Chapitre 1
Introduction à l'algorithmique

Objectifs pédagogiques du chapitre

Objectif général

A la fin du chapitre, l'étudiant est capable de comprendre ce qu'est un algorithme et de derouler la démarche de sa réalisation

Objectifs pédagogiques du chapitre

Objectifs spécifiques

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

- de comprendre ce qu'est un algorithme
- de dérouler le processus d'écriture d'un algorithme
- de comprendre ce qu'est un programme
- de comprendre le processus de réalisation d'un programme (cycle de développement du logiciel)

Contenu

- Notions d'algorithmique et d'algorithme
- 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

Introduction à l'algorithmique

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

2.1 - Introduction

- Les programmeurs sont amenés à concevoir des logiciels de plus en plus complexes pour répondre aux besoins des utilisateurs (applications en réseau, base de données réparties, …)
- $\Rightarrow$maîtriser le développement des logiciels est très important

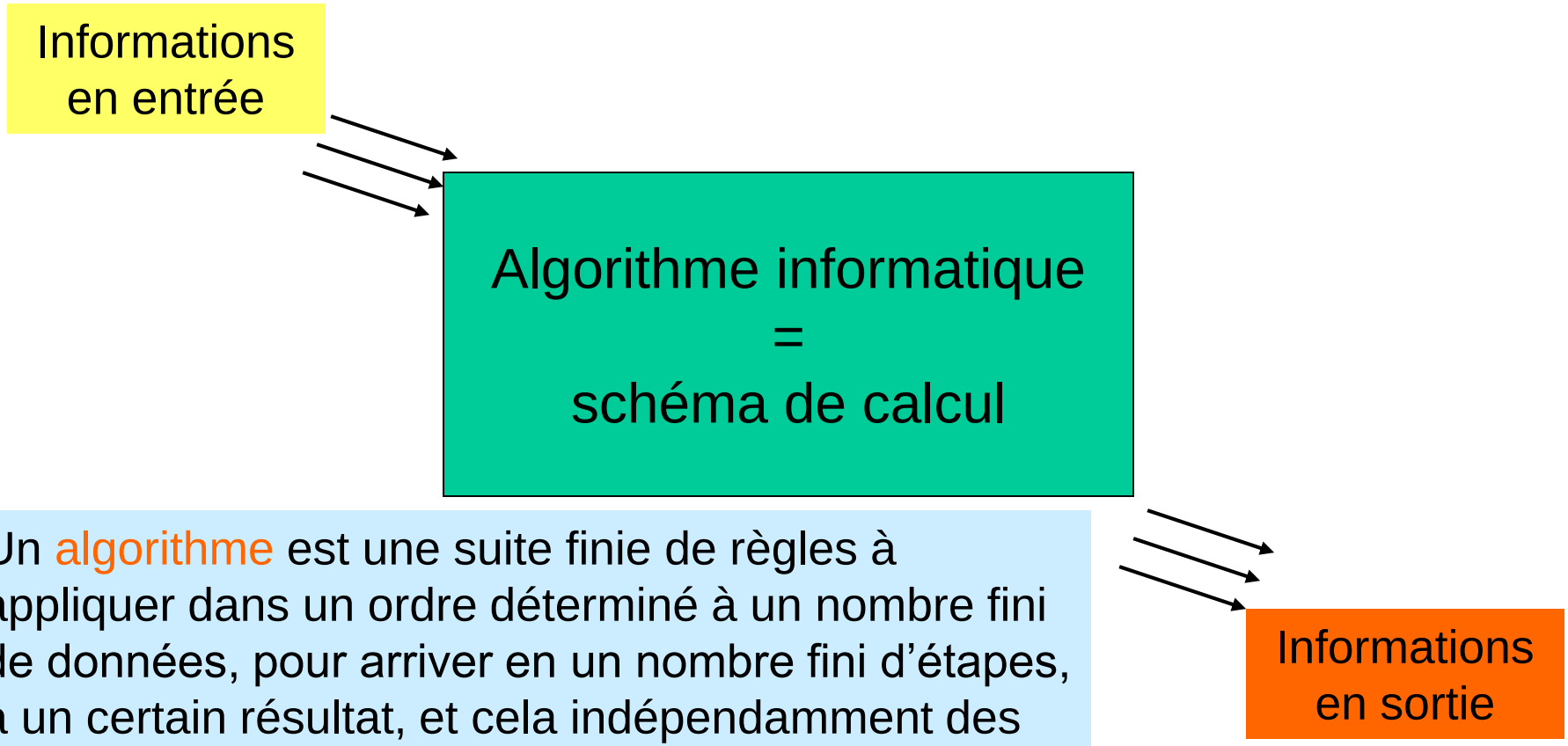
2.2 - Algorithmes et programmes

- Notion précise d'algorithme a été découverte en 825 par le mathématicien arabe Muhammmad ibn Musa al-Kharezmi
  - Moyen d'automatisation et d'économie de la pensée
- Petit Larousse : suite d'opérations élémentaires constituant un schéma de calcul ou de résolution de problème.

