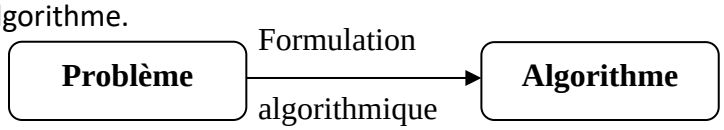
CHAPITRE II: DÉVELOPPEMENT D'UN PROGRAMME

I/ Introduction :

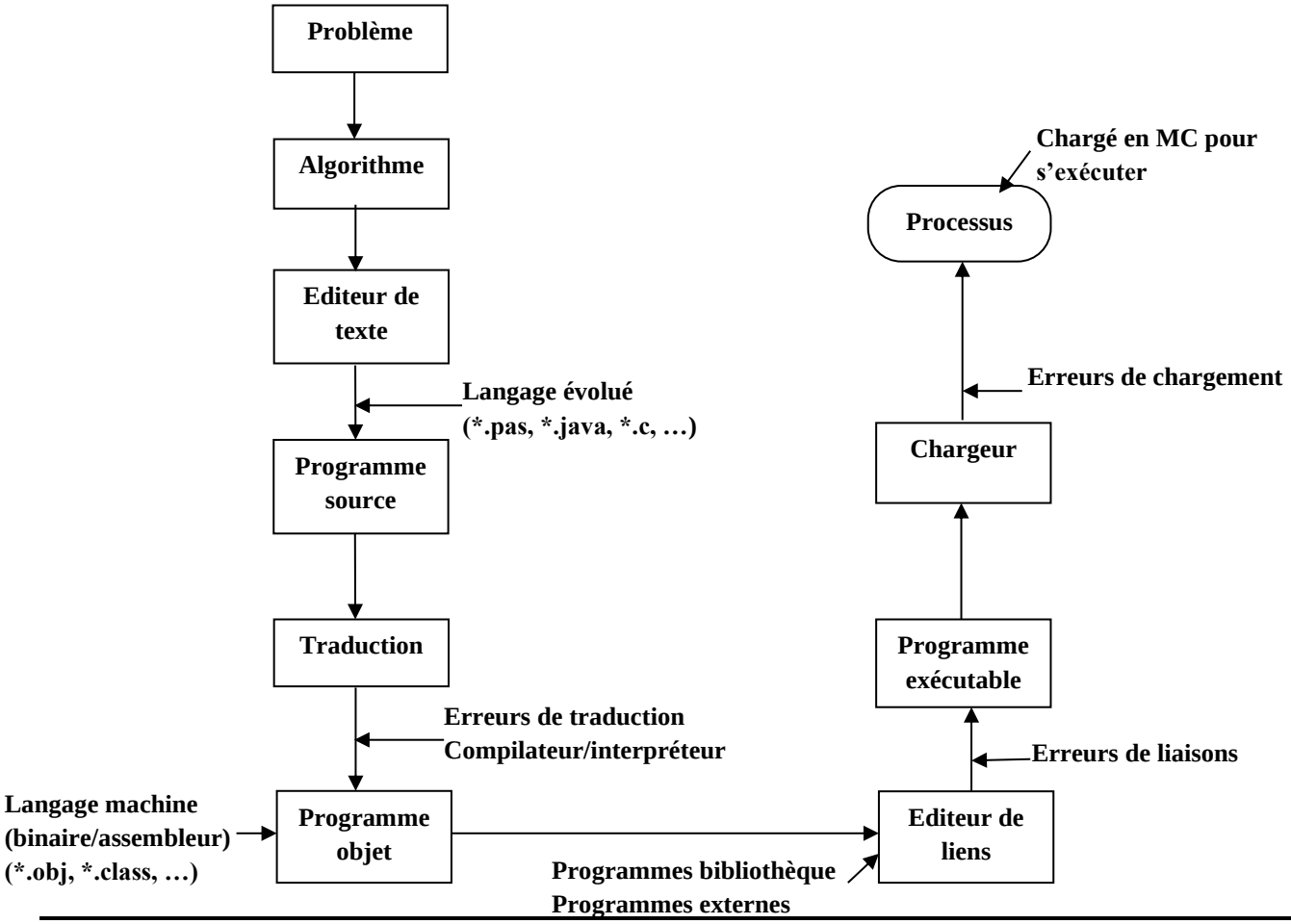
Pour qu'un problème puisse être interprété, exécuté et donne les résultats attendus par le processeur de la machine, il faut qu'il soit formulé dans un langage compréhensible par la machine et il faut que le système d'exploitation pilote son exécution.

