**

Débogage

- **Débogueur = programme ayant pour rôle d'*aider à la mise au point d'un autre programme* en cours de développement (*détection des erreurs de logique* appelés en anglais « *bugs* »), en fournissant la possibilité de suivre pas à pas :**
  - **le déroulement de chacune des instructions constituant le programme testé**
  - **l'évolution du contenu des variables, des structures de données et autres zones en mémoire centrale employées dans le programme**

Environnement de développement intégré

- **Il existe des environnements de développement intégrés EDI où tous ces outils (éditeurs de texte, compilateurs, éditeurs de liens, chargeurs, débogueurs, *gestionnaire de dépendances*) sont mis à la disposition des programmeurs pour leur faciliter le travail**

- **Exemples : Microsoft Visual C++, Borland Jbuilder, CodeBlocks, etc.**

QUESTIONS ?