2.2 - Algorithmes et programmes

- Pour nous :
  - Un algorithme est une séquence précise et non ambiguë d'une suite d'étapes pouvant être exécutées de façon automatique par un ordinateur pour résoudre un problème
  - Spécification du schéma de calcul sous forme d'une suite finie d'opérations élémentaires obéissant à un enchaînement déterminé.

2.2 - Algorithmes et programmes



Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données, pour arriver en un nombre fini d'étapes, à un certain résultat, et cela indépendamment des données.

2.2 - Algorithmes et programmes

- Programme :
  - codage d'un algorithme afin que l'ordinateur puisse exécuter les actions décrites
  - doit être écrit dans un langage compréhensible par l'ordinateur
    - → langage de programmation
- Un programme est donc une suite ordonnée d'instructions élémentaires codifiées dans un langage de programmation

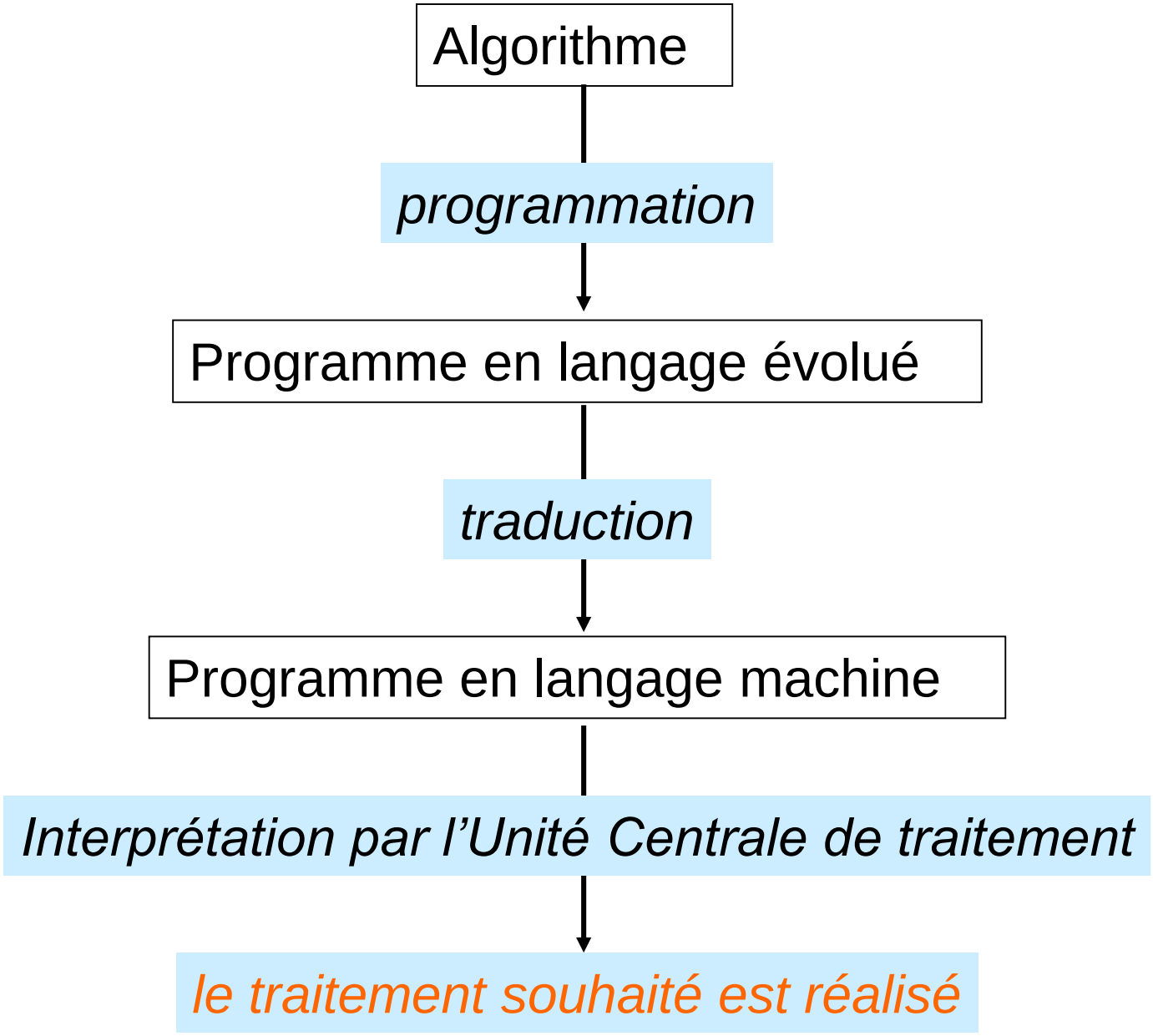
2.3 - Langages de programmation

- RAPPEL: Langage machine
  - langage binaire
  - ses opérations sont directement compréhensibles par l'ordinateur
  - propre à chaque famille d'ordinateur