Pour le faire, le programmeur commence au début d'écrire son problème d'une manière structurée sous forme d'un algorithme.



Cet algorithme est écrit en langage humain et pour le rendre exécutable par l'ordinateur, le programmeur fait le passer par plusieurs étapes tout en utilisant des outils de langage de programmation et des outils système.

II/ Les étapes de cheminement d'un programme :



Ces étapes permettent de bien traduire le code source en code machine tout en permettant au programmeur de corrige les erreurs apparues dans chaque étape.

1/ L'édition de programme :

C'est l'étape dont on saisit l'algorithme établit pour résoudre le problème tout en le traduisant à un programme écrit dans un langage évolué (exemple : Java, Pascal, Delphi, C++, …) ou dans le langage Assembleur. L'éditeur de texte est l'outil qui nous permet de réaliser cette étape, c'est un logiciel interactif soit associé au système, soit associé au langage de programmation. Il nous permet d'appliquer toutes les opérations nécessaires sur un fichier nommé « **Code source** »

Ce code source à une extension spécifique tout dépend du langage utilisé (*exemple* : *.pas, *.java, *.cpp, …)

Exemple : Bloc-notes, Wordpad, Edit de MS-DOS, Editeur du compilateur C, JEDPlus pour java, …

2/ La traduction en langage machine :

Le traducteur est un outil système qui permet de traduire les instructions écrites dans un langage de programmation vers des instructions écrites dans le langage machine (Binaire), on distingue deux types de traducteurs :

- *Le compilateur* :

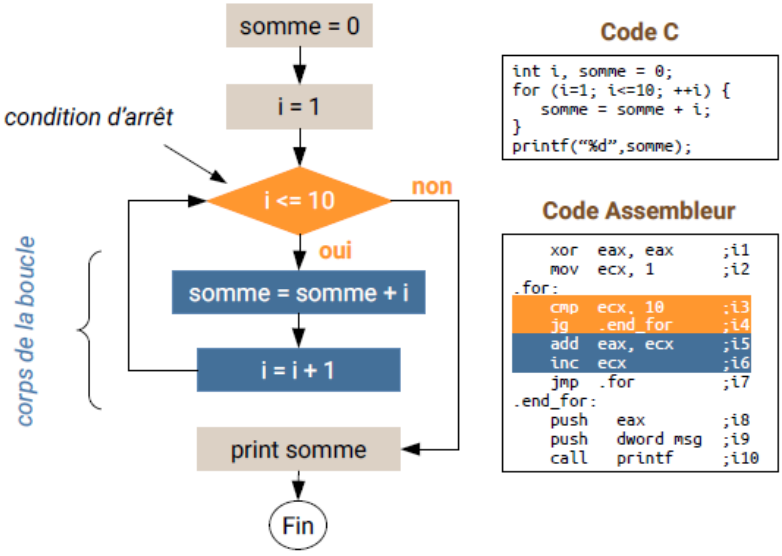
Est un logiciel de complexité grandissante, permettant de traduire un programme source vers le langage machine en passant par :

