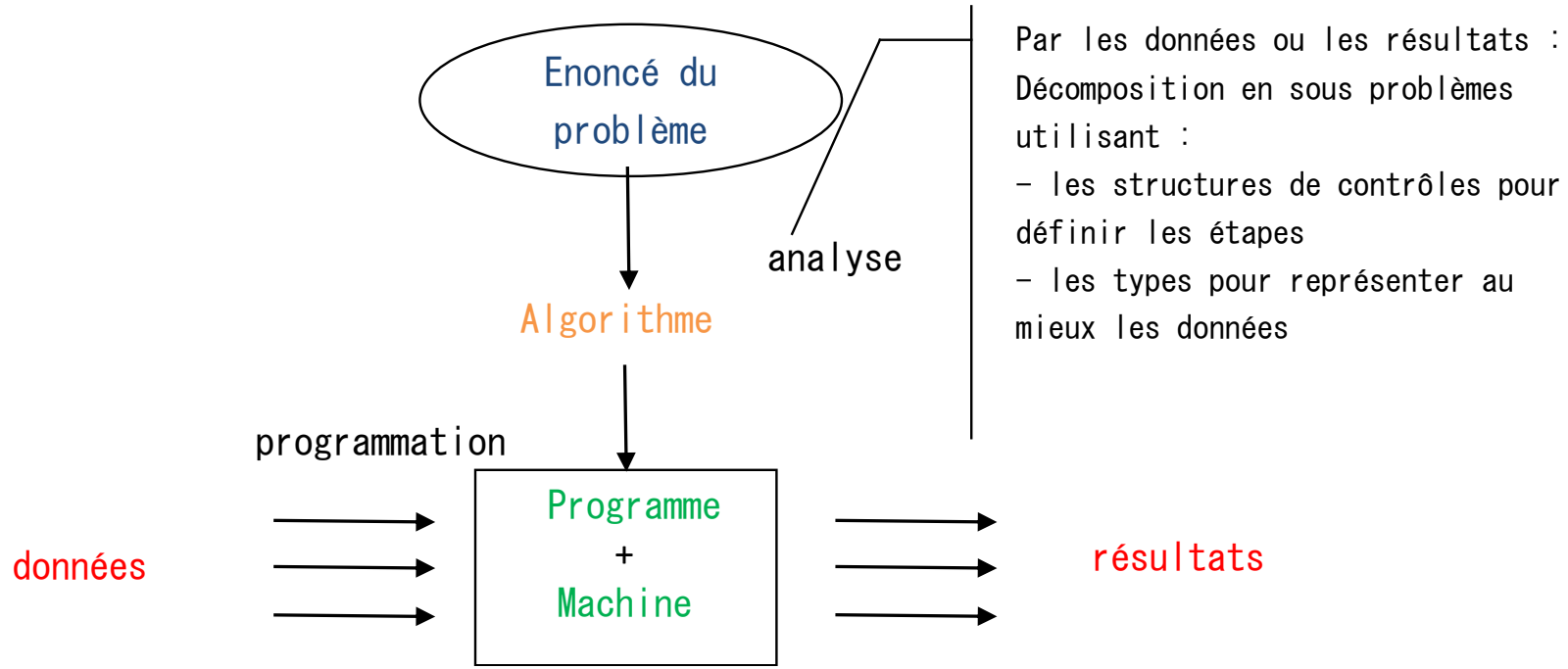
Chapitre 1

De l’Algorithme au Programme : Utilisation du langage C

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

I Introduction

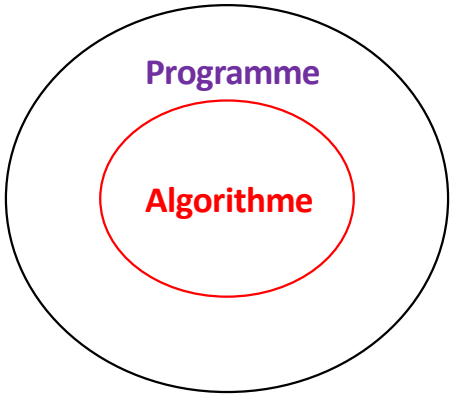


Par les données ou les résultats :
Décomposition en sous problèmes utilisant :
- les structures de contrôles pour définir les étapes
- les types pour représenter au mieux les données

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

II Introduction

- La traduction n'est possible que si les concepts et structures utilisés dans l'algorithme sont compris dans le langage de programmation (compris par la machine).
- Les deux fonctions, algorithme & programme, peuvent être schématisées comme suit :



Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

- Par rapport au langage algorithmique, le langage de programmation offre des concepts supplémentaires et qui lui sont spécifiques.