- Ecriture des premiers programme en langage machine

2.4 - Importance des algorithmes

- Pour mener à bien un traitement sur un ordinateur il faut :
  1. Concevoir un **algorithme** qui décrit comment le traitement doit être fait
  2. Exprimer l'algorithme sous la forme d'un **programme** dans un langage de programmation adéquat
  3. Faire en sorte que l'ordinateur exécute le programme : **compilation**


Algorithme
programmation
Programme en langage évolué
traduction
Programme en langage machine
Interprétation par l'Unité Centrale de traitement
le traitement souhaité est réalisé

Introduction à l'algorithmique

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

- Un **algorithme** : description d'un cheminement (succession d'étapes dans le temps) conduisant à la résolution d'un problème donné

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 :

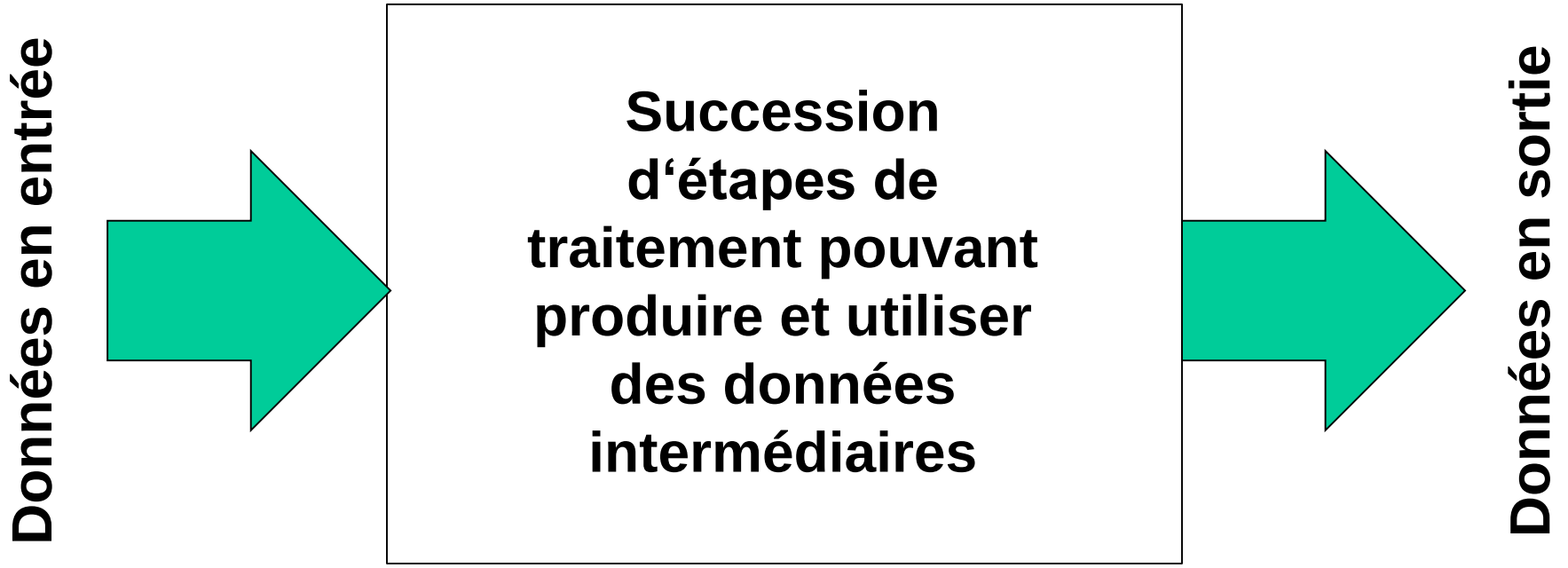
➢ les **données en entrée** du problème : ce qui est connu au départ, c'est-à-dire les hypothèses

➢ les **données en sortie/résultat** : ce qui est fourni comme résultats à la fin de la résolution du problème