- **L'analyse lexicale** : le rôle de cette phase est la reconnaissance des unités lexicales (tokens), en inspectant le code source caractère par caractère. Ces unités peuvent être des nombres, des identificateurs, des mots clés, des opérateurs, … Chaque unité est décrite par un type (mot clé, identificateur, constante, opérateur, …) et une valeur (le nom qui figure dans le code source). Alors le résultat de cette étape :
  - La création de la table de symbole.
  - Elimination de blancs et de commentaires.
  - Signalisation des erreurs lexicales.
- **L'analyse syntaxique** : vérifie que le code source à compiler respecte bien les règles syntaxiques du langage. Le résultat est la création de l'arbre syntaxique en se basant sur les unités lexicales et l'ensemble des règles syntaxiques du langage. Elle met à jour la table des symboles en ajoutant les informations manquantes comme les types des identificateurs, et elle signale les erreurs dues au non-respect de la syntaxe du langage.
- **L'analyse sémantique** : qui vérifie la sémantique du programme. Cet analyseur utilise l'arbre syntaxique pour identifier les opérandes et les opérations et il utilise la table des symboles pour

identifier les types des opérandes. Il signale les erreurs sémantiques qui peuvent être faites dans le code.

- **L'optimisation de code**: L'optimisation du code est une stratégie de modification du programme qui s'efforce d'améliorer le code intermédiaire, de sorte qu'un programme utilise le moins de mémoire possible, minimise son temps d'exécution et offre une vitesse élevée.
  - Il s'agit de la quatrième étape d'un compilateur, qui vous permet de choisir d'optimiser ou non votre code, ce qui la rend réellement optionnelle.
  - Il permet de réduire l'espace de stockage et d'augmenter la vitesse de compilation.
  - Il prend le code source en entrée et tente de produire un code optimal.
  - Le fonctionnement de l'optimisation est fastidieux ; il est préférable d'employer un optimiseur de code pour accomplir cette tâche.
- **Transformation en code objet** : c'est la partie qui réalise effectivement la traduction du code source en code objet.

Lors de ces étapes le compilateur signale des erreurs à corriger qui peuvent apparaître dans chaque étape. Le résultat de cette phase est un fichier nommé code objet a une extension spécifique selon le langage de programmation (*exemple* : *.obj Pascal, *.class Java, …)



- *L'interpréteur* :

L'interpréteur est un logiciel qui permet de traduire le programme vers le langage machine et de l'exécuter en même temps.

Ce type de traducteur ne résulte pas un fichier contenant le code objet ce qui impose l'interprétation à chaque fois d'exécution (*exemple* : l'interpréteur HTML).

N.B : le compilateur définit deux types de codes dans le code objet :

- Le code absolu qui a une adresse fixe dans la mémoire.
- Le code translatable qui peut se mettre dans n'importe quel emplacement dans la mémoire.

L'opération de translation consiste à ajouter à chaque emplacement qui contient une adresse, la valeur de l'adresse effective de l'implantation finale dans la mémoire.

3/ <u>L'édition de liens</u> :

Lors de la traduction d'un programme en code objet, le compilateur associe à chaque instruction son type, dont on peut avoir :