- Il existe différentes formes de langages de programmation qui sont adaptés à différentes formes de raisonnement.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

1) Les langages ***impératifs :***

- ✓basés sur la notion de variable
- ✓la variable peut changer d'état
- ✓la variable est représentée par une ou plusieurs cases mémoires

Exemple : Ada, C, Pascal, Fortran, Cobol, C++, Perl, PHP, Java, Python,…

Les formes d'algorithmes utilisées jusqu'à présent sont dites impératives. Cela implique que les algorithmes ne peuvent être traduits aisément que dans des langages à forme impérative.

2) Les langages ***fonctionnels :***

✓basés sur la notion de fonction et de variable (au sens mathématique)

✓la variable ne peut avoir qu'une seule valeur (donc ne change pas d'état)

Exemple : Lisp, ML, CAML, …

3) Les langages ***logiques :***

✓basés sur la notion de logique

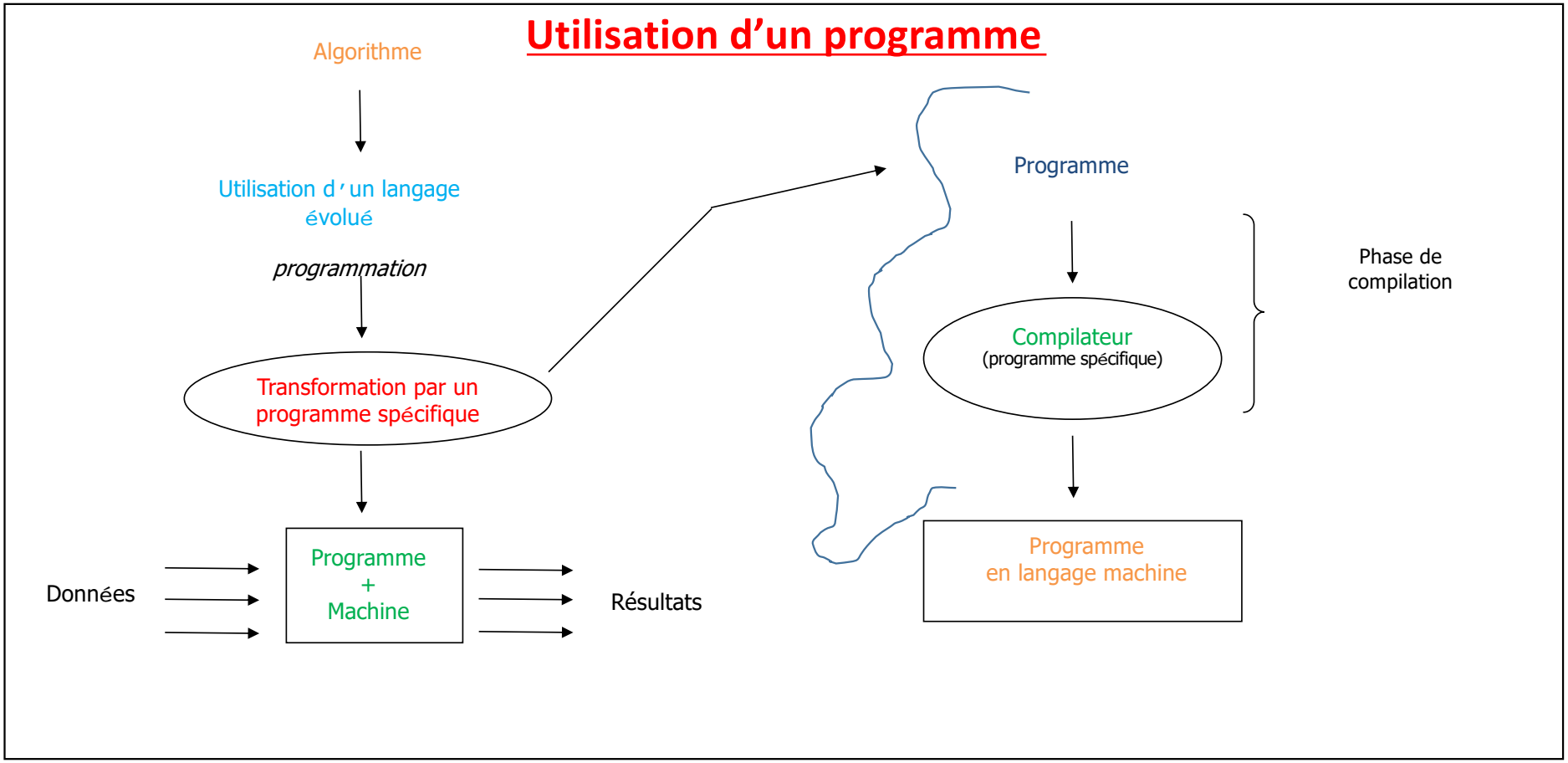
✓effectuent des raisonnements sur un ensemble de prédicats logiques

Exemple : Prolog, OZ, Clips, Python:PyPy

Ils se concentrent sur ce qui doit être calculé et non comment le calcul doit être fait (*programmation déclarative*)

D'autres formes de langages existent comme les langages à objets qui sont des extensions des langages impératifs comme C++, Smalltalk, Java, C#, Python, ...