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

Introduction à l'algorithmique

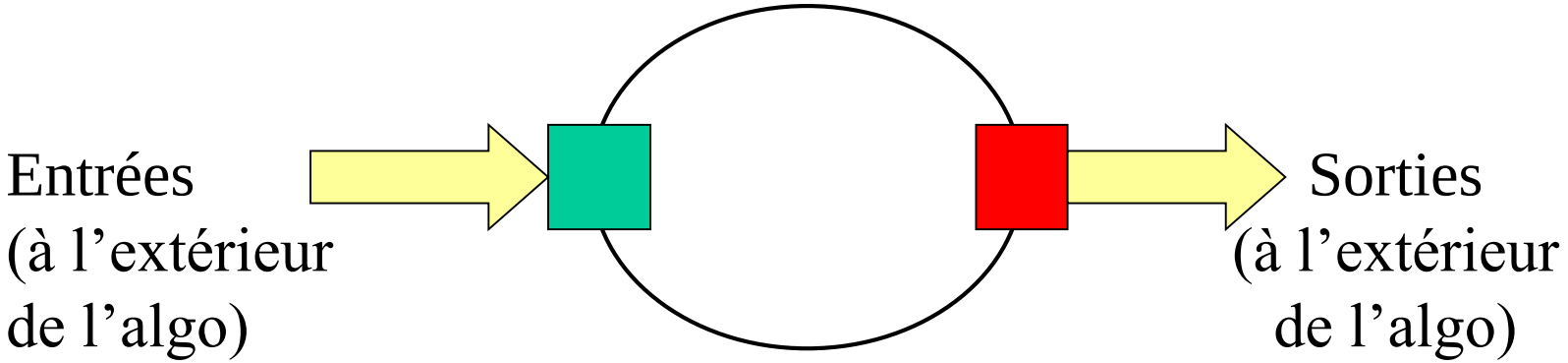
Algorithme



NB : le nombre de données varie en fonction du problème à résoudre

Introduction à l'algorithmique

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



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

Introduction à l'algorithmique

• **La variable en entrée** est généralement employée au sein de l'algorithme pour calculer des valeurs intermédiaires ou des valeurs en sortie/résultat

• **La variable en sortie/résultat** est généralement employée pour communiquer en dehors de l'algorithme un résultat calculé dans l'algorithme à partir des variables en entrée et des variables intermédiaires

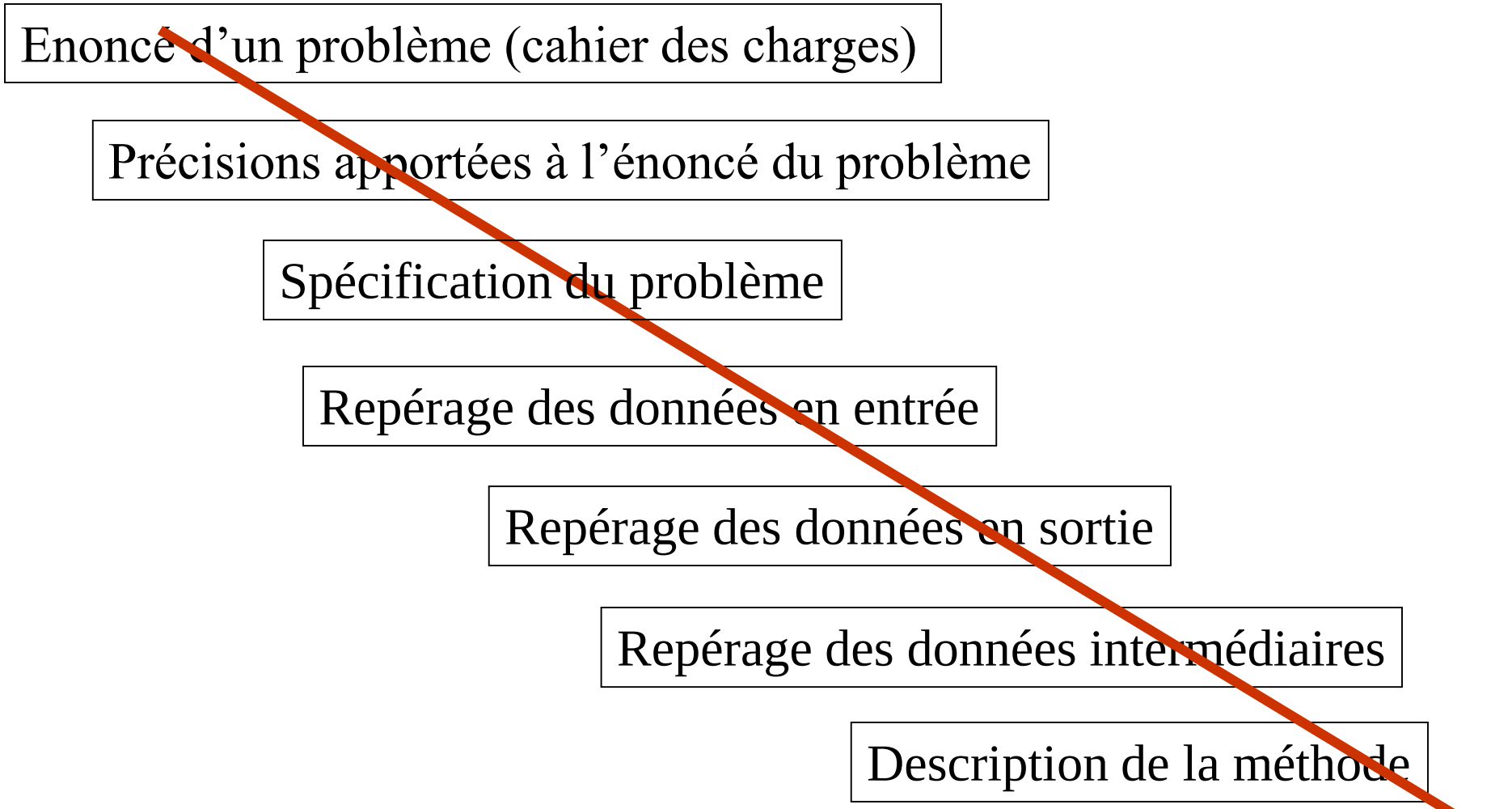
• **La variable en entrée/sortie** est généralement employée pour servir successivement à communiquer une valeur à l'algorithme et un résultat calculé envoyé a l'extérieur de l'algorithme