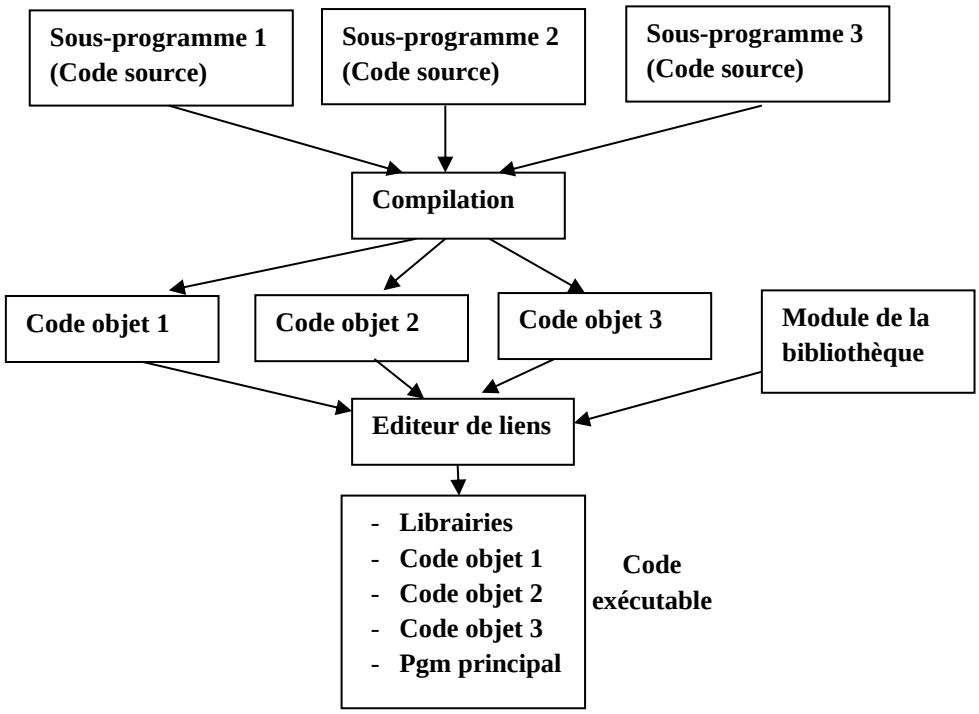
- Des données variables
- Des données constantes
- Des instructions
- Des appels à des procédures, des modules ou des variables externes.

Dans le dernier type, le programmeur peut référencer des structures ou des modules externes par rapport au module en question (il s'agit des références externes de la bibliothèque du langage ou personnelles). Alors le compilateur marque ces références sans les ramener et les ajouter au code objet.

Donc il associe à chaque référence soit :

- Un lien interne au programme, mais qui peut être accédé par d'autre modules → lien utilisable (définition externe).
- Un lien externe appartient à un autre module appelé dans ce programme → lien à satisfaire (référence externe).

L'éditeur de lien est un outil système qui a comme objectif d'accomplir la tâche du compilateur en ajoutant les codes objets de tous les liens à satisfaire dans le code objet principal. Alors l'éditeur de liens cherche l'origine de chaque référence externe et établit une table globale des modules contenant les informations : nom du module, taille et adresse d'implantation.



Le travail de l'éditeur de liens est basé sur la table de fonctions qui contient toutes les informations concernant la bibliothèque. On peut distinguer entre deux types des éditeurs de liens :

Editeur de lien statique : c'est l'édition de liens qui s'établit une fois pour toute et elle engendre un fichier résultat liant toutes les références externes avec le programme principal, nommé le fichier exécutable. Ce type des éditeurs de liens ne se refait pas à chaque exécution.

Exemple : l'éditeur de liens du langage Pascal → des fichiers *.exe.

Editeur de liens dynamiques : c'est l'édition de liens qui s'établit à chaque demande d'exécution du programme, elle n'engendre pas un fichier exécutable.

Exemple : l'éditeur de liens du langage Java

4/ Le chargement :

Après l'édition de liens, le programme est prêt à s'exécuter, pour faire il faut le charger dans la mémoire centrale. Cette phase est faite par un outil système nommé le chargeur. Cette étape consiste à lire les instructions en mémoire secondaire, et les transférer en mémoire centrale tout en lisant au début l'adresse de chargement fournit par l'éditeur de liens. Dans ce cas le programme devient un processus.

On distingue deux types de chargement :

Le chargement absolu : le code chargé ne doit être pas met dans un autre bloc de données différent de celui indiqué dans le code objet, (d'une autre façon recopier le code).

Le chargement relogeable (translatable) : le chargement peut se réaliser à n'importe quelle adresse dans la mémoire centrale, la façon de translater les adresses est basée sur les informations fournies par l'éditeur de liens.

5/ Le débogueur :

C'est un logiciel qui permet d'exécuter le programme pas à pas, d'afficher les valeurs des variables à tout moment et mettre en place des points d'arrêts sur des conditions ou des lignes du programme. Il offre au programmeur la possibilité de contrôler l'exécution et de détecter l'origine des erreurs non corrigées.