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C



Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

III Le langage C

III.1 Historique

- Le langage **C** a été développé par Dennis Ritchie au début des années 70, pour servir de langage de développement au système d'exploitation UNIX (laboratoire Bell) créé par Kenneth Thompson.
- Pourquoi le nom **C**? A cause du langage B, initialement utilisé, qui s'est révélé assez limité pour l'écriture du S.E UNIX.
- UNIX est écrit à 90% en C.
- C'est un langage simple d'utilisation :
  - ✓ ensemble d'instructions réduit, syntaxe facilement maîtrisable.
  - ✓ Il possède un jeu très riche d'opérateurs. Cela permet un accès à la quasi-totalité des ressources-machine.
  - ✓ On peut écrire des programmes presque aussi efficaces qu'en assembleur tout en programmant de manière structurée.
  - ✓ C'est un langage faiblement typé ce qui le rend très permissif et peut conduire à adopter de très mauvaises habitudes de programmation.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Remarque :

- Le laxisme du langage et du compilateur n'est pas sans danger car on peut aboutir à des erreurs d'exécution parfois difficiles à déceler.

Exemple :

- un débordement de tableau non vérifié,
- une ponctuation mal placée ou oubliée

peuvent entraîner des comportements parfois déroutants.

- De ce fait, il faut programmer de manière **précise** et **claire**.
- Si aucun effort n'est fait au niveau de la **présentation** et de la **rigueur**, un programme **C** a tendance à être très rapidement **illisible**.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

III.2 Concepts généraux du langage

III.2.1 Structure générale d'un programme C

- Un programme source **C** est un ensemble de **modules (morceaux)**, qui peuvent être compilés séparément.
- Chaque module comporte un certain ensemble de **fonctions** et, éventuellement, des définitions de **variables globales** pouvant être spécifiées dans des **fichiers séparés** => *compilation séparée*
- Dans cet ensemble de modules, le programme source **C** comprend une fonction particulière qui constitue le ***programme principal*** et qui a toujours pour nom ***main***.
- L'exécution de tout programme **C** entraîne un appel à cette fonction : ***main*** est le point d'entrée.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

III.2.2 Exemple de programme simple écrit en C

Soit le programme suivant qui calcule et affiche la moyenne d'un étudiant à partir de deux notes.

```
/* importation des bibliothèques */

#include <stdio.h>   /* bibliothèque standard d'entrée-sortie */

int main()

{

 /* déclarations de variables */

 float noteM, noteF;

 float moy;

 /* corps du programme principal */

   printf("Saisir la note de Math : \n");

   scanf("%f", &noteM);

   printf("Saisir la note de Français : \n");

   scanf("%f", &noteF);

   moy = (noteM + noteF)/2.0;

   printf(" moyenne est de : %f", moy);

}
```

stdio.h : standard input/output, contient entre autres les instructions printf et scanf

main : nom du programme principal

{} : un bloc d'instructions est toujours défini entre 2 accolades

/* : marque de début d'un commentaire

***/** : marque de fin d'un commentaire

printf : instruction d'affichage à l'écran

scanf : instruction de saisie depuis le clavier

%f : format de saisie ou d'affichage d'une valeur de type réel

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

III.2.3 Composants d'un programme C

Un programme C est composé entre autres des groupes d'éléments suivants :

- Des identificateurs : un identificateur permet de nommer un objet du programme : une variable, une fonction, un type. Il est construit à partir de lettres (majuscules ou minuscules, non accentuées), de chiffres, du caractère _. Le premier caractère est une lettre.

- Des mots-clés : ce sont des mots réservés pour le langage et ne peuvent être utilisés comme identificateurs. Exemple : les noms de types, les structures de contrôle ou instructions.