Introduction à l'algorithmique

• Parfois l'**énoncé du problème** (**cahier des charges**) est incomplet et mérite qu'on y ajoute des **précisions** (compléments d'informations) avant qu'une solution ne puisse être apportée

• Le problème posé par un utilisateur ou un client dans ses termes à lui doit être traduit dans des termes compréhensibles par les informaticiens : c'est la phase de **spécification du problème** donnant lieu 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

- Pour résoudre **un problème**, généralement on va le découper/décomposer en problèmes plus petits à résoudre : c'est la politique du « **diviser pour régner** » qui est employée
- Deux approches de réalisation de la solution :
  - ✓ **approche descendante** (top-down)
  - ✓ **approche ascendante** (bottom-up)

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 : on dit qu'on procède par **raffinements successifs**

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

– Affinement des algorithmes

- Un algorithme doit décrire précisément le traitement qu'il doit exécuter et s'assurer que **tous les cas de figures possible** ont bien été prévus.
- Exemple : algorithme permettant de calculer la durée d'un voyage à partir du tableau d'affichage des aéroports

1. Consulter l'heure de départ
2. Consulter l'heure d'arrivé
3. Soustraire l'heure de départ de celle d'arrivée

– Affinement des algorithmes

- Problèmes :
  - fuseaux horaires différents
  - Si un point applique l'heure d'Été et pas l'autre
- Pour éviter de telles erreurs le concepteur doit suivre une démarche rigoureuse et méthodique :
  - Affinement progressive de l'algorithme
  - Démarche descendant, top down
  - Technique du « diviser pour mieux régner »

– Affinement des algorithmes

- Exemple : robot domestique avec un algorithme de préparation d'une tasse de café soluble

1. Faire bouillir l'eau
2. Mettre le café
3. Ajouter l'eau dans les tasses

– Affinement des algorithmes

- 1. faire bouillir l'eau

peut être affinée en

  - 1.1. remplir la bouilloire d'eau
  - 1.2. brancher la bouilloire sur le secteur
  - 1.3. attendre l'ébullition
  - 1.4. débrancher la bouilloire

– Affinement des algorithmes

- 2. mettre le café dans la tasse

pourrait être affiné en

  - 2.1. ouvrir le pot à café
  - 2.2. prendre une cuiller à café
  - 2.3. plonger la cuiller dans le pot
  - 2.4. verser le contenu de la cuiller dans la tasse
  - 2.5. fermer le pot à café

– Affinement des algorithmes

- 3. ajouter de l'eau dans la tasse

pourrait être affinée en

3.1. verser de l'eau dans la tasse jusqu'à ce que celle-ci soit pleine

– Affinement des algorithmes

- 1.1. remplir la bouilloire d'eau

- peut nécessiter les affinements suivants:

- 1.1.1. mettre la bouilloire sous le robinet
- 1.1.2. ouvrir le robinet
- 1.1.3. attendre que la bouilloire soit pleine
- 1.1.4. fermer le robinet

– Affinement des algorithmes

- Affinement ne se fait pas dans le vide
- Savoir où s'arrêter
- **Connaître les capacités du processeur**
- Exemple :
  - Brancher la bouilloire, activité interprétable
  - Remplir la bouilloire, activité non interprétable
    $\Rightarrow$ affinement

– Affinement des algorithmes

- Processeur informatique : ordinateur, capacités d'interprétation connues
- **Le concepteur d'un algorithme doit donc affiner ce dernier jusqu'à ce que les étapes puissent être écrites à l'aide d'un langage de programmation**

2.1 – Ecrire un algorithme

- Calcul de l'intérêt et de la valeur acquise par une somme placée pendant un an
- L'énoncé du problème indique
  - Les données fournies: deux nombres représentant les valeurs de la somme placée et du taux d'intérêt
  - les résultats désirés: deux nombres représentant l'intérêt fourni par la somme placée ainsi que la valeur obtenue après placement d'un an.