- Des opérateurs : de comparaison, logiques, arithmétiques, …

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Convention d'écriture :

- la casse doit être respectée : il y a une différence entre majuscules et minuscules dans les identificateurs
- une ligne commentaire est définie entre /* et */ (selon le principe des parenthèses)
- On peut définir des commentaires sur plusieurs lignes
- Le symbole // permet de mettre en commentaire toute la ligne qui le suit
- Il est cependant préférable d'utiliser la 1° notation (plus jolie et plus propre). La notation // est surtout utilisée lors de la mise en œuvre d'un programme pour mettre en commentaire une ligne d'instruction temporairement mise de côté.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Quelques règles à respecter pour les commentaires :

- on place un commentaire n'importe où dans le fichier source
- un commentaire ne peut contenir la marque de fin (*/)
- les commentaires ne peuvent être imbriqués
- les commentaires ne peuvent pas couper un mot du programme en deux

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Syntaxe du programme principal :

[*directives au préprocesseur*] => *chaque directive commence par #*

[*déclarations de variables externes*]

[*déclarations de fonctions*] => *peuvent être placées après la fonction principale main*

type main ([arguments]) => *entête de la fonction principale*

{

déclarations des variables internes => *correspond au lexique*

liste d'instructions => *corps du programme principal, correspond aux définitions*

}

Remarque :

- le programme principal ***main*** est une fonction
- Comme toute fonction, elle doit renvoyer un résultat indiquant le code de terminaison : 0 en cas de succès, différent de 0 en cas d'anomalies ou d'erreurs fatales. Par conséquent le type de main est toujours un entier : ***int***. Et le renvoi du résultat est facultatif.
- Elle est généralement sans paramètres(optionnels)
- Il est toutefois possible de lui passer des paramètres directement à l'appel lorsqu'on l'exécute (*à voir éventuellement plus tard dans la partie dédiée aux fonctions)*

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Création et Compilation d'un programme simple sous linux en mode commande :

1ère étape :

- ❑ Saisie du programme : on dispose d'un éditeur
- ❑ Le nom du fichier de saisie aura pour extension : **.c**

Exemple : first_pg.**c**

2ème étape :

- ❑ Compilation :

gcc first_pg.**c**

- ❑ S'il y n'y a pas d'erreurs => création d'un fichier exécutable : **a.out** (nom généré par défaut)

- ❑ Pour donner un nom à son programme exécutable :

gcc -o first_pg first_pg.**c**

où first_pg est le nom du fichier exécutable qui peut être exécuté comme une commande système.

- ❑ Implicitement, le compilateur exécute 3 étapes : pré-compilation, compilation, edition de liens.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

La pré-compilation :

Cette phase est assurée par le *pré-processeur*.

Il effectue des transformations d'ordre purement lexical (textuel) dans le *fichier source C*.

Il prend en compte les directives qui lui sont spécifiques et qui commencent par le symbole **#** : inclusion d'autres fichiers sources, de librairies, remplacement des macro-définitions.

La compilation :

Elle permet la traduction du fichier généré par le *pré-processeur* dans le langage d'assemblage, qui est un *langage machine* de bas niveau ayant une forme lisible.

Elle est suivie d'une étape qui est l'assemblage qui va traduire toutes les instructions en *instructions machine directement compréhensibles par le processeur.*

Généralement, la compilation et l'assemblage se font dans la foulée l'une de l'autre. On obtient en résultat un *fichier objet* : **.o**

L'édition de liens :

Un programme peut être composé de plusieurs fichiers sources, dont, entre autres, des librairies de fonctions standards déjà écrites. Il faut lier entre eux les différents fichiers objets. La phase édition de liens produit le *fichier exécutable* composé de tous ces morceaux.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Syntaxe générale de la commande de complilation :

gcc [options] nom_fichier_source**.c** [**-l** nom_librairies]

([ ] : signifie que c'est optionnel)

Options du compilateur :

Les plus importantes sont :

- -o nom_fichier : spécifie le nom du fichier exécutable (par défaut : a.out).
- -c : supprime l'édition de liens; seul le fichier objet est généré.
- -I nom_répertoire : spécifie le répertoire de recherche des fichiers à inclure (fichiers d'entêtes), en plus du répertoire courant.
- -L nom_répertoire : idem pour les librairies pré-compilées, en plus du répertoire habituel où elles sont stockées.
- -E : seul le pré-processeur est activé. Le résultat est envoyé sur la sortie standard.
- -Wall : affiche tous les messages d'avertissements : warnings. Un warning n'interdit pas la génération d'un exécutable.
- -W : affiche les messages d'avertissements supplémentaires.

Chapitre 1 De l'Algorithme au Programme : Utilisation du langage C

Les librairies :

Si des librairies spécifiques sont utilisées dans le programme, chacune d'elles est définie par :

-l nom_librairie

Le système les recherche dans le répertoire des librairies pré-compilées.

A défaut, si elles ne s'y trouvent pas, on précise leur chemin d'accès avec l'option **–L**.

Exemple : l'utilisation de la librairie mathématique se fait avec : **-lm**

Remarque : **man gcc**

permet d'avoir toute la documentation en ligne sur la commande gcc.