2.1 - Exemple

- Formalisation de l'algorithme:En français
  1. prendre connaissance de la somme initiale et du taux d'intérêt
  2. multiplier la somme par le taux; diviser ce produit par 100; le quotient obtenu est l'intérêt de la somme
  3. additionner ce montant et la somme initiale; cette somme est la valeur acquise
  4. afficher les valeurs de l'intérêt et de la valeur acquise.

  - SI=somme initiale
  - T=taux d'intérêt (ex: 3 pour 3%)
  - I=intérets=S*T/100
  - SF=somme finale=S+I

2.1 - Exemple

- Formalisation de l'algorithme
  - En langage de description : pseudo code, LDA (Langage de Description Algorithmique)

écrire " Introduisez la somme initiale (en francs): "
Lire SI
écrire " Introduisez le taux d'intérêt (ex: 3 pour 3%): "
lire T
I ← SI* T / 100
SF ← SI + I
écrire " L'intérêt fourni est de " , **I** , "francs "
écrire " La somme après un an sera de " , **SF** , "francs "

Introduction à l'algorithmique

• Exemple 1 : Alice et Bob ont chacun une parcelle. La parcelle de Bob est carrée tandis que celle d'Alice est circulaire. Connaissant la taille du côté de la parcelle de Bob, quel doit être le rayon de celle d'Alice pour qu'on puisse parler d'égalité des pourtours ?

a) Spécification : Soit **c** la longueur du côté de la parcelle de Bob et soit **r** le rayon de la parcelle d'Alice. Il faut calculer la valeur que devrait avoir le rayon de la parcelle d'Alice pour qu'on puisse parler d'égalité des pourtours.

Introduction à l'algorithmique

b) Données/variables en entrée :

- **c** est de type réel, correspond à la longueur du côté de la parcelle carrée de Bob ;

c) Données/variables en sortie/résultat

- **r** est de type réel, correspond à la longueur du rayon que devrait avoir la parcelle d'Alice pour qu'il y ait égalité des pourtours ;

d) Données/variables intermédiaires

- **P** est de type réel, correspond au pourtour de la parcelle carrée de Bob ;

Introduction à l'algorithmique

e) Description de l'enchaînement logique

Début

P←c*4 /* calcul du périmètre du carré */

r ← P/(2*π) /* calcul du rayon du cercle dont la circonférence correspond au périmètre du carré */

Fin

ATTENTION !!!

Les instructions sont exécutées du début à la fin les unes après les autres sauf lors de la rencontre des instructions de rupture de séquence.

Introduction à l'algorithmique

•**PROGRAMME = 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.)

Introduction à l'algorithmique

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

Compilation & interprétation

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

- **Ces programmes sources sont ensuite :**

  - **soit traduits en PROGRAMMES EXECUTABLES pour pouvoir être exécutés par les ordinateurs, après une succession de traitements**

  - **soit directement fournis à des INTERPRETEURS**

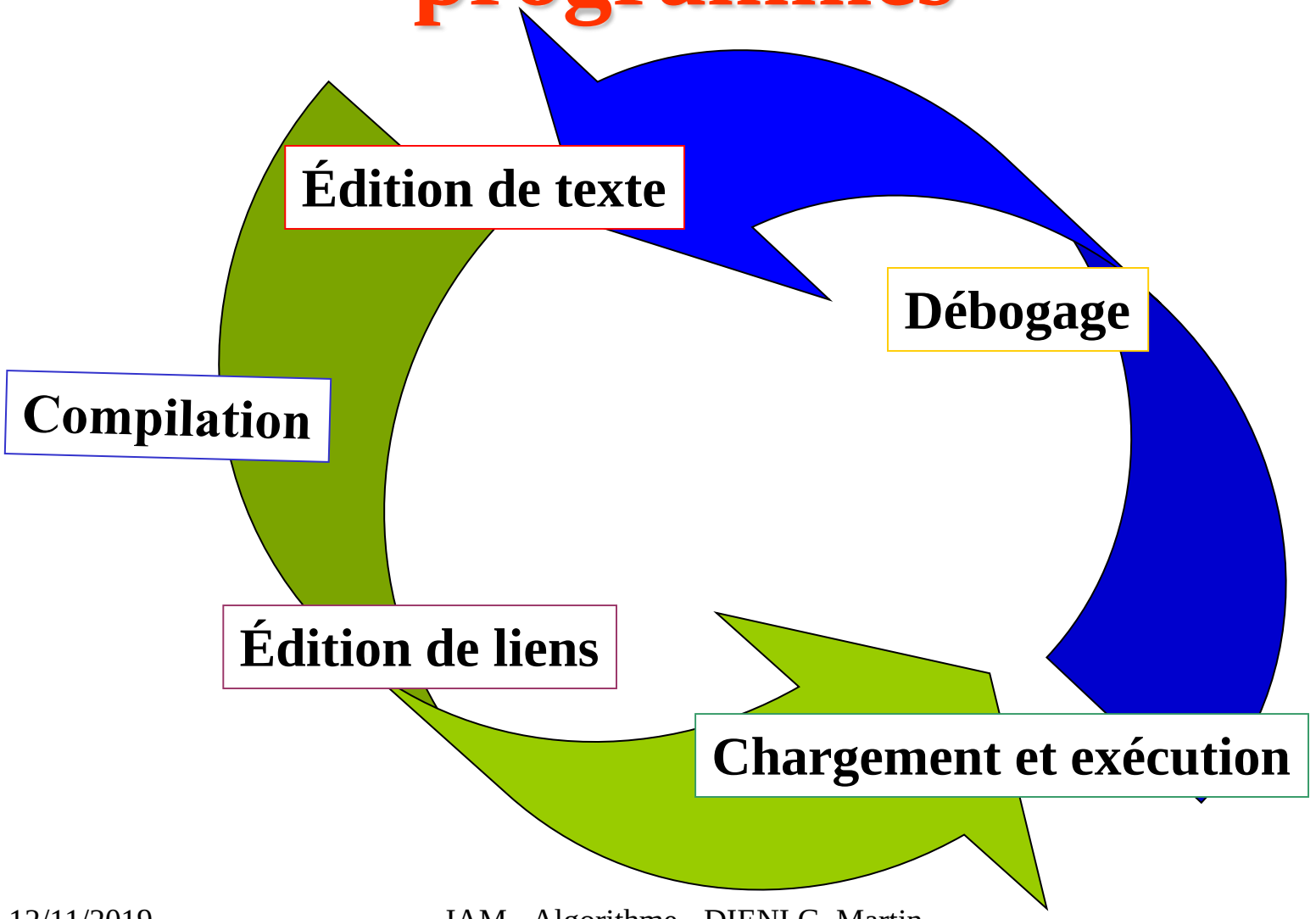
Production du code exécutable

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

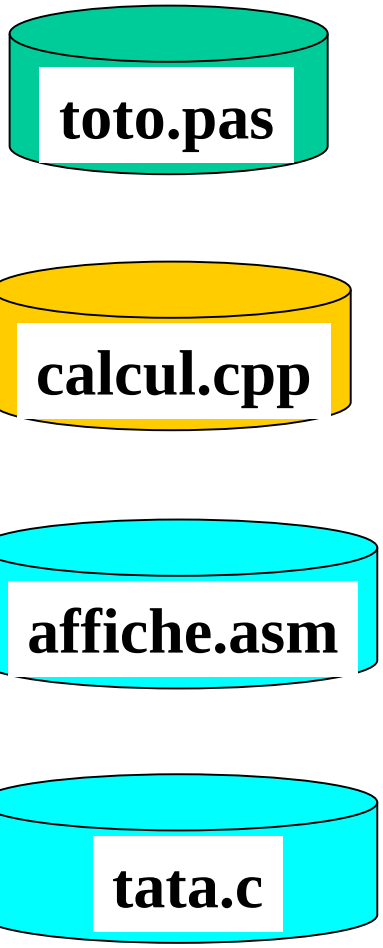
Interprétation

- **Le processus d'interprétation est employée lorsque**
  - **l'utilisateur souhaite écrire rapidement un petit programme jetable (maquette)**
  - **le source peut sans aucun problème être communiqué à l'utilisateur**
- **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**

Phases du développement de programmes

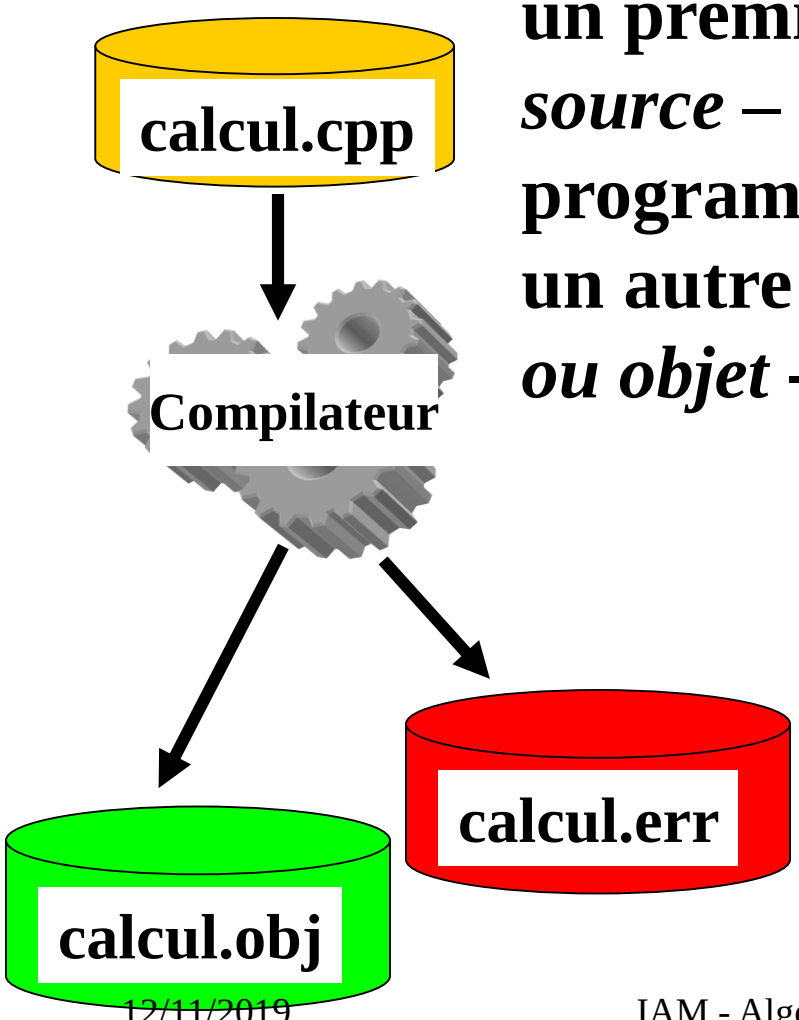


Édition de texte



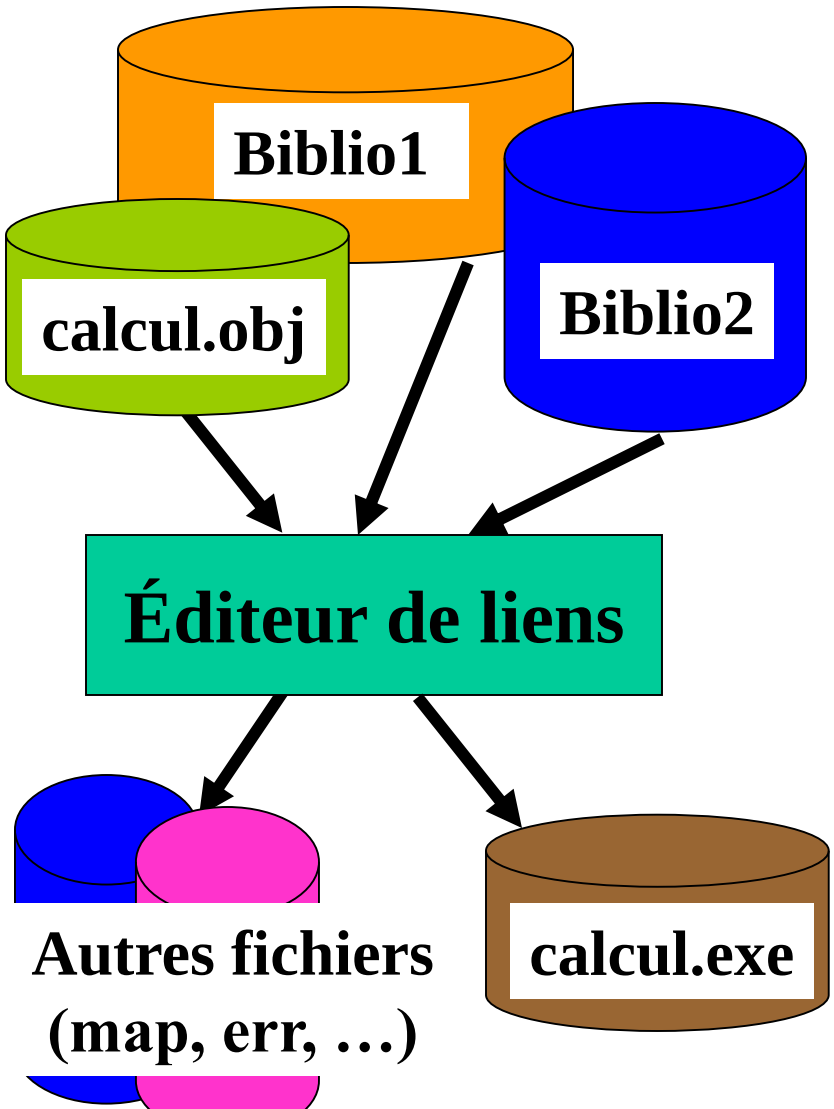
- **Éditeur de texte = *programme de saisie de texte* par l'utilisateur *sans mise en forme*. Ce texte va correspondre à un programme écrit dans un langage de programmation (respect de la syntaxe d'écriture du programme et des instructions)**
- **Un *fichier source* résulte de cette opération et porte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

Compilation



- Compilateur = programme lisant un programme écrit dans un premier langage – *langage source* – et le traduisant en un programme équivalent écrit dans un autre langage – *langage cible ou objet* -
- Lors de ce processus les erreurs rencontrées sont signalées à l'utilisateur dans un *fichier d'erreurs de syntaxe*
- A chaque langage de programmation est associé 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**

Introduction à l'algorithmique

- **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**

Introduction à l'algorithmique

QUESTIONS ???